T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HÜCRESEL SİNİR AĞI KÜTÜPHANESİNİN DONANIM GERÇEKLEMESİNE YÖNELİK OLARAK İNCELENMESİ

OLCAY KORKMAZ

YÜKSEK LİSANS TEZİ
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ ANABİLİM DALI
ELEKTRONİK PROGRAMI

DANIŞMAN
YRD. DOÇ. DR. NERHUN YILDIZ

İSTANBUL, 2015

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

HÜCRESEL SİNİR AĞI KÜTÜPHANESİNİN DONANIM GERÇEKLEMESİNE YÖNELİK OLARAK İNCELENMESİ

Olcay KORKMAZ tarafından hazırlanan tez çalışması 27.05.2015 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Yrd. Doç. Dr. Nerhun YILDIZ

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Yrd. Doç. Dr. Nerhun YILDIZ

Yıldız Teknik Üniversitesi

Prof. Dr. Tülay YILDIRIM

Yıldız Teknik Üniversitesi

Doç. Dr. Berna ÖRS YALÇIN

İstanbul Teknik Üniversitesi

ÖNSÖZ

Tez çalışmam süresince bana rehberlik eden tez danışmanım Nerhun Yıldız'a, bu zorlu yolda beni yalnız bırakmayan Serhat Gül, Serkan AKBULUT, Sercan SATIR ve Begüm ÖZYILMAZ'a ve hiçbir zaman benden desteğini esirgemeyen aileme sonsuz teşekkürler.

Mart, 2015

Olcay KORKMAZ

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

i	İki boyutlu HSA dizisinin satır indeksi
j	İki boyutlu HSA dizisinin sütun indeksi
k	Etki alanı içinde kalan hücrelerin satır indeksi
l	Etki alanı içinde kalan hücrelerin sütun indeksi
r	Etki alanının yarıçapı
t	Sürekli zaman değişkeni
n	Ayrık zaman değişkeni
M	İki boyutlu HSA dizisinin satır sayısı
N	İki boyutlu HSA dizisinin sütun sayısı
N	İterasyon sayısı
$C(i,j)$	i. satır ve j. sütunda yer alan hücre
$C(k,l)$	k. satır ve l. sütunda yer alan hücre
x_{ij}	$C(i,j)$ hücresinin durum değişkeni
y_{ij}	$C(i,j)$ hücresinin çıkış değişkeni
u_{ij}	$C(i,j)$ hücresinin giriş değişkeni
z_{ij}	$C(i,j)$ hücresinin eşik değeri
a_{kl}	Sabit geri besleme katsayıları
b_{kl}	Sabit giriş kontrol katsayıları
T_s	Örnekleme periyodu
$A(i,j;k,l)$	Çıkış geriliminden gelen geri besleme katsayıları
$B(i,j;k,l)$	Giriş kontrol geriliminden gelen kontrol katsayıları
$A_{i,j;k,l}$	HSA'nın geri besleme sinaptik ağırlıkları
$B_{i,j;k,l}$	HSA'nın giriş kontrol sinaptik ağırlıkları
$\hat{A}_{ij;kl}$	Doğrusal olmayan HSA'nın geri besleme sinaptik ağırlıkları
$\hat{B}_{ij;kl}$	Doğrusal olmayan HSA'nın giriş kontrol sinaptik ağırlıkları
$A^{\tau}{}_{ij;kl}$	Zaman gecikmeli HSA'nın geri besleme sinaptik ağırlıkları
$B^{\tau}{}_{ij;kl}$	Zaman gecikmeli HSA'nın giriş kontrol sinaptik ağırlıkları
$\boldsymbol{A}$	HSA'nın geri besleme şablonu
$\boldsymbol{B}$	HSA'nın giriş kontrol şablonu
$\boldsymbol{Y}$	HSA'nın çıkış şablonu
$\boldsymbol{U}$	HSA'nın giriş şablonu
$\overline{\boldsymbol{A}}$	Ayrık zamanlı HSA'nın geri besleme şablonu

$\bar{\boldsymbol{B}}$	Ayrık zamanlı HSA'nın giriş kontrol şablonu
$\mathbf{U}$	Giriş matrisi
$\mathbf{Y}$	Çıkış matrisi
$\mathbf{G}$	Ara değer matrisi
C	Lineer kapasite
R_x, R_y	Lineer dirençler
$I_{xu}(i,j;k,l)$	Gerilim kontrollü akım kaynağı
$I_{xy}(i,j;k,l)$	Gerilim kontrollü akım kaynağı
I_{yx}	Gerilim kontrollü akım kaynağı
v_{xij}	HSA elektriksel devresinin durum gerilimi
v_{uij}	HSA elektriksel devresinin giriş kontrol gerilimi
v_{yij}	HSA elektriksel devresinin çıkış gerilimi
$f(.)$	Çıkış fonksiyonu
$\circledast$	Şablon nokta çarpım operatörü

KISALTMA LİSTESİ

APU	A Processing Unit
ASIC	Application-Specific Integrated Circuit
AZ-HSA	Ayrık Zamanlı Hücresel Sinir Ağları
BPU	B Processing Unit
CNN	Cellular Neural Network
DVI	Digital Visual Interface
FPGA	Field Programmable Gate Array
FSR	Full Signal Range
HD	High-Definition
HSA	Hücresel Sinir Ağları
RAM	Random Access Memory
RTCNNP	Real-Time CNN Processor
SZ-HSA	Sürekli Zamanlı Hücresel Sinir Ağları
TSA	Tüm Sinyal Aralığı
UART	Universal Asynchronous Receiver/Transmitter
VLSI	Very-Large-Scale Integration
YSA	Yapay Sinir Ağları
xPU	x Processing Unit

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

HÜCRESEL SİNİR AĞI KÜTÜPHANESİNİN DONANIM GERÇEKLEMESİNE YÖNELİK OLARAK İNCELENMESİ

Olcay KORKMAZ

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Yrd. Doç. Dr. Nerhun YILDIZ

Görsel veri kullanımının günden güne yaygınlaşması bu verilerin analizinde rol oynayan görüntü işlemenin önemini de arttırmaktadır. Görüntü işleme yönteminin yüksek performansı, verilerin doğru olarak analiz edilebilmesinin de önünü açacaktır. Bu nedenle kullanılacak görüntü işleme aracının nasıl bir sonuç ortaya koyacağını önceden bilmek karşılaşılacak olası sorunların çözümünde önemli rol oynayacaktır.

Bu tez çalışmasında hücresel sinir ağlarıyla (HSA) görüntü işlemede referans bir kaynak olan şablon kütüphanesi İkinci Nesil Hücresel Sinir Ağı Emülatörüne (Real-Time Cellular Neural Network (CNN) Processor (RTCNNP-v2)) yönelik olarak incelenmiştir. RTCNNP-v2 literatürdeki full-HD görüntü işleyebilen tek donanımsal gerçekleme olmasının yanı sıra bugüne kadar rapor edilmiş en yüksek hıza sahip HSA emülatörüdür.

Ayrık zamanda gerçeklenen RTCNNP-v2 mimarisinin şablon kütüphanesiyle uyumluluğu incelenirken HSA'nın matematiksel modelleri ortaya konmuştur. Şablon kütüphanesinde bulunan uygulamalar tüm sinyal aralığı (TSA) modeli altında incelenmiştir. İncelemeler sonucu kütüphanede yer alan uygulamaların TSA model ile uyumluluğu çizelgeler halinde verilmiştir. Son olarak RTCNNP-v2'nin bir sonraki versiyonuna yönelik önerilere yer verilmiştir.

Anahtar Kelimeler: Hücresel sinir ağları, tüm sinyal aralığı modeli, RTCNNP-v2, şablon

ABSTRACT

HARDWARE-ORIENTED CELLULAR NEURAL NETWORK SOFTWARE LIBRARY INVESTIGATION

Olcay KORKMAZ

Department of Electronics and Communications Engineering

MSc. Thesis

Adviser: Assistant Professor Nerhun YILDIZ

Widespread use of visual data increases the importance of image processing in order to analyse images. It is obvious that the performance of image processing technics are directly related to accurate analysis. Thus, understanding behaviour of these technics plays an important role in solving potential problems.

In this thesis, template library, which is a reference software library in image processing with cellular neural networks, is analyzed in terms of one of the hardware implementation of cellular neural networks, Real-Time Cellular Neural Network (CNN) Processor-v2 (RTCNNP-v2). RTCNNP-v2 is unique with its capabaility of processing full-HD video streams. Beside that, it is fastest CNN impelementation reported to date.

First, mathematical models of CNN are explained in order to introduce full signal range model which is used at RTCNNP-v2 architecture. Then, templates are illustrated based on full signal range model (FSR). Results are given in tables and templates are classified according to compability of FSR. Finally, future work is discussed for the next generation of Real-Time Cellular Neural Network Processor.

Keywords: Cellular neural networks, FSR model, RTCNNP-v2, template library

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Yapay sinir ağları (YSA) 1943 yılında Pitts ve McCulloch tarafından ortaya atılmıştır [1]. Yapay sinir ağlarının özel bir alt kümesi olan Hopfield YSA ise 1982 yılında Hopfield tarafından önerilmiştir [2].

Hopfield YSA modeli temel alınarak ortaya atılan Hücresel Sinir Ağları 1988 yılında Leon O.Chua ve Lin Yang tarafından 1988 yılında ortaya atılmıştır [3].

Hücresel sinir ağlarının tüm sinyal aralığı (TSA) modeline yönelik ilk donanımsal tasarımı Rodriguez v.d. tarafından 1993 yılında önerilmiştir [4]. Bu yayında analog olarak önerilen TSA donanımsal modelinin yanında aynı zamanda sürekli zaman HSA uygulamalarına yönelik donanımlar da önerilmiştir.

Hücresel sinir ağları kütüphanesine yönelik donanımsal tasarım incelemesi 2008 yılında Garcia v.d. tarafından yapılmıştır [5]. Bu yayında şablon kütüphanesinin 2.1 versiyonu kullanılmış ve donanım gerçeklemelerine yönelik verimli konfigürasyonlar sunulmuştur. Konfigürasyonlar şablonlardaki katsayıların istatistiksel analizleri temel alınarak oluşturulmuş ve en az sayıda matematiksel hesaplama kullanılarak donanımın oluşturulmasına yönelik önerilerde bulunulmuştur.

Hücresel sinir ağlarının matematiksel modelleri Cimagalli ve Balsi tarafından 1993 yılında toplu olarak verilmiştir [6]. Bu yayında sürekli ve ayrık tüm HSA modelleri bir arada yer almıştır.

Linan v.d. tarafından geliştirilen ACE4K gerçeklemesine yönelik şablon kütüphanesini 2000 yılında yayınlanmıştır [7]. Bu kütüphane belirli bir donanıma özel olarak oluşturulmuş ilk kütüphanedir.

Software Library for Cellular Wave Computing Engines HSA ile görüntü işleme uygulamalarının toplandığı referans bir kütüphanedir ve 3. versiyonu 2010 yılında yayınlanmıştır [8]. Kütüphanenin 4. sürümünün alfa versiyonu ise Aralık 2014 tarihinde yayınlanmıştır [9].

Kayaer ve Tavşanoğlu hücresel sinir ağlarının FPGA ile gerçeklenmesine yönelik iş hattı modeli mimarisini 2008 yılında ortaya atmıştır [10]. Mimarinin ikinci versiyonu Yildiz tarafından [11,12] önerilmiştir.

1.2 Tezin Amacı

Hücresel sinir ağları birçok matematiksel model ile ifade edilebilir. Matematiksel modeller ifade edilirken ağın türüne göre veya giriş görüntüsü özelliğine göre denklemler oluşturulabilir. Şablon kütüphanesinde, görüntü işleme uygulamalarının hangi matematiksel modeller altında çalışabileceğine dair net bir bilgiye yer verilmez. Diğer taraftan donanımsal gerçeklemeler yapılırken tümleştirme açısından genellikle HSA'nın tüm sinyal aralığı (TSA) modeli referans alınır. İkinci Nesil Hücresel Sinir Ağı Emülatörü (RTCNNP-v2) TSA model referans alınarak tasarlanmış bir gerçeklemedir. Ancak bu tez çalışmasına kadar şablon kütüphanesindeki hangi uygulamaların RTCNNP-v2 üzerinde çalışıp çalışmayacağı araştırılmamıştır. Bu tez çalışması ile şablon kütüphanesinde yer alan uygulamalardan hangilerinin İkinci Nesil Hücresel Sinir Ağı Emülatörü üzerinde çalışıp çalışmayacağının tespiti amaçlanmıştır. Tüm sinyal aralığı modeli altında yapılacak bu incelemeler sonucu RTCNNP-v2'nin bir sonraki versiyonunda olması gereken özelliklerin bulunması da tezin bir diğer amacıdır. Aynı zamanda bu çalışma ile şablon kütüphanesindeki eksik veya yanlış tanımlamaların da ortaya çıkartılması amaçlanmıştır.

1.3 Orjinal Katkı

Tezin amacında da belirtildiği üzere bu tez kapsamında incelenen şablon kütüphanesi uygulamalarının İkinci Nesil Hücresel Sinir Ağı Emülatörü ile uyumluluğu konusu daha önce literatürde yer almamıştır. Bu inceleme literatürde ilk kez yer alacaktır.

Hücresel sinir ağlarının tüm sinyal aralığı modeli uygulama bazında daha önce incelenmiş olmasına rağmen kütüphanenin tamamının TSA modeli özelinde incelenmesi araştırmalara daha önce konu olmamıştır. Şablon kütüphanesinde yer alan uygulamaların tüm sinyal aralığı modeli ile uyumluluğu bu tez kapsamında yapılan çalışmalar sonucu literatürde ilk kez toplu olarak yer almıştır.

