YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HSA'LARIN FPGA ÜZERİNDE GERÇEKLENMESİ ve GÖRÜNTÜ İŞLEME UYGULAMALARI

Elektrik-Elektronik Mühendisi Sinem FIRAT

FBE Elektronik ve Haberleşme Anabilim Dalında Elektronik Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Prof. Dr. Vedat TAVŞANOĞLU (YTÜ)

İSTANBUL, 2011

İÇİNDEKİLER

SİMGE LİSTESİ

A	Geri besleme şablonu.
$A(i, j;k, l)$	i. satır ve j. sütundaki hücreye giren, k. satır ve l. sütundaki hücre çıkısının ağırlık değeri.
B	İleri besleme (veya giriş) şablonu.
$B(i, j;k, l)$	i. satır ve j. sütundaki hücreye giren, k. satır ve l. sütundaki hücre girişinin ağırlık değeri.
C	HSA hücresindeki lineer kapasite.
$C(i, j)$	i. satır ve j. sütunda yer alan hücre.
$C(k,l)$	$C(i, j)$ hücresinin etki alanı içerisinde bulunan hücreler.
$f(.)$	Hücrenin doğrusal olmayan (nonlinear) çıkış (aktivasyon) fonksiyonu.
i, j	İki boyutlu HSA dizinin satır ve sütun indisleri.
k,l	S_r (i, j) etki alanının içinde kalan hücrelerin indisleri.
M, N	İki boyutlu HSA dizisinin satır ve sütun sayısı.
r	Etki alanı yarıçapı (radius of sphere of influence).
R_x	HSA hücresindeki kapasiteye paralel bağlı olan lineer direnç.
S_r (i, j)	$C(i, j)$ hücresinin etki alanı (sphere of influence).
T_s	Örnekleme aralığı (Sampling interval).
$x(i, j)$	$C(i, j)$ hücresinin durum değişkeni.
$y(i, j)$	$C(i, j)$ hücresinin çıkış değişkeni.
I	Hücrelerin eşik (bias) değeri.

KISALTMA LİSTESİ

ASIC	Application Specific Integrated Circuit (Uygulamaya Özel Tümleşik Devre)
AZ-HSA	Ayrık Zamanlı HSA (Discrete Time CNN (DT-CNN))
CNN	Cellular Neural Networks (Hücresel Sinir Ağları)
CNN-UM	CNN Universal Machine (HSA Evrensel Makinesi)
CSFT	Continuous Space Fourier Transform (Sürekli Uzaysal Fourier Dönüşümü)
DSFT	Discrete Space Fourier Transform (Ayrık Uzaysal Fourier Dönüşümü)
DSP	Dijital Signal Processor (Dijital Sinyal İşlemcisi)
DT-CNN	Discrete Time CNN (Ayrık Zamanlı HSA (AZ-HSA))
DVI	Digital Visual Interface (Sayısal Görüntü Arayüzü)
EDIF	Electronic Design Interchange Format (Elektronik Tasarım Değişim Formatı)
FPGA	Field Programmable Gate Array (Alanda Programlanabilen Kapı Dizisi)
Gİ	Görüntü İşleme
HDL	Hardware Description Language (Donanım Tanımlama Dili)
HSA	Hücresel Yapay Sinir Ağları
HSA-ÇKM	HSA Çok Kapsamlı Makinesi
IC	Integrated Circuits (Entegre Devreler)
LTI	Linear time-invariant (Lineer zamanla değişmeyen)
MUX	Multiplexer (Çoğullayıcı)
PAR	Place and Root
RAM	Random Access Memory (Rastgele Erişimli Bellek)
SRAM	Static RAM (Statik Rastgele Erişimli Bellek)
VHDL	VHSIC Hardware Description Language
VLSI	Very Large Scale Integration (Çok Büyük Ölçekli Tümleştirme)

ŞEKİL LİSTESİ

ÖNSÖZ

Tez danışmanım Prof. Dr. Vedat TAVŞANOĞLU hocama, bilgi ve deneyimlerini benimle paylaşan Nerhun YILDIZ' a, çalışmam süresince desteğini ve sabrını benden esirgemeyen aileme, her zaman yanımda olan arkadaşım Burak ADA' ya teşekkür ederim.

ÖZET

Görüntü işleme bir görüntünün analog veya dijital elektriksel işaretlere dönüştürüldükten sonra çeşitli yöntemlerle istenilen şekle getirilmesidir. Birçok görüntü işleme tekniği görüntünün iki boyutlu bir sinyal olarak değerlendirilerek, çeşitli bilgisayar algoritmalarının uygulanması ile gerçekleştirilir. Görüntü işleme uygulamaları biyomedikal uygulamalar, uydu görüntüsü tanıma gibi birçok alanda kullanılmaktadır.

Hücresel Sinir Ağı modeli 1988 yılında L. O. Chua ve L. Yang tarafından ortaya atılmıştır. ACE4K ve ACE16K gibi analog yapıdaki HSA gerçeklemelerinin işlem hızı, sayısal HSA gerçeklemelerinin işlem hızlarına erişememektedir. Günümüzde dijital olarak tasarlanan sistemler, görüntüleri gerçek zamanlı olarak işleyebilecek kadar hızlı hale gelmiştir. Bunun yanı sıra, dijital tasarımda değişik ihtiyaçlara göre farklı düzenlemeler yapılabilir. Bu çalışmada 3x3 boyutunda çeşitli fitrelerin kullanılabileceği bir HSA yapısı üzerinde konvolüsyon bloğu dijital olarak tasarlanmıştır.

Görüntü işlemede sık kullanılan işlemlerden birisi filtrelemedir. Bu çalışmada filtreleme işlemi için HSA yapısının FPGA ile dijital emülasyonunda kullanılabilecek bir konvolüsyon bloğu tasarlanmıştır. Bloğu test etmek amacıyla gri tonlamalı bir görüntü örneği filtrelenip incelenmiştir. Kullanılan tasarım ortamı Xilinx firmasının ISE yazılımıdır. Konvolüsyon bloğu, Altera firmasının Cyclone-III FPGA Starter Kit ve Bitech HSMC Dijital Video Daughter Card (DVI in-out) kullanılarak gerçeklenmiştir.

İlk bölümlerde HSA yapısından genel olarak bahsedildikten sonra FPGA ve VHDL ile ilgili özet bilgilere yer verilmiştir. Daha sonraki bölümlerde tezin özgün kısmını oluşturan, HSA' ların sayısal yapısına yönelik iki boyutlu konvolüsyon bloğunun gerçeklenmesi anlatılmıştır. Son bölümde ise elde edilen sonuçlar yorumlanmış ve çalışmanın devamında gelecekte yapılacak geliştirmeler için bazı önerilerde bulunulmuştur.

Anahtar Kelimeler: Konvolüsyon, Hücresel Sinir Ağı (HSA), VHDL, FPGA.

ABSTRACT

Image processing can be defined as modifying an image by means of various techniques after converting the image to analog or digital electrical signals. Many image processing techniques include the application of computer algorithms to the images which are considered as two dimensional signals. Image processing applications are used in many areas such as biomedical applications, satellite image recognition...etc

Cellular network model was first introduced in 1988 by L.O. Chua and L. Yang. Analog CNN implementations such as ACE4K and ACE16K can not reach the processing speeds that digital implementations. Nowadays, digitally designed systems are fast enough to be able to process the images in real-time. Furthermore digital design is flexible since the system can be easily altered anytime for different needs. In this work, the convolution block of the CNN is digitally designed for 3x3 dimensional filters.

One of the most used operations in image processing is filtering. Spatial filtering of an image in time domain is done by convolution. In this thesis, a convolution block which can be used both for general purpose and for the digital emulation of CNN structure on FPGA is designed. For testing purposes, a digital grayscale two dimensional signal, that is obtained from an image, is filtered and analyzed. The used design environment ise ISE software of Xilinx firm. The convolution block is implemented on Cyclone-III FPGA Starter Kit and Bitech HSMC Dijital Video Daughter Card (DVI in-out).

After the general information about CNN, general information about FPGA and VHDL is given on the first chapter and the following chapters. After these chapters, the genuine implemetation of the two-dimensional convolution block is explained on the further chapters. Finally, the obtained results were interpreted and some suggestions are given for future improvements as ongoing work in the last chapter.

Keywords: Convolution, Cellular Neural Network (CNN), VHDL, FPGA.

1. GİRİŞ

Görüntü işleme bir görüntünün dijital işaretlere dönüştürülmesi ve bilgisayarda işlenmesi konularını içerir. Birçok görüntü işleme tekniği görüntünün iki boyutlu bir sinyal olarak değerlendirilmesini ve çeşitli bilgisayar algoritmalarının uygulanması ile gerçekleştirilir. Görüntü boyutu ve görüntü çözünürlüğü arttıkça, görüntüyü bilinen yöntemlerle işlemek hayli zaman almaktadır. Görüntüyü analog olarak işleyen ve sadece zaman sabiti kadar gecikmenin ardından sonuç veren bir mimariye sahip olan Hücresel Sinir Ağı (HSA) (Cellular Neural Network), L. O. Chua ve L. Yang, Oct. (1988) tarafından ortaya atılmıştır. HSA' yı merkezi işlem birimi olarak alan ve bu birimi yerel ve evrensel (global) bellek birimleri, program belleği gibi çevre birimleriyle birlikte bir bilgisayara dönüştüren Çok Kapsamlı HSA Makinesi (HSA-ÇKM) (CNN Universal Machine) CNN-UM, 1993 yılında Roska ve Chua tarafından önerilmiştir. Tasarlanan HSA-ÇKM yapısı önce küçük ölçekli (64×64, ACE4k), sonra daha büyük ölçekli (128×128, ACE16k) analog tümdevreler üzerinde gerçeklenmiştir. Bu yapıya analog ve lojik kelimelerinin birleşmesinden oluşan "analogic computer" adı verilmiştir. Bu yapıda, işlem zamanı görüntü boyutundan bağımsızdır. HSA-ÇKM' i tek yonga üzerinde gerçeklemek hayli zor olduğu için HSA' yı gerçekleyen alternatif yollar üzerinde durulmuştur. (Wen vd., 1994; Doan vd., 1994; Ikenaga vd., 1996; Keresztes vd.,1999)

Analog tasarımın zorluğuna karşın, sayısal tasarımın gürültüye olan bağışıklılığı ve eleman parametrelerine toleransı gibi avantajları göz önüne alınırsa, HSA' yı gerçekleyen dijital yapılar hem görüntü işleme hızı açısından hem de tasarım kolaylığı açısından dikkate değerdir. Tasarlanan bir hücresel işlemci kolaylıkla paralel çalıştırılabilir. Bu hücresel işlemcilerin kontrolü birbirinin aynıdır. Ayrıca hücresel işlemcilerin kendileri olabildiğince değiştirilebilirdir. İşlemci bloklarının içindeki toplayıcı, çarpıcı ve karşılaştırıcı gibi yapılar sistemin veri çözünürlüğüne göre analog tasarımla kıyas edilemeyecek bir hızda yeniden tasarlanabilir. Tasarım kolaylığı ve tasarım esnekliği değerlendirilirse HSA' yı dijital olarak gerçeklemek önemli bir konu olarak karşımıza çıkmaktadır.

Sayısal tasarım yapılırken tasarımın ne ile yapılacağı sorusu da hayli önem arz etmektedir. Günümüzde HDL (Hardware Description Language) ile dijital tasarım yapmak hayli kolaylaşmıştır. Bu dillerin sağladığı yüksek seviyeli tasarım imkânlarıyla tasarıma olabildiğince tepeden bakıp, detayları bu dillere bırakarak ve daha az zaman harcayarak hem hız hem de işgal edilen yonga alanı açısından verimli devreler üretmek mümkündür. Bu

dillerle yapılan tasarımın bazı avantajları; yeniden kullanılabilirlik, taşınabilirlik ve tasarım esnekliğidir.

Tasarlanmış bir toplayıcı bloğu özelleşmiş bir uygulamada kullanılabileceği gibi toplayıcı gerektiren herhangi bir uygulama için de kullanılabilir (yeniden kullanılabilirlik). Bu dillerin standartları olduğu için, kurallara göre yazılan bir kod her sistemde geçerlidir (taşınabilirlik). Tasarım esnekliğinde ise, dil olabildiğince geniş imkânlar sağlamaktadır. Örneğin, sentez öncesinde kaydedicilerin büyüklüklerini sembolik sabitlerle değiştirmek oldukça kolaydır. Bütün bunlara rağmen, tasarım yapılan dili iyi bilmek gerekmektedir. Aynı tasarımın farklı kodlayıcılar tarafından kodlanması farklı büyüklükteki devrelerin oluşmasına sebep olabilir. Dili iyi bilmek ve yazılan kodun devre karşılığını en azından sezinlemek devreyi daha verimli bir şekilde tasarlamayı kolaylaştıracaktır.

Bu çalışmada HSA' nın sayısal gerçeklemesi yapılmıştır. Sayısal gerçeklemede, literatürdeki birçok devre ve işlem yöntemleri incelenmiştir. Bu yapının en önemli özellikleri, FPGA dışında bir bellek elemanı (RAM) kullanılmaması, her bir pikseli üç saat darbesinde işlemesi ve çıkışın hesaplanması için yapılan Euler iterasyonu sayısının gerçeklenen işlem birimi sayısı ile belirlenmesidir. Dış bellek kullanılmaması sistemin karmaşıklığını ve maliyetini düşürmektedir. Sistem çok sayıda hücre içeren ve 3×3 şablonlarla çalışan konumdan bağımsız bir HSA yapısının emülasyonunu, ardışık düzen (pipeline) yapısına sahip tek bir hücre ile gerçekleştirmekte ve bu hücre, paralel (eş zamanlı) çalışan ve birbirine kaskad bağlı birçok işlem biriminden oluşmaktadır. Sonuç olarak; devrenin hızını azaltmadan, daha az kaynak kullanan hücresel işlemcilerle, tam paralel çalışan bir devre tasarımı yapılmıştır. Tasarımda, bir donanım tasarımı dili olan VHDL kullanılmıştır.

2. DOĞRUSAL ZAMANLA DEĞİŞMEYEN (LTI) SİSTEM TEORİSİ

LTI (Linear Time-invariant) Sistem Teorisi, yani doğrusal zamanda değişmeyen sistem teorisi uygulamalı matematikten gelir. Elektronik mühendisliğinde, özellikle işaret işleme, kontrol teorisi gibi alanlarda kullanılır.

Doğrusal zamanla değişmeyen sistem teorisinin tanımlayıcı unsurları anlaşılacağı üzere, doğrusallık ve zamanda değişmezliktir.

2.1 Ayrık Zamanlı LTI Sistemler

Ayrık zamanlı sistemler doğrusal zamanlı sistemlerin örneklenmesi ile elde edilir.

2.1.1 Ayrık Zamanlı Sinyallerin İmpuls (Birim Örnek) Sinyalleri ile Gösterimi:

Bir ayrık (zamanlı) sinyal, her noktada ötelenmiş birim örnek sinyallerinin uygun genlik değerleri ile çarpılarak toplanmasından oluşur. Böylece,

$$x[n] = ... + x[-3]\delta[n+3] + x[-2]\delta[n+2] + x[-1]\delta[n+1] + x[0]\delta[n] + x[1]\delta[n-1] + x[2]\delta[n-2] + ...$$

ile verilir. Bu ifade,

$$x[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot \delta[n-k] \tag{2.1}$$

şeklinde de yazılabilir. Örnek olarak $x[n] = u[n]$ şeklinde tanımlanan birim basamak sinyali,

$$u[n] = \sum_{k=0}^{\infty} \delta[n-k] \tag{2.2}$$

şeklinde birim örnek sinyallerinin toplamı cinsinden ifade edilir.

2.1.2 Ayrık Zamanlı Birim Örnek (Impuls) Yanıtı ve LTI Sistemlerin Konvolüsyon Gösterimi

Doğrusal bir sistemin $x[n]$' e yanıtı her impuls bileşenin ayrı ayrı yanıtlarının süper pozisyonu olacaktır. Ayrıca TI (Time-invariant) özelliği, bir TI sistemin ötelenmiş impulsa (birim örneğe) olan yanıtı, birim örnek yanıtının ötelenmiş versiyonlarının toplamıdır. TI ayrık sistemlerde konvolüsyon bu şekilde oluşur. $h_k[n]$, ötelenmiş birim örnek ($\delta[n-k]$) sinyallerine olan yanıtı iken çıkış;

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h_k[n] \tag{2.3}$$

olur. $X[n]$ sinyali $\delta[n+1], \delta[n]$ ve $\delta[n-1]$ sinyallerine yanıtları sırasıyla $h_{-1}[n], h_0[n]$ ve $h_1[n]$ olan bir lineer sisteme uygulansın. $X[n]$ sinyali $\delta[n+1], \delta[n]$ ve $\delta[n-1]$' in süper pozisyonu olarak yazılabildiği gibi, $x[n]$' e olan yanıtı bu bileşenlere olan yanıtlarının toplamı olarak da yazılabilir.

Bir lineer sistemin *n* anındaki yanıtı, o noktadaki giriş değerinden dolayı olan, süperpozisyon işlemi sonunda elde edilen yanıttır.

Bir LTI sistemde $h_k[n] = h_0[n-k]$ olduğundan notasyon kolaylığı için $h[n] = h_0[n]$ alındığında $h[n]$ bir LTI sistemin $\delta[n]$ yanıtıdır. Böylece $y[n]$ için:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] \tag{2.4}$$

yazılır ve bu ifade konvolüsyon olarak bilinir.

2.1.3 Impuls Yanıtı ve Konvolüsyon

Yukarıdaki bölümde bahsedilen konvolüsyonda amaç, sürekli LTI bir sistemin karakterize edilmesi ve sistem yanıtının birim impuls yanıtı cinsinden bulunmasıdır.

