YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GERÇEK ZAMANLI VİDEO İŞLEYEN YENİ BİR HÜCRESEL SİNİR AĞI EMÜLATÖRÜ TASARIMI VE FPGA İLE GERÇEKLENMESİ

Elektronik Yük. Müh. Kamer KAYAER

FBE Elektronik ve Haberleşme Müh. Anabilim Dalı
Elektronik Programında Hazırlanan

DOKTORA TEZİ

Tez Savunma Tarihi : 06 Kasım 2008
Tez Danışmanı : Prof. Dr. Vedat TAVŞANOĞLU (YTÜ)
Jüri Üyeleri : Prof. Dr. Fikret GÜRGEN (BÜ)
: Prof. Dr. Tülay YILDIRIM (YTÜ)
: Doç. Dr. Sabri ARIK (İÜ)
: Doç. Dr. Müştak YALÇIN (İTÜ)

İSTANBUL, 2008

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

A	Geri besleme klonlama şablonu.
$A(i,j;k,l)$	i. satır ve j. sütundaki hücreye giren, k. satır ve l. sütundaki hücre çıkışının ağırlık değeri.
B	İleri besleme (veya giriş) klonlama 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_{ij}	$C(i,j)$ hücresinin durum değişkeni.
y_{ij}	$C(i,j)$ hücresinin çıkış değişkeni.
I	Hücrelerin eşik (bias) değeri.

KISALTMA LİSTESİ

ACE	Analogic CNN Emulator Engine (Analojik HSA Emülatörü Makinesi)
ADC	Analog-to-Digital Converter (Analogdan Dijital Çevirici)
APU	**A** template Processor Unit (**A** şablonu işlem birimi)
AZ-HSA	Ayrık Zamanlı HSA (Discrete Time CNN (DT-CNN))
bps	bit per second (saniye başına bit)
BPU	**B** template Processor Unit (**B** şablonu işlem birimi)
BRAM	Blok RAM
CLB	Configurable Logic Block (Programlanabilir Lojik Blok)
CNN	Cellular Neural Networks (Hücresel Sinir Ağları)
CNN-UM	CNN-Universal Machine (HSA Çok Kapsamlı Makinesi)
DAC	Digital-to-Analog Converter (Dijitalden Analoğa Çevirici)
DiROM	Distribute ROM (Dağıtılmış Yalnızca Okunabilen Bellek)
DSP	Digital Signal Processor (Dijital Sinyal İşlemcisi)
DT-CNN	Discrete Time CNN (Ayrık Zamanlı HSA (AZ-HSA))
EDIF	Electronic Design Interchange Format
FPGA	Field Programmable Gate Array (Alanda Programlanabilen Kapı Dizisi)
fps	frames per second (saniyedeki çerçeve sayısı)
FSR	Full Signal Range (Tüm Sinyal Aralığı)
G\Ç	Giriş\Çıkış
HDL	Hardware Description Language (Donanım Tanımlama Dili)
HMD	Head Mounted Display (Başa Takılı Ekran)
HSA	Hücresel Sinir Ağları
HSA-ÇKM	HSA Çok Kapsamlı Makinesi
I^2C	Inter-Integrated Circuit
IOB	Input/Output Block (Giriş/Çıkış Bloğu)
IP	Intellectual Property
JTAG	Joint Test Action Group
Kbit	Kilo bit
LUT	Look-Up Table (Veri Tablosu)
Mux	Multiplexer (Çoğullayıcı)
Mbps	Mega bit per second (saniye başına Mega bit)
NGD	Native Generic Database
ONR	Office of Naval Research (Bahri Araştırma Bürosu)

PROM	Programmable Read-Only Memory (Programlanabilen Yalnızca Okunabilir Bellek)
PU	Processing Unit (İşlem Birimi)
RAM	Random Access Memory (Rastgele Erişimli Bellek)
ROM	Read-Only Memory (Yalnızca Okunabilen Bellek)
RTCNNP	Real Time CNN Processor (Gerçek Zamanlı HSA İşlemcisi)
SDRAM	Synchronous Dynamic RAM (Eş Zamanlı Dinamik Rastgele Erişimli Bellek)
SRAM	Static RAM (Statik Rastgele Erişimli Bellek)
TFT	Thin Film Transistor (İnce Film Transistör)
UCF	User Constraints File (Kullanıcı Kısıtları Dosyası)
VGA	Video Graphics Array (Video Grafikleri Dizisi)
VHDL	VHSIC (Very High Speed Integrated Circuits) Hardware Description Language
VLSI	Very Large Scale Integration (Çok Büyük Ölçekli Tümleştirme)

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖNSÖZ

Bu tez çalışmasının gerçekleşmesini sağlayan değerli hocam Prof. Dr. Vedat Tavşanoğlu ve her yardıma ihtiyaç duyduğumda desteklerini benden esirgemeyen araştırma görevlisi arkadaşlarım Nerhun Yıldız, Evren Cesur ve Murathan Alpay'a teşekkür ederim. Ayrıca, tüm eğitim hayatım boyunca maddi, manevi destekleriyle daima yanımda olan annem ve babama müteşekkirim.

ÖZET

Hücresel Sinir Ağları (HSA) (Cellular Neural Networks (CNN)) kavramı Prof. Leon O. Chua ve L. Yang tarafından Kaliforniya Berkeley Üniversitesinde Aralık 1988 - Kasım 1997 arasında yapılan "Nonlinear Circuits and Neural Networks" adlı bir ONR (Office of Naval Research) projesi sırasında geliştirilmiş ve 1988 yılında yayımlanmıştır. Daha sonraları Macar Bilim Akademisinden Prof. Tamás Roska ile birlikte yine Berkeley'de yapılan bir çalışmada Prof. Chua ve Prof Roska HSA Çok Kapsamlı Makinesi (HSA-ÇKM) (CNN Universal Machine) yapısını geliştirmişlerdir. Tasarlanan HSA-ÇKM yapısı önce küçük ölçekli (64×64), sonra daha büyük ölçekli (128×128) analog tümdevreler üzerinde gerçeklenmiştir. Bu tümdevrelerin işlem hızları çok yüksek olmasına rağmen, eşdeğer dijital doğruluklarının 7-8 bit olması, maliyetlerinin çok yüksek olması, şu ana dek gerçekleştirilebilen analog işlemci sayısı en yüksek tümdevrenin 128×128 hücreli (işlemcili) ve halen üzerinde çalışılmakta olan 256×256 analog hücreli HSA tümdevresinin bitirilememiş olması, araştırmacıları HSA yapısının dijital emülasyonları üzerinde çalışmaya yöneltmiştir.

Bu tez çalışmasında, HSA yapısının FPGA tümdevresiyle dijital emülasyonu için kullanılabilecek yeni bir emülatör yapısı tasarlanmış, gerçeklenmiş ve bir kenar belirleme uygulamasıyla test edilmiştir. Bu yapı, basit taramalı (progressive) video işaretlerini gerçek zamanlı olarak işleyebilecek kadar hızlıdır ve bir VGA video işaretini alarak gerçek zamanlı olarak işleyip sonucu bir VGA monitöründe gösterebilir. Ayrıca, sistem maliyetlerinin düşürülmesi amacıyla yapı, FPGA dışında bir bellek elemanına ihtiyaç duymayacak şekilde tasarlanmıştır. Bu yeni yapı dijital görüntü işleme uygulamaları için hızlı ve genel bir çözüm oluşturmaktadır.

Bu tez çalışması, küçük hedeflere adım adım ulaşarak sonuç hedefe varma yaklaşımıyla ilerletilmiştir. Bu yaklaşım doğrultusunda, öncelikle bir HSA emülatörü işlem birimi yapısı, çalışma hızı önemsenmeden oluşturulmuştur. Sonra bu işlem birimi yapısı, ardışık düzenle (pipeline) çalışması sağlanarak hızlandırılmış ve sistemdeki işlem birimlerinin çizgisel bir dizilimle (kaskad) bağlanarak birden fazla işlem biriminin eş zamanlı olarak çalışması sağlanmıştır.

Tasarım ortamı olarak Xilinx firmasının ISE yazımı ve bu yazılımın şematik editörü kullanılmıştır. Tasarım öncelikle şematik editörle bilgisayar ortamına aktarılmış, daha sonra ISE yazılımı ile simülasyonu yapılarak tasarımdaki hatalar bulunmuş ve düzeltilmiştir. Tasarım, simülasyon programında çalışması sağlandıktan sonra FPGA kartı üzerine yüklenerek denenmiştir. Deneme işleminde, Xilinx firmasının ChipScope Pro yazılımı kullanılarak FPGA'de oluşturulmuş olan lojik analizör bloğu ile FPGA üzerindeki sinyaller çalışma esnasında bilgisayar ekranından gözlenmiş ve hata analizi yapılmıştır. Gözlenen hataların oluşma nedeni saptanarak tasarımda düzeltmelere gidilmiş ve sistemin hatasız çalışması sağlanmıştır.

Sonuç olarak, basit taramalı video işretlerini 3×3 HSA şablonları ile dış bellek kullanmadan gerçek zamanlı olarak işleyen yeni bir HSA emülatörü yapısı önerilmiş ve Celoxica RC203 kartında bulunan Xilinx Virtex-II 3000 FPGA tümdevresi üzerinde gerçeklenmiştir. Gerçeklenen sistem 640×480 piksel 60 fps monokrom VGA video girişi ile test edilmiş ve çıkış videosu bir VGA monitöründe gözlenmiştir.

Gerçeklenen emülatör yapısı CPU Turkey 2008 yarışmasının Akademik Yenilikçi Gömülü Sistem Tasarımı kategorisinde birincilik ödülü almıştır.

ABSTRACT

DESIGN AND IMPLEMENTATION OF A NEW CELLULAR NEURAL NETWORK EMULATOR ON FPGA FOR REAL TIME VIDEO PROCESSING

The Cellular Neural Network (CNN) concept was first introduced by Prof. Leon O. Chua and L. Yang at the University of California at Berkeley in 1988 during the course of an ONR (Office of Naval Research) project (N0001489J1402) entitled "Nonlinear Circuits and Neural Networks" which ran from December 1988 through November 1997. Later, as a result of the work carried out in collaboration with Prof. Tamas Roska of the Hungarian Academy of Sciences, Budapest, Hungary, during 1992-1993, Profs. Chua and Roska developed the CNN Universal Machine architecture at Berkeley. This architecture was first implemented on smaller scale (64×64) and later on larger scale (128×128) analog chips. However the development of analog chips had been rather slow and has some drawbacks. The fact that the equivalent bit accuracy of these chips is only 7-8 bits although their operation speed is quite high, and that the chip with the highest number of processors fabricated so far has only 128×128 cells and lastly the fact that the works on the 256×256 chip have not so far been completed, have led the researchers to start working on the digital emulations of CNN.

In this thesis a new FPGA architecture for the emulation of the CNN structure is designed, implemented and tested with an edge detection application. This architecture is fast enough for real-time processing of video signals and is capable of taking a VGA signal, processing it in real-time and displaying the output on a VGA monitor. Furthermore, in order to lower the system cost, the design has been carried in such a way that the system does not require an external memory (RAM). The new structure provides a fast general purpose solution for digital image processing applications.

In this thesis a step-by-step approach is using to achieve the ultimate goal. In doing so, first a CNN emulating processor structure is developed without paying any attention to speed. Then this structure is modified in order to make it function in a pipelined manner to increase speed and the processors have been laid out in the form of a 1-D array which enables the simultaneous functioning of the processing units.

As the design environment, Xilinx ISE software and the schematic editor of the software have been used. At first, the design has been transferred to the PC environment by the use of the schematic editor of ISE software, and then the design errors have been found by simulation and corrected. Having seen the simulation in working order, the design has been implemented on the FPGA board and tested. In the test process, a logic analyzer core on the FPGA, which has been generated by the Xilinx ChipScope Pro software, has been used to find out the erroneous part of the design by observing the signals of the working circuit on FPGA, on the PC monitor and the design is then corrected accordingly.

As a result, a new CNN emulator architecture for real time progressive video processing with 3×3 CNN templates is proposed and implemented on a Xilinx Virtex-II 3000 FPGA in Celoxica RC203 board, which does not require external memory. Implemented system is tested with 640×480 pixels 60 fps monochrome VGA input and the output video is observed on a VGA monitor.

The emulator architecture implemented on FPGA has been awarded the First Prize in the Academic Innovative Embedded System Design category of CPU Turkey 2008 contest.

1. GİRİŞ

Hücresel Sinir Ağları (HSA) (Cellular Neural Networks (CNN)) kavramı Prof. Leon O. Chua ve L. Yang tarafından Kaliforniya Berkeley Üniversitesinde Aralık 1988 - Kasım 1997 arasında yapılan "Nonlinear Circuits and Neural Networks" adlı bir ONR (Office of Naval Research) projesi sırasında geliştirilmiş ve 1988 yılında yayımlanmıştır (Chua ve Yang, 1988a; 1988b). Daha sonraları Macar Bilim Akademisi'nden (Hungarian Academy of Sciences, Budapest) Prof. Tamás Roska ile birlikte yine bu proje kapsamında yapılan çalışmada, Prof. Chua ve Prof. Roska Çok Kapsamlı HSA Makinesi (HSA-ÇKM) (CNN Universal Machine (CNN-UM)) yapısını geliştirmişlerdir (Roska ve Chua, 1993). Tasarlanan HSA-ÇKM yapısı önce küçük ölçekli (64×64, ACE4k), sonra daha büyük ölçekli (128×128, ACE16k) analog tümdevreler üzerinde gerçeklenmiştir (Liñán vd., 1999, 2001, 2002a; 2002b; Rodríguez-Vázquez vd., 2004).

HSA iyi tanımlanmış bir yapay sinir ağı yapısıdır ve özellikle görüntü işleme uygulamalarında kullanılmaktadır. HSA yapısında, görüntüdeki her bir piksele bir hücre karşı düşer ve hücreler görüntü işlemeyi paralel (eş zamanlı) çalışarak gerçekleştirir. Bu paralel çalışmayı sağlamak için tümdevre üzerinde görüntüdeki piksel sayısı kadar işlemci gerçeklenmesi gerekir. Bu çok sayıdaki işlemci yapısının tümdevre üzerinde gerçeklenebilmesi, dijital işlemcilere göre çok daha az yer kaplayan analog işlemciler (hücreler) kullanılarak mümkün olmaktadır. Bu şekilde gerçeklenmiş tümdevrelerin en üstün yanı işlem hızlarının çok yüksek olmasıdır. Ancak bu güne kadar gerçekleştirilmiş en gelişmiş tek katmanlı HSA analog tümdevresi (ACE16k) 128×128 hücre içermektedir ve tümdevrenin eşdeğer dijital doğruluğu 7-8 bittir. Günümüzün görüntü işleme uygulamalarının daha yüksek çözünürlüklere ihtiyaç duyması, analog gerçeklemelerin gürültü ve sıcaklık değişimlerine dijital eşdeğerlerine göre daha duyarlı ve üretim maliyetlerinin çok daha yüksek olması araştırmacıları HSA yapısının dijital emülasyonları üzerinde çalışmaya yöneltmiştir (Feher vd., 1996; Keresztes vd., 1999; Hidvégi vd., 2002, 2003; Nagy ve Szolgay, 2002, 2003; Beke vd., 2004; Toledo vd., 2005a; 2005b; Vörösházi vd., 2006; Kincses vd., 2006; Martínez vd., 2007). Emülatör yapılarının FPGA tümdevreleriyle gerçeklenmesi ise, paralel çalışan işlem birimlerinin FPGA üzerinde oluşturulmasının kolaylığı, yeniden yapılandırılabilmeleri ve maliyetlerinin düşük olması bakımından yeğlenen özellikleri içermektedir.

Bu çalışmada bugüne kadar oluşturulan HSA emülatörü yapıları incelenmiş, geliştirilmiş ve yeni bir HSA emülatörü yapısı önerilmiştir. Önerilen yapı bir FPGA geliştirme kartı üzerinde gerçeklenmiş ve gerçek zamanlı bir video işleme uygulamasında kullanılmıştır. Bu yapının en önemli özellikleri, FPGA dışında bir bellek elemanı (RAM) kullanmaması, 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.

Emülatör yapısı iki aşamada geliştirilmiştir. Öncelikle hız kaygısı güdülmeksizin bir tek işlem birimi ile çalışan ve FPGA içerisindeki blok RAM'larda yüklü 128×48 piksel 9 bit gri tonlamalı bir görüntüyü işleyerek sonucu VGA monitöründe gösteren bir donanım yapı tasarlanmış ve FPGA üzerinde gerçeklenmiştir. Bu yapının oluşturulmamsıdaki amaç HSA emülatörü tasarımı ve FPGA üzerinde gerçeklenmesi ile ilgili deneyim kazanmaktır. Daha sonra tasarlanan işlem birimi yapısı geliştirilerek ardışık düzen yapısına sahip ve birbirlerine kaskad bağlanabilir hale getirilmiş; video giriş ve I^2C blokları oluşturulmuş, ayrıca kontrol bloğu da geliştirilerek emülatör yapısı tamamlanmıştır. Tasarım, Celoxica firmasının RC203 kartı üzerinde bulunan Virtex-II 3000 FPGA tümdevresi ile gerçeklenmiş, ayrıca video giriş\çıkışı için video ADC ve DAC tümdevreleri kullanılmıştır.

Tasarımın bilgisayar ortamına aktarılması için Xilinx ISE yazılımının şematik editörü ve bu yazılımın "CORE Generator" uygulaması kullanılmıştır. Tasarımın simülasyonu için de yine ISE yazılımı kullanılmıştır. Ancak simülasyon işleminin uzun zaman alması nedeniyle, daha sonra simülasyon yerine "ChipScope Pro" yazılım kullanılarak FPGA içerisinde bir lojik analizör bloğu oluşturulmuştur. Bu blok sayesinde FPGA üzerindeki işaretler sistem çalışırken bilgisayar ekranında gözlenmiş ve doğrulanmıştır.