BÖLÜM 2

HÜCRESEL SİNİR AĞLARI

İnsanın biyolojik yapısı çağlar boyunca bilim için vazgeçilmez bir esinlenme kaynağı olmuştur. İnsan beyninin öğrenme yapısı ve öğrenme sonucu karar verme mekanizması bilim insanlarının araştırmalarına sıklıkla konu olmuş ve insan beynini taklit edebilen sistemler bilim insanlarının yanı sıra tüm insanoğlunun hayallerini süslemiştir. Bu hayalleri gerçekleştirmek adına ilk somut adım düzensizlikten nefret eden İngiliz matematikçi Charles Babbage tarafından atılmıştır. Nümerik çizelgeler oluşturulurken yapılan insan hatalarından fazlasıyla bıkan Babbage insan zekasına sahip makinalar konusunda kafa yormuş ve günümüz bilgisayarlarının da atası olan *Fark Makinesini* tasarlamıştır. *Fark Makinesi* bu alanda atılmış önemli adımlardan biri olsa da öğrenme yeteneğine sahip bir tasarım değildir. 1943 yılına gelindiğinde Pitts ve McCulloch insan beyninin nöronlarını matematiksel olarak modellemeyi başarmış [1] ve Babbage'ın *Fark Makinesine* öğrenme yeteneği kazandıracak bir buluşa imza atmıştır. İnsanoğlunun yeryüzünde var olup düşünmeye başladığı ilk andan bu yana nasıl düşündüğümüze dair ipucu veren bu keşif Yapay Sinir Ağları (YSA) olarak adlandırılmıştır. YSA'nın türlerinden biri olan Hopfield YSA ise 1982 yılında J. Hopfield tarafından önerilmiştir [2]. Hücresel Sinir Ağları ise (HSA), ağdaki bütün hücrelerin birbirine bağlanması üzerine kurulu olan Hopfield YSA'nın özel bir alt kümesidir. Leon O.Chua ve Lin Yang tarafından 1988 yılında ortaya atılan Hücresel Sinir Ağları [3], Hopfield YSA modelinden farklı olarak insan beynindeki gibi bölgesel bağlantılara sahip hücrelerden oluşmaktadır.

Görüntü işleme özellikle tasarım, imalat, güvenlik, tıp ve mimari gibi bir çok alanda kullanılmakta ve öneminini günden güne arttırmaktadır. Art arda sıralanan bu alanlarda

kullanılması şöyle dursun artık herkes Instagram gibi sosyal paylaşım siteleri sayesinde farkında bile olmadan görüntü işlemektedir. Bilgi paylaşımında ve kullanımında görsel veriye bu kadar ihtiyaç duyulduğu bir dönemde bu verilerin işlenmesi de kaçınılmaz bir araştırma konusu olarak yer almaktadır.

Bu tez çalışmasında görüntü işleme yöntemlerinden biri olan 'her pikseli komşularını da hesaba katarak işleme' HSA ile gerçekleştirilmiştir. İşlenecek görüntüler HSA ile görüntü işleme uygulamalarının toplandığı referans bir kütüphane olan şablon kütüphanesinden alınmış ve HSA donanımsal gerçeklemelerinden İkinci Nesil Hücresel Sinir Ağı Emülatörüne yönelik incelemelere tabi tutulmuştur.

Hücresel sinir ağları (HSA) uzayda tanımlanmış birbirine bağlı komşu hücreler sisteminden oluşan paralel bir hesaplama modelidir. HSA ilk olarak 1988 yılında Leon O.Chua ve Lin Yang tarafından ortaya atılmıştır [3].

Hücresel sinir ağlarının en önemli ve yaygın kullanım alanı görüntü işleme uygulamalarıdır. Diğer görüntü işleme yöntemlerine nazaran daha basit olan matematiksel ifadesi ve paralel yapısını nedeniyle HSA görüntü işlemede önemli avantajlar sağlamaktadır.

HSA ile görüntü işlerken, görüntüdeki her bir piksel hücresel ağdaki her bir hücreye karşı düşürülür. Ağdaki her kararlı denge durumu ise iki boyutlu görüntüyü tanımlar. Ağın kararlı sonuca yakınsama süresinin devrenin boyutundan bağımsız olması görüntü işlemede tercih edilmesine neden olan avantajlardan bir diğeridir.

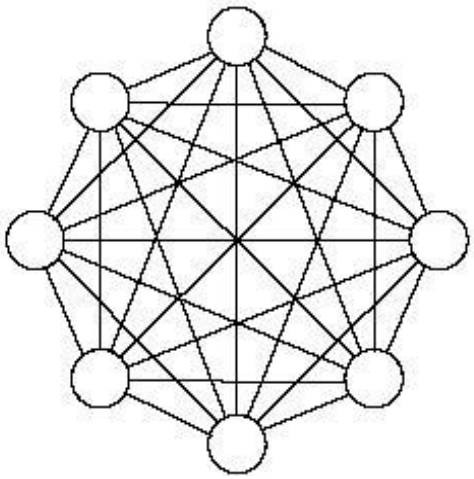
HSA'da şablonların, giriş ve sınır koşullarıyla beraber değiştirilmesi renk ve derinlik algılama, hareket kestirimi, gürültü yok etme gibi birçok görüntü işleme uygulamalarının aynı yonga seti üzerinde gerçeklenmesine olanak sağlar.

2.1 Temel Tanımlar

HSA ile ilgili temel tanımlar bu başlık altında yapılmıştır.

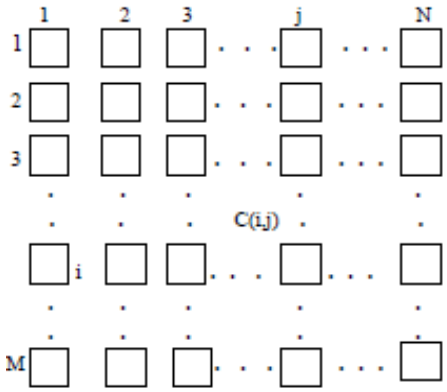
Tanım 2.1 *Standart HSA Mimarisi*, Hopfield sinir ağlarından esinlenilmiştir. Hopfield sinir ağlarındaki bütün hücreler birbiriyle doğrudan bağlantılıyken HSA sistemini

oluşturan her bir hücre en yakın komşuluğundaki hücre ile doğrudan, diğer hücrelerle dolaylı olarak bağlıdır.



Şekil 2. 1 Hopfield sinir ağı

Bir hücresel sinir ağı sistemi N-boyutta konumlanmış hücreler dizisinden oluşur. Hücreler üçgen, dikdörtgen veya altıgen düzlemlerde iki veya üç boyutta konumlanabilirler. Her bir hücre r-komşuluğunda diğer hücrelere bağlı olup çoklu girişe ve tek bir çıkışa sahiptir. Genel olarak kullanılan HSA mimarisi M harfi satırı, N harfi sütünu niteleyecek şekilde M x N boylarında, iki boyutlu kartezyan koordinat sisteminde yerleştirilmiş hücrelerden oluşur. Kartezyan koordinat indisleri $i = 1,2,3 \dots M$ ve $j = 1,2,3 \dots N$ şeklindedir ve $i.$ satır $j.$ sütundaki hücre $C(i,j)$ ile ifade edilir. Bu tez çalışmasında da referans alınacak iki boyutlu standart HSA mimarisi Şekil 2.2'de gösterilmiştir.

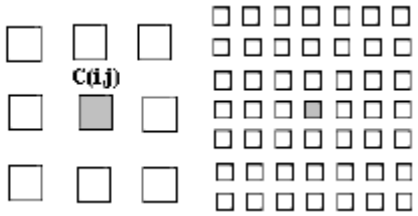


Şekil 2. 2 Standart HSA mimarisi

Tanım 2.2 *Etki küresi* $S_r(i,j)$, $C(i,j)$ hücresinin r yarıçapındaki komşularının oluşturduğu kümeyi ifade eder. r pozitif bir tam sayı olmak üzere;

$$S_r(i,j) = \{C(k,l) | \max_{1\le k\le M, 1\le l\le N}\{|k-i|, |l-j| \le r\}\} \quad (2.1)$$

koşulunu sağlayan tüm hücreler etki küresi içerisinde yer alır. Seçilen *r* yarıçapına göre hücrenin komşularıyla olan sinaptik ağ sayısı belirlenmiş olur. Etki küresi $(2r+1)$ x $(2r+1)$ şeklinde de gösterilebilir. Şekil 2.3'de farklı etki kürelerine sahip HSA yapıları gösterilmiştir.



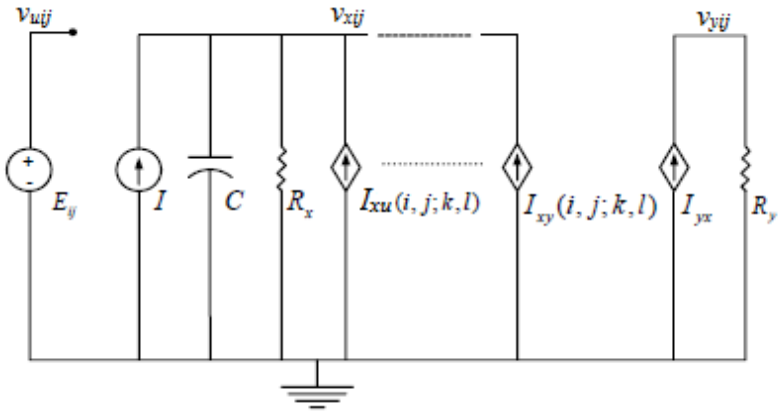
(a) (b)

(a) r=1 için etki küresi (b) r=3 için etki küresi

Şekil 2. 3 Farklı etki yarıçapları için HSA dizileri

$r > N/2$ ve $M = N$ koşulları sağlandığından HSA'daki her bir hücre ağdaki diğer hücreye de bağlanmış olur. Bu durum Tanım 2.1'de sözü edilen Hopfield sinir ağına karşılık düşer.

Tanım 2.3 *HSA hücresinin elektriksel modeli* bir lineer kapasite (C), iki lineer direnç (R_x, R_y), hücrenin eşik gerilimini belirleyen bağımsız akım kaynağı (I), p komşu hücre sayısı olacak şekilde $2p$ adet $I_{xu}(i,j;k,l)$ ve $I_{xy}(i,j;k,l)$ gerilim kontrollü akım kaynakları ve yine gerilim kontrollü bir akım kaynağı olan I_{yx}'ten oluşur. Sürekli zaman HSA'da yer alan bir hücrenin elektriksel modeli Şekil 2.4'te gösterilmiştir.



Şekil 2. 4 Bir HSA hücresinin elektriksel modeli

Devrenin çıkış gerilimi v_{yij}, devredeki tek lineer olmayan eleman I_{yx}'e bağlıdır ve çıkış fonksiyonuna bağlı olarak aşağıdaki gibi formüle edilir;

$$I_{yx} = \frac{1}{R_y} f(v_{xij}) \tag{2.2}$$

$$v_{yij} = R_y I_{xy} \tag{2.3}$$

Elektriksel modelde komşu hücrelerin etkisi iki adet gerilim kontrollü bağımlı akım kaynağıyla ifade edilmiştir. Bir hücrenin komşu hücrelerle olan ilişkisi, kontrol gerilimi v_{ukl}'den gelen $B(i,j;k,l)$ kontrol katsayıları;

$$I_{xu}(i,j;k,l) = B(i,j;k,l) v_{ukl} \tag{2.4}$$

ve çıkış geriliminden gelen geribesleme katsayıları $A(i,j;k,l)$;

$$I_{xy}(i,j;k,l) = A(i,j;k,l) v_{ykl} \tag{2.5}$$

ile ifade edilir.

2.2 Sürekli Zamanlı HSA'nın Matematiksel Modelleri

Standart HSA modelinin yanında HSA'nın diğer sürekli zamanlı modelleri de bu başlık altında verilecektir.

2.2.1 Chua-Yang Matematiksel Modeli

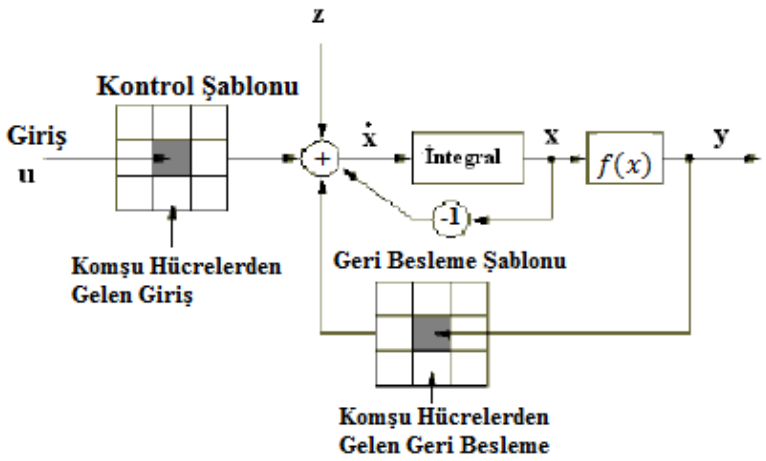
Hücresel sinir ağında yer alan bir hücrenin Şekil 2.4'te verilen analog devre modelinde Kirschoff gerilim ve akım yasaları uygulanırsa hücrenin matematiksel ifadesi

$$C\frac{dv_{xij}(t)}{dt} = -\frac{1}{R_x}v_{xij}(t) + \sum_{kl\in S_r(i,j)} A(i,j;k,l).v_{ykl} + \sum_{kl\in S_r(i,j)} B(i,j;k,l).v_{ukl} + I \quad (2.6)$$

$$v_{yij}(t) = \frac{1}{2}(|v_{xij}(t)+1| - |v_{xij}(t)-1|) \quad (2.7)$$

$1 \le i \le M, 1 \le j \le N$

şeklinde olur. Leon O.Chua ve Lin Yang HSA'nın teorisini ortaya koyarken bu modeli öne sürdüğünden (2.6) denklemi Chua-Yang matematiksel modeli olarak literatürde yerini almıştır. Bu model aynı zamanda HSA'nın orijinal matematiksel modeli olarak kabul edilir.