Bir görüntü sinyalini ideal bir filtreden geçirmek için yapılan işlem temel olarak konvolüsyon tekniğine dayanmaktadır. Konvolüsyon işlemi, çıkış görüntüsündeki her bir pikselin, giriş görüntüsündeki aynı pikselin komşuluğundaki piksellerin ağırlıklı toplamına eşit olduğu bir komşuluk operasyonudur. Filtrelenecek görüntüye uygulayacağımız konvolüsyon şablonuna konvolüsyon kerneli veya konvolüsyon kalıbı adı verilmektedir. Konvolüsyonda bir pikselin çıkış değeri kendisinin ve komşu piksellerin değerlerinin bir ağırlıklı toplamı olarak bulunur.

$$out(m,n) = in(m,n) * h(m,n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} in(k,l)h(m-k,n-l) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} in(m-k,n-l)h(k,l))$$

$$y(t) = x(t) * h(t); \stackrel{\Delta}{=} \int_{-\infty}^{\infty} x(t-\tau) \cdot h(\tau) d\tau \tag{2.5}$$

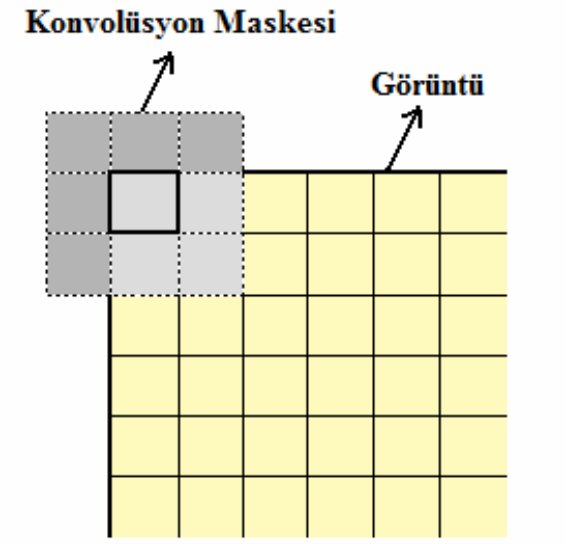
$$= \int_{-\infty}^{\infty} x(\tau) \cdot h(t-\tau) d\tau \tag{2.6}$$

Bu ifade konvolüsyon integrali veya süperpozisyon integrali olarak adlandırılır. *h(t)* sistemin impuls yanıtıdır, $x(\tau)=\delta(\tau)$. *y(t)* $x(\tau)$ giriş fonksiyonunun ağırlıklı ortalaması ile doğru orantılıdır. $y(t)$ çıkışı girişin ağırlıklı integrali olup, $x(\tau)$' nun ağırlığı $h(t-\tau)$' dur. Burada $h(t-\tau)$ bulunurken $h(t)$ ifadesinde τ değişken değişikliği yapılarak t sabit olarak alınır ve $h(t)$' den yansıma ile $h(-\tau)$ elde edilir. Daha sonra $t>0$ için sağa doğru kaydırılarak $h(t-\tau)$ bulunur. Son olarak $x(\tau)$ ile $h(t-\tau)$ çarpılarak integrali alınır.

Konvolüsyonun, LTI sisteminin çıkışını üretmesinin nedenini anlamak için, $\{x(u-\tau);u\}$ formülünün $\{x(u-\tau)\}$ fonksiyonu sabit τ ve u değişkeni ile temsil edilmesine izin verilir ve kısa gösterim $\{x(u);u\}$ nun $\{x\}$ ile temsil edilmesine izin verilir. Sonra sürekli zaman sistemi $\{x\}$ giriş fonksiyonundan $\{y\}$ çıkış fonksiyonuna dönüşür.

2.1.4 İki Boyutlu Konvolüsyon

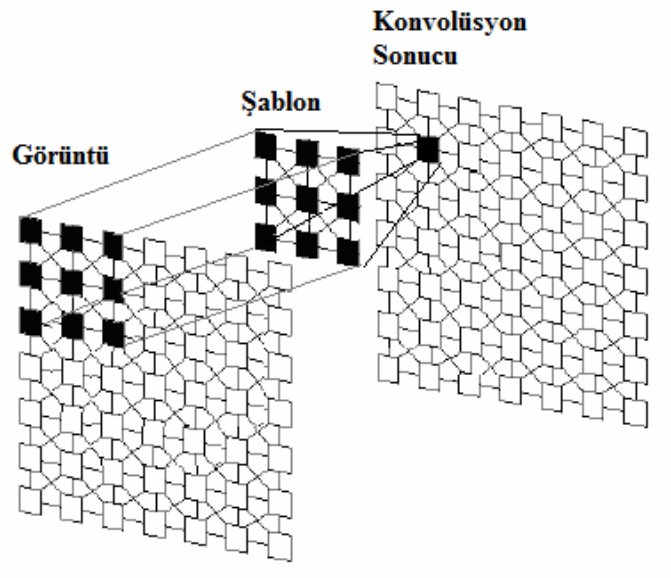
Görüntü üzerinde konvolüsyon işleminin matematiksel ifadesi (2.5) eşitliğinde gösterilmiştir. İki boyutlu konvolüsyon işlemi, piksellerin komşuluğundan faydalanarak bir filtre çekirdeğinin (konvolüsyon maskesi) resim üzerinde gezdirilmesi ile yapılır. Şekil 2.1' de gösterilmiştir.



Şekil 2. 1 2 boyutlu konvolüsyon

Konvolüsyon, yumuşatma, keskinleştirme, kenar belirleme gibi görüntü işleme fonksiyonlarını gerçekleştirmede çok sık kullanılmaktadır. Konvolüsyon işlemi uygulanan görüntünün bir pikselinin yeni değeri kendisinin ve çevresindeki piksellerin ağırlıklı ortalaması ile bulunmaktadır. Piksellerin ağırlıkları konvolüsyon şablonu (kerneli) olarak

adlandırılan bir matris ile belirlenir. Konvolüsyon şablonu uygulamaya göre farklı boyutlarda olabilmekle beraber bu tez dahilinde 3x3 boyutunda bir matristir. Şekil 2.2' de görüntü üzerindeki bir piksel için konvolüsyon işlemi görülmektedir. Şablon matris ve matrisin üzerinde bulunduğu görüntü matrisinin konvolüsyonu sonucu elde edilen piksel değeri çıkış görüntüsü üzerinde yer alır. Bu işlem görüntü üzerinde tüm piksellere uygulanarak görüntü üzerinde 2 boyutlu konvolüsyon işlemi yapılmış olur.



Şekil 2. 2 Görüntü üzerinde konvolüsyon işlemi

Tez dahilinde gerçeklenen konvolüsyon işlemi yapılmadan önce (3x3 boyutlu bir şablon için), öncelikle bir matris olan görüntünün piksellerinin sırasıyla tutulması gerekir. 3x3 boyutlu konvolüsyon şablonun işleme alınabilmesi için birbiri ardına üç satırda bulunan piksel değerlerine ihtiyaç vardır. İlk iki satırda bulunan tüm piksel değerleri ve üçüncü satırın ilk üç piksel değeri bir RAM bloğunda saklanmıştır. Görüntünün piksel değerlerini sıralama işlemi tamamlandıktan sonra konvolüsyon bloğunda işleme başlanır. Diğer işlemler ilerleyen bölümlerde anlatılacaktır.

3. HÜCRESEL SİNİR AĞLARI

Görüntü işleme, biyomedikalden uydu görüntüsü tanımaya kadar birçok alanda kullanılır. Görüntü boyutu ve görüntü çözünürlüğü arttıkça görüntüyü bilinen yöntemlerle işlemek hayli zaman almaktadır. Görüntü işleme ilk olarak analog yapılar üzerinde gerçeklenmiştir. Görüntüyü analog olarak işleyen ve sadece zaman sabiti kadar gecikmenin ardından sonuç veren bir mimariye sahip Hopfield Sinir Ağı' nın özel bir kümesi olan Hücresel Sinir Ağları (HSA) (Cellular Neural Network), 1988' de ortaya atılmıştır.

Sinir sistemimizdeki nöronların diğer nöronlarla bölgesel olarak bağlı olması, her bir nöronun 1000' e yakın komşuluk kurduğu nöron bulunduğu bilinmektedir, bu özellik HSA yapısının dayandığı en önemli özelliktir. Hopfield sinir ağlarındaki bütün hücrelerin birbirleriyle doğrudan bağlılığına karşın HSA' da her bir hücre en yakın komşuluğundaki hücre ile doğrudan, diğer hücrelerle dolaylı olarak bağlıdır. HSA' nın bu yapısı, ağın tümleşik devre gerçeklemesini olanaklı kılmış ve ilk HSA yongası (CNN-UM) Macaristan Teknik Üniversitesi' nde Prof. T. Roska ve öğrencileri tarafından gerçeklenmiştir.

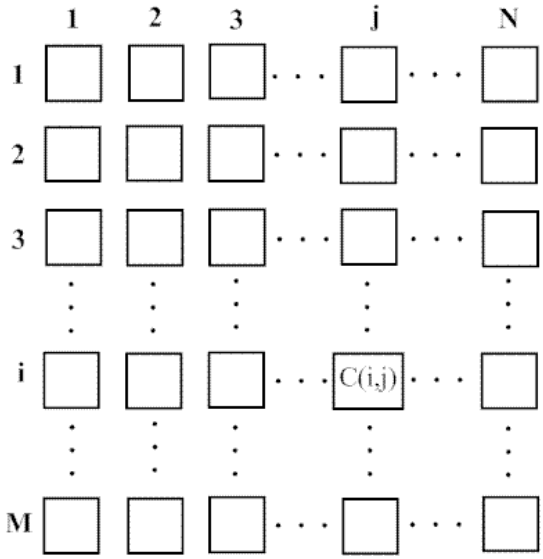
Bölgesel bağlantı sonucu, hem devrenin kapladığı alan azalmakta hem de verilerin devre üzerindeki taşıma yolları kısa olduğundan, toplam güç harcaması düşük olmaktadır. HSA' nın önemli avantajlarından bir diğeri de kararlı bir sonuca yakınsama süresinin devrenin boyundan bağımsız olması ve birkaç nöron gecikmesi ile belirlenmesidir.

3.1 Temel Gösterim ve Tanımlamalar

Bu bölümde Hücresel Sinir Ağları ile ilgili temel tanımlar yapılarak matematiksel gösterim şekilleri ortaya konmuştur.

3.1.1 Standart HSA Mimarisi

Hücresel Sinir Ağları hücre adı verilen birbirine eşdeğer analog işlemci birimlerinden oluşur. Bu hücreler genellikle iki veya daha çok boyutlu kare ızgara yapısı şeklinde dizilir. Buradaki (i, j) kartezyen koordinat indisleri $i = 1, 2, \ldots, M$ ve $j = 1, 2, \ldots, N$ şeklindedir. $M \times N$ adet hücre içeren iki boyutlu bir HSA yapısındaki hücre dizilimi Şekil 3.1' de verilmiştir.

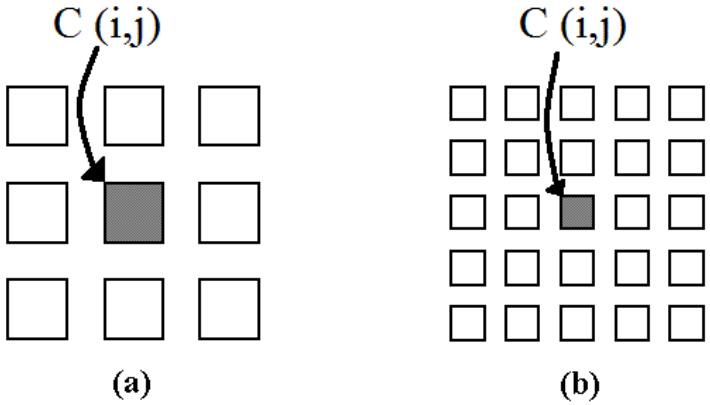


Şekil 3. 1 MxN adet hücre içeren iki boyutlu HSA yapısındaki hücre dizilimi

3.1.2 *C(i,j)* Hücresinin Etki Alanı

Her hücre kendi komşuluk sınırları içerisindeki hücrelere belli ağırlık değerleri ile bağlıdır. HSA' daki komşuluk aşağıda verilen denklemle tanımlanır.

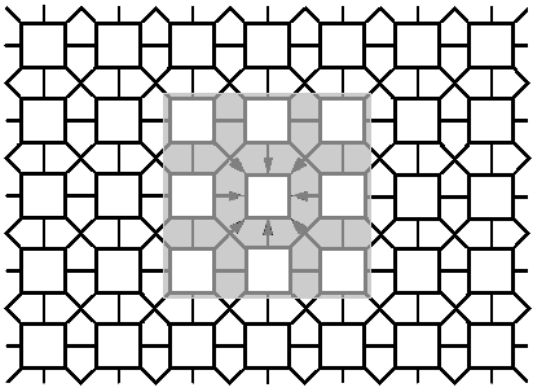
$$S_r(i,j) = \{C(k,l)\big|_{1\le k\le M,1\le l\le N} \max\{|k-i|,|l-j|\} \le r\}$$



Şekil 3. 2 (a) r = 1 ve (b) r = 2 komşuluk için elde edilen etki alanı

Burada $S_r(i,j)$ etki alanı (sphere of influence) olarak adlandırılır ve $C(i,j)$ hücresinin r komşuluğu içerisinde yer alan $C(k,l)$ hücrelerinin oluşturduğu bir kümeyi ifade eder. $C(i,j)$ hücresi $C(k,l)$ hücre kümesindeki her hücrenin çıkış değerinden doğrudan etkilenir. Etki alanı içerisinde yer alan hücre sayısı $(2r+1)\times(2r+1)$ adettir ve r komşuluklu bir yapı, $(2r+1)\times(2r+1)$ komşuluklu olarak da adlandırılır. Şekil 3.2' de sırasıyla bir ve iki komşuluk (r= 1 ve r = 2) için $C(i,j)$ hücresinin $S_r(i,j)$ etki alanı gösterilmiştir.

Gerçeklemede sağladığı kolaylıklar göz önünde tutularak HSA uygulamaları genellikle bir komşuluklu olarak oluşturulur. Bir komşuluklu HSA için hücreler arasındaki bağlantılar Şekil 3.3' te verilmiştir.



Şekil 3. 3 Bir komşuluklu HSA için hücreler arasındaki bağlantı

Her hücre, komşuluğu içerisinde bulunan hücrelerin çıkışlarından belirli ağırlık değerleriyle geri besleme alır. Bu ağırlık değerleri geri besleme sinaptik operatörü (feedback synaptik operator) olarak adlandırılır. Bu operatörün geri besleme olarak nitelendirilme nedeni, etki alanı içerisindeki hücrelerin çıkışından (sistemin çıkışından), $C(i,j)$ hücresini etkileyen girişlere işaret oluşturmasıdır. Her hücre ayrıca komşuluğu içerisinde bulunan hücrelerin giriş değerlerine de belli ağırlık değerleriyle bağlıdır, bu ağırlık değerleri ise ileri besleme veya giriş sinaptik operatörü (input synaptic operator) olarak adlandırılır. Her hücreye ayrıca sabit bir eşik (bias) girişi bağlıdır. Sinaptik operatörler (ağırlıklar) ve eşik değerleri değiştirilerek aynı HSA yapısı değişik görüntü işleme uygulamalarında kullanılabilir. Örneğin kenar belirleme, köşe belirleme, aşındırma (erosion), genişletme (dilation) vb…

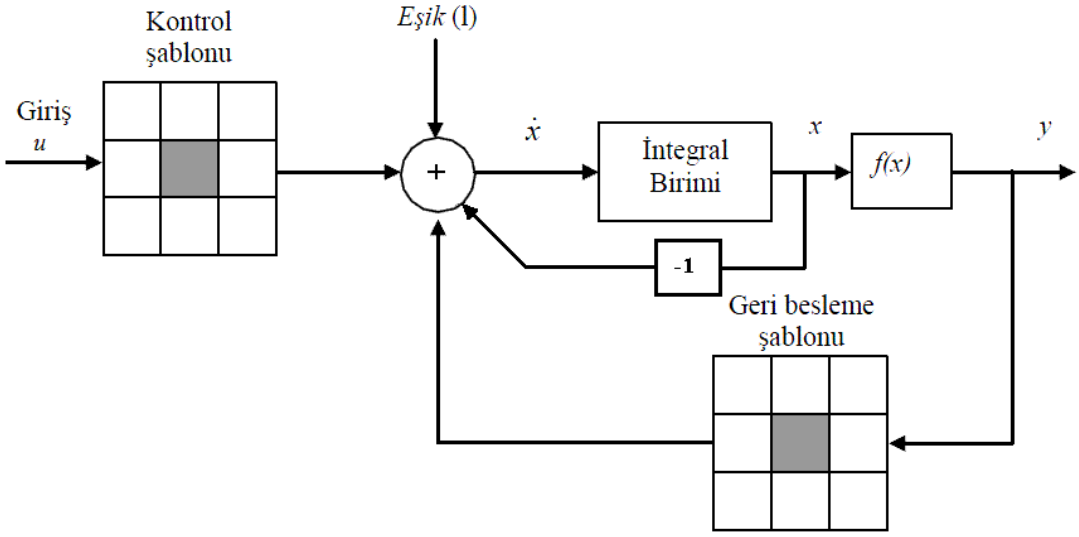
3.1.3 Bir Hücrenin Matematiksel Tanımı

3.1.3.1 Durum Denklemi ve Blok Diyagramı

$M \times N$ adet hücre içeren iki boyutlu HSA yapısındaki hücrelerin durum denklemi (3.2.a)' da verilmiştir.

$$\frac{dx(i,j,t)}{dt} = -x(i,j,t) + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)y(k,l,t) + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u(k,l,t) + I$$
$$1 \le k \le M, 1 \le l \le N \qquad (3.2.a)$$

Burada, $x(i,j)$ **C**(i,j) hücresinin durum değişkeni, $y(k,l)$ **C**(i,j) hücresinin çıkış değeri, u_{kl} girişler piksel değerlerini, I - $x(i,j)$ değerine eklenen sabit değeri göstermektedir. **A**$(i,j;k,l)$ geri besleme sinaptik operatörü ve **B**$(i,j;k,l)$ giriş (kontrol) sinaptik operatörüdür.



Şekil 3. 4 Bir hücrenin blok diyagramı

Şekil 3.4' de görüldüğü gibi bir HSA gerçeklemesi yapılırken 2 ayrı konvolüsyon işlemi yapılmaktadır. Bir kontrol şablonuyla giriş piksel değerleri ve bir geri besleme şablonu ile çıkış değeri bir konvolüsyon işlemine girer.

3.1.3.2 Çıkış Denklemi

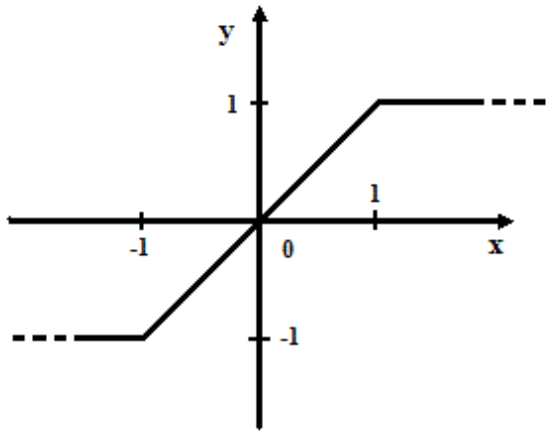
M×N adet hücre içeren iki boyutlu HSA yapısındaki hücrelerin çıkışı ile durumu arasındaki ilişkiyi ifade eden çıkış fonksiyonu ise (3.2.b)' de

$$y(i,j) = f(x(i,j)) = \frac{1}{2}\left(|x(i,j)+1| - |x(i,j)-1|\right) \qquad (3.2.b)$$

$$1 \le k \le M, 1 \le l \le N$$

Sınır koşulları, $S_r(i,j)$ etki alanının içinde yer almasına rağmen M×N' lik iki boyutlu dizinin dışında kalan eksik **C**(i,j) hücrelerinin bilinmeyen $y(\mathrm{k,l})$ ve $u(k,l)$ değişkenlerine verilen değerlerdir.

İlk koşullar, $t = 0$ anındaki $x(i,j,0)$ değerleridir.