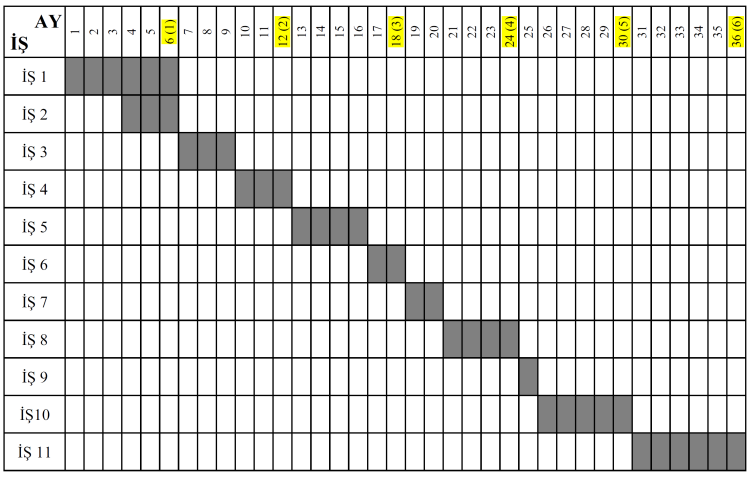
Yapılan literatür taramasında dış RAM kullanan birkaç çalışmanın mevcut olduğu görülmüş ancak dış RAM kullanmayan yalnızca bir çalışmaya rastlanmıştır. Martínez vd. (2007) tarafından gerçekleştirilen bu çalışmada Xilinx firmasının Virtex-4 ailesinde bir FPGA tümdevresi kullanılmış olup, gerçeklenen yapının standart bir VGA videoyu (640×480 piksel) gerçek zamanlı olarak işleyebileceği belirtilmiştir. Bu çalışmada gerçeklenen yapı içerisindeki her bir işlem birimi, iki blok RAM ve iki DSP48 bloğu kullanarak 17 saat darbesinde bir piksel işlemektedir.

Bu tez çalışmasında gerçeklenen yapıdaki işlem birimi ise üç blok RAM ve üç çarpıcı kullanarak üç saat darbesinde bir piksel işlemektedir. Kaba bir hesapla, kullanılan donanım miktarı bir buçuk kat arttırılmış, buna karşın hız beş buçuk kattan daha fazla artmıştır. Önerilen yapının sağladığı bu hız artışı, standart VGA videonun sadece Virtex-4 gibi gelişmiş ve yüksek maliyetli FPGA aileleri ile değil, Spartan-3 ve Virtex-II gibi daha düşük maliyetli FPGA aileleri ile de gerçek zamanlı olarak HSA yöntemiyle işlenmesine olanak sağlamıştır. Donanım yapısının tasarımında herhangi bir kaynaktan alınan bir HSA emülatörü yapısı kullanılmamış olup tamamı özgündür. Gerçekleştirilen emülatör yapısı "CPU Turkey 2008" yarışmasının "Akademik Yenilikçi Gömülü Sistem Tasarımı" kategorisinde birincilik ödülü almıştır.

Tez çalışmasına ilişkin iş planı ve iş-zaman diyagramı 1.1 bölümünde, HSA ile ilgili genel bilgiler 2. bölümde, literatürdeki bazı önemli dijital HSA emülatörü yapılarıyla ilgili genel bilgiler 3. bölümde, bu tez çalışmasında gerçekleştirilen ilk HSA emülatörü yapısı 4. bölümde, geliştirilmiş HSA emülatörü yapısı 5. bölümde, sonuçlar ise 6. bölümde anlatılmıştır. Uygulamada kullanılan HSA emülatörünün kontrol işaretleri Ek-2'de, şematik diyagramları Ek-3'de verilmiştir.

1.1 Tez Dahilinde Yapılan Çalışmalar

Tez dahiline yapılan çalışmalar 11 ayrı iş başlığı altında incelenebilir. Tez çalışmasının İş-Zaman diyagramı Şekil 1.1'de, iş tanımları ve her altı aylık dönemde yapılan işler ise şekilden sonra verilmiştir.



Şekil 1.1 İş-Zaman Diyagramı.

1.1.1 1. ve 6. Aylar Arasında Yapılan İşler

İŞ 1: Malzeme temini.

HSA emülartörü tasarımında kullanılacak olan donanım ve yazılımlar temin edilmiş ve kabaca incelenmiştir. Temin edilen donanım ve yazılımlar aşağıda verilmiştir.

- Xilinx "Spartan-3 Starter Kit" kartı
- Celoxica RC203 katı (gerçeklenen HSA yapıları RC203 kartında çalıştırılmıştır)
- Xilinx ISE ve ChipScope Pro yazılımları

Kullanılan donanım ve yazılımlar 4.2 ve 5.1 bölümlerinde anlatılmıştır.

İŞ 2: Yayım incelemesi.

FPGA üzerinde HSA yapısının oluşturulması ile ilgili bilgi içeren çalışmalar ayrıntılı olarak incelenmiş ve FPGA üzerinde gerçekleştirilecek ilk donanım yapısı kabaca belirlenmiştir. İncelenen yayımlar 3. bölümde anlatılmıştır.

1.1.2 7. ve 12. Aylar Arasında Yapılan İşler

İŞ 3: Spartan-3 FPGA tümdevresinin öğrenilmesi.

Xilinx firmasının Spartan-3 FPGA ailesinin donanım yapısı ayrıntılı olarak incelenmiş ve öğrenilmiştir. Spartan-3 FPGA ailesi tez kapsamında anlatılmamıştır (bilgi için Xilinx firmasının internet adresine (http://www.xilinx.com) bakılmalıdır).

İŞ 4: Tasarım ve simülasyon yazılımının öğrenilmesi.

FPGA üzerinde devre tasarım ve simülasyonunun yapılması için kullanılan ISE yazılımı öğrenilmiş ve bu amaçla ISE yazılımı kullanılarak "Spartan-3 Starter Kit" kartı üzerinde çalışan örnek bir donanım yapısı oluşturulmuştur.

1.1.3 13. ve 18. Aylar Arasında Yapılan İşler

İŞ 5: Donanım tanımlama dilinin öğrenilmesi.

VHDL (VHSIC (Very High Speed Integrated Circuits) Hardware Description Language) donanım tanımlama dili öğrenilmiştir. Bu dil tez çalışması dahilinde öğrenilmiş ancak tasarım aşamasında hiç kullanılmamıştır. Tasarım tamamen ISE yazılımının şematik editörü ve "CORE Generator" eklentisi (yardımcı yazılımı) kullanılarak gerçekleştirilmiştir.

İŞ 6: FPGA üzerinde çalışacak bir HSA emülatörü donanım yapısı tasarlanması.

Tez çalışmasındaki ilk HSA emülatörü donanım yapısı kağıt üzerinde tasarlanmış, tasarlanan donanım yapısının ISE yazılımının şematik editörü ile çizilmiş ve kontrol bloğunun simülasyonu yapılarak çalıştığı gözlenmiştir.

1.1.4 19. ve 24. Aylar Arasında Yapılan İşler

İŞ 7: Virtex-II FPGA donanım yapısının öğrenilmesi.

Celoxica firmasının RC203 kartı üzerinde bulunan Xilinx Virtex-II tümdevresinin donanım yapısı öğrenilmiştir. Virtex-II donanım yapısının Spartan-3'e çok benzemesinden dolayı görece kolay öğrenilmiştir.

İŞ 8: Sistem dahilinde kullanılan FPGA dışındaki tümdevrelerin öğrenilmesi.

RC203 kartı üzerinde bulunan video DAC (ADV7123) ve RC203 kartının genişleme yuvasına bağlı bir kartta bulunan video ADC (TVP7001) tümdevreleri öğrenilmiştir.

1.1.5 25. ve 30. Aylar Arasında Yapılan İşler

İŞ 9: Tez dahilinde tasarlanan ilk HSA emülatörü donanım yapısının çalıştırılması.

Büyük çoğunluğu İŞ 6'da tasarlanan, (tez dahilindeki) ilk HSA emülatörü yapısı RC203 kartında çalıştırılmıştır. Bu yapı FPGA'in iç belleğinde bulunan 128×48 piksel sabit 9 bit gri tonlamalı bir görüntüyü ardışık düzen yapısına sahip olmayan bir tek işlem bloğu ile işler ve sonucu bir VGA monitöründe gösterir. Yapının hızlı çalışması hedeflenmemiş, sadece çalışan bir HSA emülatörü gerçeklenmiştir. Böylece, hem tasarım ortamı hem de emülatör tasarımıyla ilgili deneyim kazanılmıştır. Bu yapı 5. bölümde anlatılmıştır.

İŞ 10: Gerçek zamanlı video işleyen HSA emülatörü yapısının tasarlanması ve çalıştırılması.

İŞ 9'de gerçeklenen yapı geliştirilerek ardışık düzen yapısına sahip ve kaskad bağlanabilen işlem birimleriyle çalışır hale getirilmiştir. Ayrıca "I^2C" seri haberleşme, "Adres ve Kontrol", "Video Giriş" ve "Video Çıkış" blokları tasarlanmıştır. Gerçeklenen yapı 640×480 piksel 60 fps 9 bit gri tonlamalı VGA videoyu gerçek zamanlı olarak işleyerek sonuç videoyu bir VGA monitöründe gösterir. Bu yapı 6. bölümde anlatılmıştır.

1.1.6 30. Aydan Sonra Yapılan İşler

30. aydan sonra yapılan işler (İŞ 11) aşağıda verilişitir.

- CNNA 2008 (İspanya) konferansı için bildiri yazılması, sunum ve demonstrasyonunun yapılması (Kayaer ve Tavşanoğlu, 2008a)
- Doktora tezinin yazılması
- CPU-Turkey yarışması dahilinde, CeBIT Eurasia Bilişim 2008 fuarına katılım ve gerçeklenen yapının demonstrasyonunun yapılması
- Gömülü Sistemler sempozyumu (İTÜ) için bildir özeti yazılması ve poster sunumu yapılması (Kayaer ve Tavşanoğlu, 2008b)

2. HÜCRESEL SİNİR AĞLARI VE HSA ÇOK KAPSAMLI MAKİNESİ

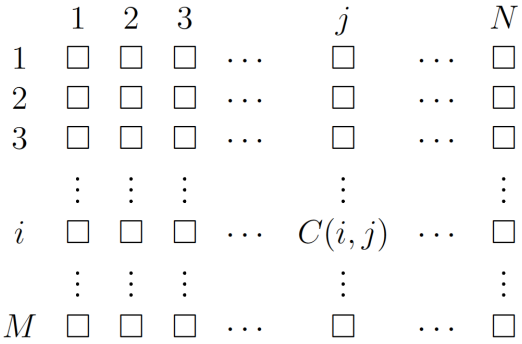
Hücresel Sinir Ağları (HSA) çok boyutlu ayrık uzayda tanımlanan güçlü bir paralel hesaplama modelidir. Chua-Yang tanımına göre:

- HSA çok boyutta düzenli sıralanmış elemanlardan (hücrelerden) oluşan bir yapıdır.
- Hücre dizilimi, örneğin düzlemsel dikdörtgen, üçgen veya altıgen ızgara, torus, üç boyutlu sonlu bir dizi veya iki boyutlu katmanlardan oluşan üç boyutlu bir dizi şeklinde olabilir.
- Hücreler, her biri bir veya birkaç parametrik fonksiyon ile tanımlanan çok girişli ve tek çıkışlı işlemcilerdir.
- Her hücrenin içerisinde, bazı durumlarda dışarıdan gözlenemeyen bir durum değişkeni mevcuttur.
- Değişik komşuluk değerleri ile çalışabilirler.
- HSA dinamik sistemi sürekli zaman veya ayrık zamanda çalışabilir.
- Sürekli zamanda HSA veri ve parametreleri sürekli değerler alır.
- Ayrık zamanda HSA birden fazla yineleme (iteration) ile sonuca ulaşır.

HSA'nın temel karakteristiği, birimler (hücreler) arasındaki bağlantıların yerel (lokal) olmasıdır. Yani verilerin sadece birbirine komşu (komşuluk içerisindeki) hücreler arasında doğrudan paylaşılır. Tabii bu yapı, tüm verinin işlenebilmesi amacıyla birbirine doğrudan bağlı olmayan hücreler arasındaki veri alışverişine de dolaylı olarak olanak sağlar.

2.1 Hücresel Sinir Ağları

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. $M \times N$ adet hücre içeren iki boyutlu bir HSA yapısındaki hücre dizilimi Şekil 2.1'de verilmiştir. Şekilde $C(i, j)$ $i.$ satır ve $j.$sütundaki hücreyi ifade etmektedir.

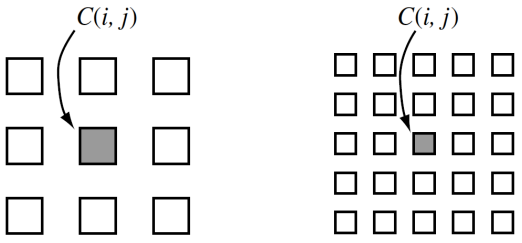


Şekil 2.1 $M \times N$ adet hücre içeren iki boyutlu HSA yapısındaki hücre dizilimi.

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

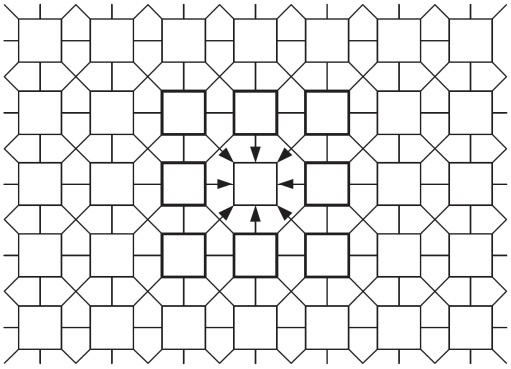
$$S_r(i,j) = \left\{ C(k,l)\big|_{1\le k\le M,\,1\le l\le N} \ \max\{|k-i|,|l-j|\} \le r \right\} \tag{2.1}$$

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 2.2'de sırasıyla bir ve iki komşuluk ($r=1$ ve $r=2$) için $C(i,j)$ hücresinin $S_r(i,j)$ etki alanı gösterilmiştir.



Şekil 2.2 $r=1$ (3×3) ve $r=2$ (5×5) komşuluk için $C(i,j)$ hücresinin etki alanı.

Gerçeklemede sağladığı kolaylıklar göz önünde tutularak HSA uygulamaları genellikle bir komşuluklu olarak oluşturulur. Bir komşuluklu HSA için hücreler arasındaki bağlantılar Şekil 2.3'de verilmiştir.



Şekil 2.3 Bir komşuluklu HSA için hücreler arasındaki bağlantılar.

Her hücre, komşuluğu içerisinde bulunan hücrelerin çıkışlarından belirli ağırlık değerleriyle geri besleme alır. Bu ağırlık değerleri geri besleme sinaptik operatörü (feedback synaptic 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) giriş değeri bağlıdır. Sinaptik operatörler (ağırlıklar) ve eşik değerleri değiştirilerek aynı HSA yapısı değişik görüntü işleme uygulamalarında kullanılabilir. Örneğin kenar belirleme, köşe belirleme, kontör belirleme, aşındırma (erosion), genişletme (dilation) vb..

$M \times N$ adet hücre içeren iki boyutlu HSA yapısındaki hücrelerin durum denklemi (2.2.a)'da, hücrenin çıkışı ve durumu arasındaki ilişkiyi ifade eden çıkış fonksiyonu ise (2.2.b)'de verilmiştir.

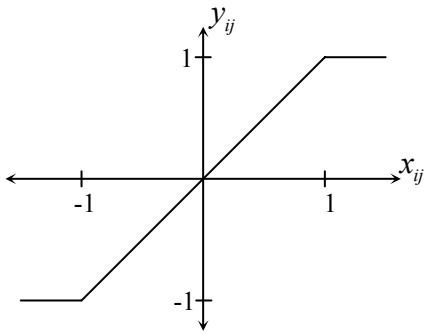
$$\frac{dx_{ij}(t)}{dt} = -x_{ij}(t) + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)y_{kl}(t) + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u_{kl} + I \qquad (2.2.a)$$
$$1 \le i \le M,\ 1 \le j \le N$$

$$y_{ij} = f(x_{ij}) = \frac{1}{2}\left(\left|x_{ij}+1\right| - \left|x_{ij}-1\right|\right) \qquad (2.2.b)$$
$$1 \le i \le M,\ 1 \le j \le N$$

Parçalı doğrusal (partially linear) doyma türü bir fonksiyon olan çıkış fonksiyonu (2.2.b), (2.3) denkleminde verilen şekliyle de ifade edilebilir. Çıkış fonksiyonunun grafiği Şekil 2.4'de verilmiştir.

$$y_{ij} = f(x_{ij}) = \begin{cases} 1, & x_{ij} > 1 \\ x_{ij}, & |x_{ij}| \le 1 \\ -1, & x_{ij} < -1 \end{cases} \tag{2.3}$$

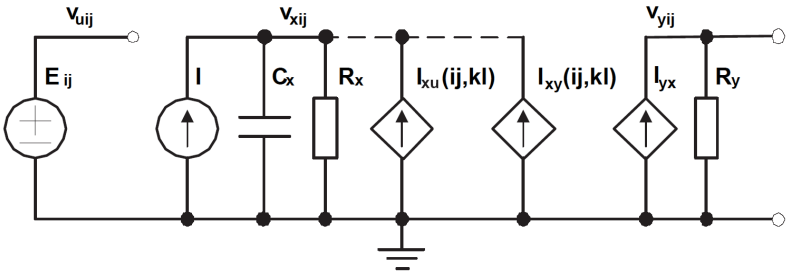
Burada x_{ij} $C(i,j)$ hücresinin durum değişkenini, y_{ij} bu hücrenin çıkış değişkenini, u_{ij} girişini (giriş piksel değerini), I_{ij} ise $-x_{ij}$ 'ye eklenen sabit bir değeri (eşik) göstermektedir. $A(i,j;k,l)$ ve $B(i,j;k,l)$ ise sırasıyla geri besleme ve giriş sinaptik ağırlıklarıdır. $A(i,j;k,l)$, $C(i,j)$ hücresinin komşuluğu içerisindeki $C(k,l)$ hücresinin çıkışı ile $C(i,j)$ hücresi arasındaki sinaptik operatörü (ağırlığı), $B(i,j;k,l)$ ise komşuluk içindeki $C(k,l)$ hücresinin girişi ile $C(i,j)$ hücresi arasındaki sinaptik operatörü (ağırlığı) gösterir.



Şekil 2.4 Hücrenin çıkış fonksiyonu.

2.2 HSA Hücresinin Devre Modeli

Bir HSA hücresinin Chua-Yang devre modeli Şekil 2.5'de verilmiştir.



Şekil 2.5 HSA hücresinin devre modeli.

Şekil 2.5'de görülen devre doğrusal (linear) dirençler, doğrusal bir kapasite, doğrusal ve doğrusal olmayan (nonlinear) kontrollü kaynaklar ve bir tane bağımsız kaynak içeren,

dinamik bir devredir. Burada *u*, *x* ve *y* indisleri sırasıyla giriş, durum ve çıkışı ifade etmektedir.