Şekil 2. 5 Hücrenin blok diyagramı

2.2.2 Konumdan Bağımsızlık ve Konuma Bağımlılık Kavramları

Chua-Yang modelinde verilen (2.6) denkleminin notasyonunu bu tezde de kullanılacağı haliyle

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} a_{k-i,l-j}y_{kl}(t) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} b_{k-i,l-j}u_{kl} + z_{ij} \quad (2.8)$$

şeklinde değiştirmek mümkündür. Burada a_{kl} bir önceki çıkışın etkisini taşıyan geri besleme katsayısını, b_{kl} mevcut girişin etkisini taşıyan giriş katsayısı, u_{kl} hücrenin

girişini, y_{kl} hücrenin çıkışını ve x_{ij} hücrenin durumunu ifade eder. Analog devre modelinde verilen bağımsız akım kaynağı I'nın yerini ise eşik değeri olarak adlandırılan z katsayısı almıştır. $k, l \in S_r(i,j)$ olmak üzere

$$k - i = \hat{k} \tag{2.9}$$

$$l - j = \hat{l} \tag{2.10}$$

eşitlikleri tanımlanırsa (2.8) denklemi

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} a_{\hat{k},\hat{l}} y_{\hat{k}+i,\hat{l}+j}(t) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} b_{\hat{k},\hat{l}} u_{\hat{k}+i,\hat{l}+j} + z_{ij} \tag{2.11}$$

haline gelir. Konvolüsyonun yer değiştirme özelliğinden yararlanıldığında terimler

$$\sum_{k=-r}^{r}\sum_{l=-r}^{r} a_{\hat{k},\hat{l}} y_{\hat{k}+i,\hat{l}+j}(t) = \sum_{k=-r}^{r}\sum_{l=-r}^{r} a_{\hat{k}+i,\hat{l}+j} y_{\hat{k},\hat{l}}(t) \tag{2.12}$$

$$\sum_{k=-r}^{r}\sum_{l=-r}^{r} b_{\hat{k},\hat{l}} u_{\hat{k}+i,\hat{l}+j} = \sum_{k=-r}^{r}\sum_{l=-r}^{r} b_{\hat{k}+i,\hat{l}+j} u_{\hat{k},\hat{l}} \tag{2.13}$$

şeklinde düzenlenmiş olur ve (2.8) denklemi

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} a_{kl} y_{i+k,l+j}(t) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} b_{kl} u_{i+k,l+j} + z_{ij} \tag{2.14}$$

olarak tekrar tanımlanabilir. (2.14) denklemi geri besleme katsayısı a_{kl}, giriş kontrol katsayısı b_{kl} ve eşik terimi z, kartezyan koordinat indisleri olan i ve j'ye bağlı olmadığından *'Konumdan Bağımsız HSA'* olarak isimlendirilir. Diğer bir deyişle ağda yer alan herhangi bir hücrenin sinaptik bağlantılarındaki ağırlıklar hücrenin ağ içerisindeki yerleşiminden bağımsız olarak belirlenebiliyorsa HSA konumdan bağımsızdır.

Hücrenin sinaptik bağlantılarındaki ağırlıklar hücrenin ağ içerisindeki yerleşiminden bağımsız olarak belirlenemiyorsa matematiksel denklemi

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} a_{i+k,j+l} y_{i+k,l+j}(t) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} b_{i+k,j+l} u_{i+k,l+j} + z_{ij} \tag{2.15}$$

halini alır. Bu tür HSA yapıları *'Konuma Bağımlı HSA'* olarak adlandırılır. Konuma bağımlı HSA (2.15)'te verilen şekilde kartezyan koordinat indislerine bağlı olabileceği gibi şablonun durum veya girişle bağlı değişmesiyle de ortaya çıkabilir. Bu şekil bir konuma bağımlı HSA yapısında, şablonda yer alan k, l indisli katsayılar duruma veya girişe bağlı olarak değişir.

2.2.3 Doğrusal Olmayan ve Zaman Gecikmeli HSA Matematiksel Modeli

Standart HSA modeli olarak kabul edilen Chua-Yang modeli doğrusal olmayan çıkış fonksiyonuna sahiptir ve bu da standart HSA modelinin aslında karakteristiğinin bir sonucu olarak doğrusal olmadığını gösterir. Ancak literatürde orijinal HSA modeli doğrusal olarak kabul edilir. Burada doğrusallıktan kasıt Tanım 2.3'te belirtildiği üzere gerilim kontrollü akım kaynaklarının doğrusallığıdır. HSA'yı gerçekten doğrusal olmayan bir modele dönüştürmek için gerilim kontrollü akım kaynakları (2.4) ve (2.5)'ten farklı olarak;

$$I_{xu}(i,j;k,l) = \hat{B}_{ij;kl}\left(v_{ukl}, v_{uij}\right) + B^{\tau}{}_{ij;kl} v_{ukl}(t-\tau) \tag{2.16}$$

$$I_{xy}(i,j;k,l) = \hat{A}_{ij;kl}\left(v_{ykl}, v_{yij}\right) + A^{\tau}{}_{ij;kl} v_{ykl}(t-\tau) \tag{2.17}$$

şeklinde yazılmalıdır. Burada $\hat{A}_{ij;kl}$ ve $\hat{B}_{ij;kl}$ doğrusal olmayan bağlantıları temsil eder ve en çok iki değişkenin sürekli fonksiyonlarıdır. $A^{\tau}{}_{ij;kl}$ ve $B^{\tau}{}_{ij;kl}$ ise zamanla değişen girişleri temsil eder ve gerçek sabitlerdir. Doğrusal olmayan ve zaman gecikmeli konumdan bağımsız HSA durum denklemi aşağıda verilmiştir.

$$\dot{x}_{ij}(t) = -x_{ij} + \sum_{k=-r}^{r}\sum_{l=-r}^{r} \hat{a}_{kl} y_{i+k,l+j}(t) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} \hat{a}_{kl} y_{i+k,l+j}(t-\tau)$$
$$+ \sum_{k=-r}^{r}\sum_{l=-r}^{r} \hat{b}_{kl} u_{i+k,l+j}(t) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} \hat{b}_{kl} u_{i+k,l+j}(t-\tau) + z_{ij} \tag{2.18}$$

Doğrusal olmayan ve zaman gecikmeli modelin kullanılması gerçek dünya problemlerinin çözümüne katkı sağlamaktadır. Örneğin; donanım gecikmelerini modellemek için veya hareketli bir nesnenin hareketini tayin edebilmek için zaman gecikmekli model, genliğe bağlı değişen sinaptik bağlantılar için doğrusal olmayan model kullanılabilir.

2.2.4 Diğer Matematiksel Modeller

HSA'nın en yaygın olarak karşılaşılan modellerinden biri de kompleks katsayılı HSA modelidir. Geri besleme katsayısı a_{kl}, giriş kontrol katsayısı b_{kl} ve eşik terimi z'nin kompleks değerler alabildiği bu modele örnek olarak Gabor benzeri süzgeçler verilebilir.

Çok katmanlı HSA yapılarında ise bir hücre içerisinde tek katmanlı yapıdan farklı olarak birden fazla durum değişkeni olabilir. Bu çeşit bir HSA'nın kullanım alanı olarak ise çok katmanlı süzgeç yapıları örnek gösterilebilir.

Yukarıda bahsedilenlerin dışında HSA'nın birçok matematiksel modeli bulunmaktadır [6]. Ancak diğer matematiksel modeller bu tezin kapsamı dışında olup çalışmanın devamında kullanılmayacaktır.

2.2.5 Tüm Sinyal Aralığı Modeli

Tüm sinyal aralığı (TSA) (Full Signal Range (FSR)) modeli hücresel sinir ağlarının analog gerçeklemelerini kolaylaştırmak amacıyla Rodriguez-Vazquez tarafından önerilmiştir [4]. Standart HSA modeli olarak kabul edilen Chua-Yang modelinden farklı olarak $x_{ij}(t) \triangleq y_{ij}(t)$ tanımıyla durum değerleri de giriş ve çıkış değerleri ile aynı aralık arasına sıkıştırılmıştır. Böylelikle SZ-HSA ve ayrık zamanlı HSA'ya göre daha az karmaşık bir hücre yapısı elde edilmiş ve VLSI (Very-Large-Scale Integration) gerçeklemelerinde güç tüketimi ve alandan tasarruf edilmiştir. Pratikte bütün sürekli zaman HSA uygulamalarında HSA'nın TSA modeli kullanılmaktadır.

TSA modeli sürekli veya ayrık zamanlı bütün matematiksel modellere uygulanabilir. Bu tez kapsamında kullanılacak olan konumdan bağımsız HSA'ya uygularsak (2.14) denklemi

$$\dot{x}_{ij}(t) = -y_{ij}(t) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} a_{kl} y_{i+k,l+j}(t) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} b_{kl} u_{i+k,l+j} + z \quad (2.19)$$

haline gelir. Tezin devamında ağırlıklı olarak bu model kullanılacaktır.

2.2.6 HSA'da Şablon Kavramı

Konumdan bağımsız HSA'da sinaptik ağırlıklar hücrenin ağ içerisindeki yerleşiminden bağımsız belirlenebildiğinden kullanılırken $\boldsymbol{A}$ ve $\boldsymbol{B}$ sinaptik ağırlıkları konumdan ve zamandan bağımsız olduğundan şablonlarla ifade edilebilir. Şablonlar çoğunlukla **3 x 3** boylarında seçilir. Böyle bir şablonda $S_1(i,j)$ etki alanı

$$S_1(i,j) = \begin{matrix} C(i-1,j-1) & C(i-1,j) & C(i-1,j+1) \\ C(i,j-1) & C(i,j) & C(i,j+1) \\ C(i+1,j-1) & C(i+1,j) & C(i+1,j+1) \end{matrix} \tag{2.20}$$

olarak yazılabilir. $C(i,j)$ hücresi **3 x 3** komşuluk içerisinde çıkıştan, girişten ve eşik teriminden gelen sinaptik bağlantılara sahiptir.

Çıkıştan gelen katkı $\boldsymbol{A}_{i,j;k,l}$ geri besleme şablonunun çıkış değişkeni ile konvolüsyonundan elde edilir. $\boldsymbol{Y}_{ij}$, y_{ij} çıkış değişkeninin etki alanı matrisini ve ⊛ simgesi şablon nokta çarpım operatörünü ifade etmek üzere

$$\sum_{C(k,l)\in S_r(i,j)} \boldsymbol{A}_{i,j;k,l} y_{kl} \triangleq \boldsymbol{A} \circledast \boldsymbol{Y}_{ij} \tag{2.21}$$

ve

$$\boldsymbol{A} \circledast \boldsymbol{Y}_{ij} \triangleq \begin{matrix} a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ a_{0,-1} & a_{0,0} & a_{0,1} \\ a_{1,-1} & a_{1,0} & a_{1,1} \end{matrix} \circledast \begin{matrix} y_{i-1,j-1} & y_{i-1,j} & y_{i-1,j+1} \\ y_{i,j-1} & y_{i,j} & y_{i,j+1} \\ y_{i+1,j-1} & y_{i+1,j} & y_{i+1,j+1} \end{matrix} \tag{2.22}$$

tanımları yapılabilir. Burada şablon nokta çarpım operatörü (⊛) uzaysal konvolüsyona karşılık gelir. Bu nedenledir ki $\boldsymbol{Y}_{ij}$ matrisi *M x N* boyutundaki çıkış görüntüsünün üzerinde 3 x 3 bir pencere gezdirilerek elde edilebilir. Bu yüzden $\boldsymbol{Y}_{ij}$ matrisine '$C(i,j)$ Hücresindeki Çıkış Görüntüsü' de denir.

Girişten gelen katkı ise $\boldsymbol{B}_{i,j;k,l}$ giriş kontrol şablonunun giriş değişkeni ile konvolüsyonundan elde edilir. $\boldsymbol{B}$, u_{ij} giriş değişkeninin etki alanı matrisi ⊛ simgesi şablon nokta çarpım operatörünü göstermek üzere

$$\sum_{C(k,l)\in S_r(i,j)} \boldsymbol{B}_{i,j;k,l} y_{kl} \triangleq \boldsymbol{B} \circledast \boldsymbol{U}_{ij} \tag{2.23}$$

ve

$$\boldsymbol{B} \circledast \boldsymbol{U}_{ij} \triangleq \begin{matrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & b_{0,0} & b_{0,1} \\ b_{1,-1} & b_{1,0} & b_{1,1} \end{matrix} \circledast \begin{matrix} u_{i-1,j-1} & u_{i-1,j} & u_{i-1,j+1} \\ u_{i,j-1} & u_{i,j} & u_{i,j+1} \\ u_{i+1,j-1} & u_{i+1,j} & u_{i+1,j+1} \end{matrix} \tag{2.24}$$

tanımları yapılabilir. Çıkıştan gelen katkıya benzer şekilde $\boldsymbol{U}_{ij}$ matrisi de *M x N* boyutundaki giriş görüntüsünün üzerinde 3 x 3 bir pencere gezdirilerek elde edilebilir. Bu yüzden $\boldsymbol{U}_{ij}$ matrisine '$C(i,j)$ Hücresindeki Giriş Görüntüsü' de denir.

Eşik teriminden gelen katkı ise z_{ij} konumdan bağımsız olduğundan $z_{ij} \triangleq z$ olarak ifade edilir.

Bu üç etkinin altında konumdan bağımsız HSA şablonlar yardımıyla

$$\dot{x}_{ij} = -x_{ij} + \boldsymbol{A} \circledast \boldsymbol{Y}_{ij} + \boldsymbol{B} \circledast \boldsymbol{U}_{ij} + z \tag{2.25}$$

olarak tanımlanabilir.