Şekil 3. 5 Lineer olmayan çıkış fonksiyonu

3.2 Konumdan Bağımsız HSA

Eğer geri besleme **A**(*i*, *j*;*k*, *l*) ve **B**(*i*, *j*;*k*, *l*) bağlantı ağırlıkları ve *I* eşik değeri, hücrenin ağ içerisindeki yerleşiminden ve zamandan bağımsız olarak belirleniyorsa bu yapıya "Zamandan ve Konumdan Bağımsız HSA" adı verilir.

Bir hücre ile bu hücrenin komşuluk hücreleri arasındaki bağlantı ağırlıklarını ifade eden matrislere şablon denir. Bu şablonlar her bir hücre için uygulandığında, aynı bağlantı ağırlıklarını kopyalar. Şablon terimi bu değişmezlik özelliğini vurgulamak için kullanılır. Bu yapıda, eşik (bias) değeri de her hücre için aynıdır (I_{ij}=*I*). Bağlantı şekli tüm dizi üzerinde kopyalandığı için çok az sayıdaki bağlantı ağırlığının belirlenmesi yeterlidir. Bu, $M \times N$ boyutlarındaki bir HSA yapısının davranışını **A**(*i*,*j*;*k*,*l*), **B**(*i*,*j*;*k*,*l*) ve *I*' yı temsil eden "$2 \cdot (2r+1)^2 + 1$" tane reel sayıda oluşan kümenin belirleyeceği anlamına gelir. Bir komşuluklu (r =1) konumdan bağımsız bir HSA yapısının durum denklemi (3.3a)' da verilmiştir.

$$\sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)y(k,l) = \sum_{|k-i|\le r} \sum_{|l-j|\le r} A(i-k,j-l)y(k,l) \tag{3.3a}$$

$$\sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u(k,l) = \sum_{|k-i|\le r} \sum_{|l-j|\le r} B(i-k,j-l)u(k,l) \tag{3.3b}$$

$$I_{ij} = I$$

HSA' nın uygulamalarında en fazla 3×3 komşuluğun (etki küresi yarıçapı r =1) bulunduğu konumdan bağımsız yapı kullanılır. Bu uygulamalarda analizini kolaylaştıracak bazı

tanımlamalar yapılmıştır. Geri besleme (**A**(*i-k,j-l*)), giriş (**B**(*i-k,j-l*)), operatörleri ve $I(i,j)$ eşik değeri, hücrenin ağ içerisindeki yerleşiminden bağımsız olduklarından, bir şablon şeklinde gösterilebilirler.

3.2.1 Geri besleme operatöründen (A(i-k , j-l)) gelen katkı

$$\sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)y(k,l) = \sum_{|k-i|\le r}\sum_{|l-j|\le r} A(i-k,j-l)y(k,l)$$

$$\begin{aligned} &= a(-1,-1)y(i-1,j-1) + a(-1,0)y(i-1,j) + a(-1,1)y(i-1,j+1) \\ &+ a(0,-1)y(i,j-1) + a(0,0)y(i,j) + a(0,1)y(i,j+1) \\ &+ a(1,-1)y(i+1,j-1) + a(1,0)y(i+1,j) + a(1,1)y(i+1,j+1) \\ &= \sum_{k=-1}^{1}\sum_{l=-1}^{1} a(k,l)y(i+k,j+l) \end{aligned} \tag{3.4}$$

$$\sum_{k=-1}^{1}\sum_{l=-1}^{1} a(k,l)y(i+k,j+l) \overset{\Delta}{=} \begin{bmatrix} a(-1,-1) & a(-1,0) & a(-1,1) \\ a(0,-1) & a(0,0) & a(0,1) \\ a(1,-1) & a(1,0) & a(1,1) \end{bmatrix} \otimes \begin{bmatrix} y(i-1,j-1) & y(i-1,j) & y(i-1,j+1) \\ y(i,j-1) & y(i,j) & y(i,j+1) \\ y(i+1,j-1) & y(i+1,j) & y(i+1,j+1) \end{bmatrix}$$

$$= A \otimes Y(i,j) \tag{3.5}$$

Burada 3×3 boyutundaki ***A*** matrisi **"Geri Besleme Şablonu"** olarak isimlendirilir.

$Y(i,j)$ matrisi *MxN* boyutundaki çıkış görüntüsünün üzerinde 3×3 bir pencere gezdirilerek elde edilebileceğinden **"C(*i,j*) Hücresindeki Çıkış Görüntüsü"** olarak adlandırılır.

$\otimes$ sembolü nokta çarpımların toplamını ifade eder ve **"Şablon Nokta Çarpım"** olarak isimlendirilir.

Nokta çarpım matrissel bir çarpım olmayıp, skaler çarpımla, matrislerin birbirleriyle aynı indisteki elamanlarının çarpılmasıyla yapılır. Ayrık matematikte bu işlem uzaysal konvolüsyona karşı düşer. Bazı uygulamalarda **A** şablonu şu şekilde ikiye ayrılabilmektedir:

$a(0,0)$**;** ***A*** şablonunun merkez elemanını göstermek üzere ($a(k,l)$; *k=l=0*);

$$A = A^0 + \widetilde{A}, \quad A^0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a(0,0) & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \widetilde{A} = \begin{bmatrix} a(-1,-1) & a(-1,0) & a(-1,1) \\ a(0,-1) & 0 & a(0,1) \\ a(1,-1) & a(1,0) & a(1,1) \end{bmatrix} \tag{3.6}$$

A^0 : Şablonun Merkez Bileşeni,

$\widetilde{A}$: Şablonun Çevre Bileşeni olarak adlandırılır.

3.2.2 Giriş (kontrol) operatöründen (B(i-k, j-l)) gelen katkı

$$\sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u(k,l) = \sum_{|k-i|\le r}\sum_{|l-j|\le r} B(i-k,j-l)u(k,l)$$

$$\begin{aligned} &= b(-1,-1)u(i-1,j-1) + b(-1,0)u(i-1,j) + b(-1,1)u(i-1,j+1) \\ &+ b(0,-1)u(i,j-1) + b(0,0)u(i,j) + b(0,1)u(i,j+1) \\ &+ b(1,-1)u(i+1,j-1) + b(1,0)u(i+1,j) + b(1,1)u(i+1,j+1) \\ &= \sum_{k=-1}^{1}\sum_{l=-1}^{1} b(k,l)u(i+k,j+l) \end{aligned} \tag{3.7}$$

$$\sum_{k=-1}^{1}\sum_{l=-1}^{1} b(k,l)u(i+k,j+l) \overset{\Delta}{=} \begin{bmatrix} b(-1,-1) & b(-1,0) & b(-1,1) \\ b(0,-1) & b(0,0) & b(0,1) \\ b(1,-1) & b(1,0) & b(1,1) \end{bmatrix} \otimes \begin{bmatrix} u(i-1,j-1) & u(i-1,j) & u(i-1,j+1) \\ u(i,j-1) & u(i,j) & u(i,j+1) \\ u(i+1,j-1) & u(i+1,j) & u(i+1,j+1) \end{bmatrix}$$

$$= B \otimes U(i,j) \tag{3.8}$$

$$\dot{x}(i,j) = -x(i,j) + A \otimes Y(i,j) + B \otimes U(i,j) + I \tag{3.9}$$

3×3 boyutundaki **B** matrisi **'Giriş (Kontrol) Şablonu'** olarak isimlendirilir.

$U(i,j)$ matrisi $M \times N$ boyutundaki giriş görüntüsünün üzerinde 3×3 bir pencere gezdirilerek elde edilebileceğinden, "**C(*i,j*) Hücresindeki Giriş Görüntüsü"** olarak adlandırılır.

Bazı uygulamalarda **B** şablonu şu şekilde ikiye ayrılabilmektedir. $b(0,0)$ **; B** şablonunun merkez elemanını göstermek üzere. ($b(k,l)$ *; k=l=0*)

B^0 **:** Şablonun Merkez Bileşeni,

$\overline{B}$:Şablonun Çevre Bileşeni olarak adlandırılır.

3.3 Ayrık Zamanlı HSA

Ayrık zamanlı HSA (AZ-HSA) (Discrete Time CNN (DT-CNN)) modeli elde edilirken Chua-Yang modelinden yola çıkılmıştır:

$$\dot{x}(i,j,t) = -x(i,j,t) + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)y(k,l,t) + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u(k,l) + I(i,j) \tag{3.10}$$

Euler ileri yaklaşıklık formülü kullanılarak, (3.10) durum denkleminin ayrık zamandaki eşdeğeri (3.11) denkleminde verildiği gibi elde edilebilir.

$$\frac{x(i,j,n+1)-x(i,j,n)}{T(s)} = -x(i,j,n) + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)y(k,l,n) + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u(k,l) + I(i,j) \qquad (3.11)$$

Burada T_s örnekleme aralığıdır. (3.11) denklemi, sadece bir sonraki durum sol tarafta kalacak şekilde düzenlenirse AZ-HSA modeli (3.12) denkleminde verildiği gibi elde edilir.

$$x(i,j,n+1) = (1-T(s))x(i,j,n) + T(s)\Big(\sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)y(k,l,n) + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u(k,l) + I(i,j)\Big) \qquad (3.12)$$

$$y(i,j) = f(x(i,j)) = \frac{1}{2}\Big(|x(i,j)+1| - |x(i,j)-1|\Big)$$

3.4 Tüm Sinyal Aralığı Modeli ile AZ-HSA

Tüm sinyal aralığı (TSA) (Full Signal Range (FSR)) modelinden elde edilen AZ-HSA ile Chua-Yang modelinden elde edilen AZ-HSA arasındaki tek fark, TSA modelinde bir hücrenin durum ve çıkış değerlerinin birbirine eşit olmasıdır. Bu eşitlik, hücrenin durum değerinin [-1,1] aralığı dışına çıktığında kırpma (truncation) işlemiyle −1 veya +1 yapılmasıyla elde edilir. Böylelikle, (3.12) eşitliğiyle verilen Şekil 3.5' teki giriş-çıkış ilişkisi göz önünde tutulduğunda, *x* ve *y* terimleri birbirine eşit alınabilir. Sonuç olarak, *x* ve *y* terimlerinin birbirine eşit alınmasıyla elde edilen TSA ile AZ-HSA modeli (3.13.a) ve (3.13.b) denklemleriyle ifade edilir (Espejo vd., 1994; Dominguez-Castro vd., 1994).

$$v(i,j,n) = (1-T_s)x(i,j,n) + T_s\Big(\sum_{C(k,l)\in S_r(i,j))} A(ij,kl)x(kl,n) + \sum_{C(k,l)\in S_r(i,j))} B(ij,kl)u(kl) + I(i,j)\Big) \qquad (3.13.a)$$

$$x(i,j,n+1) = \begin{cases} 1, & v(i,j,n) > 1 \\ v(i,j,n) & |v(i,j,n)| \le 1 \\ -1 & v(i,j,n) < -1 \end{cases} \qquad (3.13.b)$$

(3.13.a) denklemindeki T(s) ve (1-T(s)) terimlerinin $A(ij,kl)$, $B(ij,kl)$ ve $I(ij)$ değerinin içine katılmasıyla $A'(ij,kl)$ $B'(ij,kl)$ ve $I'(ij)$ değerleri elde edilir. 3x3 komşuluklu konumdan

bağımsız bir HSA yapısı için $\overset{)}{A}$, $\overset{)}{B}$ şablonları ve $\overset{)}{I}$ değeri (3.14) eşitliklerinde görüldüğü gibi elde edilir.

$$\overset{)}{A} = T(s)\cdot\begin{Bmatrix} a(-1,-1), & a(-1,0) & a(-1,1) \\ a(0,-1) & \dfrac{1+T(s)\cdot(a(0,0-1))}{T(s)} & a(0,1) \\ a(1,-1) & a(1,0) & a(1,1) \end{Bmatrix},\ \overset{)}{B} = T(s)B,\ \overset{)}{I} = T(s)I \qquad (3.14)$$

Bu değişikliklerle birlikte (3.13.a) eşitliği, (3.15.a) ve (3.15.b) eşitliklerinde görüldüğü gibi iki parçalı şekilde ifade edilebilir.

$$v(i,j,n) = \sum_{C(k,l)\in S_r(i,j))} \overset{)}{A}(ij,kl)x(kl,n) + g(ij) \qquad (3.15.a)$$

$$g(ij) = \sum_{C(k,l)\in S_r(i,j))} \overset{)}{B}(ij,kl)u(kl) + \overset{)}{I}(ij) \qquad (3.15.b)$$

(3.15.b) eşitliğinde de görüldüğü gibi g(i,j) değerinin her piksel için sadece bir kez hesaplanması yeterlidir. *v*(i,j) değeri ise her iterasyonda değişir. 3x3 komşuluklu konumdan bağımsız bir HSA yapısı için (3.15.a) ve (3.15.b) eşitlikleri sırasıyla (3.16.a), (3.16.b) eşitliklerinde görüldüğü gibi yazılabilir.

$$v(i,j,n) = \overset{)}{A} \otimes X(i,j,n) + g(i,j) \qquad (3.16.a)$$

$$g(i,j) = \overset{)}{B} \otimes U(i,j) + \overset{)}{I} \qquad (3.16.b)$$

CASTLE ve Falcon sayısal HSA emülatörü yapıları bu AZ-HSA modelini kullanır. Bu modelin kullanım sebepleri:

1. Bir sonraki durumu hesaplanma işlemi, sonucu her iterasyonda değişen (3.16.a) denklemi ile sabit kalan (3.16.b) denklemi seklinde iki parçaya ayrılmıştır. Böylece *g(i,j)* değerinin her piksel için sadece bir kez hesaplanır (her iterasyon için tekrar hesaplanmaz).
2. Her iki eşitlik de bir şablon nokta çarpımı ile bir sabitin toplamından oluşur. Böylece aynı işlem birimi (processing unit) yapısıyla her iki eşitlik de hesaplanabilir.

3.5 Sınır Koşulları

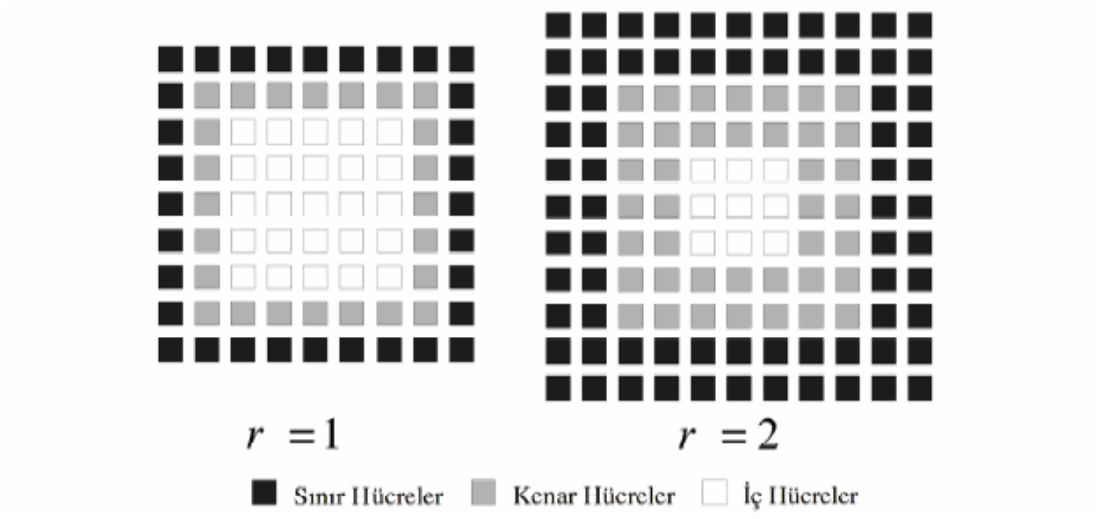
r komşuluk miktarına sahip bir HSA' da sınır koşulları göz önüne alınırken öncelikle $M \times N$ boyutundaki hücre dizisinin $(M+2r)\times(N+2r)$ boylarında bir dizinin ortasına oturtulmuş gibi düşünülmesi gerekir. Bu şekilde $M \times N$ boyutundaki dizinin kenardaki elemanlarının eksik olan sinaptik bağlantıları tamamlanmış olur. *MxN'* lik alanın dışında kalan C(*i, j*) hücrelerine

verilecek değerler sınır koşullarıdır. $r = 1$ komşuluğa ve dolayısıyla 3×3 boyutundaki etki alanına sahip bir HSA için en çok kullanılan sınır koşulları aşağıdaki şekilde sıralanabilir.

$x(i, j) = h(i, j; \tilde{x}_{ij}, t)$ $(i = 1,2, \ldots, M; j = 1,2, \ldots, N)$' deki sanal değerler sınır koşulları ile belirlenmektedir.

Pratik uygulamalarda kullanılan HSA yapıları, genellikle sınırlı ızgaralar (genellikle kare veya dikdörtgen) üzerinde tanımlanmaktadır. Bu nedenle, ızgaranın sınırlarında yer alan hücreler ızgaranın içinde yer alan hücrelerden daha az komşuya sahiptir. Komşu kümesindeki bu eksiklik, ızgaranın çevresine sanal hücreler eklenerek giderilebilir. Bu sanal hücrelere sınır hücreleri, ızgaranın içindeki hücrelere ise iç hücreler denir. En az bir tane sınır hücresi komsusu bulunan iç hücrelere ise kenar hücreleri denir. Burada sanal hücrelerin gerçek hücre olmadığına dikkat edilmelidir, çünkü (3.2.a) denklemine uymazlar. Sınır hücrelerinin durum (x) ve giriş (u) değerleri yapının sınır koşulları olarak adlandırılır. Izgaranın çevresindeki sınır hücrelerinin çerçeve genişliği *r'* ye eşit olmalıdır (Şekil 3.6).

Sınır koşullarını hesaba katmanın başka bir yolu da kenar hücrelerinin durum denklemlerini değiştirmektir. Fakat bu iki yöntem arasındaki fark sadece kavramsaldır, çünkü matematiksel olarak eşdeğer oldukları gösterilebilir.

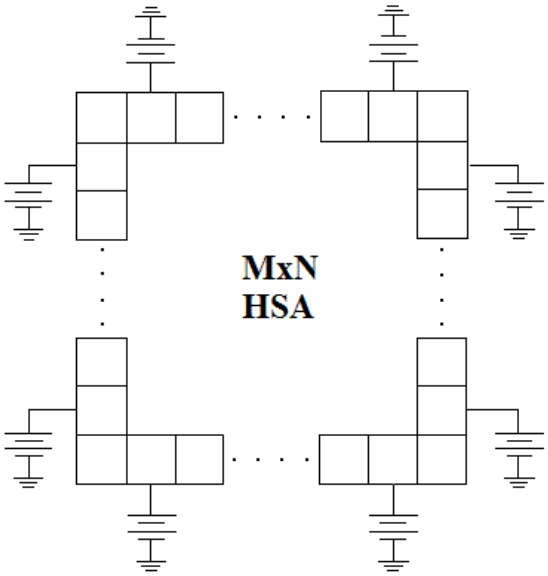


Şekil 3. 6 Komşuluk boyutu r' nin sınır hücrelerine etkisi (Saatçi,2003)

Pratikte sınırlı HSA yapılarının sınır koşullarını hesaba katmak için üç farklı yaklaşım bulunmaktadır. Şimdi bu yaklaşımlar ele alınacaktır.