Devre modeli şu özelliklere sahiptir:

- Doğrusal kapasite C_x ve doğrusal direnci R_x hücre çekirdeğini oluşturur ve devre dinamiklerinden sorumludur.
- Durum kapasitesi C_x hücrenin tek bellek elemanıdır.
- Hücrenin giriş gerilimi v_{uij}, bağımsız gerilim kaynağı E_{ij} tarafından sağlanır ve geçici hal sırasında [−1, 1] aralığında olan sabit bir değere sahiptir.
- Hücrenin durum gerilimi v_{xij}, C_x kapasitesinin gerilimidir. Bu gerilim başlangıç durumunda (t=0), herhangi bir değere sahiptir.
- Hücrenin çıkış gerilimi v_{yij}, hücrede doğrusal olmayan (parçalı doğrusal) tek eleman olan bir gerilim kontrollü akım kaynağı (I_{yx}) tarafından sağlanır. I_{yx}, v_{xij} tarafından kontrol edilmektedir ve burada $I_{yx} = \frac{1}{R_y} f(v_{xij})$ ve $v_{yij} = f(v_{xij})$ olmaktadır. $f(.)$ hücrenin çıkış fonksiyonudur.
- Komşuluk içerisindeki hücrelerin hücreye olan etkisi, doğrusal gerilim kontrollü akım kaynakları $I_{xu}(i,j;k,l)$, $I_{xy}(i,j;k,l)$ ile modellenmiştir. Bu akım kaynakları sıra ile her bir komşu hücredeki giriş gerilimi v_{ukl} ve her bir komşu hücrenin çıkış gerilimi v_{ykl}'ye bağımlıdır. Bu kaynakların karakteristikleri şöyle verilebilir: Her $C(k,l) \in S_r(i,j)$ için $I_{xy}(i,j;k,l) = A(i,j;k,l).v_{ykl}$ ve $I_{xu}(i,j;k,l) = B(i,j;k,l).v_{ukl}$. Eğer *p* komşu hücrelerin sayısını gösteriyorsa, her bir hücre en çok 2*p* tane doğrusal gerilim kontrollü akım kaynağı içerir.
- Eşik (bias) terimi *I* bağımsız akım kaynağı ile gösterilmiştir.

Daha anlaşılır olması açısından v_{uij}, v_{xij} ve v_{yij} yerine kısaca u_{ij}, x_{ij} ve y_{ij} yazılabilir. Böylece $C(i,j)$ hücresinin durum denklemi, Şekil 2.5'den Kirchoff'un akımlar yasası kullanarak (2.4) denklemlerinde verildiği gibi elde edilebilir.

$$C\frac{dx_{ij}(t)}{dt} = -\frac{1}{R_x}x_{ij}(t) + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)y_{kl}(t) + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u_{kl} + I \qquad (2.4.a)$$

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

$$y_{ij} = f(x_{ij}) = \begin{cases} 1, & x_{ij} > 1 \\ x_{ij}, & |x_{ij}| \le 1 \\ -1, & x_{ij} < -1 \end{cases} \tag{2.4.b}$$

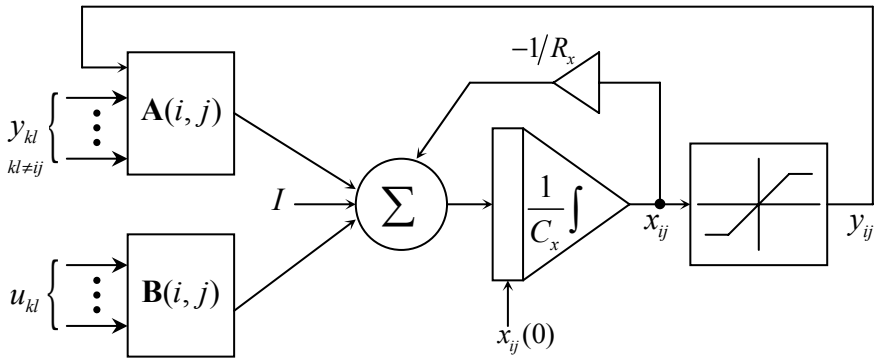
(2.4.a) denkleminin her iki yanının integrali alınırsa (2.4) denklemlerinden:

$$x_{ij}(t) = \frac{1}{C_x}\int \left[-\frac{1}{R_x} x_{ij}(t) + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l) y_{kl}(t) + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l) u_{kl} + I \right] dt \tag{2.5.a}$$

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

$$y_{ij} = f(x_{ij}) \tag{2.5.b}$$

denklemleri elde edilir. Bu denklemlerle ifade edilen bir HSA hücresinin blok diyagramı Şekil 2.6'da verilmiştir.



Şekil 2.6 Bir HSA hücresinin blok diyagramı.

HSA hücresi $C(i,j)$, bir tek durum değişkenine sahiptir ve bu nedenle birinci mertebeden bir sistem niteliğindedir. Hücrenin komşuluğu içerisindeki her bir $C(k,l)$ hücresinin çıkış ve giriş değeri, $C(i,j)$ hücresine ağırlıklı bir toplam oluşturacak şekilde sırasıyla $A(i,j;k,l)$ geri besleme ve $B(i,j;k,l)$ giriş sinaptik ağırlıkları ile bağlıdır. Bu ağırlıklı toplama bir de I eşik değeri toplanarak, ($\frac{dx_{ij}}{dt}$, x_{ij} durumunun değişim hızını gösterdiğine göre) durum değişkeninin değişim hızı elde edilir. Denklemin genelliği bozulmadan, devrenin zaman sabiti $\tau = R_x C_x$; $R_x = 1$, $C_x = 1$ seçilerek 1'e eşitlenebilir. Böylece $M \times N$ hücreye sahip Chua-Yang devre modelinin durum denklemi (2.2.a) ve (2.2.b) denklemleriyle ifade edilen HSA'nın durum denklemine eşdeğer olur.

2.3 Konumdan Bağımız HSA ve HSA Şablonları

Konumdan bağımsız HSA (Space-invariant CNN) yapısında, $A(i,j;k,l)$ ve $B(i,j;k,l)$ bağlantı ağırlıkları hücrenin konumundan bağımsızdır ve sadece hücre etrafındaki küçük bir komşuluk alanı için tanımlanmıştır. Bir hücre ile bu hücrenin komşuluk hücreleri arasındaki bağlantı ağırlıklarını ifade eden matrislere klonlama şablonları denir. Bu şablonlar her bir hücre için uygulandığında, aynı bağlantı ağırlıklarını klonlarlar. Klonlama şablonu 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 sayıdan oluşan kümenin belirleyeceği anlamına gelir. Bir komşuluklu ($r = 1$) konumdan bağımsız bir HSA yapısının durum denklemi (2.6)'de verilmiştir.

$$\dot{x}_{ij} = -x_{ij} + \sum_{k=-1}^{1}\sum_{l=-1}^{1} a_{k,l} y_{i+k,j+l} + \sum_{k=-1}^{1}\sum_{l=-1}^{1} b_{k,l} u_{i+k,j+l} + I \tag{2.6}$$

(2.6) ifadesinin kısaltılması amacıyla, (2.7) eşitliğinde tanımlanan bir operatör ($\otimes$) kullanılabilir. Bu operatör, literatürde şablon nokta çarpımı (template dot product) olarak adlandırılmıştır.

$$\begin{aligned}
\sum_{k=-1}^{1}\sum_{l=-1}^{1} a_{k,l} y_{i+k,j+l} &\triangleq \begin{bmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ a_{0,-1} & a_{0,0} & a_{0,1} \\ a_{1,-1} & a_{1,0} & a_{1,1} \end{bmatrix} \otimes \begin{bmatrix} y_{i-1,j-1} & y_{i-1,j} & y_{i-1,j+1} \\ y_{i,j-1} & y_{i,j} & y_{i,j+1} \\ y_{i+1,j-1} & y_{i+1,j} & y_{i+1,j+1} \end{bmatrix} = \mathbf{A} \otimes \boldsymbol{Y}_{ij} \\
\sum_{k=-1}^{1}\sum_{l=-1}^{1} b_{k,l} u_{i+k,j+l} &\triangleq \begin{bmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & b_{0,0} & b_{0,1} \\ b_{1,-1} & b_{1,0} & b_{1,1} \end{bmatrix} \otimes \begin{bmatrix} u_{i-1,j-1} & u_{i-1,j} & u_{i-1,j+1} \\ u_{i,j-1} & u_{i,j} & u_{i,j+1} \\ u_{i+1,j-1} & u_{i+1,j} & u_{i+1,j+1} \end{bmatrix} = \mathbf{B} \otimes \boldsymbol{U}_{ij}
\end{aligned} \tag{2.7}$$

(2.6) denkleminin şablon nokta çarpımı operatörüyle yazılmış hali (2.8) denkleminde verilmiştir.

$$\dot{x}_{ij} = -x_{ij} + \mathbf{A} \otimes \boldsymbol{Y}_{ij} + \mathbf{B} \otimes \boldsymbol{U}_{ij} + I \tag{2.8}$$

Literatürde şablonlar genellikle **A** şablonu ve **B** şablonu veya sırasıyla geri-besleme klonlama şablonu ve ileri-besleme (veya giriş) klonlama şablonu olarak adlandırılır ve birer matrisle ifade edilirler.

$r = 1$ olan bir HSA için şablonlar (2.9) eşitliğinde görüldüğü gibi verilebilir.

$$\mathbf{A}=\begin{bmatrix} A(i,j;i-1,j-1) & A(i,j;i-1,j) & A(i,j;i-1,j+1) \\ A(i,j;i,j-1) & A(i,j;i,j) & A(i,j;i,j+1) \\ A(i,j;i+1,j-1) & A(i,j;i+1,j) & A(i,j;i+1,j+1) \end{bmatrix}$$
$$\mathbf{B}=\begin{bmatrix} B(i,j;i-1,j-1) & B(i,j;i-1,j) & B(i,j;i-1,j+1) \\ B(i,j;i,j-1) & B(i,j;i,j) & B(i,j;i,j+1) \\ B(i,j;i+1,j-1) & B(i,j;i+1,j) & B(i,j;i+1,j+1) \end{bmatrix} \quad (2.9)$$

(2.9) eşitliği, (2.10) eşitliği şeklinde de ifade edilebilir.

$$\mathbf{A}=\begin{bmatrix} a_{i-1,j-1} & a_{i-1,j} & a_{i-1,j+1} \\ a_{i,j-1} & a_{i,j} & a_{i,j+1} \\ a_{i+1,j-1} & a_{i+1,j} & a_{i+1,j+1} \end{bmatrix}=\begin{bmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ a_{0,-1} & a_{0,0} & a_{0,1} \\ a_{1,-1} & a_{1,0} & a_{1,1} \end{bmatrix}$$
$$\mathbf{B}=\begin{bmatrix} b_{i-1,j-1} & b_{i-1,j} & b_{i-1,j+1} \\ b_{i,j-1} & b_{i,j} & b_{i,j+1} \\ b_{i+1,j-1} & b_{i+1,j} & b_{i+1,j+1} \end{bmatrix}=\begin{bmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & b_{0,0} & b_{0,1} \\ b_{1,-1} & b_{1,0} & b_{1,1} \end{bmatrix} \quad (2.10)$$

2.4 Ayrık Zamanlı HSA

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

$$\dot{x}_{ij}(t)=-x_{ij}(t)+\sum_{C(k,l)\in S_r(i,j)} A_{ij,kl}y_{kl}(t)+\sum_{C(k,l)\in S_r(i,j)} B_{ij,kl}u_{kl}+I_{ij} \quad (2.11)$$

Euler ileri yaklaşıklık formülü kullanılarak, (2.11) durum denkleminin ayrık zamandaki eşdeğeri (2.12) denkleminde verildiği gibi elde edilebilir.

$$\frac{x_{ij}(n+1)-x_{ij}(n)}{T_s}=-x_{ij}(n)+\sum_{C(k,l)\in S_r(i,j)} A_{ij,kl}y_{kl}(n)+\sum_{C(k,l)\in S_r(i,j)} B_{ij,kl}u_{kl}+I_{ij} \quad (2.12)$$

Burada T_s örnekleme aralığıdır. (2.12) denklemi, sadece bir sonraki durum sol tarafta kalacak şekilde düzenlenirse AZ-HSA modeli (2.13) denkleminde verildiği gibi elde edilir.

$$x_{ij}(n+1)=(1-T_s)x_{ij}(n)+T_s\Big(\sum_{C(k,l)\in S_r(i,j)} A_{ij,kl}y_{kl}(n)+\sum_{C(k,l)\in S_r(i,j)} B_{ij,kl}u_{kl}+I_{ij}\Big)$$
$$y_{ij}=f(x_{ij})=\begin{cases} 1, & x_{ij}>1 \\ x_{ij}, & |x_{ij}|\le 1 \\ -1, & x_{ij}<-1 \end{cases} \quad (2.13)$$

2.4.1 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, (2.3) eşitliğiyle verilen Şekil 2.4'daki 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 (2.14.a) ve (2.14.b) denklemleriyle ifade edilir (Espejo vd., 1994; Dominguez-Castro vd., 1994).

$$v_{ij}(n) = (1-T_s)x_{ij}(n) + T_s\Big(\sum_{C(k,l)\in S_r(i,j)} A_{ij,kl}x_{kl}(n) + \sum_{C(k,l)\in S_r(i,j)} B_{ij,kl}u_{kl} + I_{ij}\Big) \tag{2.14.a}$$

$$x_{ij}(n+1) = \begin{cases} 1, & v_{ij}(\mathrm{n}) > 1 \\ v_{ij}(\mathrm{n}), & \left|v_{ij}(\mathrm{n})\right| \le 1 \\ -1, & v_{ij}(\mathrm{n}) < -1 \end{cases} \tag{2.14.b}$$

(2.14.a) denklemindeki T_s ve $(1-T_s)$ terimlerinin $A_{ij,kl}$, $B_{ij,kl}$ ve I_{ij} değerinin içine katılmasıyla $\hat{A}_{ij,kl}$, $\hat{B}_{ij,kl}$ ve $\hat{I}_{ij}$ değerleri elde edilir. 3×3 komşuluklu konumdan bağımsız bir HSA yapısı için $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$ şablonları ve $\hat{I}$ değeri (2.15) eşitliklerinde görüldüğü gibi elde edilir.

$$\hat{\mathbf{A}} = T_s.\begin{bmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ a_{0,-1} & \dfrac{1+T_s.(a_{0,0}-1)}{T_s} & a_{0,1} \\ a_{1,-1} & a_{1,0} & a_{1,1} \end{bmatrix}, \quad \hat{\mathbf{B}} = T_s.\mathbf{B}, \quad \hat{I} = T_s.I \tag{2.15}$$

Bu değişikliklerle birlikte (2.14.a) eşitliği, (2.16.a) ve (2.16.b) eşitliklerinde görüldüğü gibi iki parçalı şekilde ifade edilebilir.

$$v_{ij}(n) = \sum_{C(kl)\in S_r(i,j)} \hat{A}_{ij,kl}x_{kl}(n) + g_{ij} \tag{2.16.a}$$

$$g_{ij} = \sum_{C(kl)\in S_r(i,j)} \hat{B}_{ij,kl}u_{kl} + \hat{I}_{ij} \tag{2.16.b}$$

(2.16.b) eşitliğinden de görülebileceği gibi g_{ij} değerinin her piksel için sadece bir kez hesaplanması yeterlidir, v_{ij} değeri ise her iterasyonda değişir. 3×3 komşuluklu konumdan

bağımsız bir HSA yapısı için (2.16.a) ve (2.16.b) eşitlikleri sırasıyla (2.17.a) ve (2.17.b)'de görüldüğü gibi yazılabilir.

$$v_{ij}(n) = \hat{\mathbf{A}} \otimes \boldsymbol{X}_{ij}(n) + g_{ij} \tag{2.17.a}$$

$$g_{ij} = \hat{\mathbf{B}} \otimes \boldsymbol{U}_{ij} + \hat{I} \tag{2.17.b}$$

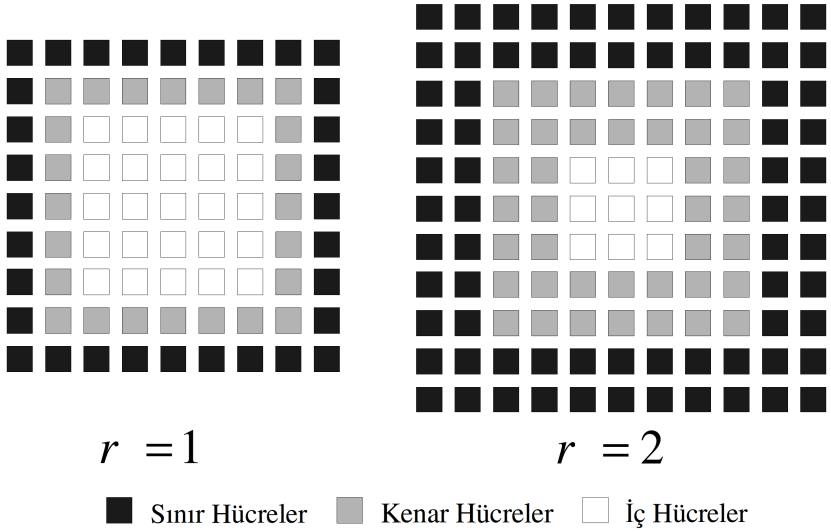
3. Bölümde anlatılan CASTLE ve Falcon dijital HSA emülatörü yapıları bu AZ-HSA modelini kullanır. Bu modelin kullanılma sebepleri:

1. Bir sonraki durumu hesaplanma işlemi, sonucu her iterasyonda değişen (2.16.a) denklemi ile sabit kalan (2.16.b) denklemi şeklinde iki parçaya ayrılmıştır. Böylece g_{ij} değerinin her piksel için sadece bir kez hesaplanır (her itersyon 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.

2.5 Sınır Koşulları

Pratik uygulamalarda kullanılan HSA yapıları genellikle sınırlı ızgaralar 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 komşusu bulunan iç hücrelere ise kenar hücreleri denir. Burada sanal hücrelerin gerçek hücre olmadığına dikkat edilmelidir, çünkü (2.2.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 2.7).

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 2.7 Komşuluk boyutu r 'nın sınır hücrelerine etkisi (Saatçi, 2003).