2.2.7 Ayrık Zamanlı HSA Modelleri

Hücresel sinir ağları hem sürekli hem de ayrık zamanda uygulanabilen dinamik bir sistemdir. Bu özelliğin bir sonucu olarak sürekli zamanda tanımlanan matematiksel modeller ayrıklaştırılarak ayrık zamanlı HSA (AZ-HSA) modelleri türetilebilir.

Bu tez kapsamında kullanılacak olan konumdan bağımsız TSA modelini ayrık zamanlı olarak ifade edebilmek için [12]'de olduğu gibi sürekli zaman eşitlikleri zaman domeninde örneklenirse;

$$x_{ij}(t)]_{t=nT_s} = x_{ij}(nT_s) \triangleq x_{ij}(n) \tag{2.26}$$

$$\dot{x}_{ij}(t)]_{t=nT_s} = \dot{x}_{ij}(nT_s) \triangleq \dot{x}_{ij}(n) \tag{2.27}$$

$$y_{ij}(t)]_{t=nT_s} = y_{ij}(nT_s) \triangleq y_{ij}(n) \tag{2.28}$$

eşitlikleri elde edilir. Elde edilen eşitlikler ileri Euler yaklaşıklığı;

$$\dot{x}_{ij}(n) \cong \frac{x_{ij}(n+1) - x_{ij}(n)}{T_s} \tag{2.29}$$

ile düzenlenirse AZ-HSA için durum denklemleri;

$$x_{ij}(n+1) = x_{ij}(n) + T_s(-x_{ij}(n) + \boldsymbol{A}\circledast \boldsymbol{Y_{ij}}(n) + \boldsymbol{B}\circledast \boldsymbol{U_{ij}} + z) \quad (2.30)$$

$$y_{ij}(n) = f\left(x_{ij}(n)\right) = \frac{1}{2}\left(\left|x_{ij}(n)+1\right| - \left|x_{ij}(n)-1\right|\right) \quad (2.31)$$

olarak elde edilir.

Ayrık zamanlı HSA modellerinde buraya kadar yazılan denklemler konumdan bağımsız sürekli zamanlı modelin ayrıklaştırılma işlemlerini ifade etmektedir. Tüm sinyal aralığı modelinin karakteristiğini taşıması için $y_{ij}(n) \triangleq x_{ij}(n)$ olarak tanımlanıp, bu tanımın bir sonucu olarak $y_{ij}(n+1) \triangleq f\left(x_{ij}(n+1)\right)$ olarak düzenlenirse (2.30) denklemi

$$x_{ij}(n+1) = (1-T_s)y_{ij}(n) + T_s\boldsymbol{A}\circledast \boldsymbol{Y_{ij}}(n) + T_s\boldsymbol{B}\circledast \boldsymbol{U_{ij}} + T_s z \quad (2.32)$$

haline gelir. Burada matematiksel eşitlikleri düzenlemenin yanı sıra $y_{ij}(n) \triangleq x_{ij}(n)$ varsayımıyla yeni bir AZ-HSA modeli ortaya konmuştur. Ortaya konan bu yeni matematiksel modelin geri besleme ve kontrol şablonu katsayılarıyla birlikte eşik değeri de

$$\bar{a}_{kl} = \begin{cases} (1-T_s) + T_s a_{kl}\ , & k,l = 0, \\ T_s a_{kl}, & diğer \end{cases} \quad (2.33)$$

$$\bar{b}_{kl} = T_s b_{kl}\ , \quad (2.34)$$

$\bar{z} = T_s z$ olarak tanımlanırlarsa (2.30) eşitliği

$$x_{ij}(n+1) = \bar{\boldsymbol{A}}\circledast \boldsymbol{Y_{ij}}(n) + \bar{\boldsymbol{B}}\circledast \boldsymbol{U_{ij}} + \bar{z} \quad (2.35)$$

haline gelir. Elde edilen (2.32) eşitliği $y_{ij}(n+1) \triangleq f\left(x_{ij}(n+1)\right)$ olarak düzenlenen eşitlikle birleştirilirse konumdan bağımsız ayrık zamanlı TSA modeli çıkış denklemi

$$y_{ij}(n+1) = f(\bar{\boldsymbol{A}}\circledast \boldsymbol{Y_{ij}}(n) + \bar{\boldsymbol{B}}\circledast \boldsymbol{U_{ij}} + \bar{z}) \quad (2.36)$$

olarak tanımlanır. Bu eşitlikten de görültüğü üzere $x_{ij}(n)$'in içerdiği tüm bilgiler TSA modelde $y_{ij}(n)$'e aktarılmış ve $x_{ij}(n)$'i saklamaya gerek kalmamıştır. Bu tezde bahsi geçen şablon kütüphanesindeki görüntü işleme uygulamaları, ayrık zamanlı TSA model referans alınarak incelenmiştir.

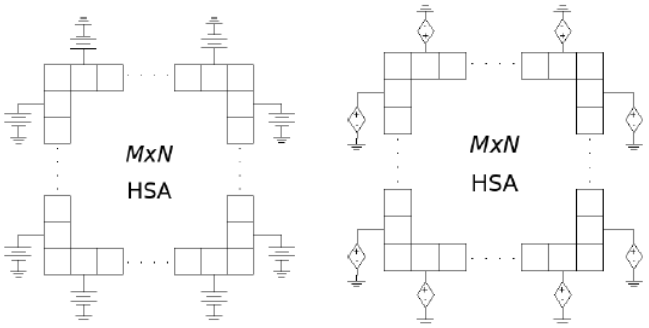
2.3 Sınır Koşulları

Hücresel sinir ağlarında yer alan hücreleri normal hücreler ve sınır hücreleri şeklinde ifade edersek $(2r+1)$ adet komşusu olan hücreler normal olarak sınıflandırılabilir. Bunun dışında kalan $(2r+1)$ adetten daha az komşuluğa sahip olan hücreler ise sınır hücreleri olarak tanımlanabilir ve M x N'lik alanın dışında kalan bu $C(i,j)$ hücrelerine verilecek değerler sınır koşullarını oluşturur. Sınır hücrelerindeki komşu hücre eksikliği yalnızca sınır hücrelerini değil dolaylı olarak ağın davranışını da etkiler. Bu problemin çözümü için r komşuluğuna sahip bir HSA $(M+2r)$ x $(N+2r)$ boylarında bir dizinin ortasına yerleştirilmiş gibi düşünülebilir. Bu şekilde bir varsayımla ağdaki sinaptik bağlantılar tamamlanmış olur. Ağa eklenen ve etki küresini tamamlayan bu sanal hücrelerin her biri için sanal durum, sanal giriş, sanal çıkış ve sanal eşik değeri tanımlanır. İşte bu sanal parametreler çeşitli sınır koşullarıyla belirlenirler. $r=1$ komşuluğa sahip bir HSA'da en yaygın olarak kullanılan sınır koşulları aşağıda açıklanmıştır.

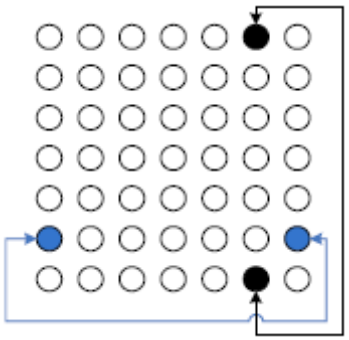
Sabit (Dirichlet) Sınır Koşulları: Her sanal hücrenin durum ve girişlerine önceden tanımlanmış sabit değerler atamasıyla sağlanır. Atanan bu sabit değerler genelde sıfır olmakla değerler keyfi bir sabit de olabilir. Donanımsal gerçeklemelerde bu sabit değerler Şekil 2.6(a)' daki gibi bir gerilim kaynağı veya toprak olarak düşünülebilir.

Zero-Flux (Neumann) Sınır Koşulları: Sanal hücrelerin durum ve giriş değerleri en yakın gerçek komşusunun durum ve giriş değerleri olarak kabul edilir.

Periyodik (Toroid) Sınır Koşulları: Ağın ilk ve son satırı ile ilk ve son sütunu birbirinin aynısı seçilir. Bu yapısı nedeniyle ağ kendi üzerine yuvarlanan bir halka şeklini alır.



(a) Sabit (Dirichlet) sınır koşulları (b) Zero-Flux sınır koşulları



(c) Periyodik sınır koşulları

Şekil 2. 6 HSA'da yaygın kullanılan sınır koşulları

BÖLÜM 3

İKİNCİ NESİL HÜCRESEL SİNİR AĞI EMÜLATÖRÜ

Klasik yapay sinir ağları sistemlerinde bağlantı sayısının fazla ve bağlantılar arası mesafelerin uzak olması tümleştirme açısından ciddi sıkıntılar doğurmaktadır. Hücresel sinir ağlarının ise yerel olarak birbirine bağlı hücrelerden oluşması, çok geniş ölçekli tümleşimleri (Very-Large-Scale Integration (VLSI)) kolaylaştırmakta ve tasarlanan hücrenin ağın istenen noktasına taşınmasına imkan vermektedir. Kari tarafından ortaya atılan 'Cellular Automata' teorisi [13] hücrelerin birbirinden eşit uzaklıklarla yerleştirilmesinin yolunu açmakta ve devre üzerinde hücreler arasındaki bağlantı yollarını fiziksel olarak daha uygulanabilir hale getirmektedir.

HSA'nın literatürdeki ilk tümleşik devre uygulaması Şekil 2.4'te verilen analog hücre modeli üzerinden geliştirilmiştir [14]. Analog devreler, doğası gereği paralel yapıya sahip olduğundan dijital gerçeklemelere göre daha yüksek yakınsama oranı sağlamaktadır. Bu bilgiden yola çıkıldığında sürekli zamanda tasarlanmış bir donanım ayrık zamanda tasarlanmış bir donanıma göre ilk bakışta daha cazip görünmektedir. Ancak analog devre yapısı her ne kadar avantajlı görünse de HSA ile görüntü işleme özelinde incelendiğinde giriş görüntüsünün boyutu hatrı sayılır bir sorunu da beraberinde getirmektedir. Bugüne kadar yapılabilmiş en yüksek hücre sayısına sahip analog tasarımın 176 x 144 boyutunda olması sorunun boyutu hakkında da fikir vermektedir. 176 x 144 boyutundan daha büyük görüntüler parçalanarak ayrı ayrı işlenebilir ancak bu durumda da çözünürlük ayrı bir performans kaybı olarak gün yüzüne çıkacaktır. Literatürde yer alan önemli SZ-HSA uygulamalarına ACE16K [15] ve Eye-RIS [16] örnek verilebilir.

Ayrık zamanlı uygulamalar ise sürekli zamanlı HSA uygulamalarına göre daha geniş boyutta giriş görüntüsü işleyebilirler. Dijital donanım tasarımlarında ise RAM gibi paylaşımlı birimlerin kullanımı, sınır koşullarının yaratılması ve yüksek adette giriş/çıkış sinyallerinin kontrolü HSA'nın donanımsal gerçeklemelerine olumsuz yönde etki etmektedir. Ancak dijital donanım tasarımı, HSA'nın matematiksel denklemini yazılımsal olarak hesaplayabilme şansı vermekte ve analog tasarıma göre daha kolay gerçekleme imkanı vermektedir. FPGA (Field Programmable Gate Array) ise paralel yapıya sahip olmasından dolayı dijital tasarımlarda ön plana çıkmakta ve ASIC (Application-Specific Integrated Circuit) devre üretimine göre daha az maliyetli olduğundan tercih edilmektedir. Önemli AZ-HSA uygulamaları arasında ise Zarandy tarafından geliştirilen CASTLE [17], Nagy ve Szolgay tarafından geliştirilen Falcon [18], Kayaer ve Tavşanoğlu tarafından geliştirilen RTCNNP-v1 [10,19] ve RTCNNP-v1'in ikinci versiyonu olan RTCNNP-v2 [11,12] uygulamaları ön plana çıkmaktadır.

Bu bölümde FPGA ile gerçeklenmiş AZ-HSA uygulamalarından İkinci Nesil Hücresel Sinir Ağı Emülatörü (Real-Time Cellular Neural Network (CNN) Processor-v2 (RTCNNP-v2)) mimarisi anlatılacaktır. Çalışmanın devamında yer verilecek olan şablon kütüphanesi incelemeleri RTCNNP-v2 gerçeklemesine yönelik olarak yapılacaktır.

RTCNNP-v2 mimarisi literatürdeki tek full-HD çözünürlükte gerçek zamanlı görüntü işleme yeteneğine sahip bir mimaridir. 60 Hz çerçeve hızını destekleyen ve 124.4 MHz görünür piksel oranına sahip İkinci Nesil Hücresel Sinir Ağı Emülatörü yüksek performanslı Altera Stratix IV GX 230'da ve düşük maliyetli Cyclon III C 25'de gerçeklenmiştir.

3.1 TSA Model Çıkış Denkleminin İki Aşamalı Olarak Değerlendirilmesi

RTCNNP-v2 mimarisinde görüntü işleme bloğunun döngüsel bir çalışma prensibi yerine iş hattı mantığıyla tasarlanmış olması ayrık zamanlı HSA'nın paralelleştirilememe olumsuzluğunu telafi etmiştir. İş hattı modelini uygulamak ve böylelikle sistem hızını arttırmak için HSA'nın TSA modelindeki ifadesi olan (2.28) denklemi

$$y_{ij}(n+1) = \underbrace{f(\overline{\boldsymbol{A}} \circledast \boldsymbol{Y}_{ij}(n) + \overbrace{\overline{\boldsymbol{B}} \circledast \boldsymbol{U}_{ij} + \overline{z}}^{B\ işlemi})}_{A\ işlemi} \tag{3.1}$$

şeklinde A ve B işlemlerine ayrılmıştır.

B işlemi, HSA çıkış denkleminin ayrık zaman değişkeni n'e bağlı olmayan kısmıdır ve

$$g_{ij} = \bar{\boldsymbol{B}} \circledast \boldsymbol{U}_{ij} + \bar{z} \tag{3.2}$$

şeklinde tanımlanır. g_{ij}'nin ayrık zamana bağlı olmaması yalnızca bir sefer hesaplanmasının yeterli olduğunu gösterir ve A işlemini

$$y_{ij}(n+1) = f(\bar{\boldsymbol{A}} \circledast \boldsymbol{Y}_{ij}(n) + g_{ij}) \tag{3.3}$$

haline getirir.