3.5.1 Sabit (Drichlet) Sınır Koşulları

Eksik hücrelerin *x* ve *u* değişkenlerine sabit değerler vermek şeklinde uygulanır. Genelde sıfır olarak alınmakla beraber bu konuyla ilgili bir sınırlama yoktur. Sabit gerilim kaynağı veya toprak olarak düşünülebilecek sinaptik bağlantılardır (Şekil 3.7). Daha çok işlemlerin basitleştirilmesi amacıyla kullanılır.



Şekil 3. 7 Sabit (Dirichlet) Sınır Koşullarının Devre Gösterimi

Soldaki eksik hücreler: $y(i,0) = \alpha(i,0)$; $u(i,0) = \beta(i,0)$; $i = 1, 2, \ldots, M$

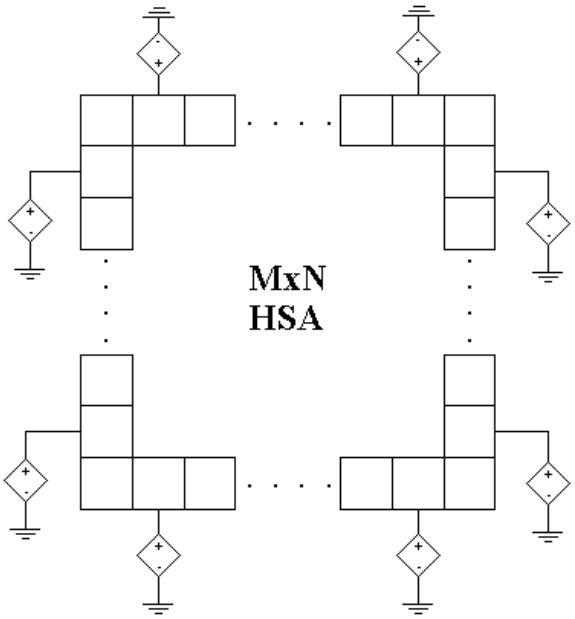
Sağdaki eksik hücreler: $y(i, N+1) = \alpha(i, N+1)$; $u(i, N+1) = \beta(i, N+1)$; $i = 1, 2, \ldots, M$

Üstteki eksik hücreler: $y(0, j) = \alpha(0, j)$; $u(0, j) = \beta(0, j)$; $j = 1, 2, \ldots, N$

Alttaki eksik hücreler: $y(M+1, j) = \alpha(M+1, j)$; $u(M+1, j) = \beta(M+1, j)$; $j = 1, 2, \ldots, N$

3.5.2 Sıfır Akı (Zero Flux (Neumann)) Sınır Koşulları

Gereken her yerde eksik hücrelerin yerine eksik hücrenin en yakın gerçek komşusu kullanılması şeklinde uygulanır. Örneğin sol kenarın dışındaki eksik bir hücrenin değerlerinin yerine doğrudan sol kenardaki hücrenin değerleri alınır (Şekil 3.8). Özellikle termodinamik yasalarına göre enerji sızıntısından sönümlenme riski taşıyan sistemlerde kullanılır.



Şekil 3. 8 Sıfır Akı Sınır Koşulları Devre Gösterimi

Soldaki eksik hücreler: $y(i,0) = y(i,1)$; $u(i,0) = u(i,1)$; $i = 1, 2, \ldots, M$

Sağdaki eksik hücreler: $y(i, N+1) = y(i, N)$; $u(i, N+1) = u(i, N)$; $i = 1, 2, \ldots, M$

Üstteki eksik hücreler: $y(0, j) = y(1, j)$; $u(0, j) = u(1, j)$; $j = 1, 2, \ldots, N$

Alttaki eksik hücreler: $y(M+1, j) = y(M, j)$; $u(M+1, j) = u(M, j)$; $j = 1, 2, \ldots, N$

3.5.3 Periyodik (Toroidal) Sınır Koşulları

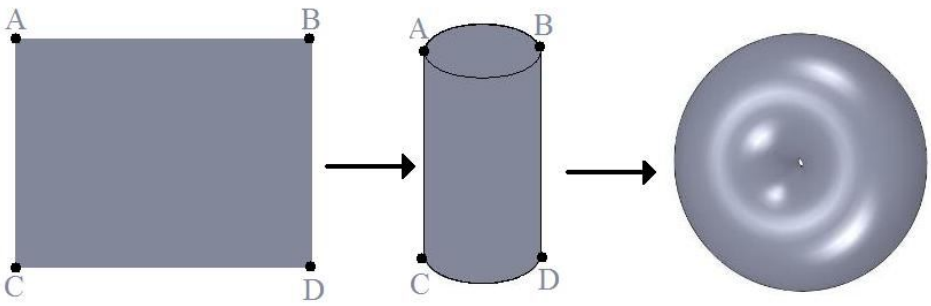
İki boyutlu HSA dizisi bir simidin yüzeyi olarak düşünülür. İki boyutlu dizi sol kenarı sağ kenarına komşu olacak şekilde silindir şekline getirilir, silindirin üst ve alt yüzeyleri de silindir bükülerek birleştirilirse iki boyutlu bir toroid, yani bir simit şekli elde edilir.

Soldaki eksik hücreler: $y(i,0) = y(i, N)$; $u(i,0) = u(i, N)$; $i = 1, 2, \ldots, M$

Sağdaki eksik hücreler: $y(i, N+1) = y(i,1)$; $u(i, N+1) = u(i,1)$; $i = 1, 2, \ldots, M$

Üstteki eksik hücreler: $y(0, j) = y(N, j)$; $u(0, j) = u(N, j)$; $j = 1, 2, \ldots, N$

Alttaki eksik hücreler: $y(M+1, j) = y(1, j)$; $u(M+1, j) = u(1, j)$; $j = 1, 2, \ldots, N$



Şekil 3. 9 Periyodik (Toroidal) sınır koşulları gösterimi

4. HSA' LARIN SAYISAL OLARAK GERÇEKLENMESİ

4.1 CASTLE ve Ardışık Düzen Yapısı İle Gerçeklenen HSA Simülatörü

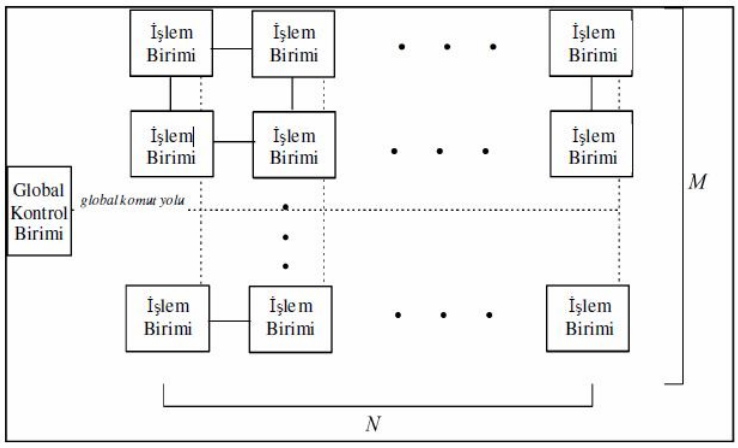
Farklı yapılarda HSA gerçeklemeleri mevcuttur, bunlardan iki tanesi CNN-UM ve CASTLE Mimarisi' dir. Bir sayısal gerçekleme olan CASTLE Mimarisi analog sistem olan CNN-UM ile aynı sonuçları verir, sadece hız farkı vardır. Eğer geçici hesaplamalar önemli ise hızdaki kayıp büyük bir eksikliktir. CASTLE pipeline tekniğini kullanarak, sayısal gerçeklemede sözü edilen hızı arttırmıştır. Bu bölüm altında önemli bazı dijital HSA emülatörü yapıları kısaca anlatılacaktır.

4.1.1 Orijinal CASTLE Mimarisi

CASTLE, HSA-EM (CNN-UM) sayısal bir kopyasıdır. Analog HSA hücresi geri Euler metodu kullanılarak fark denklemi haline getirilmiştir. CASTLE, CMOS teknolojisi kullanılarak ASIC bir tümdevre olarak tasarlanmıştır. Bu tümdevre bünyesindeki işlem birimleri, tümdevre üzerinde düzlemsel (matrissel) bir dağılım göstermektedir. Tümdevrenin hızı, 12 bit doğrulukta, bir iterasyon için işlem birimi basına 24 ns/hücre' dir (Keresztes vd., 1999).

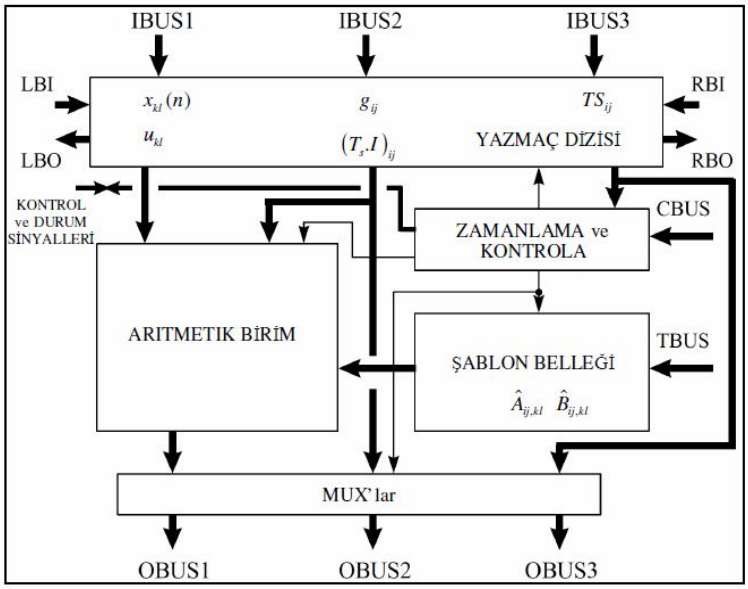
2002 yılında, CASTLE yapısındaki işlem birimi ardışık düzen yapısına sahip hale getirilerek hızı arttırılmıştır. Tasarlanan yapı Xilinx firmasının Virtex FPGA ailesinde denenmiştir. FPGA üzerinde gerçeklenen CASTLE, ardışık düzenli CASTLE, çarpıcı ve toplayıcıların da ardışık düzenli CASTLE yapıları için elde edilen hızlar sırasıyla, 6 bit doğrulukta, bir iterasyon için işlem birimi basına 35 ns/hücre, 9 ns/hücre ve 6 ns/hücre' dir (Hidvégi vd., 2002, 2003).

CASTLE yapısı matris şeklinde dizilmiş işlem birimleri ve bu işlem birimlerinin kontrolünü sağlayan bir kontrol birimden oluşur. Bu yapı Şekil 4.1' de (Keresztes vd., 1999) gösterilmiştir.



Şekil 4. 1 CASTLE yapısının blok diyagramı

Yatay olarak sıralanmış işlem birimleri görüntünün dikey olarak kesilmiş şeritleri üzerinde işlem yaparlar. Bir işlem biriminin işleyeceği resmin y ekseni üzerindeki piksel sayısı orijinal resimdeki kadardır, yani işlem birimi sayısı ile değişmez. Dikey olarak sıralanmış işlem birimleri, yatay eksende sıralanmış işlem birimlerinin çıkış değerlerini alarak bir sonraki iterasyon işlemini gerçekleştirirler. İşlem biriminin yapısı Şekil 4.2' de (Keresztes vd., 1999) verilmiştir.

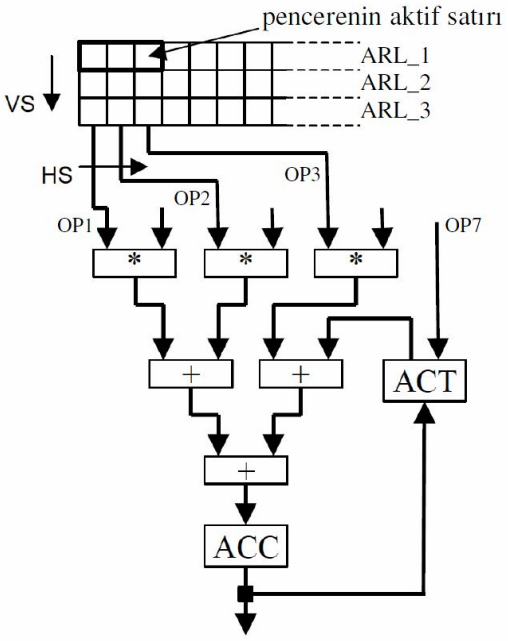


Şekil 4. 2 CASTLE işlem biriminin yapısı

IBUS1, IBUS2, IBUS3, CBUS, TBUS, LBI ve RBI yollarıyla (bus) veriler ve kontrol komutları işlem birimine girer. OBUS1, OBUS2, OBUS3, LBO ve RBO yollarıyla da veriler işlem biriminden dışarı çıkar. IBUS1 *u(k,l)* giriş ve *x(k,l,n)* durum değerlerini, IBUS2 T(s), I(i,j) ve g(i,j) değerlerini, IBUS3 T(s), I(i,j) kullanılacak şablon numarasını işlem birimine taşır. LBI ve RBI soldaki ve sağdaki işlem birimlerinden, islenen resmin sınır değerlerini

işlem birimine taşır. Bu sırada LBO ve RBO soldaki ve sağdaki işlem birimlerine görüntünün sınır değerlerini gönderir. Zamanlama ve kontrol işaretleri işlem birimine CBUS üzerinden taşınır. Şablondaki sayı değerleri işlem birimindeki şablon belleğine TBUS üzerinden yazılır. Şablon belleği, resim işlenmeye başlamadan önce işlem birimine yüklenir. İşlem biriminden çıkan OBUS1, OBUS2 ve OBUS3, sıra ile yeni durum değerlerini, g(i,j) değerlerini ve kullanılacak şablon numaralarını bir sonraki (altındaki) işlem birimine iletir.

Dikey konumda yerleştirilmiş işlem birimleri, verilerini (*x(k,l)*, *g(i,j)*, *T(s)(i,j)*) bir önceki işlem birimi dizisinden alırlar ve piksellere ait durumlar için bir iterasyon gerçekleştirerek yeni durumları bulurlar (3.15). Bu işlem birimlerinin işleme başlayabilmesi için durumların üç satır için oluşmasını beklemeleri gerekir ve bu satırlar da tümdevre içerisinde saklanır. Dikey işlem birimi sayısını sınırlayan durum çip alanı içinde oluşturulabilecek işlem birimi miktarıdır. Dikey işlem birimi sayısı ne kadar fazla olursa, çıkış değerleri dış belleğe yazmadan önce o kadar fazla iterasyon gerçekleştirilir. Böylece dış beleğe erişim miktarı azaltılmış olur. Aritmetik birimin basitleştirilmiş blok diyagramı Şekil 4.3' te verilmiştir. ARL_1, ARL_2 ve ARL_3 kaydırmalı yazmaçları aritmetik birim tarafından işlem görecek verileri saklar.



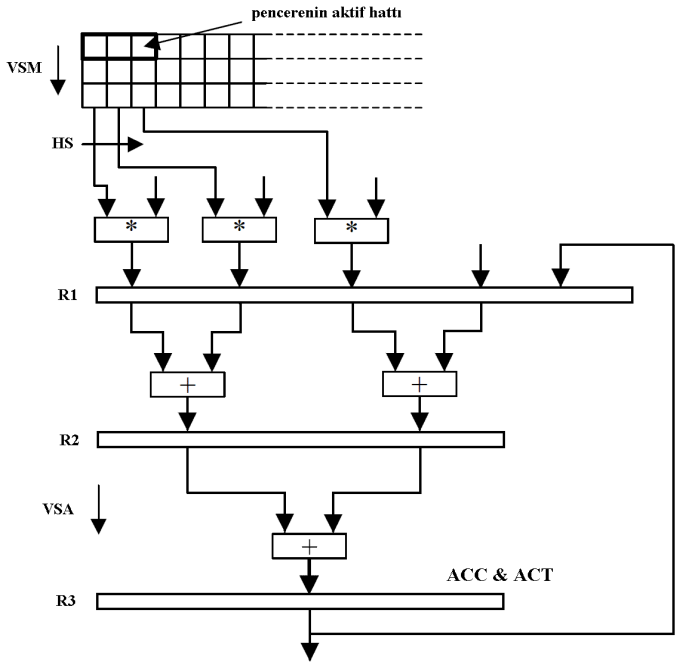
Şekil 4. 3 CASTLE yapısının aritmetik birim yapısı

Aritmetik birim üç adet çarpıcı üç adet toplayıcı ve iki adet akümülatörden (ACC, ACT) oluşmaktadır. Öncelikle, hesaplamada kullanılacak sabit değer g(i,j) veya T(s)I(i,j) geçici akümülatöre (ACT) yüklenir. Üç adet çarpıcı önce ilk satırdaki üç durumu bu durumlara ait şablon değerleri ile çarpar, çıkan sonuçlar ve ACT' ye yüklenen sabit, üç adet toplayıcı

tarafından toplanarak sonuç ACC' ye oradan da ACT' ye yazılır. Sonra işlem birimi bir alt satırdaki üç durumu bu durumlara ait şablon değerleri ile çarpar. Bu işlem için kaydırmalı yazmaçlardaki soldan ilk üç durumun değeri yukarıdan aşağıya doğru döndürülür (vertical shift (VS)). Bu döndürme işlemi üç kere tekrarlandığında hücrenin (pikselin) bir sonraki durum değeri hesaplanmış olur. Bu işlem tamamlandığında bir yandaki hücrenin bir sonraki durumunun hesaplanması amacıyla tüm kaydırmalı yazmaçlar sola döndürülür (horizontal shift (HS)) ve bir sonraki hücrenin yeni durumunun hesaplama işlemi başlar. IBUS2 ve IBUS3 de bu sola döndürme esnasında kaydırılır ve yeni hesaplanacak piksel için g(i,j) ve kullanılacak şablonun numarasını işlem birimine iletir.

4.1.2 Ardışık (Pipeline) yapılı CASTLE Mimarisi

Orijinal CASTLE aritmetik çekirdeği geçici kayıtçılar ile daha etkili hale gelebilir (Arato vd., 2000). R_1, R_2 ve R_3 kayıtçı grubu köle sahip yapısındadır. Orijinal yapının ACC ve ACT kayıtçıları R_3 grubunda birleştirilmiştir. Sonuç olarak işlem hızı önemli şekilde artar. Pipeline çözümündeki pencere hareketi orijinal CASTLE mimarisindekinden farklıdır. Aktif hattın bir piksel düşey yönde hareketinden sonra aynı anda aktif hat üç defa yatay yönde ve hesaplamanın ara sonucunda bir defa düşey yönde kayar (Şekil 4.4).



Şekil 4. 4 Ardışık düzenli CASTLE mimarisi

4.2 Falcon Yapısı

FPGA tümdevresi üzerinde çalışmak üzere tasarlanan, tek ve çok katmanlı HSA emülasyonu yapabilen Falcon yapısı 2002 yılında önerilmiştir. Bu yapının tek katmanlı hali işlem birimi diziliminin matrissel (düzlemsel) olması açısından CASTLE' a benzemektedir. (Nagy ve Szolgay, 2002, 2003, Beke vd. 2004) yayımları Falcon yapısı üzerinde yapılan gelişmeleri ele almaktadır.