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

2.5.1 Dirichlet Sınır Koşulları

Bu tip sınır koşulları sabit veya sıfır sınır koşulları olarak da bilinir. Sınır hücrelerinin girişleri ve çıkışlarına sabit değerler atanmaktadır. (2.2.a) denkleminden de görüleceği gibi bunun yapılması her bir kenar hücresine sadece sabit bir değer eklenmesi anlamındadır. Bu sabit, kenar hücresine ek bir giriş olarak dahil edilebilir ve sınır hücrelerinin sıfır giriş ve çıkış değerlerine sahip olduğu düşünülebilir. Ayrıca, kenar hücrelerinin durum denklemleri, bu sabit değerler eklenerek değiştirilebilir ve böylece hesaplamaya hiçbir sınır hücresi dahil edilmemiş olur.

Bu sınır koşulu yöntemi görüntü işleme uygulamalarında bir kenar etkisine sahiptir. Bu etki, sabit değerli sınır hücrelerinin neden olduğu süreksizlikten kaynaklanır ve ızgaranın kenarında bazı istenmeyen sonuçların oluşmasına neden olabilir.

2.5.2 Dairesel Sınır Koşulları

Bu tip sınır koşulları katlamalı veya toroidal olarak da bilinir. Sınır hücrelerine ızgaranın diğer tarafındaki kenar hücrelerinin giriş ve durum değerleri verilir. Örneğin, ızgaranın en sol tarafında bulunan bir sınır hücresi her zaman, aynı satırdaki, ızgaranın sağ tarafında bulunan kenar hücresinin değerini alır. Benzer olarak, ızgaranın üst tarafında bulunan bir sınır hücresi her zaman, aynı sütundaki, ızgaranın alt tarafında bulunan kenar hücresinin değerini alır.

2.5.3 Neumann Sınır Koşulları

Bu tip sınır koşulları ayna veya yansıyan sınır koşulları olarak da bilinir. Bu kavramda sınır hücrelerine ızgaranın aynı tarafında olan kenar hücrelerinin değerleri verilir. Başka bir deyişle, ızgaradaki kenar hücreleri ızgaranın kenarlarında bir ayna görmektedir. Örneğin, ızgaranın sol kenarındaki bir kenar hücre sol komşusu olarak kendisini görmektedir. Bu yöntemin diğer iki yönteme göre VLSI gerçekleme kolaylığı açısından üstünlüğü olduğu gibi ayrıca, ızgara kenarında bulunan süreksizlikleri de yok edici etkisi vardır.

2.6 HSA Çok Kapsamlı Makinesi

HSA yapısının bir VLSI tümdevresi üzerinde gerçeklenmesi çok yüksek hesaplama hızlarına ulaşılmasını sağlamaktadır. Gerçeklenen yapının programlanabilir olması ise kullanılabilirliğinin arttırılması açısından gereklidir. Bu amaçla 1993 yılında HSA Çok Kapsamlı Makinesi (HSA-ÇKM) (CNN Universal Machine (CNN-UM)) yapısının geliştirildiği bir çalışma yapılmıştır (Roska ve Chua, 1993). Bu çalışmada, analog işlemciler ile lojik elemanlarından oluşan, analog ve lojik birimlerin bir arada kullanıldığı (analojik) bir yapı önerilmiştir. Bu yapıda görüntü işleme işlemleri analog, kontrol ve programlama işlemleri ise lojik birimler tarafından gerçekleştirilmektedir. Daha sonraları Angel Rodrigez-Vazquez'in de katkılarıyla HSA-ÇKM yapısını kullanan ACE4k ve ACE16k analojik tümdevreleri gerçeklenmiştir. HSA-ÇKM yapısı Şekil 3.1'de verilmiştir.

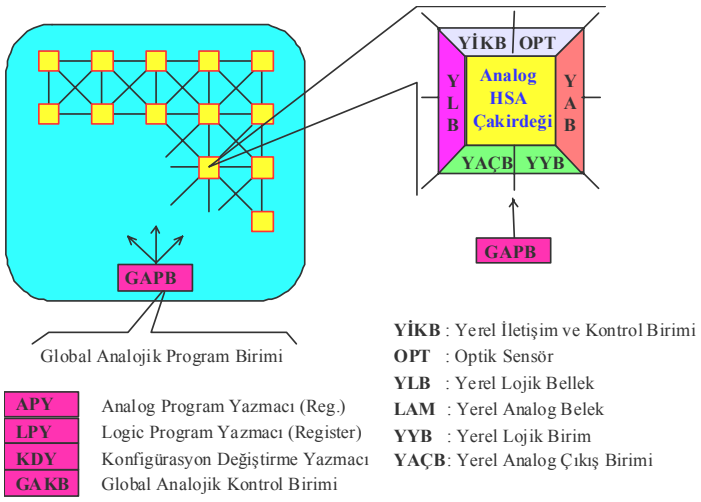
ACE4k analojik tümdevresi 1999 yılında gerçeklenmiştir (Liñán vd., 1999, 2002a). Bu tümdevrede 64×64 HSA hücresi mevcut olup HSA işlemlerini analog olarak yapılmaktadır. ACE4k tümdevresi geliştirilerek 2001 yılında ACE16k tümdevresi oluşturulmuştur (Liñán vd., 2001, 2002b; Rodríguez-Vázquez vd., 2004). Bu tümdevrede 128×128 HSA hücresi mevcut olup HSA işlemleri analog olarak yapılmaktadır.

ACE16k tümdevresinin bazı özellikleri:

- 128×128 boyutundaki bir görüntüyü yaklaşık 8μs de 7-8 bite eşdeğer doğrulukla işler.
- Görüntü bilgisini dijital veri yolu veya üzerindeki her bir hücrede bulunan optik sensör ile alabilir.
- Dijital veri giriş\çıkışı için 32 bitlik iki yönlü bir veri yolu kullanır, bu veri yolu 32 MHz'de çalışır ve giriş\çıkış hızı 128 MByte/s'dir.
- Dijital veri yolu üzerinden aldığı 640×480 piksel gri tonlamalı VGA görüntüsünü 100

fps hızla işleyebilir.

- Dijital veri yolu üzerinden aldığı (tümdevredeki hücreler boyutunda) 128×128 piksel gri tonlamalı görüntüyü 2500 fps hızla işler. Giriş görüntüsü optik olarak alındığında daha yüksek fps hızlarında çalışabilir.
- Günümüze dek üretilmiş HSA tümdevreleri arasında en fazla hücreye sahip (128×128) tek katmanlı HSA tümdevresidir.



Şekil 3.1 HSA Çok Kapsamlı Makinesi yapısı.

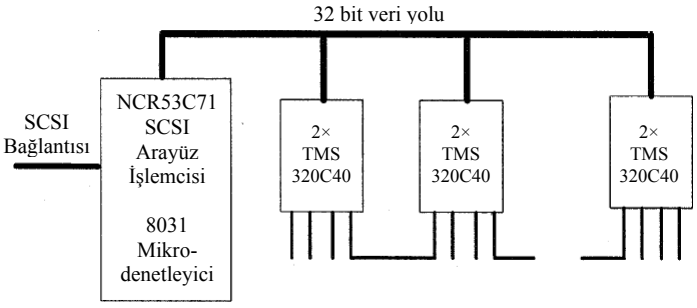
Bu tümdevrenin sakıncalı yanları, çok pahalı olması ve doğruluğunun 7-8 bitin üzerine çıkamamasıdır. Ayrıca, tümdevredeki hücre sayısından daha fazla piksele sahip bir görüntü işlenecekse, görüntü parçalara ayrılarak işlenmek zorundadır. Bu durum, tümdevrenin gerçek çalışma hızını ortaya koymasını engellemekte ve hızın veri yolunun hızıyla sınırlanmasına neden olmaktadır. Tümdevrenin bu güne kadar daha fazla hücreye sahip bir sürümü (versiyonu) üretilememiştir.

3. DİJİTAL HSA EMÜLATÖRÜ YAPILARI

Analojik tümdevrelerin işlem hızları çok yüksek olmasına rağmen eşdeğer dijital doğruluklarının 7-8 bit olması, maliyetlerinin çok yüksek olması, şu ana dek gerçekleştirilebilen analog işlemci sayısı en yüksek tümdevrenin 128×128 hücreli (işlemcili) ve halen üzerinde çalışılmakta olan 256×256 analog hücreli HSA tümdevresinin bitirilememiş olması, araştırmacıları HSA yapısının dijital emülasyonları üzerinde çalışmaya yöneltmiştir (Feher vd., 1996; Keresztes vd., 1999; Hidvégi vd., 2002; 2003; Nagy ve Szolgay, 2002; 2003; Beke vd., 2004; Toledo vd., 2005a; 2005b; Vörösházi vd., 2006; Kincses vd., 2006; Martínez vd., 2007). Bu bölüm altında önemli bazı dijital HSA emülatörü yapıları kısaca anlatılacaktır.

3.1 ACE Yapısı

Tasarımı planlanan ACE4k tümdevresinin oluşturacağı sonuçların karşılaştırılması amacıyla, 1996 yılında ACE adı verilen bir dijital HSA emülatörü yapısı oluşturulmuştur. Bu yapıda, 32 bit kayan noktalı aritmetikle işlem yapan TMS320C40 DSP tümdevreleri kullanılmıştır. Yapı 2 ila 16 adet arasında DSP'yi paralel olarak çalıştırabilmektedir. Sistemde 512 MByte'a (32MB×16) kadar dış RAM kullanılabilmekte ve hızı bir iterasyon için işlemci başına 2.87 μs/hücre'dir (Feher vd., 1996). Sistemin yapısı Şekil 3.1'de verilmiştir.



Şekil 3.1 ACE HSA emülatörü yapısı (Feher vd., 1996).

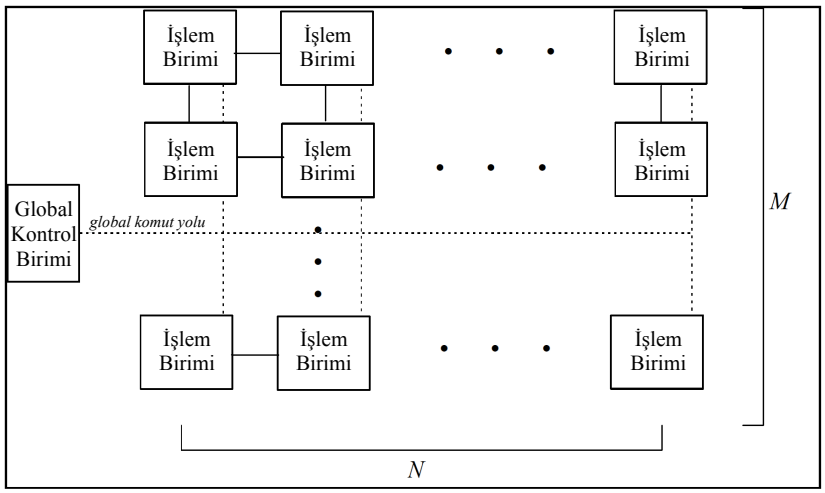
3.2 CASTLE Yapısı

CASTLE 1999 yılında 0.35μm, üç metal katman CMOS teknolojisi kullanılarak ASIC bir tümdevre olarak tasarlanmıştır. Bu tümdevre bünyesindeki 24 adet işlem birimi (processor unit) ile HSA'nın dijital emülasyonunu gerçekleştirmektedir. İşlem birimleri tümdevre üzerinde düzlemsel (matrisel) bir dağılım göstermektedir. Tümdevrenin hızı, 12 bit doğrulukta, bir iterasyon için işlem birimi başına 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 başına 35 ns/hücre, 9 ns/hücre ve 6 ns/hücre'dir (Hidvégi vd., 2002, 2003). 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.

CASTLE ve Falcon yapılarındaki işlemler 2.4.1 bölümünde anlatılan TSA ile AZ-HSA modeli temel alınarak gerçekleştirilir.

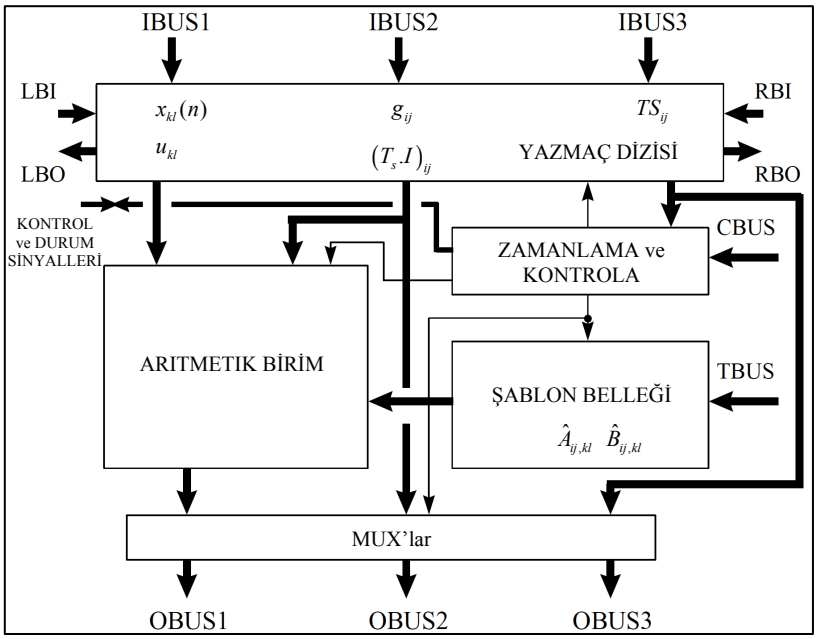
CASTLE yapısı matris şeklinde dizilmiş işlem birimleri ve bu işlem birimlerinin kontrolünü sağlayan bir kontrol birimden oluşur. Bu yapı Şekil 3.2'de gösterilmiştir.



Şekil 3.2 CASTLE yapısının blok diyagramı (Keresztes vd., 1999).

Yatay olarak sıralanmış işlem birimleri görüntünün dikey olarak kesilmiş şeritleri üzerinde işlem yaparlar. Örneğin, elimizdeki görüntünün genişliği x ekseninde (yatay eksende) 240 piksel olsun ve x eksenindeki işlem birimi sayısı da 6 olsun. Bu durumda her işlem birimi 240/6=40 piksel genişliğindeki bir resim üzerinde işlem yapacaktır. 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 3.3'de verilmiştir.

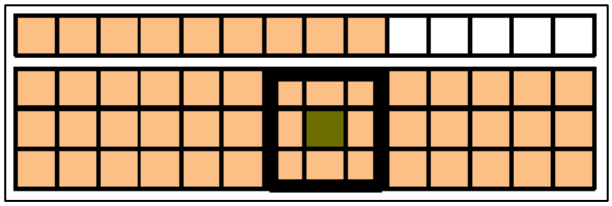


Şekil 3.3 CASTLE işlem biriminin yapısı (Keresztes vd., 1999).

IBUS1, IBUS2, IBUS3, CBUS, TBUS, LBI ve RBI yollarıyla (bus) veriler ve kontrol komutları işlem bitimine girer. OBUS1, OBUS2, OBUS3, LBO ve RBO yollarıyla da veriler işlem biriminden dışarı çıkar. IBUS1 u_{kl} giriş ve $x_{kl}(n)$ durum değerlerini, IBUS2 $(T_s.I)_{ij}$ ve g_{ij} değerlerini, IBUS3 TS_{ij} kullanılacak şablon numarasını işlem birimine taşır. LBI ve RBI soldaki ve sağdaki işlem birimlerinden, işlenen resmin sınır değerlerini işlem birimine taşırken, LBO ve RBO soldaki ve sağdaki işlem birimlerine işledikleri resimlerin sınır değerlerini gönderir. CBUS zamanlama ve kontrol işaretlerini işlem birimine taşır. TBUS ise şablondaki sayı değerlerini işlem birimindeki şablon belleğine yazar. Şablon belleği, resim işlenmeye başlamadan önce işlem birimine yüklenir. İşlem biriminden çıkan OBUS1, OBUS2 ve OBUS3, sıra ile yeni durum değerlerini, g_{ij} değerlerini ve kullanılacak şablon numaralarını bir sonraki (altındaki) işlem birimine iletir.

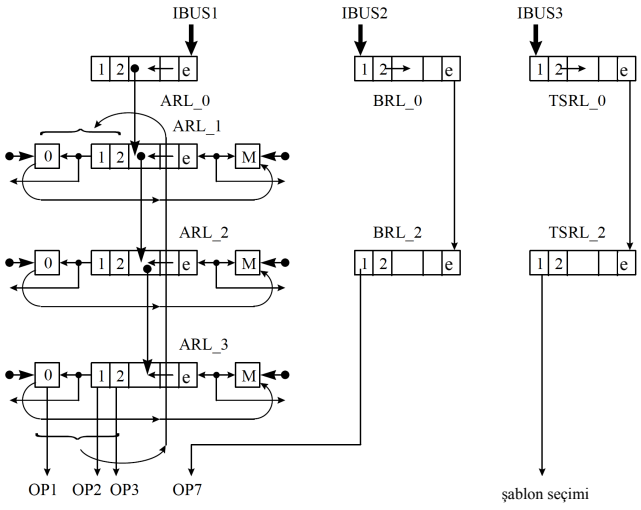
HSA emülatörünün hızlı çalışabilmesi için izlenen strateji, tümdevre (veya FPGA) dışındaki belleğe erişimin olabildiğince az sayıda tutulmasıdır. Bunun nedeni, belleğe erişim hızının tümdevrenin çalışma hızına göre oldukça yavaş olmasıdır. Bu nedenle bir iterasyon için gerekli olan bütün durum (x_{kl}) ve g_{ij} değerleri bu değerlere olan ihtiyaç sona erene kadar tümdevre içinde saklanmalıdır. Bu amaçla öncelikle işlenecek resme ait ince yatay bir kesit tümdevrenin iç belleğine alınır. Bu kesit şablonun 3×3 veya 5×5 olmasına göre değişim

gösterir. 3×3 şablon için resmin 3 satırının durum (x_{kl}) değerleri işlem biriminin içine alınır ve bu değerler işlem görürken 4. satırın durum (x_{kl}) değerleri boş bir yazmaç (register) bölgesine dış bellekten yüklenir. Bu durum Şekil 3.4 de gösterilmiştir.



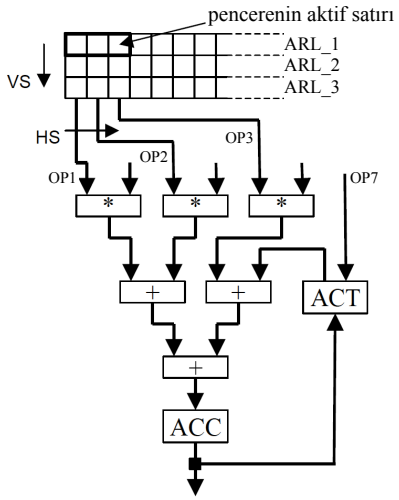
Şekil 3.4 Durum değerlerinin tümdevrenin iç belleğinde saklanışı ve bir pikselin bir sonraki durum değerinin hesaplanması için gerekli bölge (Keresztes vd., 1999).