Bu durumda N iterasyon sayısını ifade edecek şekilde, TSA modelin matematiksel hesabı şu adımlarla yapılır:

B işlemi: $g_{ij} = \bar{\boldsymbol{B}} \circledast \boldsymbol{U}_{ij} + \bar{z}$ (3.4a)

$A(1)$ işlemi: $y_{ij}(1) = f(\bar{\boldsymbol{A}} \circledast \boldsymbol{Y}_{ij}(0) + g_{ij})$ (3.4b)

$A(2)$ işlemi: $y_{ij}(2) = f(\bar{\boldsymbol{A}} \circledast \boldsymbol{Y}_{ij}(1) + g_{ij})$ (3.4c)

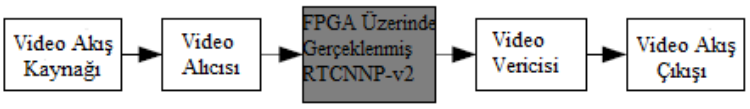
$\vdots \quad : \vdots \quad \vdots \quad \vdots$

$A(N)$ işlemi: $y_{ij}(N) = f(\bar{\boldsymbol{A}} \circledast \boldsymbol{Y}_{ij}(\mathrm{N}-1) + g_{ij})$ (3.4d)

TSA model matematiksel ifadesi üzerinde yapılan bu düzenlemelerle model ileri beslemeli bir işlem dizisine dönüştürülmüş ve iş hattı prensibine uygun hale getirilmiştir. Buna paralel olarak işlem dizisindeki her bir adım farklı bir işlemciye yaptırılmış ve emülatörün çalışma hızı arttırılmıştır.

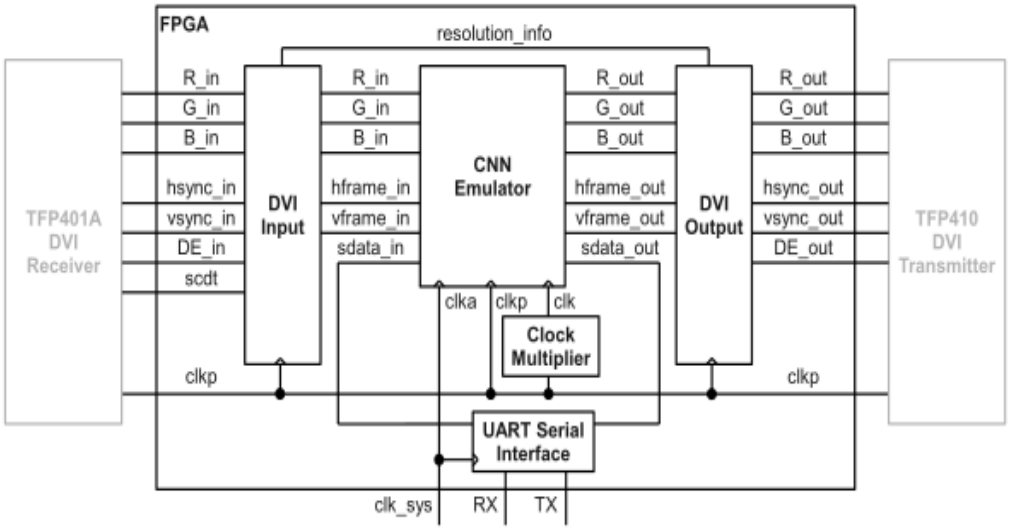
3.2 RTCNNP-v2 Mimarisi

İkinci Nesil Hücresel Sinir Ağı Emülatörü akan bir video yayını yakalayıp hücresel sinir ağlarıyla işledikten sonra tekrar akan bir video yayını olarak çıkışa vermek üzere tasarlanmıştır. Şekil 3.1'de sistemin blok diyagramı verilmiştir.



Şekil 3. 1 Sistemin blok diyagramı

RTCNNP-v2 temel olarak sayısal görüntü giriş bloğu, giriş görüntülerinin işlendiği HSA emülatör bloğu ve işlenen görüntülerin çıkışa aktarıldığı sayısal görüntü çıkış bloğundan oluşur. Sayısal görüntü giriş bloğu DVI sinyallerinin çözünürlüğünü ve çerçeve hızını tayin etmek için tasarlanmıştır. Bu bloktan geçen sinyaller HSA emülatör bloğuna RGB (red,green,blue) pikseller olarak verilir. Sayısal görüntü çıkış bloğu, giriş bloğundan orjinal görüntünün çözünürlüğü ve çerçeve hızı hakkında bilgi alır ve bu bilgilere göre çıkış görüntüsünü yaratır.



Şekil 3. 2 RTCNNP-v2'nin FPGA gerçeklemesi

3.2.1 HSA Emülatör Bloğu

HSA emülatör bloğunda görüntü işleme bir dizi işlemci tarafından gerçekleştirilir. Her bir saat darbesinde yeni bir giriş görüntüsü emülatöre ulaştırılırken aynı anda işlenmiş bir görüntü de sayısal görüntü çıkış bloğuna aktarılır. Bu sayede zaman gecikmelerinin önüne geçilir.

HSA emülatör bloğunda görüntü işlemeyi gerçekleştiren bir dizi işlemcinin yanı sıra RGB piksellerin gri ton piksellere dönüştürüldüğü bir blok ve akabinde gri ton piksellerin siyah-beyaz piksellere dönüştürüldüğü bir başka blok daha bulunur. Hücresel sinir ağların uygulamalarının neredeyse tamamı gri tonlu veya siyah-beyaz görüntü üzerinde çalıştığından sinyal görüntü işleme bloğuna ulaşmadan önce bu bloklardan geçirilir. Eğer görüntü işleme gri ton pikseller üzerinde yapılacaksa siyah-

beyaz çevirici bloğu pas geçilir. Siyah-beyaz çevirici bloğu programlanabilme özelliği ile buna izin verecek şekilde tasarlanmıştır.

Görüntü işlemenin gerçekleştiği işlemci dizisi ise ayrık zaman değişkenine bağlı olmayan B işleminin (3.4a) gerçekleşeceği BPU (B-Processing Unit) ile başlar. B işleminin sonucu tüm iterasyon adımlarında sabit kalacağı için tek bir BPU kullanmak yeterlidir. Bu blok aynı zamanda orjinal giriş görüntüsü olan U'yu gerektiği durumlarda ilk değer olarak kullanabilmesi için kendisinden sonra gelen ilk APU'ya verir.

Ayrık zaman değişkenine bağlı olduğundan her iterasyon adımında sonucu değişecek olan A işlemi (3.4b) APU (A-Processing Unit) tarafından hesaplanır. Her bir APU bloğu kendisinden sonra gelene sabit $\mathbf{G}$ girişini de iletmektedir. Birbiri ardına sıralanmış APUlar birer iterasyon hesaplamakla görevli olduklarından APU sayısının toplamı iterasyon sayısına eşit olacaktır. Her bir iterasyonun farklı APU'ya yaptırılması ile iş hattı modelini oluşturulmuş ve bu model paralel işleme metoduyla performans açısından karşılaştırılabilecek bir hale getirilmiştir.

Son APU bloğundan sonra ise yalnızca kontrast arttılması ihtiyacında kullanılan kontrast ayarlama bloğu bulunur. Minimum ve maksimum piksel değerleri kullanıcı tarafından ayarlanabildiği gibi otomatik mod sayesinde sistem tarafından da hesaplanabilir. Bu bloğun çıkışı aynı zaman HSA emülatör bloğunun da çıkışını oluşturur ve çıkış sinyallerinin yaratılması için sayısal çıkış görüntü bloğuna verilir.

APU bloğunda gerçekleştirilen A işlemi ile BPU bloğunda gerçekleştirilen B işleminin tek farkı (3.4) denklemlerinden de görüldüğü üzere $f(.)$ fonksiyonudur. Toplama ve şablon nokta çarpım işlemlerinin A ve B'nin her ikisinde yer alması APU ve BPU bloklarının tek bir programlanabilir işlemciyle tasarlanabilmesine zemin oluşturmuştur. x-Processing Unit (xPU) olarak adlandırılan blok donanımsal veya yazılımsal olarak değiştirilerek A veya B işlemlerini gerçekleştirebilecek şekilde tasarlanmıştır. xPU bloğu aynı zamanda programlanabilir sınır koşullarına sahiptir. Zero flux veya sabit sınır koşulları kullanılacak görüntü yöntemine göre seçilebilir. Periyodik sınır koşulları ise hücresel sinir ağı emülatörünün ikinci versiyonunda yer almamaktadır.

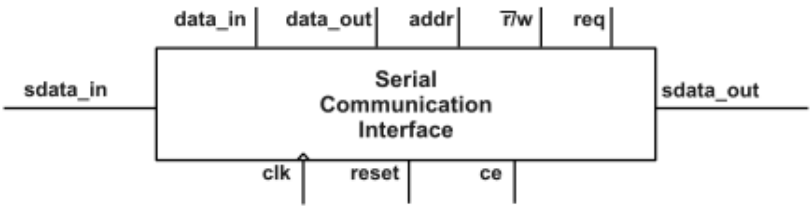
3.2.2 Yerel Kontrol Yapısı

RTCNNP-v2 mimarisinde yer alan her bir blok kendi içerisinde yerel bir kontrol yapısına sahiptir. Bloklar içerisine yerleştirilen yerel kontrol ünitesi kendisinden önceki bloktan gelen sinyalleri çözümler, bu sinyallerle kendi yerel kontrol sinyallerini üretir ve yine aynı sinyalleri uygun bir süre geciktirerek bir sonraki bloğa iletir. Yani her bir blok bir önceki blok tarafından kontrol edilirken bir sonraki bloğu da kontrol etmekle görevlidir. Kontrol mekanizması ilk bloktan son bloğa kadar devam ettirilerek bloklardaki giriş-çıkış gecikmeleri kontrol edilmiş ve böylelikle iş hattının senkronizasyonu korunmuş olur.

3.2.3 Seri Programlama Arayüzü

Seri programlama arayüzü işlem bloklarının katsayılarını programlamak ve sistem çalışırken xPU bloğunun APU ya da BPU olarak seçilmesine olanak vermek için geliştirilmiştir. Özgün olarak tasarlanan haberleşme arayüzü, sistemdeki her bir bloğu bağımsız olarak adreslenebilir ve programlanabilir kılmaktadır. Çalışan sistemde harici arayüz her ne kadar RS232 olsa da UART (Universal Asynchronous Receiver/Transmitter (Evrensel Asenkron Alıcı Verici)) bloğu yerine farklı bir arayüzün tasarlanmasıyla harici giriş-çıkış farklı bir tolla da yapılabilir.

Seri haberleşme arayüzünün blok diyagramı Şekil 3.3'de verilmiştir. Her bir blok içerisinde yer alan yerel kontrol ünitesi seri programlama arayüzüne sahiptir. Aynı zamanda bir bloktaki seri veri çıkışı (sdata_out) diğer blokta seri veri girişine (sdata_in) bağlıdır.



Şekil 3. 3 Seri haberleşme arayüzü

3.2.4 Yapılandırılabilir Özellikler

Tasarımda tekrar tekrar yapılandırılabilen en önemli blok xPU bloğudur. APU/$\overline{\text{BPU}}$ bloğundaki şablon boyutları, çarpan adetleri, görünür piksellerin maksimum sayısı, sınır koşulları gibi özellikler değiştirilebilmektedir.

Şablon boyutu ve sınır koşullarının değiştirilebilmesi sayesinde şablon kütüphanesinde yer alan görüntü işleme uygulamalarındaki farklı boyutlara ve sınır koşullarına sahip uygulamalar gerçeklenebilmektedir. Görünür piksellerin maksimum sayısının ayarlanabilmesi ise Full-HD dahil herhangi bir çözünürlükte çıkış sinyallerinin üretilebilmesi avantajını getirmektedir.

Yukarıda bahsedilen değiştirilebilir özellikler dışında xPU bloklarının sayısı da FPGA gerçeklemesinin yapılandırılabilir özellikleri arasındadır. Aynı zamanda RGB pikselleri gri tonlu piksellere çevirmek için kullanılan bloğun renk kanalı çarpanları da değiştirilebilmektedir.

3.3 RTCNNP-v2'nin Özellikleri

$\text{APU}/\overline{\text{BPU}}$ bloklarının toplam sayısı yukarıda da bahsedildiği gibi iterasyon sayısına eşittir. Donanıma eklenecek her blok, sistemin üzerinde çalıştığı görüntünün bir iterasyon daha işlenmesi anlamına gelmektedir. Bu nedenle donanımda yer alacak her xPU bloğu sistemin performansı için önem teşkil etmekte bir anlamda donanımın yeteneğini belirlemektedir.

RTCNNP-v2 mimarisi yüksek performanslı Altera Stratix IV GX 230 FPGA ile gerçeklendiğinde en fazla 150 $\text{APU}/\overline{\text{BPU}}$ bloğu donanıma sığdırılabilmiştir. Bu da Altera Stratix IV GX 230 ile 150 iterasyona kadar görüntü işlenebileceğini gösterir. Mimari aynı zamanda düşük maliyetli Altera Cyclone III C 25 FPGA ile de gerçeklenmiş ve bu donanıma 3 adet $\text{APU}/\overline{\text{BPU}}$ bloğu sığdırılabilmiştir.

Her iki prototip de 60 Hz çerçeve hızı altında 1920 x 1080 çözünürlükte Full HD olarak test edilmiştir. Gerçek zamanlı görüntü işleme ile yapılan bu test 148.5 MHz piksel saat frekansı ve 124.4 Mpix/s görünür piksel oranı altında gerçekleştirilmiştir. Saniyede 338.2 giga (338.2 x10^9) çarpma ve toplama işleme yapabilme özelliğiyle Altera Stratix

IV GX 230 ile yapılan gerçekleme bugüne kadar literatürde yer almış en hızlı HSA uygulamasıdır.