CASTLE yapıları hızlı olmalarına rağmen, çalışma prensipleri kaydırmalı yazmaçlara (shift registers) dayandığından FPGA üzerindeki gerçeklemelerinde fazla donanım kullanırlar. Bu durum, FPGA üzerinde az işlem birimi gerçeklenmesine neden olur. Nagy ve Szolgay' ın (2002) yayımında verilen Falcon yapısı FPGA için daha uygundur. Falcon yapısı CASTEL yapısının FPGA için yeniden tasarlanmış ve çok katmanlı HSA emülasyonunu da gerçekleştirmesi için geliştirilmiş halidir.

4.3 Diğer Bazı Dijital HSA Emülatörü Yapıları

HSA uygulaması ile ilgili incelenen yayınlarda, HSA' nın matematiksel modeli, uzaysal boyutlar, aktivasyon fonksiyonu ve katmanların sayısına bağlı olarak değişmektedir. Önerilen mimari ise modifikasyon ve ayarlamalara maruz kalmış HSA' nın farklı dönüşümleri, işlem kapasitesi ile ortaya çıkar. Bu kısımda iki yönlü, tek katmanlı, uzayda değişmeyen HSA modeli kullanılmıştır.

Sürekli zamanda HSA modeli durum denklemi (3.2.a)**'** dan yola çıkılarak şu şekilde ifade edilebilir:

$$\frac{dx(i,j,t)}{dt} = -x(i,j,t) + A \otimes Y(i,j,t) + B \otimes U(i,j,t) + z(i,j)$$

$$y(i,j,t) = f(x(i,j,t)) = 0.5\left(|x(i,j,t)+1| - |x(i,j,t)-1|\right) \qquad (4.1)$$

$x(i,j,t)$ durum, $y(i,j,t)$ çıkış, $z(i,j)$ eşik değeri, $Y(i,j,t)$ maskelenmiş çıkış, $U(i,j,t)$ maskelenmiş giriştir. A geri besleme ve B ileri besleme şablonlarıdır. $\otimes$ şablonun nokta çarpım operatörüdür.

Ayrık zamanda HSA modeli (4.1) denkleminden yola çıkılarak;

$$x(i,j)^{n+1} = \overline{A} \otimes Y(i,j)^{n+1} + \overline{B} \otimes U(i,j)^{n+1} + \overline{z}(i,j) \qquad (4.2)$$

(4.2) eşitliğinden yola çıkılarak;

$$g(i,j) = \overline{B} \otimes U(i,j)^{n+1} + \overline{z}(i,j) \tag{4.3}$$

$$x(i,j)^{n+1} = \overline{A} \otimes Y(i,j)^{n+1} + g(i,j) \tag{4.4}$$

elde edilir.

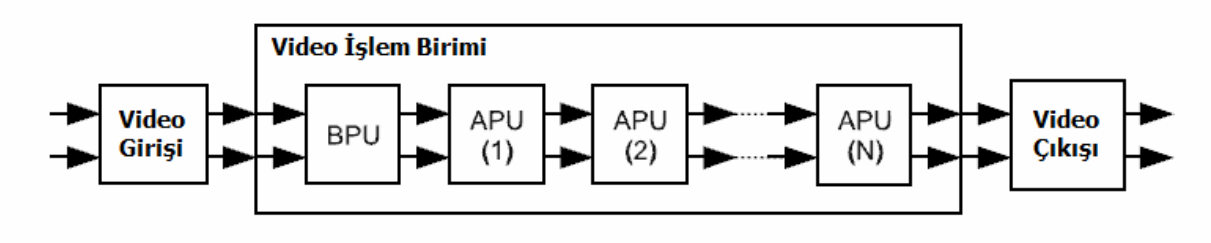
(4.3) ve (4.4) eşitlikleri sırasıyla BPU (B-Processing Unit) ve APU (A-Processing Unit) olarak gösterilirse;

$$BPU{=}g(i,j) = \overline{B} \otimes U(i,j)^{n+1} + \overline{z}(i,j)$$

$$APU(1) = f(\overline{A} \otimes Y(i,j)^{0} + BPU)$$

$$APU(n+1) = f(\overline{A} \otimes APU(n) + BPU) \tag{4.5}$$

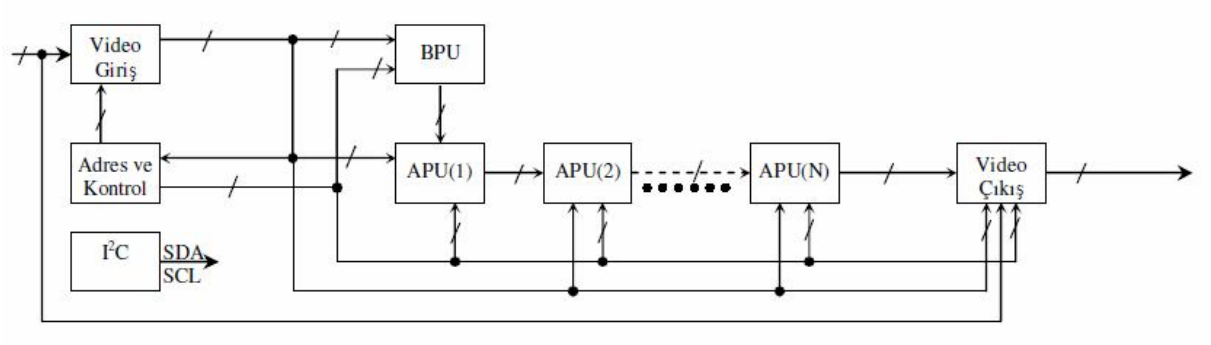
(4.5) eşitliğinde anlaşılacağı üzere BPU ve APU birimlerini gerçeklemek için konvolüsyon işlemini yapan bir bloğa gerek duyulmaktadır. Bu çalışmada gerçeklenen yapı bunu temel almaktadır. Bu sayede hem BPU birimini, hem de APU birimini gerçekleştirmek kolaylaşır.



Şekil 4. 5 FPGA gerçeklemesi blok diyagramı

2009 yılında Kamer Kayaer' in çalışmasında gerçeklenen HSA Emülatörü, FPGA tümdevresinin içinde bulunan devrenin blok diyagramı Şekil 4.6' da gösterilmiştir. İşlem birimlerinin kaskad bağlanabilmesi sağlanmış ve sistem VGA işareti alabilecek hale getirilmiştir.

Gerçeklenen sistem 640×480 piksel 60 fps 9 bit gri tonlamalı VGA işaretini alır, dış RAM kullanmadan gerçek zamanlı olarak HSA yöntemiyle işler ve elde ettiği çıkış görüntüsünü bir VGA monitöründe gösterir.



Şekil 4. 6 FPGA içerisindeki devrenin blok diyagramı (Kayaer, 2008)

Bu yapı içerisinde iki adet blok tez dahilinde incelenmiştir. Bunlar BPU ve APU(n) bloklarıdır**.** Bu bloklar ayrık zamanlı HSA (DT-CNN) emülasyonu yapan işlem birimi bloklarıdır. BPU bloğu "Video Giriş" bloğundan aldığı video girişini **B** şablonu ile işler. Elde ettiği sayıyı (sabit (*g(i,j)*)) APU(1) bloğuna iletir. Bu bloktan bir tane olma nedeni hesaplanan değerlerin (*g(i,j)*) o giriş görüntüsü için aynı (sabit) olmasıdır.

BPU bloğunun hesapladığı değerler APU(n)' ler tarafından kullanılır ve değiştirilmeden APU(n)' den APU(n+1)' e taşınır. APU(1) ilk **A** şablonu (durum) iterasyonunu gerçekleştirir. Bu iterasyon gerçekleştirilirken, ilk durum giriş görüntüsü veya herhangi bir sabit değer olarak alınabilir. APU(1) ilk iterasyonu gerçekleştirdikten sonra elde ettiği yeni durumu APU(2)' ye aktarır. Diğer APU' lar **A** şablonu (durum) iterasyonu yapmaya devam ederler. Daha sonra buradan elde edilen sonuçlar Video Çıkış Bloğuna aktarılır, sonuç elde edilir.

5. TEZ DAHİLİNDE GERÇEKLENEN HSA EMÜLATÖRÜ İÇİN KULLANILAN ARA BİRİMLER

Tez dahilinde yapılan gerçeklemede kullanılan ISE yazılımı VHDL (.hdl), Verilog (.v) veya yazılımın içindeki grafik editörüyle hazırlanan şematik (.sch) kaynak kodlarını derleyebilir. Ayrıca IP (Intellectual Property) adı verilen ve "CORE Generator" isimli, grafik arayüzlü bir yazılım ile oluşturulan donanım blokları (.xco, .xaw) da tasarıma eklenebilir. Projenin temel (ana) dosyası VHDL, Verilog veya şematik olabilir. Tez' de gerçeklenen yapılar VHDL kullanılarak oluşturulmuştur. Projenin gerçeklenmesi üç ana aşama ile yapılır. Sentezleme (synthesize), gerçekleme (implement) ve programlama dosyası oluşturmadır. Bu aşamalar sıra ile işlem penceresinde görülmektedir.

Sentezleme işlemi yazılan kodların doğruluğunu denetler, VHDL kodlarını sentezleme yazılımının anlayabileceği birimlere dönüştürür (derleme (compile)) ve bu işlemden sonra tasarlanan devreyi, kullanılan FPGA modeli içerisindeki yapılar ile gerçekleştirilebilecek hale getirir. Sentez işlemi sonucunda tasarımdaki her VHDL kodu "Xilinx Implementation" yazılımının anlayacağı bir EDIF (.edn) veya NGC (.ngc) devre dizgisi (netlist) dosyasına dönüştürülür. Gerçekleme (Implement) işlemi üç asamadan oluşmaktadır. Bunlar, "Translate", "Map" ve "Place & Route" işlemleridir.

Programlama Dosyası Oluşturma işleminde, FPGA' in programlanması için kullanılacak olan BIT dosyası ve bu dosyanın PROM içine yüklenecek hali olan MCS dosyası oluşturulur.

5.1 Donanım Tanımlama Dilleri (Hardware Description Languages) (HDLs)

VHDL en çok kullanılan yapısal tanımlama dilidir. HDL' ler sayısal sistemin yapısını veya davranışını tanımlamada kullanılır. HDL' de yapılan devre tanımı kullanılarak, bir benzetim programı ile yazılan tanımlamanın istenilen fonksiyonu sağlayıp sağlamadığı kontrol edilir. Tanımlamanın doğruluğu görüldükten sonra, bir sentezleme programı ile devrenin fiziksel yapısı elde edilir.

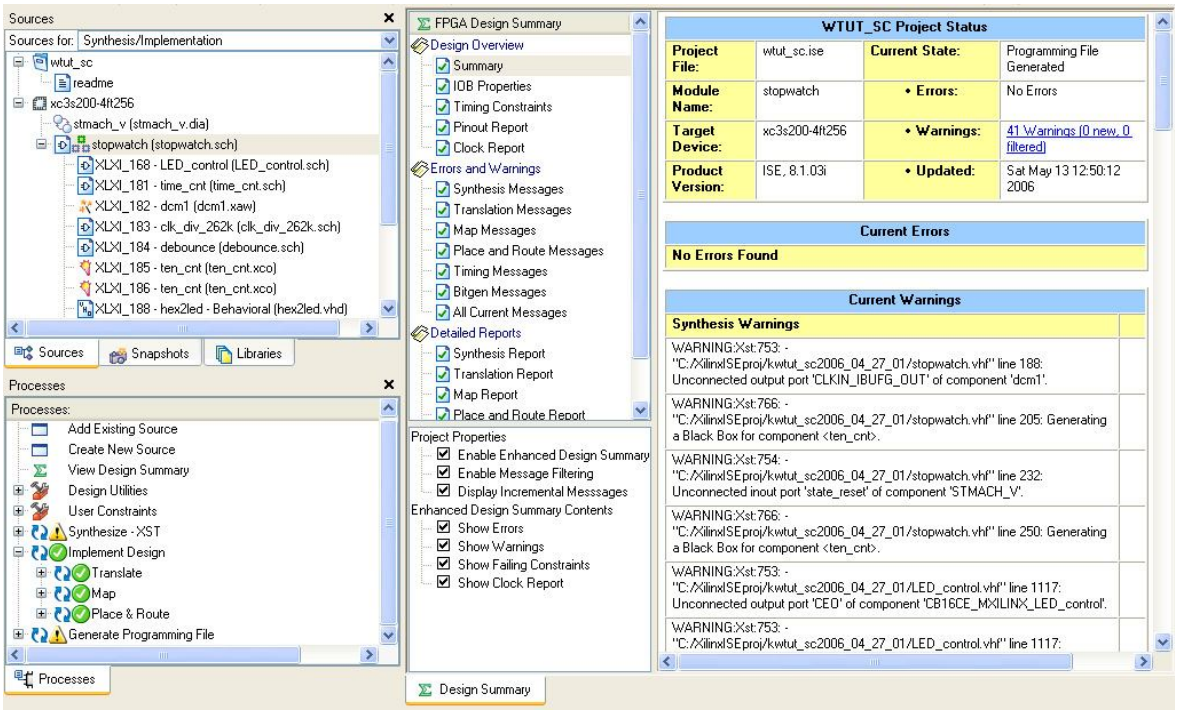
HDL tabanlı tasarım yöntemlerinin geleneksel kapı seviyesinde tasarım yöntemlerine göre bir takım üstünlükleri vardır. İlk olarak, tasarımcının daha az zamanda ve devre tasarımı hakkında çok fazla bilgi sahibi olmadan tasarım yapmasını sağlamaktadırlar. İkinci olarak, aynı tanımlama ile devrenin teknolojilerdeki yapısı elde edilebilmektedir. Bir sentezleme programından yararlanarak, HDL ile yapılan tanımlama her defasında farklı teknolojilerdeki elemanlardan oluşan devre şemasına çevrilebilir.

VHDL kullanılarak, sayısal tasarımın algoritmik, RTL ve lojik kapı seviyeleri gibi farklı seviyelerde tanımlanması mümkündür. Yukarıdan aşağıya tasarım yönteminde, sistem yüksek seviyede tanımlanır ve sonra daha alt seviyedeki bloklara ayrılır. Davranışsal mimari ile bir sistemden beklenen fonksiyon geleneksel programlara benzer şekilde, tanımlanır. Fakat sistemin nasıl gerçekleneceği konusunda ayrıntılı bilgi verilmez. Veri akışı şeklindeki mimari (RTL) ile kontrol ve veri işaretlerinin devre içindeki akışı tanımlanır. Bu işlem yapılırken, toplayıcılar, karşılaştırıcılar, kod çözücüler gibi daha önceden tanımlanmış kombinezonsal devre fonksiyonları ve basit lojik kapılar kullanılır. Yapısal mimari ile tasarımın alt sistemlerini oluşturan daha önce tanımlanmış modüllerin birbirleriyle olan bağlantıları tanımlanır (Örs, 1998).

5.2 Kullanılan Yazılım

ISE yazılımı bütün Xilinx FPGA ve CPLD aileleri üzerinde devre tasarımı için kullanılan bir tasarım geliştirme yazılımıdır. Bünyesinde tasarım, simülasyon ve JTAG kablosu yardımı ile tasarımı hedef bir FPGA veya FLASH tümdevresine yükleme işlemini yapan programlar barındırmaktadır. Tasarım geliştirme ve gerçekleme için tek basına yeterli olmakla beraber, kendisine tümleştirilen yardımcı yazılımlarla beraber de çalışabilmektedir. Bu yazılımlara, simülasyon için kullanılan Mentor Graphics firmasının ürettiği ModelSim yazılımı örnek gösterilebilir. Tez dahilinde gerçekleme sırasında yazılan VHDL kodlarının, tasarlanan yapının özelliklerini sağlayıp/sağlamadığını kontrol etmek için ModelSim simülasyon yazılımı kullanılmıştır.

ISE yazılımı proje yaklaşımı ile çalışır. Proje için kullanılan ve yazılım çalıştırılırken oluşturulan bütün dosyalar bir proje klasörü içinde saklanır. Projeye dahil edilen dosyalar ekranın sol üst tarafındaki Kaynaklar (Sources) penceresinde görünür. Dosyalar hiyerarşik bir şekilde sıralanmıştır. Bu dosyalardan biri seçildiğinde Kaynaklar penceresinin altındaki İşlemler (Processes) penceresinde bu kod için yapılabilecek işlemler görülür ve kod üzerinde yapılmak istenen işlemler bu pencereden seçilerek gerçekleştirilebilir. ISE yazılımının ekran görünüşü Şekil 5.1’ de verilmiştir.



Şekil 5. 1 ISE 11.1i yazılımının ekran görünüşü

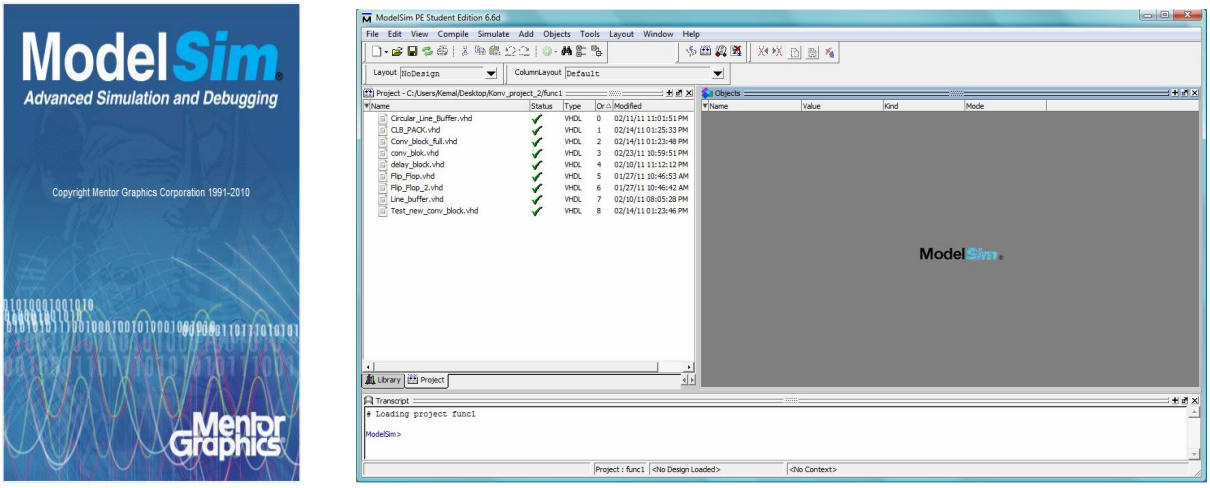
5.2.1 VHDL ile Benzetim

VHDL' de yazılan bir tasarım tanımı, bir VHDL benzetim programından faydalanılarak test edilir. Tanımlamanın benzetimini yapabilmek için benzetim programına bir takım test girişleri verilir, program daha önceden belirlenen aralıklarla bu test girişlerini tasarımın modeline uygular ve çıkışları üretir. Bu sonuçlar tasarımcı tarafından gözlenerek modelin istenildiği gibi çalışıp çalışmadığına karar verilir.