İşlem birilerinin diziliminde, yatay eksende bulunan işlem birimlerinin sayısı arttıkça birim zamanda dış bellekten alınması gereken veri miktarı da (birim zamanda işlenen hücre sayısındaki artıştan dolayı) artar. Bu artış tasarlanan sistemin yapısına göre bir noktada sınırlanır, yani dış belleğin giriş\çıkış hazının sınırına ulaşılır. Bu sınıra ulaştıktan sonra, dikey konumda yerleştirilmiş işlem birimlerinin sayısını arttırmak gerekir. Dikey konumda yerleştirilmiş işlem birimleri, verilerini (x_{kl}, g_{ij}, TS_{ij}) bir önceki işlem birimi dizisinden alırlar ve piksellere ait durumlar için bir iterasyon gerçekleştirerek yeni durumları bulurlar (2.16). 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. Sistemin yazmaç (register) yapısı Şekil 3.5'de verilmiştir.

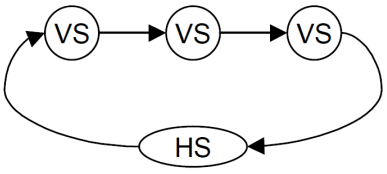


Şekil 3.5 CASTLE işlem biriminin yazmaç (register) yapısı (Keresztes vd., 1999).

Sistemde temel olarak üç adet kaydırmalı yazmaç (shift register) yolu mevcuttur. IBUS1 yolunun bağlı olduğu ilk yazmaç hesaplamada kullanılmayan tampon yazmaçtır ve işlem biriminin işleyeceği resim alanında bir satırdaki e adet durumu içerir. Hesaplama işlemi bittiğinde bu yazmaçtaki değerler (e adet) bir alttaki yazmaç gurubuna yüklenir. IBUS2 hesaplamada kullanılacak g_{ij} veya $(T_s.I)_{ij}$ değerlerini taşır. Bu değerler, sıra ile bir sonraki durum veya g_{ij} değeri hesaplanacak her piksel için bir tanedir. Örneğin, her piksel için ayrı bir g_{ij} değeri mevcuttur. IBUS3 ise bir sonraki durum değeri hesaplanacak hücrenin hangi şablona göre işlem göreceği bilgisini taşır. Sistem içinde 16 ayrı şablon seçilebilmektedir ve bu şablonlar her hücre için ayrı ayrı seçilebilir. ARL_1, ARL_2 ve ARL_3 kaydırmalı yazmaçları aritmetik birim tarafından işlem görecek verileri saklar. Aritmetik birimin basitleştirilmiş blok diyagramı Şekil 3.6'da, durum diyagramı ise Şekil 3.7'de verilmiştir.



Şekil 3.6 CASTLE'ın aritmetik birim yapısı.



Şekil 3.7 Aritmetik birimin durum diyagramı.

Aritmetik birim üç adet çarpıcı üç adet toplayıcı ve iki adet akümülatörden (ACC, ACT) oluşmaktadır. Öncelikle, hesaplamada kullanılacak sabit değer (g_{ij} veya $(T_s.I)_{ij}$) 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_{ij} ve kullanılacak şablonun numarasını işlem birimine iletir. Şekil 3.5'deki yazmaç şemasındaki "0" ve "M" pikselleri, sol ve sağdaki işlem birimlerine ait bölgelerdeki sınır piksellerinin durum değerlerini içerir.

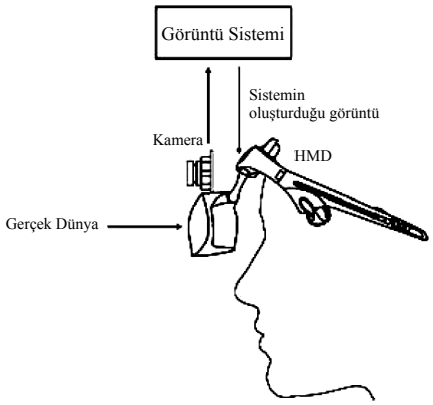
3.3 Falcon Yapısı

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 matrisel (düzlemsel) olması açısından CASTLE'a benzemektedir. Bu yapının, Virtex-300 FPGA tümdevresinde tek katmanlı HSA için çalışma hızı, 24 bit doğrulukta, işlem birimi başına 640×480 piksel bir görüntü için 178 iterasyon/s'dir. Bu çalışmada karşılaştırma amacı ile verilen CASTLE yapısının hızı ise işlem birimi başına 640×480 piksel bir görüntü için bir saniyede 135 iterasyon/s'dir (Nagy ve Szolgay, 2002, 2003). Beke vd. (2004), Vörösházi vd. (2006) ve Kincses vd. (2006) yayımları Falcon yapısı üzerinde yapılan gelişmeleri ele almaktadır ve Falcon yapısının daha iyi anlaşılması amacıyla incelenebilir.

3.4 Diğer Bazı Dijital HSA Emülatörü Yapıları

2005 yılında yayımlanan Toledo vd.'nin (2005a; 2005b) çalışmalarında bir HSA uygulaması anlatılmaktadır. Uygulamada görme alanında daralma kusuru olan göz hastaları için kullanılmak üzere tasarlanmış bir sistem geliştirilmiştir. Bu sistemde hasta bir HMD (Head Mounted Display) takmaktadır. HMD ufak saydam bir ekrandır ve hasta bu ekranın saydamlığı sayesinde hem gerçek dünyayı (baktığı yeri), hem de ekran üzerinde oluşturulan görüntüyü görür. Sistemde bir kameradan alınan görüntü, dış RAM kullanan bir FPGA üzerinde HSA ile işlenip (kenar belirleme) HMD üzerinde gösterilir. Böylece hasta, görme alanı içerisinde hem baktığı dar alanı hem de geniş alandaki cisimlerin kenar belirlenmiş halini görür. Bu durum hastaların yaşam kalitesini arttırmakta ve günlük işlerini daha kolay

yapmalarını sağlamaktadır. Özetle sistem, hastaların dar bakış alanı içerisine geniş görüntü alanının kenar belirlenmiş halini yerleştirir. Sistemin genel yapısı Şekil 3.8'de, hastaların kısıtlı görüş alanlarıyla çevrelerini nasıl gördükleri Şekil 3.9'da, hastaların HDM üzerinden çevrelerini nasıl gördükleri ise Şekil 3.10'da gösterilmiştir (Toledo vd., 2005a; 2005b).



Şekil 3.8 Göz hastaları için tasarlanmış sistemin genel yapısı (Toledo vd., 2005a; 2005b).

Şekil 3.9 Hastaların kısıtlı görüş alanıyla çevrelerini görüş şekli (Toledo vd., 2005a; 2005b).

Şekil 3.10 Hastaların HDM üzerinden çevreleri görüş şekilleri (Toledo vd., 2005a; 2005b)

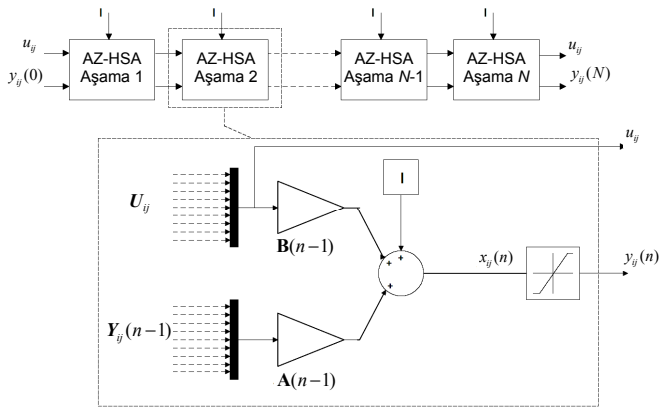
Toledo vd.'nin (2005b) çalışması, amaç ve kullanılan donanım olarak Toledo vd.'nin (2005a) çalışması ile aynı olup, sadece kullanılan ayrık zamanlı HSA (AZ-HSA) (Discrete Time CNN

(DT-CNN)) modeli farklıdır. Toledo vd.'nin (2005b) çalışmasında kullanılan AZ-HSA modeli, Martínez vd.'nin (2007) çalışmasında kullanılan modelle aynıdır. Bu model TSA ile AZ-HSA modeline benzemektedir. TSA ile AZ-HSA modeli bölüm 2.4.1'de anlatılmıştır.

Martínez vd.'nin (2007) yayımında, video işaretlerini dış RAM kullanmadan, FPGA üzerinde, HSA ile gerçek zamanlı olarak işleyen bir donanım yapısı önerilmektedir. Önerilen yapı Xilinx firmasının XC4VLX25-11 (Virtex-4) FPGA tümdevresi üzerinde gerçeklenmiş ve 640×480 piksel 78 çerçeve video işaretini gerçek zamanlı olarak işleyebileceği belirtilmiştir. Önerilen işlem birimi yapısı, video işaretini işlemek için iki adet blok RAM ve iki adet DSP48 bloğu kullanmaktadır. Bir pikselin bir sonraki durumunun hesaplanması için gereken saat frekansı, piksel frekansının 17 katıdır. İşlem birimi için FPGA'de kullanılan donanım miktarı (XC4VLX25-11 FPGA tümdevresi için) Çizelge 3.1'de, sistemin genel yapısı ise Şekil 3.12'de verilmiştir.

Çizelge 3.1 İşlem birimi için FPGA'de kullanılan donanım miktarı (Martínez vd., 2007).

FPGA Kaynaklar	Birim (Adet)	% Birim (Adet)
Alan (slice)	150	1.4
Flip-flop	172	0.8
DSP48	2	8
Blok RAM	2	2.78
Maks. iç frek. (MHz)	410	---
Maks. piksel frek. (MHz)	24.11	---

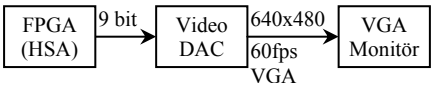


Şekil 3.12 Sistemin genel yapısı (Martínez vd., 2007).

Yapının bu çalışmadaki gerçeklemesinde bazı dezavantajlar mevcuttur. Bunlar, bir pikselin işlenmesi için gerekli olan saat darbesi sayısının 17 gibi yüksek bir değerde olması ve DSP48 bloğuna ihtiyaç duyulmasıdır. Bu iki dezavantaj, sistemin çalışması için Virtex-4 gibi hızlı ve pahalı bir FPGA tümdevresi kullanılmasını gerektirmektedir. Sistem DSP48 bloğu içermeyen düşük hızlı FPGA tümdevreleri (ör. Spartan-3, Virtex-II) için uygun değildir.

4. TEZ DAHİLİNDE GERÇEKLENEN İLK HSA EMÜLATÖRÜ

Bu tez çalışmasında gerçeklenen ilk HSA emülatörü, FPGA tümdevresinin içinde bulunan blok RAM modüllerine yüklenmiş (sınır koşuları dahil) 128×48 piksel 9 bit gri tonlamalı bir görüntüyü istenen iterasyon sayısı kadar işler ve çıkış görüntüsünü bir VGA monitöründe gösterir. Emülatör yapısı hızlı çalışması için tasarlanmamış olup sadece çalışan bir HSA emülatörü gerçeklenmesi, böylece emülatör tasarımı ve gerçeklemesiyle ilgili deneyim kazanılması hedeflenmiştir. Sistemin genel blok diyagramı Şekil 4.1'de verilmiştir.



Şekil 4.1 Gerçeklenen ilk emülatör sisteminin genel blok diyagramı.

4.1 Kullanılan Donanım

Sistem Xilinx firmasının Spartan-3 FPGA tümdevresini içeren "Spartan-3 Starter Kit" kartı üzerinde çalışacak şekilde tasarlanmaya başlanmış, ancak Xilinx firmasının Virtex-II FPGA tümdevresini içeren Celoxica firmasının "RC203" kartının temin edilmesiyle tasarım bu kart üzerinde çalıştırılmıştır. RC203 kartına ilişkin açıklamalar aşağıda verilmiştir.

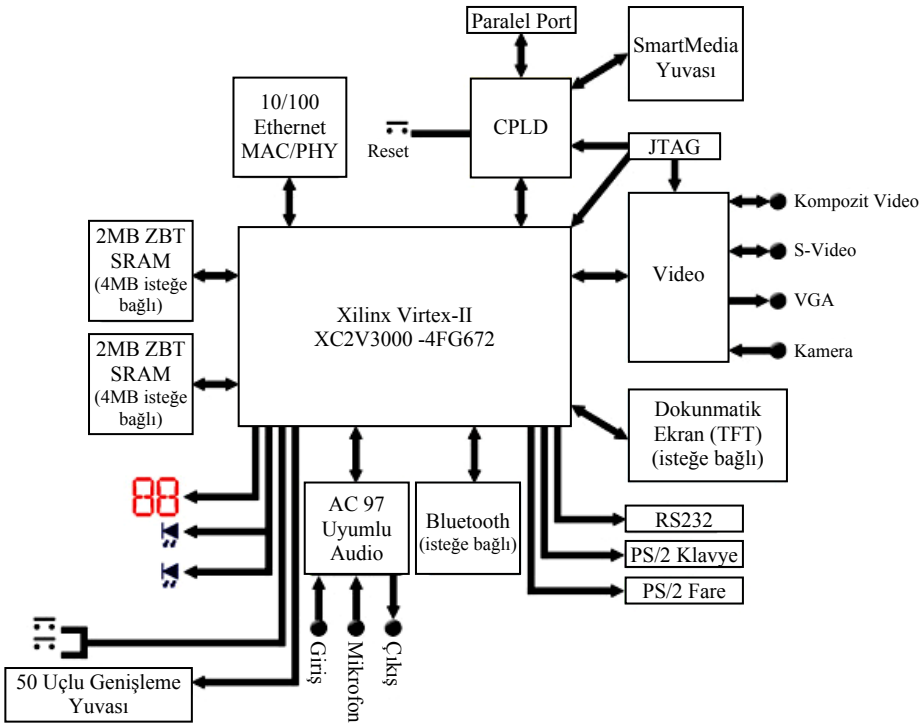
4.1.1 Celoxica RC203 FPGA Geliştirme Kartı

RC203 kartı, Xilinx Virtex-II 3000 FPGA tümdevresi, video ve audio giriş/çıkışları ve 100 Mbit Ethernet portuyla güçlü bir geliştirme ortamı sağlamaktadır. Kartın genel özellikleri aşağıda verilmiştir.

- Virtex-II XC2V3000-4 FPGA
- 10/100Mbit Ethernet
- Her biri 2 MByte olan 2 bank halinde, toplam 4 MByte SRAM
- Video desteği
  - Kompozit (composite) video giriş/çıkışı
  - S-Video giriş/çıkışı
  - VGA çıkışı
  - Kamera girişi
- AC'97 uyumlu audio
  - Mikrofon girişi

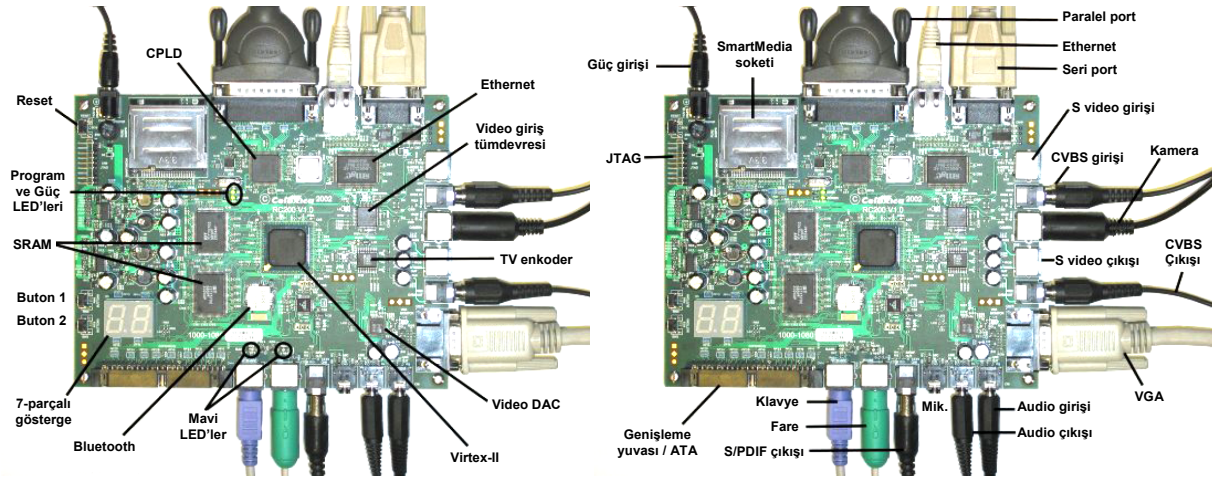
  - Stereo girişi (Line in)
  - Stereo (kulaklık) çıkışı (Line out)
- FPGA'in yapılandırılmasında (configuration) kullanılabilen SmartMedia Flash hafızası için soket (yuva)
- Yapılandırma/Yeniden yapılandırma ve SmartMedia kontrolü için kullanılan CPLD
  - Güç açıldığında SmartMedia'dan yüklenme
  - SmartMedia takıldığında yüklenme
  - İstendiği taktirde FPGA'in kendisini yeniden yapılandırabilmesi
- FPGA'in yapılandırılması ve PC ile haberleşmede kullanılan Paralel port
- RS232 portu
- PS/2 klavye ve fare (mouse) portları
- 2 adet 7-parçalı gösterge, 2 mavi LED
- 2 adet anlık bağlantı sağlayan anahtar
- 50 uçlu genişleme yuvası
  - 33 adet genel amaçlı giriş/çıkış ucu
  - 3 adet besleme ucu (+12V, +5V, +3.3V)
  - 2 adet saat işareti ucu
- JTAG bağlantısı

RC203 kartının blok diyagramı Şekil 4.2'de verilmiştir.



Şekil 4.2 RC203 kartının blok diyagramı.

RC203 kartının eleman ve bağlantı yerleşimi Şekil 4.3'de verilmiştir.



Şekil 4.3 RC203 kartının eleman ve bağlantı yerleşimi.