Mimarinin ileri beslemeli iş hattı modeli ile tasarlanmış olması istenen adette FPGA'in art arda eklenmesine olanak sağlar ve iterasyon limiti sorununu da ortadan kaldırmış olur. FPGAlerin sisteme eklenmesi herhangi bir optimizasyon geliştirme ve optimizasyon gerektirmez. Ayrıca eklenecek FPGAlerin marka ve modelleri de farklılık gösterebilir. Bu özellikleriyle mimari modüler bir yapı sunarak gerektiği durumlarda iş gücünün bölünmesine ortam hazırlamaktadır.

Bu tez çalışmasında şablon kütüphanesinde yer alan görüntü işleme uygulamaları RTCNNP-v2 ile tasarlanmış iki prototip üzerinde çalışıp çalışmayacağına göre incelenmiştir.

BÖLÜM 4

ŞABLON KÜTÜPHANESİ İNCELEMELERİ

Şablon kütüphanesi HSA ile görüntü işleme uygulamalarının toplandığı referans bir kütüphanedir. Bu tez çalışmasında kullanılan kütüphane üçüncü versiyon olup 2010 yılında Pazmany P. Katolik Üniversitesi tarafından yayınlanmıştır [8].

MATLAB ortamında yapılan yazılımsal incelemeler, görüntü işleme uygulamalarının HSA'nın TSA modeli kullanılarak yapılmış ve öncelikle uygulamaların TSA modelde çalışıp çalışmadığına bakılmıştır. Kütüphane üzerinde yapılan analizler sonucu HSA'nın TSA modelinde çalıştırılamayan görüntü işleme uygulamaları ayrı bir konu olarak ele alınmış ve hangi sebeplerden ötürü çalıştırılamadığı dokümante edilmiştir. Aynı zamanda bu bilgilerden yola çıkılarak sorunun kütüphaneden mi yoksa RTCNNP-v2'nin donanımsal tasarımından mı kaynaklandığına dair sonuçlara ulaşılmıştır.

4.1 RTCNNP-v2 Üzerindeki Kısıtlamalar

Şablon kütüphanesi, standart matematiksel model olarak kabul edilen Chua-Yang modeline sahip uygulamaların yanında doğrusal olmayan, zaman gecikmeli ve çok katmanlı matematiksel modellere sahip uygulamaları da içermektedir. Bu yapılara ek olarak bazı uygulamalardaki şablon katsayıları uzayla değişen biçimde tanımlanmıştır.

İkinci nesil hücresel sinir ağı emülatörü şu ana kadar önerilmiş en hızlı AZ-HSA gerçeklemesi olmasına rağmen donanımsal olarak bazı kısıtlamalara sahiptir.

Mevcut gerçeklemelerin harici bir hafıza birimi olmamasından dolayı (2.18) denklemindeki zaman gecikmeli terimler saklanamaz. HSA'nın TSA modelinin denklemi

uzaysal domende ayrı ayrı değerlendirilmediğinden aynı anda farklı koordinatlardaki görüntüler işlenemez. Diğer bir deyişle doğrusal olmayan HSA modeli bu mimari üzerinde gerçeklenemez.

Şablon katsayılarının uzaysal olarak değişmesi matematiksel modeli değiştirmez ancak hesaplama yöntemine etki eder. RTCNNP-v2 üzerinde bulunan xPU bloklarının hesaplama yöntemi uzaysal olarak değişen katsayıları hesaplayamadığından şablon katsayıları uzaysal olarak değişen görüntü işleme uygulamaları da RTCNNP-v2 üzerinde gerçekleştirilemez. Aynı zamanda xPU bloğu kompleks katsayılarla işlem yapamadığından çok katmanlı HSA yapıları da RTCNNP-v2'nin gerçekleyebileceği yapılar arasında yer almaz.

Yukarıda açıklanan kısıtlamalardan ötürü şablon kütüphanesindeki doğrusal olmayan, zaman gecikmeli ve çok katmanlı matematiksel modellere sahip uygulamalarla birlikte şablon katsayıları uzaysal olarak değişen görüntü işleme uygulamaları bu tez çalışması kapsamının dışında tutulmuştur.

4.2 Şablon Kütüphanesinin Kullanımı

Şablon kütüphanesinde yer alan görüntü işleme uygulamaları özelliklerine göre başlıklar altında sınıflandırılmıştır. Temel görüntü işleme uygulamaları, uzaysal lojik ve hareket bu sınıflardan bazılarıdır.

Kütüphanede yer alan her bir görüntü işleme uygulaması için geri besleme ve giriş kontrol şablonları, giriş ve ilk durum atama koşulları sınır koşullarıyla birlikte verilmiştir. Her bir uygulama için öncelikle değişkenler ve sınır koşulları verilir devamında ise bu görüntü işleme uygulamasına özel bir örnek eklenir. Şekil 4.1'de şablon kütüphanesinden bir alıntı verilmiştir. Kütüphanede verilen bilgilerin tanımları ve kullanım şekli sırasıyla aşağıda verilmiştir.

Uygulamanın İsmi: Bu kısımda görüntü işleme uygulamasının ismi belirtilir. Eğer var ise uygulamanın şablon kütüphanesine kazandırılmasında yararlanılan yayın referans verilir.

A, B şablonları ve z katsayısı: Geri besleme şablonu $\boldsymbol{A}$ ile giriş kontrol şablonu $\boldsymbol{B}$ matrisel formda, eşik değeri z ise sabit bir değer olarak verilir.

Verilen (Given): Görüntü işlemenin gri ton veya siyah-beyaz görüntü üzerinde yapılacağı belirtilir.

Giriş (Input): Ağın girişi olan $U(t)$'ye görüntünün mü yoksa bir sabitin mi atanacağı bilgisini verir.

İnitial State (İlk Durum): Durum değişkeni $\mathbf{X}$'in ilk durumu olan $\mathbf{X}(0)$ belirlenir.

Sınır Koşulları (Boundary Conditions): Uygulamada hangi sınır koşulunu kullanacağını belirtir.

Çıkış (Output): Çıkışın hangi yakınsama durumunda ve hangi özel koşullar altında elde edileceğini belirtir. Buna ek olarak dikkate alınması gereken durumlar gerektiğinde ayrı bir not satırı ile verilir.

Örnek: Uygulama özelinde giriş ve çıkış görüntüleri boyutlarıyla birlikte verilir. Eğer var ise CANDY ve ALADDIN hücresel sinir ağı simülatörlerinde görüntülerin kolaylıkla bulunmasını sağlayan kod isimleri uygulamanın kod ismiyle birlikte verilir.

GlobalConncetivityDetection1: ***Detects the one-pixel thick closed curves and deletes the open curves from a binary image*** **[61]**

$\mathbf{A} =$ | 6 | 6 | 6 |
|---|---|---|
| 6 | 9 | 6 |
| 6 | 6 | 6 |

$\mathbf{B} =$ | -3 | -3 | -3 |
|---|---|---|
| -3 | 9 | -3 |
| -3 | -3 | -3 |

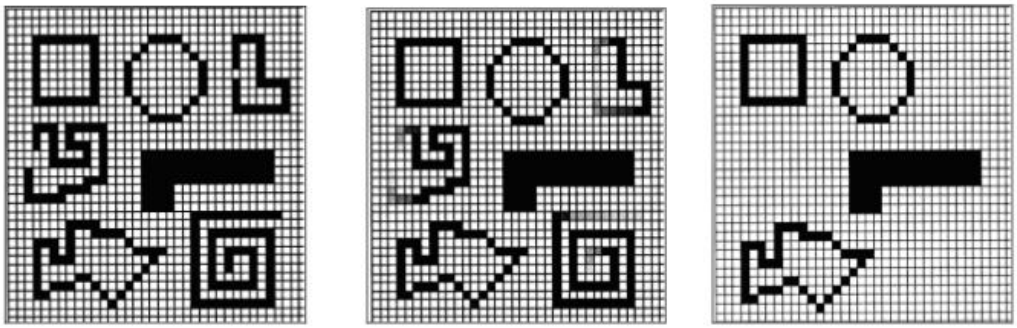
$z =$ -4.5

I. Global Task

Given:	static binary image **P**
Input:	**U(t) = P**
Initial State:	**X**(0) = **P**
Boundary Conditions:	Fixed type, $y_{ij} = 0$ for all virtual cells, denoted by [Y]=0
Output:	**Y(t)⇒Y(∞)** = Binary image which contains all closed curves present in the initial image **P**

II. Example: image size: 36x36.

Remarks: *The binary image* **P** *containing closed and open curves (one-pixel thick) is applied both at the input and loaded as initial state. If one pixel is removed from a closed curve, it becomes an open curve and is deleted, as shown in the second image. The compact (solid) objects from the image are not modified.*



Şekil 4. 1 Şablon kütüphanesinden bir alıntı [8]

4.3 Şablon Kütüphanesinde Yer Alan Görüntü İşleme Uygulamalarının TSA Model ile Uyumluluğunun İncelenmesi

Kütüphanede yer alan görüntü işleme uygulamalarının TSA model ile uyumluluğunun incelenmesi için (2.30) denklemi MATLAB ortamında yazılmıştır. Her bir görüntü işleme uygulaması için geri besleme ve giriş kontrol şablonları kütüphaneden alınmış ve ilk durum, giriş ve sınır koşulları atamalarına tanımlandığı şekliyle sadık kalınmıştır. Kütüphanede verilen giriş görüntüsü mevcut olduğu takdirde [20] sayfasından alınmış ve yazılan MATLAB koduna giriş olarak verilmiştir. Programın çıkış görüntüsü, kütüphanedeki çıkış görüntüsü ile karşılaştırılmıştır. Kütüphanede verilen giriş görüntülerinin [20] sayfasında olmadığı durumlarda, bilgisayar ortamında görüntüler oluşturulmuş ve giriş görüntüsü olarak kullanılmıştır. Ayrıca kütüphanedeki orjinal görüntünün yanında farklı bir giriş görüntüsü oluşturularak işlenmiş ve çıkış görüntüsünün kullanılan uygulamanın özelliklerini taşıyıp taşımadığına bakılmıştır. Bu sayede kütüphaneden kaynaklı tanımlama eksiklikleri de ortaya çıkarılmıştır. TSA model ile uyumluluğun incelendiği örnek bir uygulama aşağıda verilmiştir.

Smoothing ismiyle anılan görüntü işleme uygulaması gri tonlu olarak verilen bir giriş görüntüsünü siyah-beyaz çıkış görüntüsüne çevirir. Bu uygulama için kütüphanede verilen geri besleme şablonu $\boldsymbol{A}$ ve giriş kontrol şablonu $\boldsymbol{B}$ ve eşik değeri z aşağıdaki gibidir.

$$\boldsymbol{A}=\begin{bmatrix}0 & 1 & 0\\ 1 & 2 & 1\\ 0 & 1 & 0\end{bmatrix}, \quad \boldsymbol{B}=\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}, \quad z=0 \tag{4.1}$$

Verilen bu değerler koda eklenmiş ve bu uygulama için belirlenen sabit sınır koşullarına göre görüntü ayarlanmıştır. Kütüphanede 'madonna.bmp' olarak verilen görüntü [20] sayfasından alınmış ve tanımlandığı şekliyle $\mathbf{X}(0)$ ilk durumuna atanmıştır. Giriş değişkeni ise verilen şekliyle keyfi bir sabit olarak alınmıştır. Bu verilerle çalıştırılan programın çıkış görüntüsünün kütüphanede verilen görüntü ile aynı olduğu saptanmıştır. Bir sonraki aşamada kütüphanede verilenden farklı bir görüntü oluşturularak yine aynı uygulamada çalışıp çalışmadığı gözlenmiştir. Kütüphanede yer alan uygulama sayfası Şekil 4.2'de, kütüphaneden farklı olarak verilen giriş görüntüsü program çıktısıyla beraber Şekil 4.3'te verilmiştir.

Smoothing: ***Smoothing with binary output*** **[1]**

Old names: AVERTRSH, AVERAGE

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z = \boxed{0}$$

I. Global Task

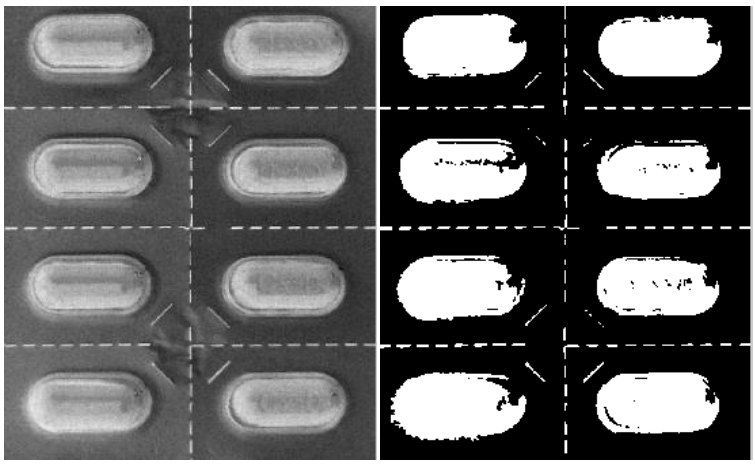
Given:	static grayscale image **P**
Input:	**U(t)** = Arbitrary or as a default U(t)=0
Initial State:	**X**(0) = **P**
Boundary Conditions:	Fixed type, $y_{ij} = 0$ for all virtual cells, denoted by [Y]=0
Output:	**Y**(t)⇒**Y**(∞) = Binary image where black (white) pixels correspond to the locations in **P** where the average of pixel intensities over the r=1 feedback convolution window is positive (negative).