Benzetim tasarımın her aşamasında kullanılabilir. Tasarımın yüksek seviyelerinde yapılan benzetimde sadece tasarımın fonksiyonel davranışı hakkında bilgi elde edilir. Bu aşamada benzetim çok hızlıdır, fakat benzetim sonuçlarından tasarımın gerçek devre elemanları ile çalışması ve zamanlaması konusunda çok fazla bilgi elde edilemez. Daha düşük tasarım aşamalarına gidildikçe benzetim daha fazla zaman alacaktır, fakat benzetim sonuçlarından tasarımın çalışması ve zamanlaması konusunda daha fazla bilgi elde edilebilir.

5.2.2 Modelsim Benzetimi

Modelsim Mentor Graphics Şirketi tarafından tamamen FPGA benzetimi için geliştirilmiş bir benzetim aracıdır. Modelsim. FPGA üzerinde gerçekleme yapmadan önce yazılan VHDL ve Verilog kodlarının benzetimini tıpkı donanım üzerindeki gibi yapar. Modelsimde yapılan benzetime geçmeden önce Modelsimin kısa bir tanıtımı yapılmıştır.



Şekil 5. 2 Modelsim giriş sayfası

Modelsim açıldığında ilk olarak Şekil 5.2' deki sayfa görülür. Bu sayfanın solundaki alanda benzetimini yaptığım basit bir giriş işaretinin sabit bir sayı ile çarpılıp çıkış işareti oluşturulmaktadır. Giriş işareti metin dosyası halinde tutulmaktadır. Benzetim sırasında metin dosyası bir işaret kaynağı görevi görmektedir. Çıkış ise başka bir metin dosyasına yazılmaktadır.

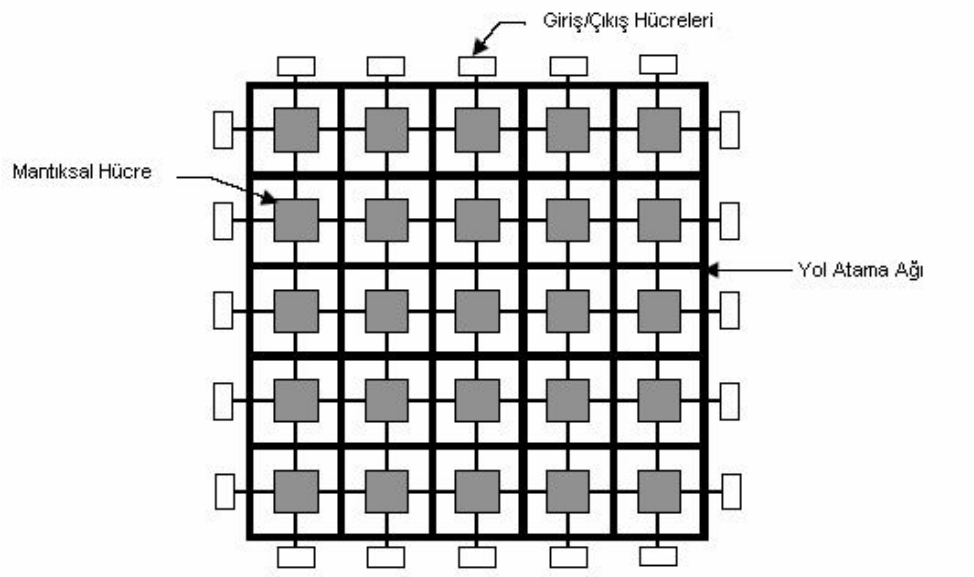
Yazılan VHDL kodları önce derlenmekte ve yazım hatalarına taranmaktadır. Herhangi bir hatada derleyici uyarıyı Şekil 5.2' deki resmin alt kısmında görüldüğü gibi vermektedir. Herhangi bir uyarı olmadığı zaman derlenme işlemi başarılmış sayılmaktadır. Derleme işlemi başarılı olduktan sonra sıra benzetim işlemine gelmektedir. Benzetim işlemi için ayrı bir pencere açılır ve bu pencerede tanımlanmış olan bütün işaretler gözükmektedir.

Xilinx ISE üzerinde VHDL kodu ile yazılan tasarım tanımı, Modelsim programı ile test edilerek tasarlanan modelin uygunluğuna karar verilir. İstenilen sonuçları veren tasarım artık FPGA üzerinde çalıştırılmaya hazırdır.

5.2.3 VHDL ile FPGA Programlaması

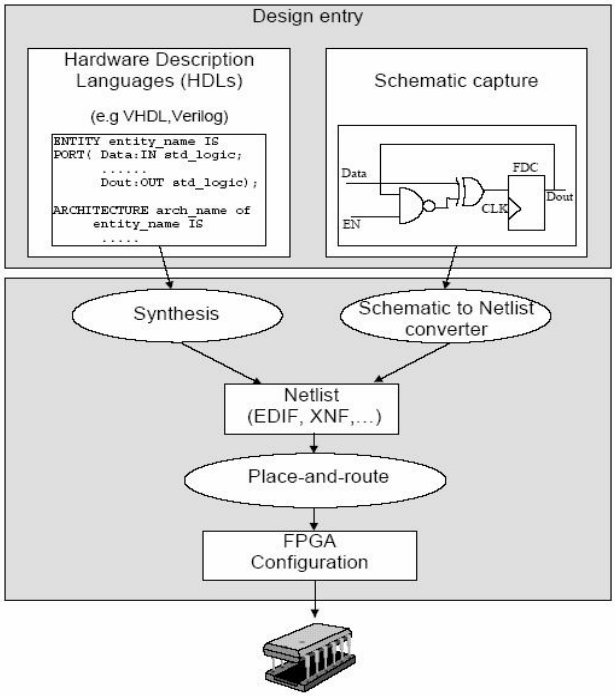
FPGA' ler programlanabilen anahtarlar yardımı ile birbirine bağlanabilen mantık hücreleri dizisi içerir. Mantıksal hücreler paket pinlerine programlanabilen giriş çıkış hücreleri ile bağlıdır.

Tekrar ayarlanabilen donanım çözümlerinden, elektriksel olarak diğer bir algoritmanın tekrar programlanabileceği FPGA' ler yüksek performans sağlarlar. Her ne kadar ilk FPGA' ler basit mantık devreleri ve kontrol uygulamalarında kullanılsalar da, günümüzdeki FPGA' ler milyon adet kapılarla ifade edilen bir yapıya kavuşmuşlardır. Üstelik tümleşik devre üretimindeki gelişmeler FPGA' lerin çalışabileceği maksimum hızları arttırmıştır. Görüntü ve video işleme uygulamaları için geliştirilen günümüz FPGA' leri 500 MHz' lik bir bant genişliği ile çevresindeki diğer elemanlarla haberleşebilmektedir.



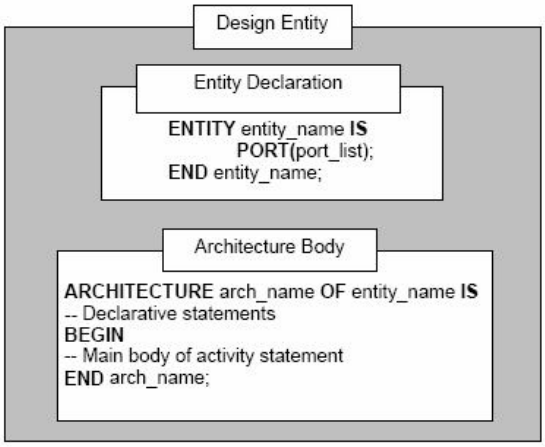
Şekil 5. 3 FPGA' in genel yapısı

Tipik bir FPGA tasarım akış diyagramı Şekil 5.3' te verilmiştir.



Şekil 5. 4 FPGA tasarım çevriminin genel görünüşü

VHDL dili donanım modeli için varlık tasarımı (design entity) kavramını kullanır. Her varlık tasarımı (design entity) bir varlık bildirimi ve yapı bloğundan (Şekil 5.4) oluşur.



Şekil 5. 5 VHDL temel yapısı

Varlık bildirimi varlığın dış dünya bağlantılarını belirtir, tıpkı bir devrenin giriş ve çıkışları gibi. Yapı bloğu (Architecture body) ise giriş ve çıkışlar arsındaki ilişkinin nasıl olduğunu tanımlamaktadır. Bir varlık birbiri ile ilişkili farklı yapılara sahip olabilir. Bunun bir yolu varlık tasarımının alt modüller ile yapılmasıdır. Alt modüllerin hepsi bir varlıktır ve sinyaller kullanılarak birbirlerine bağlanırlar. Çoğu durumda, varlık tasarımı yapısal olara ayrılmadan tanımlanır. Örnek olarak, eğer bir tasarım (IC - Integrated Circuits) tümdevreler ile yapılacaksa, tümdevrelerin içyapılarını bilinmesine gerek yoktur. Bu gibi durumlarda modül

tarafında gerçeklenen fonksiyonun tanımı gerekir. İçyapısına gerek duyulmamaktadır. Bu gibi tanımlamalara fonksiyonel (functional) ya da davranışsal (behavioural) tanımlama denir.

Kompleks yapılar bir fonksiyonun girişi olarak tanımlanamayabilirler. Geri dönüşümlü bir sistem, çıkış zamanının bir fonksiyonu olabilir. VHDL bu problemi çalıştırılabilen bir program formunda izin verilmiş davranışsal tanımlama yardımı ile çözer. Bu gibi durumlar için VHDL' e ekstra programlama yetenekleri eklenmiştir. Eklenenler sırası ile veri tipleri (integers, floating points, physical, enumeration, arrays), veri nesneleri (constants, variable and signals), ardışık ifadeler (if and case statements), alt programlar (procedures and functions) , paketlerdir.

Kayıtçılar ve kayıtçılar arası mantık elemanları ile yapılan tanımlamalar RTL (Register Transfer Level) tanımlamalardır. Bu tip tanımlamalar FGPA den bağımsız ve taşınabilirlerdir. VHDL kodu derleyici (synthesiser) tarafında derlenerek FPGA netlists dosyası oluşturulur. Şu anki derleyiciler toplama "+" ve çarpma "*" gibi yüksek seviyeli tanımlamaları optimize ederek FPGA uygulaması için derleyebiliyorlar. VHDL kodu derleyici tarafında farklı FPGA modelleri için derlenebilmektedir.

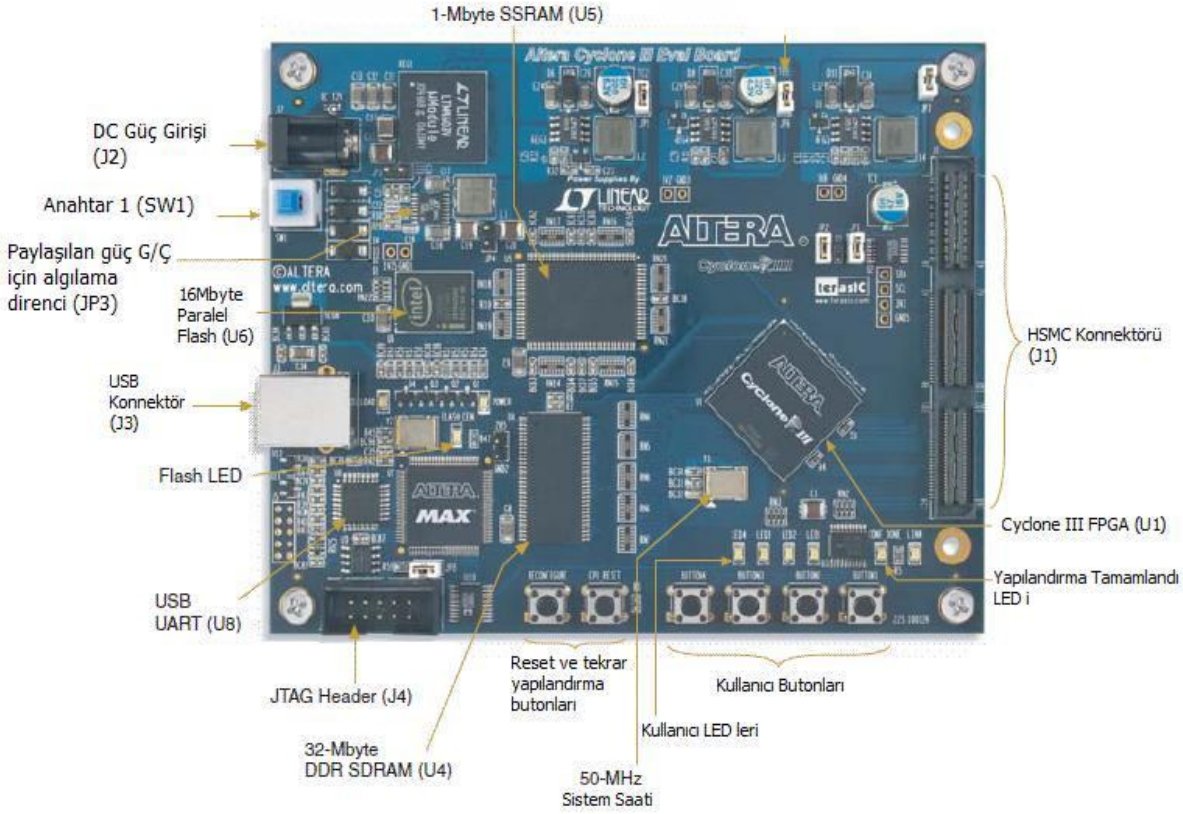
5.3 Kullanılan Donanım

Sistem Altera firmasının Cyclone III FPGA tümdevresini içeren "Cyclone III Starter Kit" kartı ve Bitec DVI I/O yardımcı kartı ile birlikte çalışacak şekilde tasarlanmıştır.

Cyclone III FPGA Starter Kit Geliştirme Kartı üzerinde Altera firmasının Cyclone III FPGA tümdevresi bulunmaktadır.

- Cyclone III starter kartı (Şekil 5.6 da gösterilmiştir.)
    - Cyclone III EP3C25F324 FPGA
    - Yapılandırma
        - Gömülü USB-Blaster™ devresi
    - Hafıza
        - 256 Mb DDR SDRAM
        - 1 MB senkron SRAM
        - 16 MB Intel P30/P33 flash

- Saat
  - 50-MHz on-board osilatör
- Anahtarlar ve göstergeler
  - Altı adet button
  - Yedi adet LED
- Konnektörler
  - HSMC
  - USB Type B



Şekil 5. 6 Cyclone III FPGA Starter Kit

HSMC konektörü yardımıyla Bitec DVI I/O yardımcı kartına bağlanarak görüntü değerlerini işlem yapacak FGPA pinlerine iletir. DVI I/O yardımcı kartı Şekil 5.2’ de gösterilmiştir.

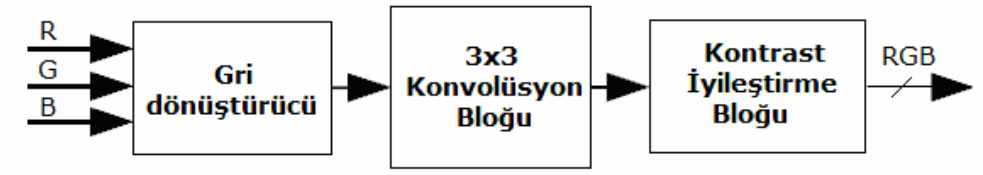
Şekil 5. 7 Bitec DVI Yardımcı Kart

6. HSA' LARIN SAYISAL GERÇEKLEMESİNE YÖNELİK İKİ BOYUTLU KONVOLÜSYON BLOĞU GERÇEKLEMESİ

Bu tezde bir görüntünün istenilen bir filtre bloğu ile konvolüsyonu gerçeklenmiştir. Hazırlanan bu tez Cyclone-III FPGA Starter Kit ve Bitech HSMC Dijital Video Daughter Card (DVI in-out) kullanılarak gerçeklenmiştir.

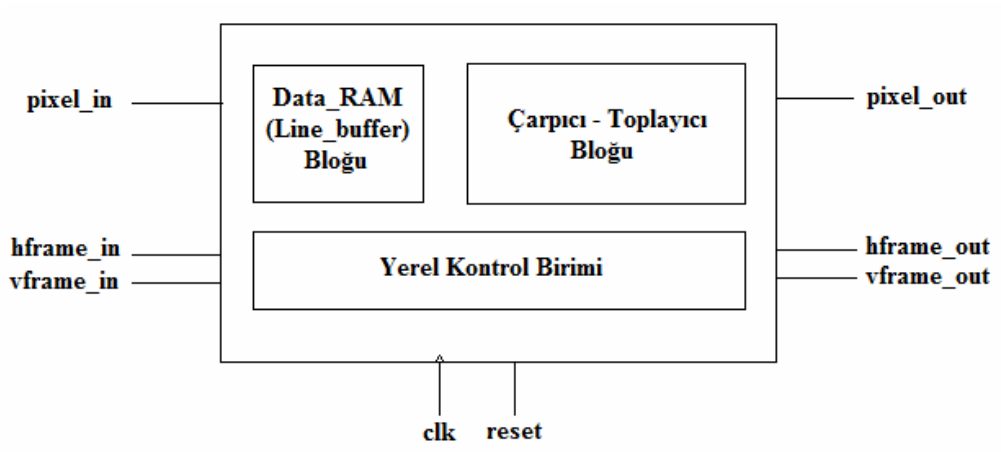
Şekil 6. 1 Sistemin genel blok diyagramı

Şekil 6.1' de gösterilen yapı içerisinde FPGA - HSA emülatörü bloğu gerçeklenmiştir. Gerçeklemede, girişten gelen görüntünün piksel değerleri, konvolüsyon işlemine girmeden önce 2 satır ve 3. satırın ilk 3 pikseli şeklinde, bir bellek elemanında saklanmıştır. Daha sonra konvolüsyon bloğu ile işleme alınmıştır. Bu bloktan elde edilen sonuçlar VGA çıkışına aktarılmıştır.