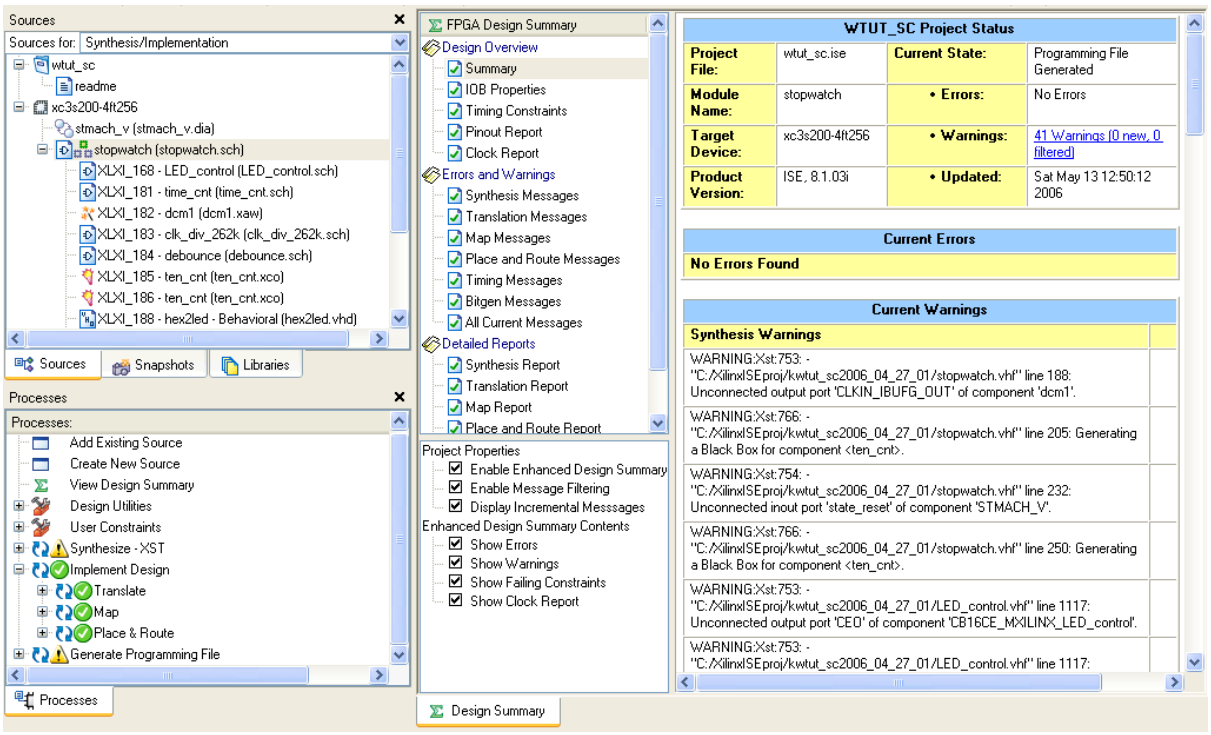
4.2 Kullanılan Yazılım

Sistemin tasarım ve simülasyonu için sadece Xilinx firmasının ISE 8.1i yazlımı kullanılmıştır.

4.2.1 ISE 8.1i Yazılımı

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 başına yeterli olmakla beraber, kendisine tümleştirilen yardımcı yazılımlarla beraber de çalışabilmektedir. Bu yazılımlara, simülasyon için kullanılan Mentor Graphics firmasının ürettiği ModelSim yazılımı örnek gösterilebilir.

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 4.4'de verilmiştir.



Şekil 4.4 ISE 8.1i yazılımının ekran görünüşü.

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 şematik editörü ve "CORE Generator" yazılımı kullanılarak oluşturulmuştur.

Projenin gerçeklenmesi üç ana aşama ile yapılır. Bunlar sentezleme (synthesize), gerçekleme (implement) ve programlama dosyası oluşturmadır. Bu aşamalar sıra ile işlem penceresinde görülmektedir.

Sentezleme işlemi yazılan kodların doğruluğunu denetler, HDL 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 HDL kodu "Xilinx Implementation" yazılımının anlayacağı bir EDIF (.edn) veya NGC (.ngc) devre dizgisi (netlist) dosyasına dönüştürülür.

Gerçekleme (Implement) işlemi üç aşamadan oluşmaktadır. Bunlar "Translate", "Map" ve "Place & Route" işlemleridir. "Translate" işlemi, tasarımdaki devre dizgisi (netlist) dosyalarını tek bir NGD devre dizgisi dosyasında birleştirir. Oluşturulan bu yeni dosya,

tasarımın yanısıra zamanlama ve yerleşim kısıtlarını (constraints) da içerir. Kısıtlara kullanıcı kısıtları (user constraints) dosyası (UCF) da dahildir. Ayrıca zamanlama ve lojik tasarım kurallarının doğruluğu da bu aşamada denetlenir. "Map" işlemi, tasarımdaki her bir lojik birim için gerekli olan CLB'leri ve IOB'ları paylaştırır (allocate). "Place & Route" işlemi, kullanılacak CLB'lerin ve IOB'ların yerlerini ve aralarındaki bağlantıların hangi hatlar üzerinden gerçekleştirileceğini tam olarak belirleyerek, tasarımı FPGA üzerindeki son haline getirir.

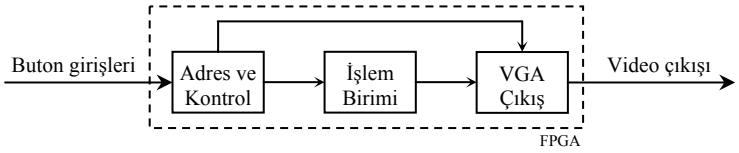
Programlama Dosyası Oluşturma işleminde, FPGA'in programlanması için kullanılacak olan BIT dosyası ve bu dosyanın PROM içine yüklenecek hali olan MCS dosyası oluşturulur.

4.3 Tez Dahilinde Gerçeklenen İlk HSA Emülatörünün Yapısı

Donanım yapısının tasarımında herhangi bir kaynaktan alınan bir HSA emülatörü yapısı kullanılmamış olup tasarım tamamen özgündür. Tasarlanan donanım yapısı üç blok halinde incelenebilir:

1. Adres ve Kontrol bloğu (3×3 **A** ve **B** şablonları için),
2. İşlem Birimi bloğu (3×3 **A** ve **B** şablonları için),
3. VGA Çıkış bloğu.

FPGA içerisindeki yapının genel blok diyagramı Şekil 4.5'de verilmiştir.



Şekil 4.5 FPGA içerisindeki yapının genel blok diyagramı.

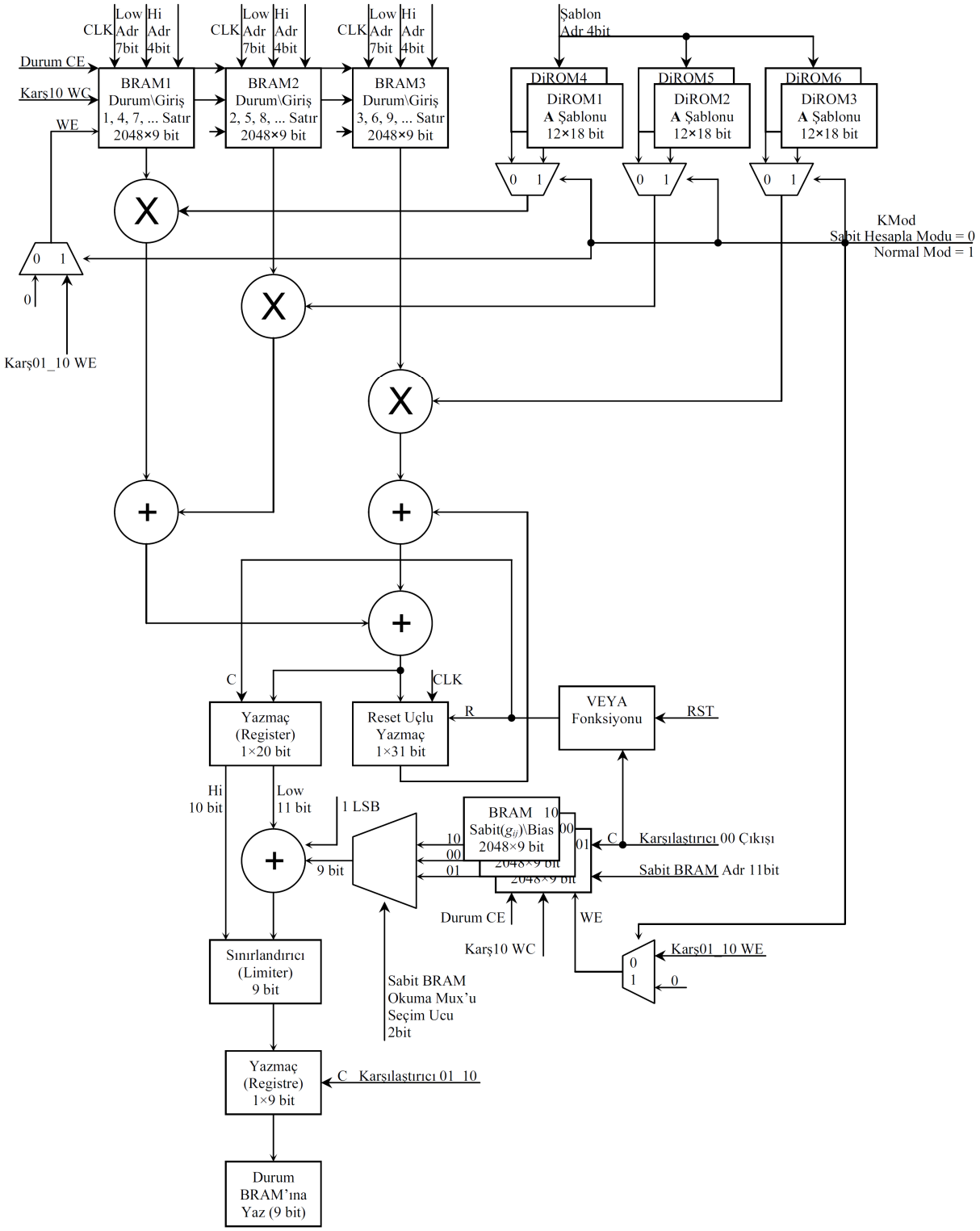
Adres ve Kontrol bloğu, İşlem Birimi bloğuna iletilecek olan verilerin adres değerlerinin üretilmesi, işlem süresinin (iterasyon sayısı) kontrolü, "Reset" ve "Tekrar" butonu girişlerinin değerlendirilmesi, İşlem Birimi bloğunun çıkışındaki verilerin saklanması için gerekli olan adres ve işaretlerin üretilmesinde kullanılır.

İşlem Birimi bloğunun iki görevi vardır. Bunlardan ilki FPGA içindeki blok RAM'larda bulunan görüntü bilgisi ve eşik (bias) değerini kullanarak o giriş görüntüsü için sabit olan g_{ij} değerlerini hesaplamak ((2.17.b) denklemi). Bu işlem tekrarlı (iteratif) bir işlem olmayıp değerler $\hat{\mathbf{B}}$ şablonunun giriş görüntüsü üzerinde gezdirilmesi ve eşik değerlerini kullanılması

ile elde edilen (her piksel için ayrı) sabit değerlerdir. Hesaplanan g_{ij} değerleri, FPGA'deki blok RAM'larda bulunan eşik değerlerinin üzerine yazılır. İşlem Birimi bloğunun yaptığı ikinci iş, önceden hesapladığı sabit (g_{ij}), o anki durum ve $\hat{\mathbf{A}}$ şablonu değerlerini kullanarak yeni durum değerini hesaplamaktır ((2.17.a) denklemi). Bu hesaplama yönteminin ayrıntıları 2.3 bölümünde verilmiştir.

Tasarlanan donanımda, FPGA tümdevresinin içinde bulunan blok RAM'lardan (BRAM) 6 adeti kullanılmaktadır. Bu BRAM'ların üç tanesi durum\giriş değerlerini, diğer üç tanesi ise o giriş görüntüsü için kullanılacak olan sabit\eşik (g_{ij} veya bias) değerlerini taşır. Durum\Giriş değerlerini taşıyan BRAM'lardaki durum dizilişi üçer satır atlayarak gider. Örneğin, birinci BRAM'da 1, 4, 7, 10, …; ikinci BRAM'da 2, 5, 8, 11, …; üçüncü BRAM'da 3, 6, 9, 12, … (görüntü) satırları bulunmaktadır. Bu diziliş sabit\bias BRAM'ları için de aynı şekildedir. Verilerin diziliş şekli aynı anda üç adet çarpma işlemi yapma imkanı sağlar. Sınır koşulları, durum\giriş verisinin içine dahildir ve sabit BRAM'ında da aynen mevcuttur. Yani sabit ve durum BRAM'larındaki veri dizilimi aynı şekildedir (adrestedir). Adres ve Kontrol bloğunun blok diyagramı Ek 1'de, İşlem Birimi bloğunun blok diyagramı ise Şekil 4.6'da verilmiştir.

İşlem Birimi bloğunda, şablon değerleri FPGA üzerindeki lojik bloklar (CLB) kullanılarak oluşturulmuş 6 ayrı ROM'dan (distributed ROM (DiROM)) okunur. Bu ROM'ların üç tanesi (DiROM4-6) sabitlerin (g_{ij}) hesaplanması için kullanılan $\hat{\mathbf{B}}$ şablonunun, diğer üçü (DiROM1-3) ise yeni durumların hesaplanması için kullanılan $\hat{\mathbf{A}}$ şablonunun değerlerini içerir. Bu iki şablonun her birinde 9 adet değer bulunmaktadır ve üçlü gruplardaki her bir ROM (içinde bulunduğu gruba göre) $\hat{\mathbf{B}}$ veya $\hat{\mathbf{A}}$ şablonunun bütün (9) değerlerini içerir. Ancak her bir ROM'daki veri dizilimi (adres yerleri) farklıdır. Bu fark, aynı şablon değerinin işlenen pikselin bazen üst satırına, bazen aynı satırına, bazen de alt satırına gelmesinden, yani şablonun görüntünün her satırında kaydırılmasından kaynaklanır. Ayrıca ROM'lar her üç değerde bir kullanılmayan bir değer içerir. Örneğin 0, 1, 2 adreslerindeki değerler kullanılırken 3 adresinin değeri kullanılmaz. Bunu nedeni ROM adresinin yüksek ve düşük ağırlıklı ikişer bitten oluşturulmasının kontrolde kolaylık sağlamasıdır. Bu yaklaşımla yüksek iki birlik adres bir arttırıldığında, ROM çıkışında şablon matrisinin bir sonraki satırındaki değer gözlenir. Böylece ROM adresindeki yüksek ağırlıklı iki bit şablonun hangi satırındaki, düşük ağırlıklı iki bit ise şablonun hangi sütunundaki verinin okunacağını belirlemiş olur. Bu yöntemde her şablon (9 veri) 12 adres alanı yer kaplar. Şablon ROM'larındaki (DiROM) veri dizilimi 5. bölümde ayrıntılı olarak anlatılmış ve gösterilmiştir (Çizelge 5.1).



Şekil 4.6 İşlem Birimi bloğunun blok diyagramı.

Adres ve Kontrol bloğundaki “KMod” çıkışı İşlem Birimi’nin sabit değerini mi yoksa durum değerini mi hesaplayacağını kontrol eder. Sabit değerleri sadece “Reset” butonuna basıldıktan sonra bir kere hesaplanır (KMod=0) ve Sabit BRAM’larına yazılır. Sabit hesaplama işleminde durum BRAM’larında giriş görüntüsü, sabit BRAM’larında ise eşik değer(ler)i mevcuttur. Hesaplama sonunda Sabit BRAM’ları g_{ij} sabitlerini, durum BRAM’ları ise ilk durumları (giriş görüntüsü ile aynı) saklamaktadır. Sonraki iterasyonlarda (KMod=1) ise yeni durum değerleri hesaplanır ve önceden belirlenmiş bir iterasyon sayısı sonunda hesaplama işlemi durur. Sonuçlar (9 bit) durum BRAM’larında saklanır ve VGA monitöründe gösterilir. Eğer işlem bittikten sonra “Tekrar” butonuna basılırsa iterayon işlemi (KMod=1) kaldığı yerden, tekrar önceden belirlenmiş iterasyon sayısı kadar devam eder ve yeni sonuçlar VGA monitöründe gösterilir.

Tez dahilinde tasarlanan ilk HSA emülatörü yapısı 5. bölümde anlatılan gelişmiş HSA emülatör yapısı için bir hazırlık niteliğinde olduğundan daha ayrıntılı olarak açıklanmamıştır. Ayrıntılı açıklamalar 5. bölümde yapılmıştır.

5. GELİŞTİRİLMİŞ HSA EMÜLATÖR

Geliştirilmiş HSA emülatöründe ilk tasarlanan yapıdaki işlem birimi ardışık düzen (pipeline) yapısında çalışacak hale getirilerek hızı arttırılmış, İşlem birimlerinin kaskad bağlanabilmesi sağlanmış ve sistem VGA işareti alabilecek hale getirilmiştir. Bu yapı ayrıca FPGA üzerinde gerçeklenmiş ve çalışması bir kenar belirleme uygulamasında gözlenmiştir. 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. Sistemin genel blok diyagramı Şekil 5.1'de verilmiştir.

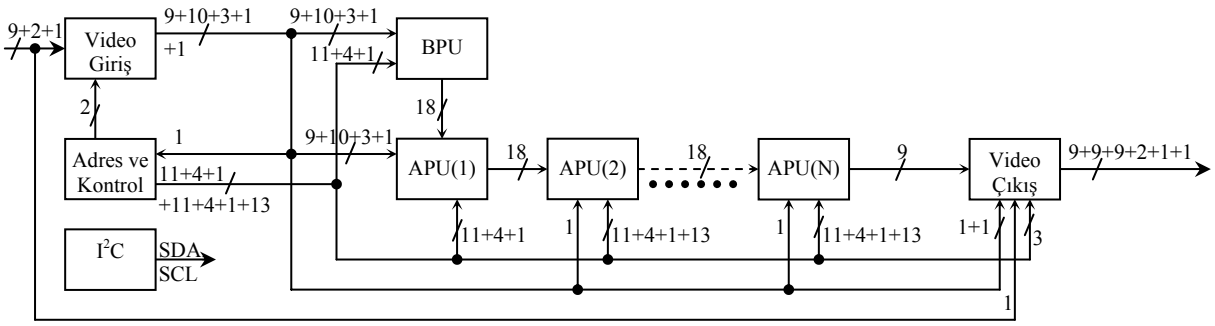
Şekil 5.1 Geliştirilmiş HSA emülatörü sisteminin genel blok diyagramı.