II. Example: image name: madonna.bmp, image size: 59x59; template name: avertrsh.tem .

input

output

Şekil 4. 2 Şablon kütüphanesindeki Smoothing uygulaması [8]



(a) Programın giriş görüntüsü (b) Programın çıkış görüntüsü

Şekil 4. 3 Şablon kütüphanesindekinden farklı bir görüntüyle Smoothing uygulaması

Yukarıda açıklanan şekliyle yapılan incelemeler sonucu kütüphanede tanımlandığı şekliyle TSA modelde çalıştırılan uygulamalar Çizelge 4.1'de verilmiştir. Bu çizelge

hazırlanırken uygulamanın kütüphanede verilen hiçbir değişken üzerinde oynamadan çalışmış olmasına dikkat edilmiştir. Bunun dışında kalan uygulamalar ise ayrı olarak değerlendirilmiştir.

Çizelge 4. 1 Şablon kütüphanesinde tanımlandığı şekliyle çalışan görüntü işleme uygulamaları

Smoothing	Hole-Filling
Masked CCD	3x3 Inverse Halftoning
Concentric Contour Detector	5x5 Inverse Halftoning
Global Connectivity Detection	Local Southern Element Detector
Global Connectivity Detection1	Left Peeler
Corner Detection	Right Edge Detection
Diagonal Line Remover	Masked Shadow
Vertical Line Remover	Vertical Shadow
Thin Line Remover	Directed Growing Shadow
Approx. Diagonal Line Detector	Threshold
Diagonal Line Remover	White Filler
Rotation Detector	Black Filler
Heat Diffusion	Black Propagation
Edge Detection	White Propagation
Optimal Edge Detector	Concave Location Filler
Point Extraction	Concave Arc Filler
Point Removal	Junction Extractor
Selected Object Extraction	Junction Extractor1
Filled Counter Extraction	LE7pixel Vertical Line Remover
3x3 Halftoning	LE3pixel Line Detector
5x5 Halftoning2	

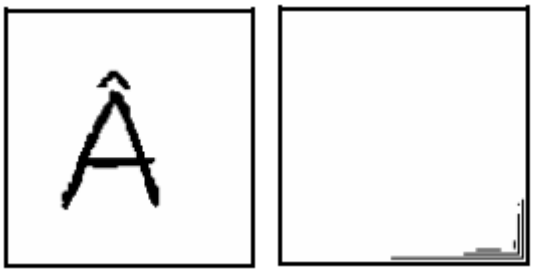
4.4 TSA Modelde Çalıştırılamayan Uygulamalar

Kütüphanede tanımlandığı şekliyle TSA modelde çalıştırılamayan uygulamalarda temel olarak üç farklı sorun saptanmıştır. Çıkış görüntüsü uyuşmazlığı, eksik tanımlanmış uygulama ve yanlış tanımlanmış uygulama olarak isimlendirilen bu sorunlar aşağıda açıklanmıştır.

4.4.1 Çıkış Görüntüsü Uyuşmazlığı

Kütüphanede verilen orijinal giriş görüntüsü MATLAB programına giriş olarak verildiğinde kütüphanedeki çıkış görüntüsüyle programın çıkış görüntüsünün aynı olmadığı durumlar 'çıkış görüntüsü uyuşmazlığı' olarak sınıflandırılmıştır.

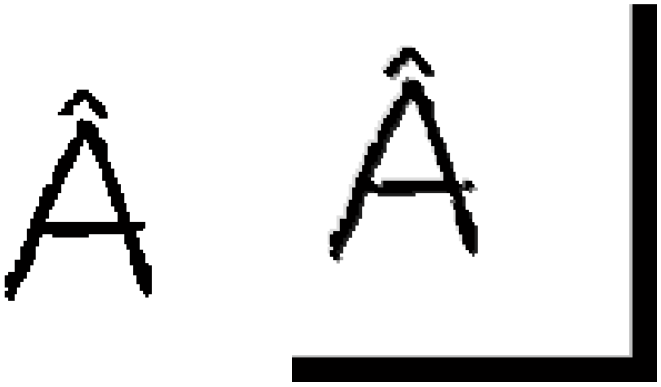
Diagonal Hole Detection, uygulandığında çıkış görüntüsü uyuşmazlığı yaşanan görüntü işleme uygulamalarından biridir. Bu uygulama görüntü üzerindeki her çapraz çizgiden elde edilen çapraz deliklerin sayısını algılar. Kütüphanede verilen giriş ve çıkış görüntüleri Şekil 4.4'de yer almaktadır.



(a) Giriş görüntüsü (b) Çıkış görüntüsü

Şekil 4. 4 Kütüphanede verilen Diagonal Hole Detection görüntüleri

Uygulamanın orjinal görüntüsü MATLAB programına giriş olarak verildiğinde elde edilen çıkış ise Şekil 4.5'te verilmiştir.



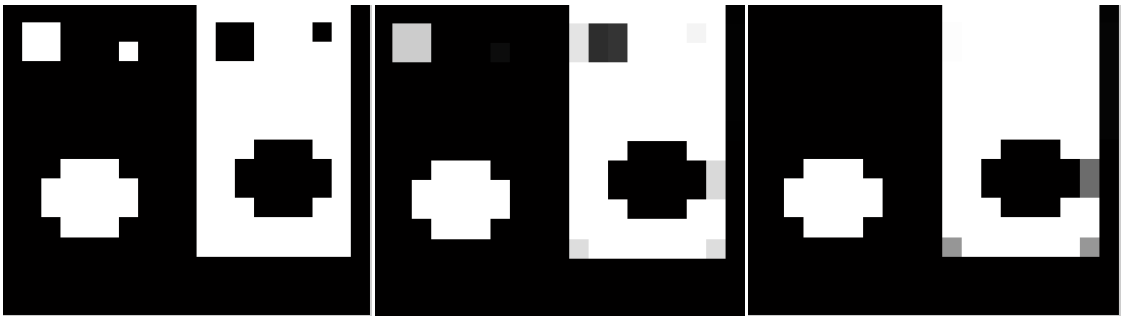
(a) Giriş görüntüsü (b) Çıkış görüntüsü

Şekil 4. 5 Diagonal Hole Detection MATLAB giriş ve çıkış görüntüleri

4.4.2 Eksik Tanımlanmış Uygulama

HSA'nın TSA modeli daha önce de bahsedildiği üzere ayrık zamanlı bir modeldir. Şablon kütüphanesinde bazı uygulamaların yalnızca ayrık zamanlı modelde çalışabileceği belirtilmiş olsa da çalışması için ek bilgilerin gerektiği görüntü işleme uygulamaları 'eksik tanımlanmış uygulama' olarak sınıflandırılmıştır.

Çalışması için ek bilgi ihtiyacı duyulan uygulamalara *Small Object Remover* verilebilir. *Small Object Remover*, görüntü üzerinden küçük nesnelerin silindiği bir uygulamadır. Ancak kütüphanede silinecek küçük nesnelerin boyutuna dair bilgi verilmemiştir. Zira iterasyon sayısı yeteri kadar arttırıldığında görüntü tamamen kaybolmaktadır. Küçük objelerin silinmesi için kullanılan bu uygulamada iterasyon sayısı sınırlanmazsa uygulamanın çalışmadığı yanılgısına düşülebilir.



(a) Giriş görüntüsü (b) 120 iterasyon sonrası (c) 300 iterasyon sonrası

Şekil 4. 6 Small Object Remover giriş ve farklı iterasyonlardaki görüntüleri

4.4.3 Yanlış Tanımlanmış Uygulama

Kütüphanede verilen görüntü işleme uygulaması tanımlandığı halinden farklı bir biçimde çalışıyorsa 'yanlış tanımlanmış uygulama' olarak sınıflandırılmıştır.

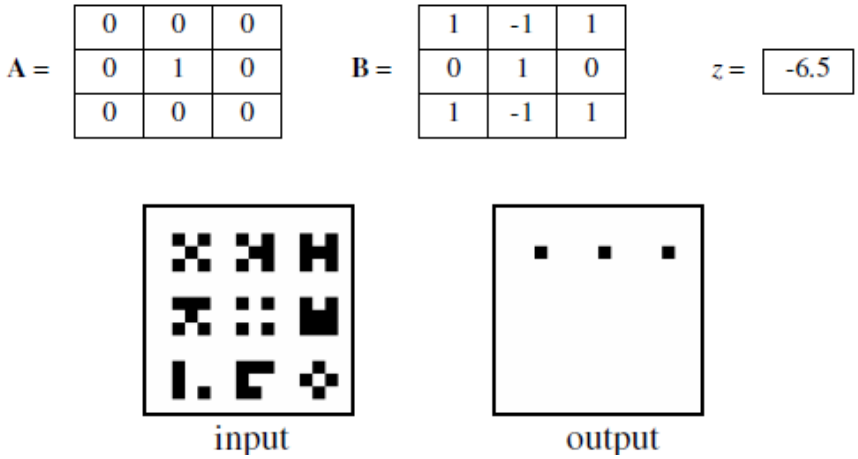
Pattern Matching Finder, bu sınıfta verilebilecek örneklerden biridir. Bu uygulamanın giriş kontrol şablonu üzerinden verilen örüntüyü algılayıp buna eş örüntüleri tespit etmesi beklenmektedir. Uygulamanın geri besleme şablonu, giriş kontrol şablonu ve eşik değeri kütüphanede şu şekilde tanımlanmıştır:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} \mathrm{b} & \mathrm{b} & \mathrm{b} \\ \mathrm{b} & \mathrm{b} & \mathrm{b} \\ \mathrm{b} & \mathrm{b} & \mathrm{b} \end{bmatrix}, \qquad z = -N + 0.5 \tag{4.2}$$

$$\mathrm{b} = \begin{cases} 1, \text{örüntüde eş olması istenen piksel siyah ise} \\ 0, \text{örüntüdeki piksel dikkate alınmayacak ise} \\ -1, \text{örüntüde eş olması istenen piksel beyaz ise} \end{cases}$$

$N =$ siyah ve beyaz piksellerin toplamı, $\boldsymbol{B} =$ şablonun sıfır olmayan değerleri.

Yukarıdaki gibi kütüphanede tanımlanan bu görüntü işleme uygulaması için verilen orjinal örnek ise Şekil 4.7'de verilmiştir.



Şekil 4. 7 Pattern Matching Finder şablon kütüphanesinde tanımlanmış hali

Verilen örneğe göre siyah olması istenen pikseller: $(1,1);\ (1,3);\ (2,2);\ (3,1);\ (3,3)$

Verilen örneğe göre beyaz olması istenen pikseller: $(1,2);\ (3,2)$

Toplam siyah ve beyaz piksel sayısı: $7;\ z = -7 + 0.5 = -6.5$

Giriş kontrol şablonu üzerinden verilen örüntünün kuralını sağlayan ilk üç örüntü çıkışta saptanmıştır. Ancak **B** şablonu için verilen kurala riayet edilerek hazırlanan farklı giriş görüntüleri için her zaman doğru sonuç alınamamıştır. Örneğin referans alınacak örüntü aşağıdaki gibi tanımlanacak olsun.

Siyah olması istenen pikseller: $(1,1);\ (2,2);\ (3,3)$

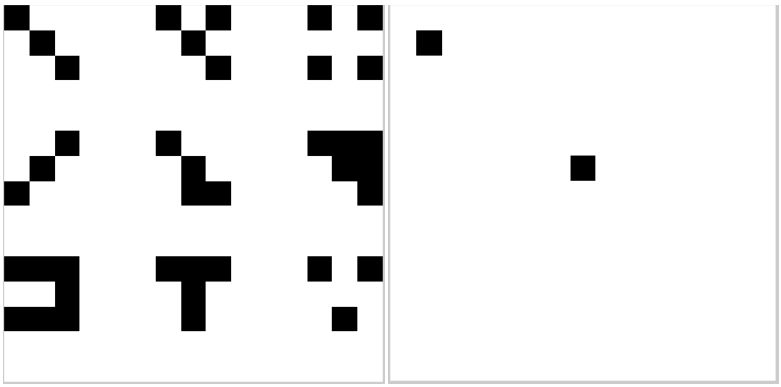
Beyaz olması istenen pikseller: $(3,1);\ (3,2)$

Toplam siyah ve beyaz piksel sayısı: 5

Bu durumda giriş kontrol şablonu ve eşik değeri

$$\boldsymbol{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & 1 \end{bmatrix} \quad , \quad z = -5 + 0.5 = -4.5 \tag{4.3}$$

olarak hesaplanır. Bu koşullar altında verilen giriş görüntüsü için kütüphanede verilen tanım doğrulanamamıştır. Programın giriş ve çıkış görüntüleri Şekil 4.8'de verilmiştir.



(a) Giriş görüntüsü (b) Çıkış görüntüsü

Şekil 4. 8 Pattern Matching Finder giriş görüntüsü ve program çıktısı

Programın giriş kontrol şablonu üzerinden verilen referans örüntüye göre 1,2 ve 6 numaralı örüntülerin çıkış vermesi beklenirken, 1 ve 5 numaralı örüntüler çıkış görüntüsü üzerinde yer almıştır. Bunun dışında denenen farklı örneklerde de benzer durumlar gözlenmiştir. Uygulamanın hangi durumlarda kütüphanede tanımlanan haliyle doğru sonuç verdiği ilişkilendirilememiş ancak her zaman doğru sonuç vermediği kanıtlanmıştır.

Yukarıda açıkça tanımlanan örnekler haricinde kütüphanede tanımlandığı haliyle çalışmayan görüntü işleme uygulamaları da tez süresi boyunca dokümante edilmiştir. Bu uygulamaların tam listesi açıklamalarıyla beraber Çizelge 4.2'de verilmiştir.