Şekil 6. 2 FPGA-HSA Emülatörü blok diyagramı

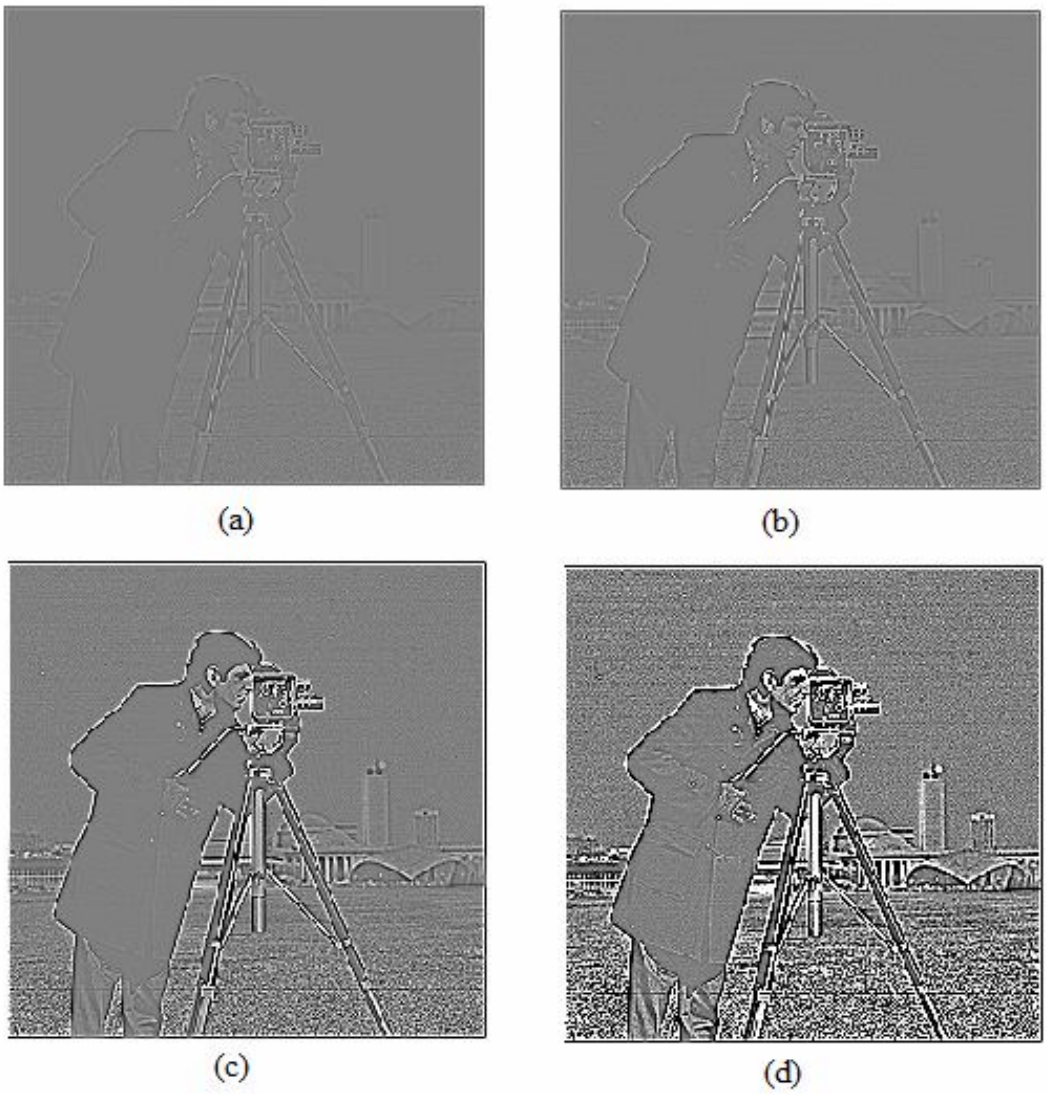
Şekil 6.2' de gösterilen blok yapısı üzerinde 3x3 konvolüsyon bloğu bu tez dahilinde gerçeklenmiştir, bu bloğu kit üzerinde çalıştırmak için giriş çıkış birimleri olarak belirtilen gri dönüştürücü ve kontrast iyileştirme bloğu 108E023 no' lu Tübitak_1001 Projesi kapsamında hazırlanan bloklardır, hazır kullanılmıştır.



Şekil 6. 3 3x3 Konvolüsyon Bloğunun Giriş/Çıkış Yapısı

Yapılan gerçeklemede girişe uygulanan siyah beyaz bir görüntü için 0-255 değerleri şu anlama gelmektedir; 0 en siyah piksel değeri, 255 ise en beyaz piksel değeridir. Çıkış görüntüsünde kontrast iyileştirme yapılmıştır. Yani görüntünün histogramı alınmış ve en küçük değer siyah-beyaz görüntüdeki 0 değerine; en büyük değer ise 255 değerine eş değer kabul edilmiştir. Bu aralıkta tekrar düzenlendikten sonra elde edilen görüntü yakın gri tonlarında göze silik görünen bir görüntüdür.

Şekil 6. 4 (a) Giriş görüntüsü (b) Gri tonlamalı görüntü (c) Laplacien ile konvolüsyon sonucunda elde edilen görüntü

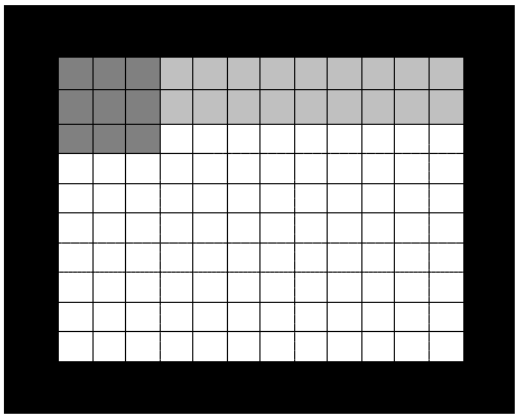


Şekil 6. 5 (a) Konvolüsyon sonucunda elde edilen görüntü, (b) x2 ile kontrast iyileştirmesi, (c) x4 ile kontrast iyileştirmesi, (d) x8 ile kontrast iyileştirmesi

Şekil 6.5(a)' da konvolüsyon sonucunda elde edilen görüntü verilmiştir. Şekil 6.2' de gösterilen blok diyagramda kontrast iyileştirme bloğunda yapılan iyileştirmeler Şekil 6.5(b), Şekil 6.5(c), Şekil 6.5(d)' de gösterilmiştir.

6.1 Piksel Değerlerinin Filtre Şablonundan Okunması

Görüntüyü oluşturan piksel değerlerinin, seçilen filtre şablonu ile işleme alınmadan önce okunabilmesi için RAM bloklarına ihtiyaç vardır. Konvolüsyon işlemini gerçekleştirmek için seçilen filtre şablonu 3x3 boyutunda bir kare şablondur. Bu şablonların işleme alınabilmesi için birbiri ardına üç satırda bulunan piksel değerlerine ihtiyaç duyar. İlk iki satırda bulunan tüm piksel değerlerinin ve üçüncü satırın ilk üç piksel değerinin bir RAM bloğunda saklanması gerekmektedir (Şekil 6.6).



Şekil 6. 6 RAM de saklanması gereken minimum piksel satırları

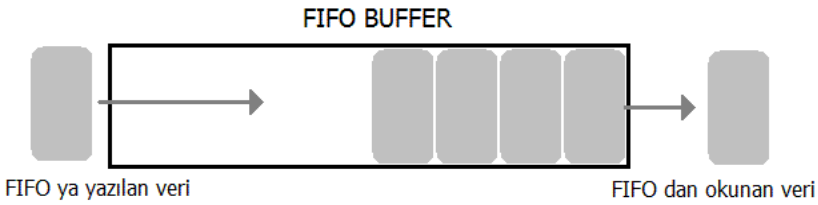
FPGA' de kullanılan RAM bloklarının büyüklüğü, giriş görüntüsünün çözünürlüğüne bağlı olarak değişir. Tez uygulamasında gerçeklenen yapı 1024x768 çözünürlükteki gerçek zamanlı gri seviyeli görüntüyü filtrelemektedir. Piksel değerlerinin 3x3' lük filtre şablonu ile işleme girmesi için iki adet 10240x9 bit büyüklüğünde RAM bloklarına ihtiyaç duyulur. Elde edilen bu RAM yapısı sayesinde 3x3 şablonun piksel değerleri elde edilip, elde edilen bu değerler seçilen filtre bloğunun girişine uygulanır.

6.2 RAM Yapıları

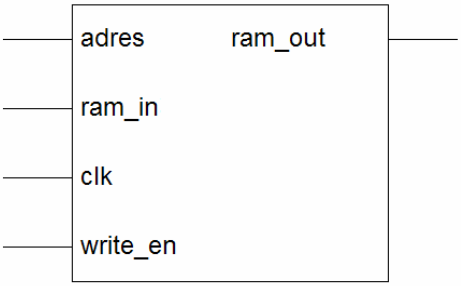
FPGA' lerde ayrı olarak tasarlanmış RAM bölgeleri yer almaktadır. Bu RAM bölgeleri Block RAM olarak adlandırılır ve tüm FPGA' lerde sayıları ve kapladıkları alan farklıdır. Bir de dağılmış (Distributed) RAM adı verilen FPGA yapısının içerisinde derleyici ve sentezleyici tarafından, yazılan VHDL koduna bağlı olarak RAM alanları da oluşturulabilir. Block RAM' den farkları FPGA içerisinde adından da anlaşılacağı üzere dağınık bir şekilde sentezlenmesidir. Tanımlanan RAM bölgeleri derleme ve sentezleme esnasında FPGA içerisinde yerleşim açısından neresi uygunsa oraya yerleştirilir. Hatta birbirlerinden dağınık bir şekilde de bulunabilir. Bu da FPGA' i ekonomik kullanmak isteyen biri için kontrolü zor bir parametredir. Uygulama bir proje olduğundan ve üzerinde çalışılan yapı da bu projenin bir parça bloğu olduğundan mümkün olduğunca yazılan kodlarda FPGA yapısını en ekonomik kullanmak amaç edinilmiştir. Bu nedenle kullanılan RAM yapıları Block RAM olarak seçilmiştir.

6.3 FIFO (First In First Out) Buffer

Fifo Buffer yapısı (Şekil 6.7' deki buffer yapısı) Block RAM' ler ile gerçeklenmiştir. Yapıda bir Ram_out çıkışı ve Ram_in, Clk, Write_en, adres girişleri yer almaktadır. Yapı, her bir saat (clk) darbesiyle darbesi ile ram_in girişinde bulunan veriyi adres giriş uçları ile bildirilen ram bölgesinin içine, bildirilen ram bölgesi içinde önceden yer alan veriyi ise Ram_out çıkışına aktarır.



Şekil 6. 7 FIFO Buffer yapısı



Şekil 6. 8 Block RAM Gösterimi

Şekil 6.8' deki yapı genel bir yapıdır. Uygulamaya göre farklılık gösterebilir. İlerleyen bölümde daha açıkça görüleceği gibi bu çalışmada görüntü pikselleri filtre bloğuna üç ayrı giriş üzerinden üç saat darbesi ile gider. Gerçeklenen FIFO Buffer bu bloğu sürebilecek yapıda tasarlanmıştır.

6.4 Yerel Kontrol Birimi

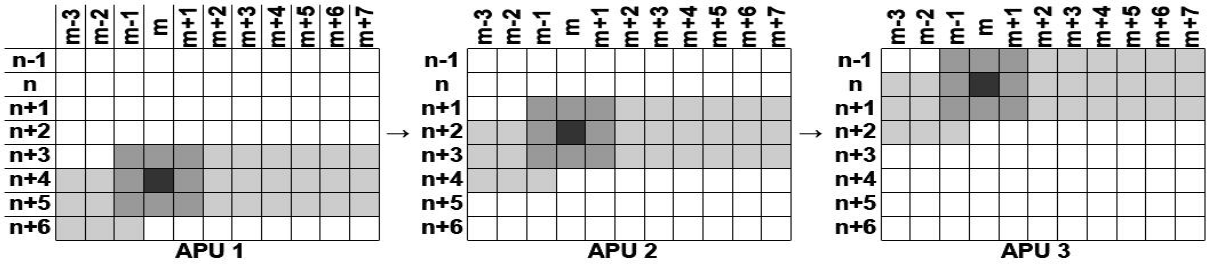
Bir işlem dizisini kontrol etmenin temel sorunu, merkezi kontrol biriminin karmaşıklığıdır. FIFO bloğunda ve konvolüsyon bloğunda kullanılan kontrol sinyalleri, adres girişleri, çoklayıcı (multiplexer) seçim uçları kontrolü bağımsız ve senkron olmalıdır. Bu senkronizasyonu sağlamak için iki adet kontrol sinyali kullanılmıştır. Yatay çerçeve sinyali Hframe ve dikey çerçeve sinyali Vframe olarak isimlendirilen iki adet kontrol sinyali bulunmaktadır. Video standartlarına göre tanımlanan gerçek video sinyalinin etrafında boş bir alan bulunmaktadır. Bu sinyal çerçevesinde 9 farklı bölge vardır. Hframe ve Vframe kontrol sinyalleri sadece görünür bölgeyi değil, aynı zamanda kontrol için ihtiyaç duyulan

çerçevenin ve satırın başlangıç ve bitiş noktalarını gösterir. Böylece sınır koşulları kontrolü sağlanır.

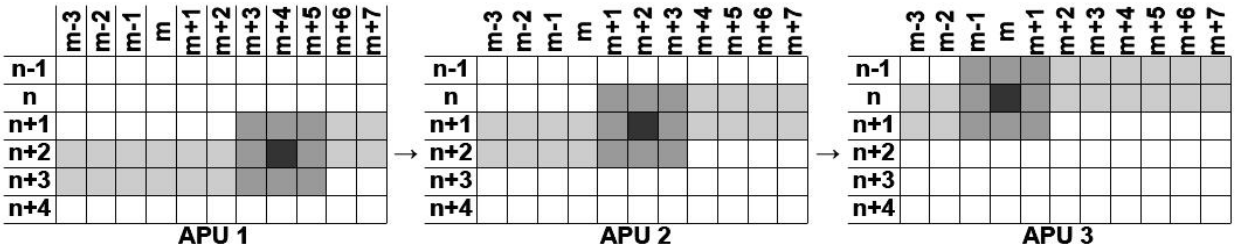
hframe = 0 vframe = 0	hframe = 1 vframe = 0	hframe = 0 vframe = 0
hframe = 0 vframe = 1	hframe = 1 vframe = 1 (Görünür Bölge)	hframe = 0 vframe = 1
hframe = 0 vframe = 0	hframe = 1 vframe = 0	hframe = 0 vframe = 0

Şekil 6. 9 Hframe ve Vframe ile kontrol edilen bölgeler (Yıldız N, vd. 2010)

Görünür bölge belirlendikten sonra bu bölge üzerinde işlem yapmak için gereken komşuluklu HSA yapısı konvolüsyon işlemine başlar. Bunu Şekil 6.10 ve 6.11' de verilen 1-komşuluklu APU yapısı ile gösterebiliriz. 1-komşuluklu HSA' da 3 ardışık APU üzerinde, iyileştirilmiş ve iyileştirilmemiş veri RAM yapıları değerlendirildiğinde elde edilen sonuçlar aşağıdaki şekilde net olarak görülmektedir.



Şekil 6. 10 İyileştirilmemiş APU yapısı (Yıldız N, vd. 2010)

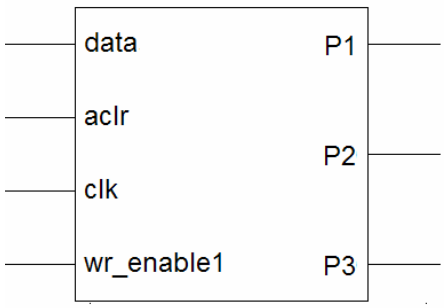


Şekil 6. 11 İyileştirilmiş APU yapısı (Yıldız N, vd. 2010)

Yerel kontrol birimi, sistemin reset ve enable sinyallerine bağlanır böylece, devam eden işlem sırasında, giriş görüntüsü RAM de sıralanırken konvolüsyon bloğu çıkışından elde edilen değerin senkron olması sağlanır.

6.5 Konvolüsyon Bloğu İçin Gereken Piksel Değerlerinin Şablondan Okunması

Görüntüyü oluşturan piksel değerleri, tasarlanan bloğun girişi üzerinden seri bir şekilde bloğun içine aktarılır. Bloğun girişine gelen bu veriler, 3x3 kare filtre edilecek şablonunun değerleri haline gelebilmesi için blok içerisinde Block RAM bölgelerine ihtiyaç duyulur. Tasarlanan yapı içerisinde her bir satır için, piksel değerlerini içerisinde barındıran Block RAM bölgelerinin yer alması gerekir. Görüntünün birinci satırı için bir, ikinci satırı için ise ikinci bir Block RAM yapısı kullanılmıştır.

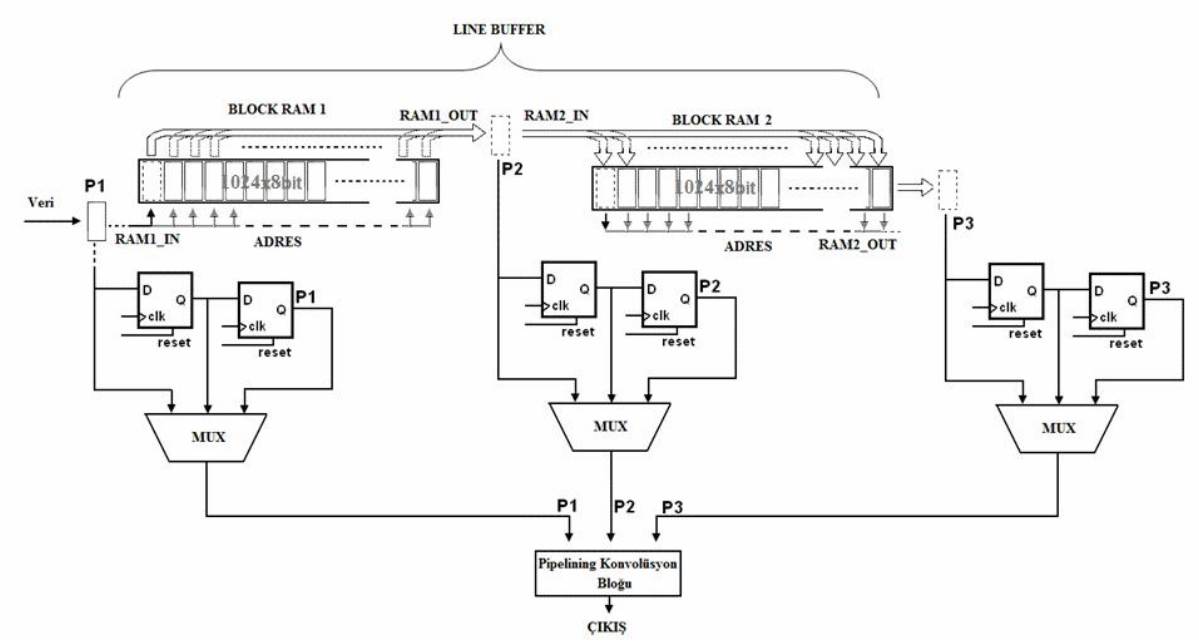


Şekil 6. 12 FIFO (Line) Buffer Blok Yapısı

Bloğun çıkışları P1, P2 ve P3, oluşturulan konvolüsyon bloğu girişlerine göre ayarlanmıştır. P1, P2 ve P3 çıkışları, giriş görüntüsünü filtrelemede kullanılacak olan 3x3 filtre şablonunun piksel değerleri, oluşturulan konvolüsyon bloğunun girişlerine uygulanır.

Şekil 6.13' teki yapının çalışma prensibi kısaca şu şekildedir. Read First modunda çalışan RAM yapısında, veri girişine her bir saat darbesiyle gelen görüntünün piksel büyüklükleri RAM1_IN sinyaline atanır. Block RAM-1' in adres yazmacının göstermiş olduğu RAM bölgesinde bulunan piksel değeri saat darbesiyle RAM1_OUT sinyaline atanarak ilgili RAM bölgesi yeni veri yazılması için boşaltılmış olur. Böylelikle RAM1_IN sinyaline gelen yeni piksel değeri o adres yazmacı ile gösterilen RAM bölgesine yazılır.