5.1 Kullanılan Yazılımlar

Bu tasarımın geliştirilmesinde ilk HSA emülatörü yapısında kullanılan ISE yazılımına ek olarak Xilinx firmasının "ChipScope Pro" adlı yazılımı kullanılmıştır. Bu yazılım FPGA üzerinde bir lojik analizör bloğu oluşturarak, devre çalışırken oluşan dijital işaretlerin bilgisayar ekranında gözlenmesini sağlar, böylece tasarlanan devrede hata ayıklama (debug) işlemi kolaylaşır ve ayrıntılı simülasyonlar yapmak için zaman harcanmasına gerek kalmaz.

5.2 Geliştirilmiş HSA Emülatörü yapısı

FPGA içerisindeki devrenin blok diyagramı Şekil 5.2'de verilmiştir. Blok diyagramda bloklar arasındaki veri yolu genişlikleri ve yönleri de görülmektedir. Bloklara ilişkin açıklamalar diyagramdan sonra verilmiştir.



Şekil 5.2 FPGA içerisindeki devrenin blok diyagramı.

5.2.1 Video Giriş Bloğu

"Video Giriş" bloğu video piksel verisinin "BPU" (B şablonu işlem birimi) bloğuna yazılması, "Adres ve Kontrol" bloğunun çalışmasının kontrolü (enable) ve "Video Çıkış" bloğunun senkronizasyonu için kullanılır. Blok girişinde 9, 2, 1 olarak görülen bağlantılar sırasıyla, video ADC den gelen video piksel verisi (9) (bu veri sadece VGA işaretinin yeşil renk kanalından gelen veridir), referans (senkronizasyon) işaretleri (2) (VGA Horizontal ref. ve Vertical ref.) ve videonun piksel saati işaretidir (1). Bloğun altında bulunan 2 genişlikli giriş ise şablon senkronizasyon (1) ve PU pasif (Processing Unit disable) girişleridir. Bloğun çıkışındaki 9, 10, 3, 1, 1 bağlantıları ise sırasıyla video piksel verisi (9), verinin yazılacağı RAM adresi (10), verinin yazılacağı blok RAM'ların yazma aktif (write enable) (3), PU aktif (enable) (1) ve "Video Çıkış" bloğu senkronizasyon (1) işaretleridir.

5.2.2 Adres ve Kontrol Bloğu

"Adres ve Kontrol" bloğu PU (Processing Unit) bloklarının çalışmasını kontrol eder. Bu bloğun çıkışındaki 11, 4, 1, 11, 4, 1, 13 bağlantıları sırasıyla BPU bloğundaki RAM'ın okuma adresi (11), **B** şablonu (DiROM) okuma adresi (4), Mux seçim (1) uçlarıdır. Bu uçlardan sonraki 11, 4, 1, uçlarındaki işaretler bu uçlardaki işaretlerin sadece 6 saat darbesi geciktirilmiş halleridir ve APU(n) bloklarındaki RAM'ın okuma adresi (11), **A** şablonu (DiROM) okuma adresi (4), Mux seçim (1) uçlarıdır. En sondaki 13 bitlik bağlantı ise APU(n) blokları arasında iletilen verilerin yazılacağı APU(n>1) içindeki blok RAM'ların yazma aktif (write enable) (3) ve yazma adresi (10) uçlarıdır. Bu bloğun üstünden çıkan 2 çıkış bağlantısının ne olduğu "Video Giriş" bloğunda açıklanmıştır. "Adres ve Kontrol" bloğunun içyapısı ayrıntılı olarak 5.4 bölümünde anlatılmıştır.

5.2.3 BPU ve APU(n) Blokları

Bu bloklar ayrık zamanlı HSA (DT-CNN) emülasyonu yapan işlem birimi bloklarıdır. BPU bloğu "Video Giriş" bloğundan aldığı video girişini **B** şablonu ile işleyerek elde ettiği 18 bitlik sayıyı (sabit (g_{ij})) APU(1) bloğuna iletir. Bu bloktan bir tane olma nedeni hesaplanan değerlerin (g_{ij}) 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 iteasyon 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. Ne kadar iterasyon yapılması gerekiyorsa, o kadar APU sisteme eklenebilir. APU(n)'ler arasında durum (9 bit) ve sabit (18 bit) iletimi söz konusudur. APU(n>1)'ler içindeki blok RAM'lar hem durum hem de sabit verilerini saklar. APU(1) ve BPU ise blok RAM'ları içinde sırasıyla sadece ilk durum (9 bit) ve video görüntüsü (9 bit) verilerini saklar. Bu blokların içyapıları ayrıntılı olarak 5.3 bölümünde anlatılmıştır.

5.2.4 Video Çıkış Bloğu

"Video Çıkış" bloğu işlenmiş görüntüyü 640×480 piksel 60 fps VGA işareti formatına uygun bir şekilde video DAC'a gönderir. Bloğun çıkışındaki 9, 9, 9, 2, 1, 1 uçları sıra ile kırmızı (9), yeşil (9) ve mavi (9) renk uçları olup çıkışın gri tonlamalı olması nedeniyle aynı değerdedir. Sonraki iki uç (2) referans (senkronizasyon) işaretleri (horizontal ref., vertical ref.), sonraki bir uç piksel saati (1), diğer bir uç da boşluk (1) (blank) işaretidir. Boşluk işareti aktif video işareti dışında kalan bölgelerde aktif edilerek DAC çıkışlarını lojik 0 yapar. Bloğun girişine ise PU aktif (1), video çıkış senkronizasyon (1), piksel saati (1) ve 13 bitlik yolun üç adet ucu bağlıdır. Bu üç uç herhangi bir APU(n>1) bloğundaki üç adet blok RAM'a giden yazma aktif (write enable) (3) uçlarıdır.

5.2.5 I^2C Bloğu

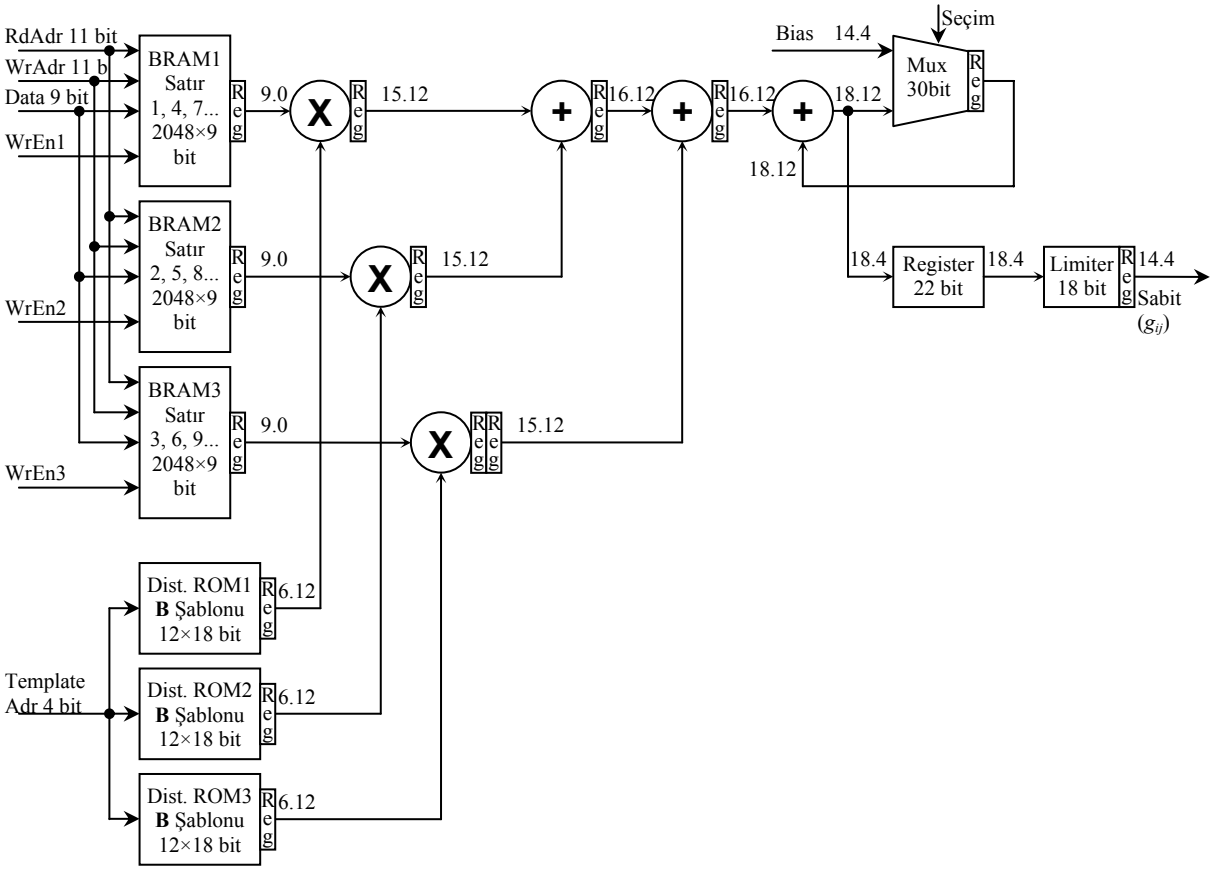
Video ADC tümdevresinin çalışmaya başlaması için tümdevre içindeki yazmaçlara (register) bazı çalışma parametrelerinin yazılması gerekir. Bu parametrelerin yazımı için ise I^2C seri veri aktarım protokolü kullanılmalıdır. Bu blok, I^2C protokolünü kullanılarak ADC'nin yazmaçlarına değer atama işlemini gerçekleştirir. Bu işlem, sistem çalışmaya başlarken sadece bir kez yapılır.

5.3 BPU ve APU(n) İşlem Birimi Bloklarının İç Yapısı

BPU, APU(1) ve APU(n>1) bloklarının içyapılarını gösteren blok diyagramlar ve açıklamaları bu bölümde anlatılacaktır.

5.3.1 BPU Bloğu

BPU (B şablonu işlem birimi) bloğu g_{ij} değerlerinin hesaplanması amacıyla kullanılır (denklem 2.17.b). BPU bloğunun blok diyagramı Şekil 5.3'de görülmektedir.



Şekil 5.3 BPU bloğunun blok diyagramı.

BPU ardışık düzen yapısıyla çalışacak şekilde tasarlanmıştır. En sağdaki toplayıcı bloğu dışındaki tüm blokların çıkışı yazmaçlıdır (registered). Bu toplayıcı bloğunun bağlı olduğu Mux yazmaçlı olduğundan toplayıcının çıkışında yazmaç kullanılmamıştır.

Şekil 5.3'de görülen üç adet blok RAM (BRAM) giriş görüntüsünün üç satırını saklar ve bu değerleri çarpıcılara iletir. Giriş görüntüsünün üç satırından oluşan görüntü şeridinin her bir satırı bir BRAM'a yazılır: ilk satır BRAM1'e, ikinci satır BRAM2'ye, üçüncü satır BRAM3'e... İlk iki satır ve üçüncü satırın ilk 3 pikseli BRAM'lara yazıldıktan sonra işlem birimi çalışmaya başlar. Üçüncü satır BRAM'ın B portundan yazılırken aynı zamanda da işlem birimi BRAM'ların A portundan veri okuyarak çalışmaya devam eder (BRAM'lar iki portludur (dual port)). Üçüncü satır BRAM3'e yazıldıktan sonra, gelen dördüncü satır BRAM1'e, birinci satırın üzerine yazılır. Bu yazma işlemi yapılmaya başladığı zaman artık birinci satırın değerlerine ihtiyaç kalmamıştır. Veri yazma işlemi bu şekilde devam eder, yani

her BRAM görüntünün sadece bir satırını saklar ve yeni gelen satırlar bu verilerin üzerine yazılır. Zaman içinde BRAM1 görüntünün 1, 4, 7, …; BRAM2 görüntünün 2, 5, 8, …; BRAM3 görüntünün 3, 6, 9, … numaralı satırlarını saklar ve zamanı geldiğinde sakladıkları bu verileri çarpıcılara iletir. İletilen piksel verisinin bit genişliği ve noktanın bulunduğu yer Şekil 5.3'de de görüldüğü gibi 9.0, yani noktadan önce 9 bit noktadan sonra 0 bit şeklindedir. Her bir BRAM'ın sakladığı piksel sayısı VGA görüntüsünün satır piksel sayısı (640) artı 2 sınır koşuludur, sistemin yapısından dolayı her sınır koşulu iki görüntü pikseli yer kaplamaktadır. Sonuç olarak her BRAM'da sınır koşulları dahil 644×9 bit veri saklanmaktadır. Her bir BRAM'ın kapasitesi 2048×9 bittir. Bu durumda belleklerin %68.5'i kullanılmamaktadır. Bu bir dezavantaj ve düzeltilmesi gereken bir durum gibi görünmesine rağmen FPGA üzerindeki blok RAM ve çarpıcı sayıları birbirine eşit ve yapıda kullanılan blok RAM ve çarpıcı sayısı da birbirine eşit olduğu için ciddi bir dezavantaj oluşturmamaktadır. Örneğin, giriş görüntüsünü daha az sayıda BRAM kullanarak sakladığımızı düşünelim. İşlemlerin üç saat darbesinde sonuçlanması için üç adet çarpıcı kullanılacaktı ve kullanılmayan BRAM'lar çarpıcı kalmadığı için sistemde başka bir yerde kullanılamayacaktı. Ayrıca bu durum sadece BPU ve APU(1)'de mevcut olup APU(n>1) bloklarındaki BRAM'lar hem durum (644×9 bit) hem de görüntünün 18 bit sabit (g_{ij}) (640×18 bit) değerlerini sakladıkları için APU(n>1) bloklarındaki belleklerin (BRAM'ların) 1924×9 bitlik bölümü (%93.9'u) kullanılmaktadır. Her BRAM için kullanılan okuma ve yazma adresleri BRAM'lar arasında aynıdır.

Video çerçevesinin sağındaki ve solundaki sınır koşulları BRAM'ların ilk değerlerinin (initial value) istenen herhangi bir sabit değere atanması ile elde edilir (bölüm 2.4.1 "Dirichlet Sınır Koşulu"). Böylece, VGA görüntü satırı BRAM'a yazıldığında (ör. 2 adresinden başlayarak) satırın sağındaki ve solundaki sınır hücrelerinin değerleri BRAM'ın ilk değerlerine eşit olur. Video çerçevesinin üstündeki ve altındaki sınır koşulları ise, VGA işaretinin aktif görüntü alanı içerisindeki ilk satırından önce ve son satırlarından sonra gönderdiği siyah satırların değerlerine eşittir (9 bit işaretli değer için -256 (siyah)). Üst ve alt sınır koşullarının gerçeklenen sistemde değiştirilememesine rağmen sistemde yapılacak bazı küçük geliştirmelerle bu sınır değerlerinin de değiştirilebilmesi sağlanabilir. Bunu gerçekleştirmek için, aktif görüntü alanı başlamadan önceki ve bittikten sonraki satır değerinin VGA işaretinden değil FPGA içerisinde bir yazmaçtan alınmasını sağlamak yeterlidir.

HSA'da kullanılan **B** şablonu üç adet 12×18 bit Distributed ROM (DiROM) yapısı içinde saklanmaktadır. Her ROM **B** şablonundaki 9 adet sayıyı 6.12 (noktadan önce 6 noktadan

sonra 12 bit) formatında 18 bit olarak saklar. Yani her ROM'un sakladığı sayılar aynıdır ancak adres sıralamaları farklıdır (Çizelge 5.1). BRAM'lardaki piksel değerleri ile çarpılacak olan DiROM1, DiROM2 ve DiROM3'deki **B** şablonu değerleri sıra ile **B** şablonunun 1., 2., ve 3. satırlarının ilk değeri ($b_{-1,-1}$, $b_{0,-1,}$ $b_{1,-1}$) ile başlar. Böylece, örneğin DiROM 0 adresi tüm DiROM'lara uygulandığında 1., 2. ve 3. DiROM uçlarında **B** şablonunun 1., 2. ve 3. satırının ilk değerleri (1. sütün değerleri $b_{-1,-1}$, $b_{0,-1,}$ $b_{1,-1}$) okunur ve piksel değerleri (1. sütun) ile çarpılır. Bir sonraki saat darbesiyle DiROM ve BRAM adresleri bir artırıldığında, DiROM'ların çıkışlarında **B** şablonunun 1., 2. ve 3. satırlarının 2. değerleri (2. sütun değerleri $b_{-1,0}$, $b_{0,0,}$ $b_{1,0}$) görülür ve yeni piksel değerleri (2. sütun) ile çarpılır. Bir sonraki saat darbesiyle DiROM ve BRAM adresleri bir artırıldığında, DiROM'ların çıkışlarında **B** şablonunun 1., 2. ve 3. satırlarının 3. değerleri (3. sütun değerleri $b_{-1,1}$, $b_{0,1,}$ $b_{1,1}$) görülür ve yeni piksel değerleri (3. sütun) ile çarpılır. Bir sonraki saat darbesiyle DiROM adresi tekrar 0 yapılır (1. sütun), BRAM adresi ise bir azaltılır (2. sütun) ve bir sonraki piksel değeri için hesaplama işlemine başlanır. Bütün bir satır için çarpma işlemleri bu şekilde devam eder. Satır bitince ve yeni satırın ilk üç pikseli BRAM1'e yüklendiğinde, bir sonraki satır işlenmeye başlar ve DiROM'lar, satır boyunca 4, 5 ve 6 adreslerinden (döngü içinde) okunmaya devam edilir. Yine tüm satır bittiğinde ve yeni satırın ilk üç pikseli BRAM2'ye yüklenince DiROM'lar, satır boyunca 8, 9 ve 10 adreslerinden (döngü içinde) okunmaya devam edilir. Bu satır da bittiğinde DiROM'lar, satır boyunca tekrar 0, 1 ve 2 adresinden okunmaya başlanır. Bu hesaplama şekli böyle devam eder ve bütün piksel değerleri gerekli **B** şablon değeri ile çarpılır. Bu hesaplama yöntemi, durum değerleri ve **A** şablonu ile APU blokları içerisinde de aynen uygulanır (Şekil 5.6).

DiROM'lar az yer kapladığı için bütün PU birimlerinde olmalarına rağmen, eğer istenirse tek bir üçlü DiROM gurubu konularak çıkışları PU bloklarına dağıtılabilir. DiROM ve BRAM'da saklanan tüm değerler işaretli ve sabit noktalı (fixed point) sayılardır. DiROM'lardaki **B** şablonu değerlerinin adreslere göre yerleşimi Çizelge 5.1'de verilmiştir.

Çizelge 5.1 **B** şablonu değerlerinin DiROM'lardaki yerleşimi.