Çizelge 4. 2 Şablon kütüphanesinde tanımlandığı şekliyle çalışmayan görüntü işleme uygulamaları

Uygulama İsmi	**Problem Tanımı**	**Açıklama**
Diagonal Hole Detection	Çıkış görüntüsü uyuşmazlığı	Kütüphanede belirtildiği gibi her çapraz sütundaki delikler algılanır ancak çıkış görüntüsü kütüphanedeki gibi çizgiler halinde değil, siyah gölge şeklinde olur.
Horizontal Hole Detection	Çıkış görüntüsü uyuşmazlığı	Kütüphanede belirtildiği gibi her yatay sütundaki delikler algılanır ancak çıkış görüntüsü kütüphanedeki gibi çizgiler halinde değil, siyah gölge şeklinde olur.
Vertical Hole Detection	Çıkış görüntüsü uyuşmazlığı	Kütüphanede belirtildiği gibi her dikey sütundaki delikler algılanır ancak çıkış görüntüsü kütüphanedeki gibi çizgiler halinde değil, siyah gölge şeklinde olur.
Center Point Detector	Yanlış tanım & Eksik tanım	Giriş kontrol şablonları ile geribesleme şablonlarının katsayıları yer değiştirildiğinde oluşan yeni giriş kontrol şablonu B(0,0) hücresine 1 yerine 0 yapıldıktan sonra çalıştırılabilmiştir. Ayrıca merkezleri bulmak için verilen adımlarda tek sayılardaki adımların kendi içerisinde, çift sayılardaki adımların da kendi içerisinde saat yönüyle değiştiği bilgisi verilmemiştir.
Masked Object Extractor	Yanlış tanım	Maske olarak kullanılan giriş görüntüsü ile ilk durum görüntüsü yer değiştirmelidir.
5x5 Halftoning1	Eksik tanım	TSA modelinde görüntü işlenirken örnekleme periyodu Ts'in 0.4'ten büyük alınmaması önerilir. Şablon kütüphanesinde bu bilgi verilmemiştir.
Pattern Matching Finder	Yanlış tanım	Şablon kütüphanesinde bu yöntem için giriş kontrol şablonu katsayıları siyah ve beyaz piksellere göre parametrik olarak verilmiştir. Ancak kütüphanede verilen görüntü örneği dışında giriş görüntülerine göre katsayılar belirlendiğinde her zaman doğru sonuç vermemektedir.
Shadow Projection	Çalıştırılamayan uygulama	Bu görüntü işleme yöntemi kütüphanede verilen bilgilerle çalıştırılamamıştır.
Surface Interpolation	Çalıştırılamayan uygulama	Bu görüntü işleme yöntemi kütüphanede verilen bilgilerle çalıştırılamamıştır.
Local Concave Place Detector	Çıkış görüntüsü uyuşmazlığı	Kütüphanede verilen çıkış görüntüsünde yalnızca iç bükeyliğin başladığı noktada bir piksel çıkış oluşurken, bu çalışmada elde edilen çıkış görüntüsünde iç bükeyliğin hem başlayıp hem de bittiği noktada birer piksel çıkış oluşmaktadır.
Small Object Remover	Eksik tanım	Giriş görüntüsü üzerindeki küçük objelerin silindiği bu yöntemde tekrarlama sayısı yeteri kadar arttırılırsa görüntünün tamamı silinir.
Pixel Search	Çalıştırılamayan uygulama	Bu görüntü işleme yöntemi kütüphanede verilen bilgilerle çalıştırılamamıştır.
Bipolar Wave	Çıkış görüntüsü uyuşmazlığı	Bu görüntü işleme yönteminde giriş görüntüsü üzerinde yer alan siyah ve beyaz pikseller gri arka plan üzerinde yayılmalı ve çıkış görüntüsü yalnızca siyah ve beyaz piksellerden oluşmalıdır. Bu bilgi şablon kütüphanesinde verilmiştir. Ancak kütüphanede örnek olarak verilen çıkış görüntülerinden biri bu bilgiyle çelişmekte ve çıkış görüntüsü üzerinde gri pikseller de bulunmaktadır. Bu çalışmada yapılan incelemede kütüphanede yer alan çıkış görüntüsünün yanlış verildiği saptanmıştır.

BÖLÜM 5

SONUÇ VE ÖNERİLER

Bu tez çalışmasında, şablon kütüphanesinde yer alan görüntü işleme uygulamalarının HSA'nın tüm sinyal aralığı modeli ile uyumlulukları incelenmiş ve bu incelemeler sonucunda hangi uygulamaların İkinci Nesil Hücresel Sinir Ağı Emülatörü üzerinde çalışıp çalışmayacağı anlaşılmıştır. Bu kapsamda verilen Çizelge 4.1 ve Çizelge 4.2 ile uygulamaların TSA model ile uyumluluğunun sonuçları literatürde ilk kez toplu olarak verilmiştir.

Elde edilen sonuçlar değerlendirildiğinde şablon kütüphanesinin halen düzeltilmeye ve geliştirilmeye olan ihtiyacı açıkça görülmüştür. Kütüphanede yer alan uygulamalar daha açık biçimde aktarılmalı ve görsel örnekler arttırılmalıdır. Nitekim bu tez çalışması devam ederken Aralık 2014 tarihinde kütüphanenin dördüncü versiyonunun alfa sürümü yayımlanmıştır. Yeni versiyonda uygulamalar detaylandırılmış ancak hali hazırda her uygulama için görsel örnekler sunulmamıştır. Bu çalışmadan sonra yapılacak şablon kütüphanesi incelemelerinde, hazır olması durumunda, şablon kütüphanesinin dördüncü versiyonunun kullanılması daha doğru sonuçlar alınmasına zemin hazırlayacaktır.

Çalışma sonucundaki bulgular RTCNNP-v2 açısından değerlendirildiğinde, emülatörün bir sonraki versiyonu için bazı geliştirmelere ihtiyaç duyduğu saptanmıştır. Bu çalışmada inceleme dışı tutulan uzayla değişen katsayılı ve çok katmanlı HSA yapılarının gerçeklenebilmesi için xPU bloğunun tekrar gözden geçirilmelidir. xPU bloğunun hesaplama metodu uzayla değişen katsayılı uygulamaları da destekleyecek şekilde oluşturulmalıdır. Diğer taraftan kompleks sayılarla işlem yapabilme özelliği eklenerek

çok katmanlı HSA yapılarını da gerçekleyebilecek hale getirilmelidir. Yine bu çalışmanın kapsamı dışında zaman gecikmeli HSA yapılarının gerçeklenebilmesi için sisteme harici bir hafıza birimi eklenmelidir. RTCNNP-v2'de HSA'nın TSA modelinin denklemi zaman domeninde A ve B işlemleriyle hesaplanmıştır. Gerçeklemenin üçüncü versiyonunda bu denklem uzaysal domende de birden fazla işlemle ifade edilirse doğrusal olmayan HSA yapıları da donanım üzerinde gerçeklenebilir duruma gelmiş olacaktır.

Bu çalışmanın devamında şablon kütüphanesinde yer alan uygulamaların tek bir donanım gerçeklemesi üzerinde çalışmasına yönelik incelemeler yapılabilir. RTCNNP-v2'nin yüksek performanslı FPGA ile gerçeklemesine 150, düşük ücretli gerçeklemesine 3 xPU bloğu sığmaktadır. Mimari, gerektiği durumlarda istenen adette FGPA'in uç uca eklenmesine izin verse de kullanılacak her FPGA ayrı bir maliyet getirecektir. Her bir xPU bloğunun bir iterasyon sayısına karşı düştüğü göz önünde bulundurulursa uygulamaların tek bir donanım üzerinde çalışabilmesi için en fazla 150 iterasyon sayısında sonuç vermesi gerekecektir. AZ-HSA'da her bir hücrenin durum denklemi (2.30) denkleminden de görüldüğü üzere giriş, ilk koşul, geri besleme ve giriş kontrol şablonları ve örnekleme periyoduna (T_s) bağlıdır. T_s dışındaki etkenler şablon kütüphanesinde verilen ve değiştirilmesi mümkün olmayan etkenlerdir. Bu durumda iterasyon sayısını düşürmenin yollarından biri örnekleme periyodunu arttırmak olacaktır. Aynı zamanda piksel frekansı ve buna bağlı olarak kullanılan çarpıcı sayısı FPGA üzerine yerleştirilebilecek xPU bloklarının sayısını dolayısıyla iterasyon sayısını belirlemektedir. Bunun yanında kullanılacak blok RAM miktarı da FPGA'deki iterasyon sayısına etki eder. Gerekli blok RAM miktarının $\boldsymbol{Y}$'nin bit genişliği ile doğrudan ilişkili olduğu da unutulmamalıdır. Şablon kütüphanesindeki uygulamaların RTCNNP'nin tek bir donanımı üzerinde çalışmasına yönelik yapılacak olası incelemelerinde bu faktörler ayrı ayrı göz önünde bulundurulmalıdır.

KAYNAKLAR

[1] McCulloch, W.S. ve Pitts, W.H., (1943). "A Logical Calculus of the Ideas Immanent in Nervous Activity", Bulletin of Mathematical Biophysics 7,115-133.

[2] Hopfield, J.J., (1982). "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proceedings of the National Academy of Sciences of the United States of America, 79:2554–2558.

[3] Chua, L.O. ve Yang, L., (1988). "Cellular Neural Networks:Theory", IEEE Trans. Circuit Syst., 35:1257–1272.

[4] Rodriguez-Vazquez, A., Espejo, S., Dominguez-Castro, R., Huertas, J.L. ve Sanchez-Sinencio, E., (1993). "Current-mode techniques for the implementation of continous- and discrete time cellular neural networks", IEEE Trans. Circuits and Syst. II, 40:132–146.

[5] Fernandez-Garcia, N.A., Suarez, M., Brea, V.M. ve Cabello, D., (2008). "Template-Oriented Hardware Design based on Shape Analysis of 2D CNN Operators in CNN Template Libraries and Applicatons", 11th International Workshop on Cellular Neural Networks and Their Applications (CNNA 2008), 63–68.

[6] Cimagalli, V. ve Balsi, M., (1993). "Cellular Neural Networks: A Review", Proceedings of Sixth Italian Workshop on Parallel Architectures and Neural Networks, 12-14 May 1993, Vietri sul Mare.

[7] CNN Software Library for ACE4K chip (Templates and Algorithms), Analogic Computers Ltd., Budapest, 2000.

[8] Software Library for Cellular Wave Computing Engines, version 3.1., MTA-SZTAKI, Budapest, 2010.

[9] Software Library for Cellular Wave Computing Engines, version 4.0 alpha 1, MTA-SZTAKI, Budapest, 2014.

[10] Kayaer, K. ve Tavsanoglu, V., (2008). "A New Approach to Emulate CNN on FPGAs for Real Time Video Processing", 11th International Workshop on Cellular Neural Networks and Their Applications (CNNA 2008), 23–28.

[11] Yildiz, N. (2013), Design of a Cellular Neural Network Emulator and Its Implementation on an FPGA Device, Doktora Tezi, YTÜ Fen Bilimleri Enstitüsü, İstanbul.

[12] Yildiz, N., Cesur, E., Kayaer, K., Tavsanoglu, V. ve Alpay, M., (2015). "Architecture of a Fully Pipelined Real-Time Cellular Neural Network Emulator", IEEE Trans. Circuits Syst., 62:130–138.

[13] Kari, J., (2005). "Theory of cellular automata: A survey", Theoratical Computer Science, 334:3–33.

[14] Yang, L., Chua, L.O. ve Krieg, K.R. (1990). "VLSI Implementation of Cellular Neural Networks", IEEE International Symposium on Circuits and Systems (ISCAS-90), 3:2425–2427.

[15] Rodriguez-Vazquez, A., Linan-Cembrano, G., Carranza, L., Roca-Moreno, E., Carmona-Galan, R., Jimenez Garrido, F., Dominguez-Castro, R. ve Meana, S., (2004). "ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs", IEEE Trans. Circuits Syst., 51:851–863.

[16] Alba, L., Castro, R.D., Jimenez Garrido, F., Espejo, S., Morillas, S., Listan, J., Utrera, C., Garcia, A., Pardo, M.D., Romay, R., Mendoza, C., Jimenez, A. ve Rodriguez-Vazquez, A., (2009). "New Visual Sensors and Processors", Cognitive Systems Monographs, 1:351–369.

[17] Zarandy, A., Keresztes, P., Roska, T. ve Szolgay, P., (1998). "CASTLE: an Emulated Digital CNN Architecture Design Issues, New Results", Electronics, Circuits and Systems IEEE International Conference, September 1998, Lisbon.

[18] Nagy, Z. ve Szolgay, P., (2003). "Configurable Multilayer CNN-UM Emulator on FPGA," IEEE Trans. Circuits. Syst. I, 50:774–778.

[19] Kayaer, K. (2008), Gerçek Zamanlı Video İşleyen Yeni Bir Hücresel Sinir Ağı Emülatörü Tasarımı ve FPGA ile Gerçeklenmesi, Doktora Tezi, YTÜ Fen Bilimleri Enstitüsü, İstanbul.

[20] CNN technology, http://cnn-technology.itk.ppke.hu/, 22 Mart 2015.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı :Olcay KORKMAZ

Doğum Tarihi ve Yeri : 31.05.1990 / İskenderun

Yabancı Dili : İngilizce

E-posta : olcaykorkmaz1@gmail.com

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Lisans	Elektronik ve Hab. Müh.	YTÜ	2012
Lise	Fen Bilimleri	Bandırma Anadolu Öğretmen Lisesi	2008

İŞ TECRÜBESİ

Yıl	**Firma/Kurum**	**Görevi**
2014 – Devam ediyor	Enersis	Elektronik Donanım Tas. Müh.
2012 – 2014	Empa Elektronik	Saha Uygulama Mühendisi

YAYINLARI

Bildiri

1. Korkmaz, O. ve Yildiz, N., (2015). “Hücresel Sinir Ağı Kütüphanesinin Donanım Gerçeklemesine Yönelik Olarak İncelenmesi”, Çankaya Üniversitesi Mühendislik ve Teknoloji Sempozyumu Bildiriler Kitabı, 243-247.