RAM1_OUT sinyaline atanan bir önceki piksel değeri de Block RAM-2 de RAM2_IN sinyaline iletilmiş olur. Sonraki saat darbesiyle beraber Block RAM2' nin adres yazmacının göstermiş olduğu yerde bulunan piksel büyüklüğü de RAM2_OUT sinyaline atanarak ilgili ram bölgesindeki veriler boşaltılır, takip eden saat darbesiyle de RAM2_IN sinyalindeki veri ilgili ram bölgesine yazılmış olur. Bu işlem her bir saat darbesiyle adres yazmaçları bir arttırılarak devam ettirilir.



Şekil 6. 13 FIFO BUFFER İç Yapısı

Kontrol devreleriyle RAM bölgelerinin adres yazmaçları ram blokların girişlerine sürekli olarak gelen verileri ilgili adreslerdeki ram bölgelerine yazar, çıkışları da P1, P2 ve P3 uçlarına atayarak bloğun düzenli şekilde çalışmasını sağlar. Örneğin, görüntünün ikinci satırında bulunan 9. piksel değeri Block RAM-1 bölgesinin 9. bölgesine yazılırken, o bölgede bulunan birinci satırın 9. piksel değeri de Block RAM-2 bölgesinin 9. bölgesine yazılır. Bu bölgede bulunan bir önceki satıra ait piksel büyüklüğü de P3 çıkışına aktarılır. Böylelikle amaçlanan, görüntünün satır genişliği büyüklüğünde iki adet RAM bölgesini birbiri ardına bağlayarak, saat darbesiyle çalışan bir shift register (kayan yazmaç) elde etmektir.

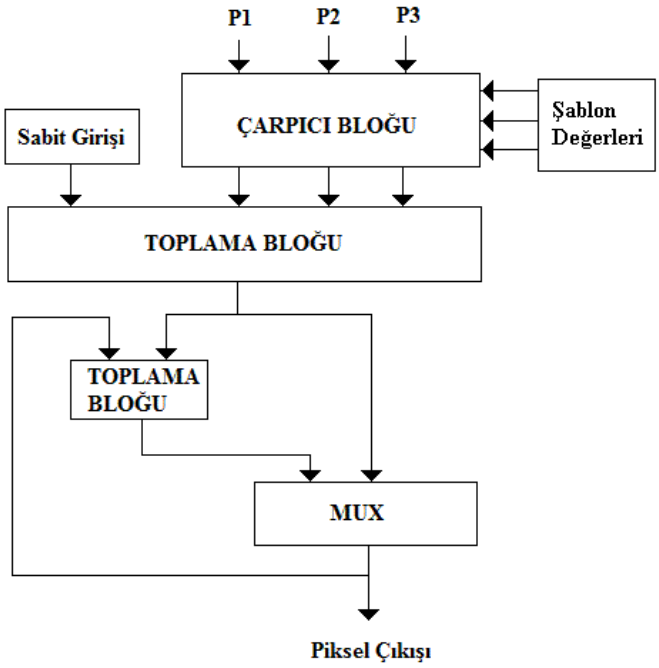
6.6 Konvolüsyon Bloğu

Bu tezde üzerinde çalışılan özgün yapı konvolüsyon bloğudur. Konvolüsyon bloğuna gelecek görüntü pikselleri, önce bir buffer yardımı ile belli bir süre tutulur. Bu sayede, girişe gelen piksel değerleri her saat darbesinde işlem bloğuna iletilir.

FIFO buffer üzerinden alınan veriler çarpıcı bloğuna alınır. HSA uyumlu çalışabilmesi için bir sabit değer girişi yer almaktadır.

FIFO çıkışından alınan P1, P2, P3 girişleri çarpıcı bloğunda işlem gördükten sonra ile toplama bloğuna alınırlar. Bu blokta işlem gören ilk piksel değeri sabit değer girişi ile toplanarak MUX üzerinden çıkışa alınır. İkinci saat darbesinde toplama bloğundan gelen piksel değeri ile işleme alınmış ilk piksel değeri ikinci toplama bloğunda işleme alınır. MUX' un diğer seçim ucuna giren ikinci toplama bloğu çıkışı, üçüncü saat darbesinde MUX' un

çıkışına alınır. Üçüncü saat darbesinde toplama bloğundan gelen piksel değeri ikinci toplama bloğuna alınır ve MUX' un çıkışında yer alan ikinci toplam değeri ile toplanır. 3 saat darbesi sonunda 3x3 boyutundaki bir filtre için ilk işlem sonucu elde edilmiş olur. Aynı çevrim tüm piksel değerleri bitene kadar tüm saat darbesi ile devam eder.



Şekil 6. 14 Konvolüsyon Bloğu Blok Gösterimi

Tezde önerilen devrenin giriş ve çıkışları Şekil 6.13' te gösterilmiştir. *Z(i,j)* (sabit girişi), eşik değerlerinin iletildiği uçtur. P1, P2 ve P3 adı verilen uçlardan görüntü pikselleri tek tek gönderilmektedir. Şablon bloğu olarak belirtilen uçtan şablon değer bilgisi gönderilecektir. Çıkış ucundan işlenen görüntüler alınabilecektir.

Yukarıda bölümlerde anlatıldığı gibi her bir hücresel işlemciyle giriş çıkış işlemleri yapan bir RAM vardır. Bu RAM piksel ve h*z(i,j) değerleri ve g(i,j) değerleri saklanacaktır. İşlemler iterasyon sayısı kadar tekrarlanır. Yapılan iterasyonlarda RAM içerikleri güncellenmektedir.

Genel Kontrol, RAM birimlerine adres, yazma aktif ve yonga aktif işaretlerini gönderir. Bu işlerin aynısı komşu RAM blokları için de gereklidir. Ayrıca diğer bütün birimlere kontrol işaretleri göndermektedir. Hücresel işlemciler içindeki kontrol işlemini azaltmak için, bunlara sadece basit kontrol işaretleri gönderilmiştir. Dolayısıyla karmaşık hesapları yapan blok Genel Kontrol olmuştur. Bunun getirdiği faydalar vardır: Hücresel işlemcilerin sayısı birden fazla olduğu için, bunların içindeki bütün bloklar hücresel işlemci sayısınca gerçeklenecektir, eğer işlemci içindeki kontrol birimi karmaşık veya büyük olursa bundan işlemci sayısınca gerçeklenince otomatik olarak devre boyutu büyümüş olacaktır.

Bir konvolüsyon işleminin üç saat çevriminde olduğunu daha önce belirtmiştik. Bu üç çevrimden ilkinde, dokuz tane hesaplanması gereken şablon-durum nokta çarpımının ilk üçü yapılmaktadır. İlk çevrimde ek toplam olarak MUX' un seçim ucu ile MUX' un çıkış değeri seçilir. Çarpıcılara gelen değerler şablon değerleri ile işleme alınır. İkinci ve üçüncü çevrimde ise MUX un çıkışında yer alan kısmi toplam değerleri toplama dâhil edilir. Böylece şablon ve durumun nokta çarpım işlemi gerçeklenmiş olur. Bu üç çevrim bittiğinde 3x3 boyutunda 9 piksel değerinin konvolüsyon sonucu elde edilmiş olur. RAM' den ilk ve son okuma dışındaki her veri okumada ötelemeli kaydetme işlemi yapılır. Dolayısıyla veriler kaydediciler içinde birer ötelenmiş olur. İterasyondaki üçüncü çevrimden sonraki okuma işlemi ile yeni veri yüklemesiyle bir sonraki konvolüsyonun ilk çevrimini oluşturmaktadır. Böylece konvolüsyon işlemi resme uygulanmış olur.

Aritmetik Birim içindeki ÇARPICI BLOĞU birimleri şablon ve durum çarpımını gerçekleştirmektedir. TOPLAMA BLOĞU birimleri ise toplayıcılardır. Fixed point (sabit nokta) olarak tanımlanan (8.0) 8 bit piksel değeri ile (8.8) 16 bit işaretli şablon değerleri çarpıcı bloğunda işleme girer ve sonunda 24 (16.8) bit değerine yükselir. Toplayıcılar piksel bit genişliğinden daha büyük bit genişliğine sahip olmak zorundadır. Çarpma işleminde çarpılan sayıların aynı bit genişliğinden olması zorunluluğu yoktur. Toplama bloğu ve çarpma bloğundan çıkan sonuçlar taşma olmaması için 28 bite genişletilir. Konvolüsyon bloğu sonunda elde edilen çıkış değeri sabit nokta aritmetiğine uygun olarak (8.0) 8 bite daraltılır. Bu bitlerde oluşan hassasiyet kaybını gidermek amacıyla yuvarlama yapılmıştır.

Pipeline ile ilgili önemli olan bir husus şudur: yeni veri okuma şekliyle her bir işlemci bloğu içindeki kaydedici dizisi boyunca yatayda ilerlenmektedir ve yataydaki kaydedici uzunluğu pipeline seviyesinden (aralara yerleştirilen kaydedici sayısından) az olursa ilk sonuç henüz çıkmadan dizi tamamen dolaşılmış olacaktır. Bu yüzden ilgili tasarımdaki işlemci blokları içindeki kaydedici yatay uzunluğu belirli bir sayının altına inemez. Yani yapılan tasarımdaki yatayda sıralanan işlemci blok sayısı resmin yataydaki piksel sayısının belirli bir oranından daha fazla olamaz. Böylece, sistemin kendi tasarımından dolayı oluşan sebeplerle eğer pipeline yapılmak isteniyorsa yatayda sıralanabilecek işlemci sayısı kısıtlanmış olmaktadır.

7. SONUÇ

Görüntü işleme uygulamaları HSA gibi görüntü pikselleri için bağımsızdır. Görüntüdeki her piksel HSA ağındaki bir düğüme eşlenebilir ve paralel analog hesaplama sayısal işlemcilere göre büyük bir hız üstünlüğü sağlar. Bu tez çalışması kapmasında HSA yapısına yönelik iki boyutlu bir konvolüsyon bloğu gerçeklemesi yapılmıştır. Bu konvolüsyon bloğu üzerinde yapılacak çeşitli değişiklikler sayesinde herhangi bir konvolüsyon işlemi için kullanılabilir olarak tasarlanmıştır. Bu sayede herhangi bir görüntü için şablon değerleri değiştirilerek, 3x3 boyutunda filtreleme işlemleri yapılabilmektedir.

Bir pikselin çıkış değerinin hesaplanması üç saat darbesi zaman alır ve bu değerin hesaplanması için yapılacak Euler iterasyonu sayısı gerçeklenen işlem birimlerinin sayısına bağlıdır. Simülasyonlar yapıldığında, elde edilen sonuçlar simülasyonun başarılı olduğunu göstermiştir.

Hazırlanan bu tez Cyclone-III FPGA Starter Kit ve Bitech HSMC Dijital Video Daughter Card (DVI in-out) kullanılarak gerçeklenmiştir. Tasarım ortamı olarak Xilinx firmasının ISE yazılımı kullanılmıştır. Gerçeklenen emülatör yapısı FPGA dışında bir bellek elemanı kullanmamakta ve video görüntülerini gerçek zamanlı olarak işlemektedir. Sistem 1024x768 piksel görüntü ile kenar belirleme uygulaması testini gerçekleştirmek için şablon değerleri $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ şeklinde seçilmiştir.

Giriş piksel verileri (8 bit) [-256 (siyah), 255 (beyaz)] arasında değer almaktadır gri tonlamalı piksel verilerine [-128, 127] dönüştürüldükten sonra işleme alınmıştır.

XGA (1024x768) 60Hz standardının piksel frekansı 65MHz'dir. Video çerçevesinin her bir pikseli için yapılacak hesaplama üç saat darbesi gerektirdiğinden, emülatör saniyede 21.7 Mega piksel (65/3) işleyebilmektedir. Emülatör en yüksek hızda çalıştırıldığında (65 MHz) 28fps (21.7Mega/786432) 1024×768 piksel video görüntüsünü 3x3 HSA şablonları ve ((66/3)-1) 21 Euler iterasyonu ile işleyebilir. Burada piksel değerleri 8 bit (gri tonlamalı), şablon değerleri ise 16 bit ile ifade edilmektedir.

Yapılan çalışmanın bazı temel özellikleri:

- FPGA dışında bellek elemanı (RAM) kullanılmaz.
- Her bir çıkış piksel değeri üç saat darbesinde hesaplanır.

- Çıkışın hesaplanması için yapılan Euler iterasyonu sayısı gerçeklenen işlem birimi sayısı ile belirlenir (işlem birimi sayısı – 1).
- Bu çalışma, gömülü sistem üzerinde gerçeklendiği için; televizyon, video kaydedici, dijital fotoğraf makinesi gibi ürünlerde doğrudan (FPGA kit üzerinden) kullanılabileceği gibi, yapılan tasarım ile ASIC halde üretilip de kullanılabilir.

İleride yapılacak çalışmalarda şunlar önerilmektedir:

- Bloğun çıkışında ayrı bir işlem birimi ile eşik fonksiyonu eklenebilir.
- Birçok blok ardı ardına dizilerek HSA' nın her bir Euler iterasyonu, ayrı birer blok ile gerçeklenebilir. Tam bir HSA yapısı oluşturulabilmesi için APU birimini gerçeklemek için sistemin çıkışına bir saturasyon bloğu eklenmelidir.
- Gelecek çalışmalar için MxN boyutlu şablon değerlerine göre doğru sonuçları verecek bir konvolüsyon bloğu yapısı gerçeklenebilir.

KAYNAKLAR

Arato p., Visegrady T., Jankovits I. ve Szigeti Sz., (2000), "High-Level Aynthesis of Pipelined Datapaths", Edited by P. Arato, Panem Budapest.

Arena, P., Basile, A., Bucolo, M., ve Fortuna, L., (2003), "An Object Oriented Segmentation on Analog CNN Chip", IEEE Transactions On Circuits And Systems-I: Fundamental Theory And Applications,50: 837-846.

Beke, L., Nagy, Z. ve Szolgay, P., (2004), "Low-cost CNN-UM Global Analogic Programming Unit implementation on FPGA", 8th IEEE International Workshop on Cellular Neural Networks and Their Applications (CNNA2004), 22-24 July 2004, Budapest/Hungary.

Chua L.O. ve Yang L., (1988): "Cellular Neural Networks: Applications", IEEE Transaction on Circuits and Systems,35:1273-1290.

Çelebi E. ve Güzeli G.C., (1995): "Hücresel Yapay Sinir Ağları ile Görüntü Onarımı", Sinyal İşleme ve Uygulamaları Kurultayı, Cilt-A:37–41.

Doan, M.D., Glesner, M., Chakrabaty, R., Heidenreich, M. ve Cheung, S., (1994), "Realisation of digital Cellular Neural Network for image processing", 3rd. IEEE International Workshop on Cellular Neural Networks and Their Applications Proceedings, (CNNA' 94): 85-90.

Dominguez-Castro, R., Espejo, S., Rodriguez-Vazquez, A. ve Carmona, R., (1994), "A CNN Universal Chip in CMOS Technology", Third IEEE International Workshop on Cellular Neural Networks and Their Application (CNNA' 94), 18-21 Dec. 1994, Rome / Italy.

Espejo, S., Rodriguez-Vazquez, A., Dominguez-Castro, R. ve Carmona, R., (1994), "Convergence and Stability of the FSR CNN Model", Third IEEE International Workshop on Cellular Neural Networks and Their Application (CNNA' 94), 18-21 Dec. 1994, Rome / Italy.

Hopfield, J.J., (1982), Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences USA.

Hidvégi, T., Keresztes, P. ve Szolgay, P., (2002), "An accelerated digital CNN-UM (CASTLE) Architecture by using the Pipe-Line Technique", 7th IEEE International Workshopon Cellular Neural Networks and Their Applications (CNNA' 2002), 22-24 July 2002, Frankfurt / Germany.

Hidvégi, T., Keresztes, P. ve Szolgay, P., (2003), "Enhanced Modified Analized Emulated Digital CNN-UM (CASTLE) Arithmetic Cores", Journal of Circuits, Systems, and Computers, special issue on "CNN Technology and Visual Microprocessors", 12 (5): 711-738.

Ikenaga, T. ve Ogura, T., (1996), Discrete-time Cellular Neural Networks Using Highly-Parallel 2D Cellular Automata CAM2 Proc. of Int. Symp. on Nonlinear Theory and Its Applications: 221-224.

K. Kayaer ve V. Tavşanoğlu, (Temmuz, 2008), "Gerçek Zamanlı Video İşleyen Yeni bir Hücresel Sinir Ağı Emülatörü Tasarımı ve FPGA Gerçeklemesi", 11. Uluslararası Hücresel Sinir ağları ve Uygulamaları Çalıştayı, Santiago de Compostela, Spain.

Keresztes, P., Zarándy, Á., Roska, T., Szolgay, P., Bezák, T., Hidvégi, T., Jónás, P. ve Katona, A., (1999), "An emulated digital CNN implementation, Journal of VLSI Signal Processing Systems", Kluwer Academic Publishers, 23: 291-303.

Nagy, Z. ve Szolgay, P., (2002), "Configurable Multi-Layer CNN-UM Emulator on FPGA", 7th IEEE International Workshop on Cellular Neural Networks and Their Applications (CNNA' 2002), 22-24, Frankfurt / Germany

Nagy, Z. ve Szolgay, P., (2003), Configurable multi-layer CNN-UM emulator on FPGA, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 50: 774–778.

Örs, S.B., (1998), "Sahada Programlanabilir Kapı Dizileri ile Lojik Devre Tasarımı ve VHDL Kullanılarak Bazı Devrelerin Gerçekleştirilmesi", Yüksek Lisans Tezi, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

Roska, T. ve Chua, L. O., (1993), The CNN Universal Machine: an Analogic Array Computer, IEEE Transactions on Circuits and Systems-II, 40: 163-173.

Roska T. ve Chua L. O.,(1993), The CNN paradigm, IEEE Trans. Circuits and Systems- I, 40: 147–156.

Roska T. ve Chua L.O., (2002), "Cellular Neural Networks and Visual Computing", Cambridge University Press, UK.

Saatçi, E. (2003), "Hücresel Sinir Ağları Kullanarak Görüntü İşleme", Doktora Tezi, London South Bank University.

Wen, K.A., Su, J.Y. ve Lu ,C.Y., (1994), "VLSI design of digital Cellular Neural Networks for image processing", J. of Visual Communication and Image Representation, 5, No:2: 117-126.

Yıldız N., Cesur E. ve Tavsanoğlu V., (2010) "A New Control Structure For The Pipelined CNN Processor Arrays", İstanbul, Yildiz Technical University.

İnternet Kaynakları

[1] www.digilentinc.com

[2] www.model.com

[3] www.vhdl-online.de/tutorial/

[4] www.xilinx.com

ÖZGEÇMİŞ

Doğum Tarihi	04.07.1984	
Doğum Yeri	Merzifon	
Lise	1998-2002	Antalya Anadolu Lisesi
Lisans	2002-2006	Sakarya Üniversitesi Mühendislik Fakültesi Elektrik-Elektronik Mühendisliği
Yüksek Lisans	2007-	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Anabilim Dalı, Elektronik Mühendisliği

Çalıştığı Kurumlar

2007-2008	MEDEL Elektronik Elektrik San. Ltd. Şti.
2008-2009	HTgrup AŞ. - Mikrosay Elektronik
2009-2010	Na-De Elektronik Ltd. Şti.