Adres	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**
DiROM1	$b_{-1,-1}$	$b_{-1,0}$	$b_{-1,1}$	Boş	$b_{1,-1}$	$b_{1,0}$	$b_{1,1}$	Boş	$b_{0,-1}$	$B_{0,0}$	$b_{0,1}$	Boş
DiROM2	$b_{0,-1}$	$b_{0,0}$	$b_{0,1}$	Boş	$b_{-1,-1}$	$b_{-1,0}$	$b_{-1,1}$	Boş	$b_{1,-1}$	$B_{1,0}$	$b_{1,1}$	Boş
DiROM3	$b_{1,-1}$	$b_{1,0}$	$b_{1,1}$	Boş	$b_{0,-1}$	$b_{0,0}$	$b_{0,1}$	Boş	$b_{-1,-1}$	$b_{-1,0}$	$b_{-1,1}$	Boş

Çarpıcı olarak FPGA içindeki tümleşik çarpıcı blokları kullanılmıştır. Şekil 5.3'de görülen her üç çarpıcı da aynı anda çalışır ve BRAM'dan ve DiROM'dan okunan değerleri işaretli olarak çarpar. En alttaki çarpıcının çıkış değeri, diğer çarpıcılara göre bir ardışık düzen evresi (stage) sonra toplamaya katıldığı için bir saat darbesi fazla geciktirilmiştir.

Toplayıcılar çarpıcıların çıkışlarını toplamak amacıyla kullanılmıştır. Her bir saat darbesinde üç adet çarpıcının çıkışı toplanır ve sonuç ortadaki toplayıcının çıkışında görülür. Bir piksel için yapılan hesaplama (şablon nokta çarpımı) üç saat darbesinde bitmektedir (9 çarpma ve 9 toplama işlemi).

30 bitlik yazmaçlı Mux (registered multiplexer) hesaplama işleminin bulunduğu duruma göre ya eşik (bias) veya ara toplam değerlerini saklar. Son toplam en sağdaki toplayıcı bloğunda oluştuğunda bir sonraki saat darbesi ile toplayıcıdaki veri 22 bitlik yazmaca aktarılırken Mux çıkışlarına (yeni pikselin) eşik değeri yüklenir ve böylece yeni piksel için hesaplama işlemi başlamış olur. Eşik değeri yüklenirken eşiğin en düşük ağırlıklı bitinin yanındaki düşük ağırlıklı bit sabit 1 yapılarak toplama sonucunun yuvarlanması işlemi de toplama sırasında otomatik olarak gerçekleştirilmiş olur. Bir piksel için hesaplama işlemi üç saat darbesinde tamamlanır. Bu nedenle PU saat işareti piksel frekansının üç katı civarında olmalıdır.

Elde edilen toplama sonucu son olarak sınırlayıcı (limiter) bloğundan geçirilerek, eğer sonuç 14.4 bit ile ifade edilebilecek üst ve alt sınırların dışında ise, bu sınır değerlerine çekilir (maksimum veya minimum 14.4 bit işaretli değer). Sınırlayıcıdan çıkan 14.4 bitlik değer (g_{ij}) APU(1)'in sabit girişinden 30 bit Mux'un yazmacına ve APU(2)'nin içindeki BRAM'a yazılır. Hesaplanan bu değerler (g_{ij}) değiştirilmeden APU(n)'ler arasında aktarılır. Bir sabite (g_{ij} 'ye) ilişkin piksel işlenirken APU(n)'in kendi 30 bitlik Mux'una ve APU(n+1)'in BRAM'ına o sabitin (g_{ij}) değeri aktarılır.

BPU 9.0 bitlik piksel, 6.12 bitlik **B** şablonu değerleriyle işlem yapmaktadır. Piksel değerleri [-256, 255] (100000000, 011111111), şablon değerleri ise [-32, 31.999755859375] (100000.000000000000, 011111.111111111111) arasında değer alabilmektedir. Pikseller tam sayı, şablon değerleri ise 0.000244140625 (0.000000000001) adımlarla değer alabilmektedir. Şablon değerlerindeki bu adım aralığı belirlenirken, **B** şablonunun tüm değerleri (9 adet değer) bir adım aralığı değiştirildiğinde (yükseltildiğinde veya düşürüldüğünde) en kötü durumda dahi (piksel değerleri -256) sonuç değerin değişiminin birden küçük olması hedeflenmiştir. Örneğin, işlenen piksel ve etrafındaki sekiz piksel -256 olsun, şablondaki

değerler de 0.000244140625 değiştirilsin. Bu durumda bir pikselin sonuca olan katkısı 0.0625 (-256×-0.00024414065), 9 pikselin katkısı ise toplamda 0.5625 (9×0.0625) olacaktır. Görüldüğü gibi sonuçtaki bu değişim birden küçüktür ve eğer noktadan sonraki bit sayısı bir eksiltilirse, iki katına çıkarak (1.125) birden büyük bir sayı olacaktır. Eğer uygulamada bu kadar küçük adım aralığına ihtiyaç duyulmuyorsa şablon değerlerini ifade eden bit sayısı düşürülebilir veya noktanın yeri değiştirilebilir.

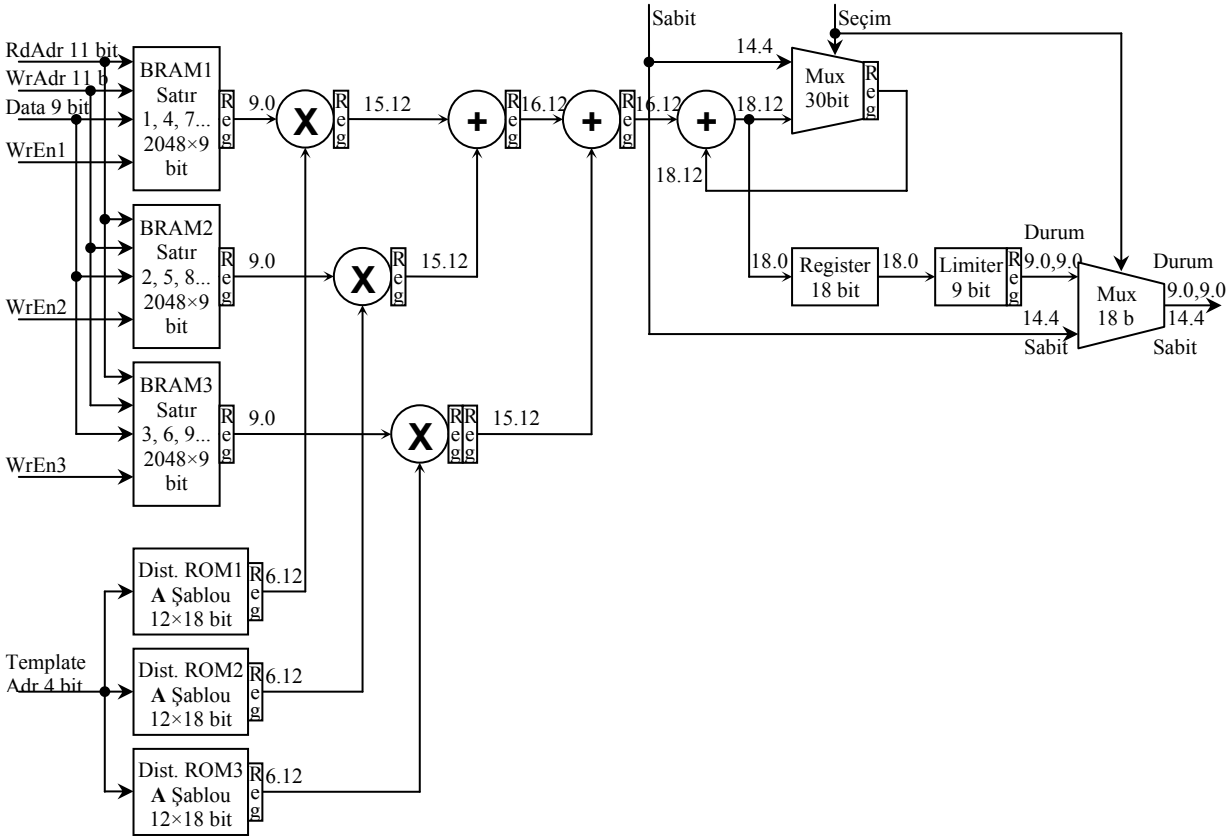
Her bir çarpıcının çıkışında oluşabilecek maksimum değer 8192'dir (-256×-32). Bu değer işaretli olarak noktadan önce 15 bitle ifade edilebilir. Noktan sonraki 12 bitle beraber bu değer toplamda 27 bit olarak 15.12 formatında (010000000000000.000000000000) yazılabilir. Soldan birinci toplayıcı iki çarpıcıdan gelen iki adet 15.12 bitlik sayıyı toplamaktadır. Bu sayıların çapıcıların çıkışında görülebilecek en yüksek sayı olan 8192 olduğunu varsayalım. Bu durumda toplayıcının çıkışında oluşabilecek maksimum sayı 16384 (8192×2) olur ve bu işaretli sayı 16.12 bitle (0100000000000000.000000000000) ifade edilebilir. İkinci toplayıcıya birinci toplayıcı ile üçüncü çarpıcıdan gelen sayıları toplar. İkinci toplayıcının çıkışında oluşabilecek maksimum sayı 24576'dır (16384+8192) ve bu işaretli sayı 16.12 bitle (011000000000000.000000000000) ifade edilebilir. Üçüncü toplayıcı ikinci toplayıcı ile işlemin bulunduğu duruma göre eşik veya ara toplam değerinin bulunduğu 30 bit Mux'dan gelen sayları toplar. 30 bir Mux'un çıkışında oluşabilecek en büyük sayı ikinci toplayıcının çıkışında üç saat darbesi süresince bulunan 24576 sayısı (24576×3=73728) ile maksimum eşik değeri 8191.9375'in (01111111111111.1111) toplamıdır. Bu toplam yaklaşık 81920'dir (73728+8192) ve işaretli olarak 18.12 bitle (010100000000000000.000000000000) ifade edilebilir. İşlem biriminde bu bit genişlikleri kullanılarak işlem süresince hiçbir yuvarlama veya sınırlama (limitleme) işlemi yapılmaması sağlanır. Ancak işlem sonucunda oluşan veri BRAM'larda saklanacağından 14.4 bitle ifade edilecek şekilde sınırlanmalıdır. Sınırlama işlemini Lmiter bloğu gerçekleştirilir. Sonuç olarak BPU bloğunun çıkışında oluşan g_{ij} sabitleri [-8192, 8191.9375] (10000000000000.0000, 01111111111111.1111) arasında 0.0625 (0.0001) adımlarla değer alırlar. g_{ij} sabitleri veya eşik değerinin alabileceği minimum ve maksimum değerler (-8192, 8191.9375), bir pikselin alabileceği minimum ve maksimum değerlerin (-256, 255) 32 katıdır. Eğer uygulamada bu oranın yükseltilmesine veya daha küçük adım aralığına ihtiyaç duyuluyorsa, sayı formatındaki noktanın yeri ihtiyaca uygun olarak değiştirilebilir.

Piksel değerlerinin 9 bit, şablon, eşik (I) ve sabit (g_{ij}) değerlerinin 18 bitle ifade edilmelerinin nedeni FPGA'deki BRAM'ların 1, 2, 4, 9, 18 veya 32 bit genişliğindeki verileri adresleyebilmesidir. Piksel (veya durum) verilerinin 9 bitle ifade edilmesi yeterliyken, şablon değerleri için adım aralığının düşük (hassasiyetin yüksek) olması, eşik (I) ve sabit (g_{ij}) değerler için ise piksellerden (veya durumlardan) daha büyük değerler alabilmeleri açısından 18 bitle ifade edilmeleri uygun görülmüştür.

Bu bit genişlikleri APU bloklarında da aynı şekilde, durum (9.0 bit) ve **A** şablonu değerleri (6.12 bit) için uygulanmıştır.

5.3.2 APU(1) Bloğu

APU(1) bloğu (A şablonu işlem birimi 1) ilk durumlardan ($x_{ij}(0)$) bir sonraki durum değerlerinin ($x_{ij}(1)$) hesaplanması (durum iterasyonu) amacıyla kullanılır (denklem 2.14). APU(1) bloğunun blok diyagramı Şekil 5.4'de görülmektedir.



Şekil 5.4 APU(1) bloğunun blok diyagramı.

APU(1)'in yapısı BPU'ya çok benzer olup küçük farklılıklar mevcuttur. BPU'daki DiROM'lar **B** şablonu değerlerini saklarken, APU(1)'deki DiROM'lar **A** şablonu değerlerini saklar. BPU ve APU(1)'deki DiROM'ların veri organizasyonu ve çalışma biçimleri aynıdır. APU(1)'deki BRAM'lar video piksel değerleri yerine başlangıç durumu değerlerini saklarlar. Başlangıç durumu video piksel değerleri veya sabit bir değer olarak seçilebilir. Başlangıç durumlarının sabit bir değer olması durumunda APU(1)'deki BRAM'ların, DiROM'ların, çarpıcıların ve soldaki iki toplayıcının kullanılmasına gerek yoktur. İlk durum ile **A** şablonunun çarpım ve toplamlarından elde edilen sabit değer sağdaki toplayıcının girişine bağlanabilir. Ayrıca 30 bit Mux da kullanılmadan, BPU'dan gelen g_{ij} değeri bir yazmaçtan geçirilerek sağdaki toplayıcının diğer girişine bağlanabilir. Böylece APU(1)'da kullanılmayan BRAM'lar ve çarpıcılar bir adet daha APU(n) bloğu gerçeklenmesi için kullanılabilir.

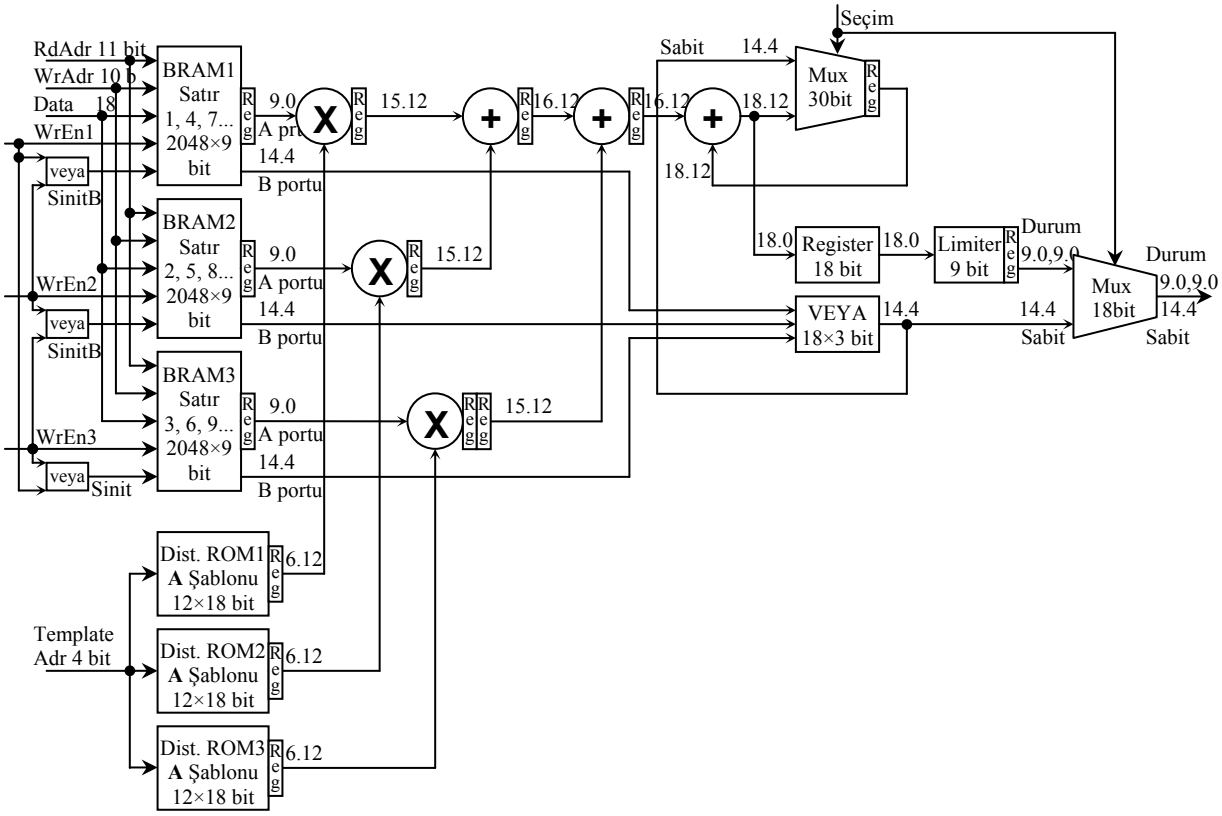
APU(1)'deki çarpıcıların ve toplayıcıların, yapısı ve yerleşimi BPU'dakiler ile aynıdır. 30 bit Mux'un sabit girişi BPU'da eşik (bias) iken APU(1)'de BPU'nun hesapladığı ve işlenen piksele ait olan sabit (g_{ij}) değeridir. APU(1)'in hesapladığı yeni durum değerleri 9.0 bit olduğu için, sınırlayıcı (limiter) ve sınırlayıcıdan önceki yazmacın bit sayısı BPU'dakinden daha düşüktür. Ayrıca APU(1)'deki sınırlayıcı yan yana duran iki piksele ait iki durum değerini bir paket halinde 18 bit olarak çıkışına yansıtır ve bu durumlar APU(2)'nin BRAM'ına yazılır. Sistemdeki bu 18 bit paketleme yapısından dolayı görüntünün sağındaki ve solundaki sınır koşulları BRAM'da 18 bit yer kaplar. g_{ij} değerleri (18 bit), durumların iletildiği aynı 18 bitlik yol üzerinden APU(2)'nin BRAM'ına yazılır.

5.3.3 APU(n>1) Blokları

APU(n>1) blokları (A şablonu işlem birimi n>1) bir sonraki durum değerlerinin ($x_{ij}(n+1)$) hesaplanması (durum iterasyonu) amacıyla kullanılır (denklem (2.14)). APU(n>1) bloklarının blok diyagramı Şekil 5.5'de görülmektedir.

Sistemdeki APU(n) sayısı istenen durum iterasyonu sayısına göre değiştirilebilir. Örneğin 10 iterasyon sonraki sonuç gözlenmek isteniyorsa, 10 adet APU(n) kullanılarak 10 iterasyon sonraki sonuç elde edilebilir. APU(n>1) ile APU(1) blokları arasındaki fark APU(n>1)'in, g_{ij} sabitlerinin BRAM'lara yazılması ve okunmasını sağlayan bir yapıya sahip olmasıdır. APU(1) ve BPU'daki BRAM'lara sadece A portundan okuma ve sadece B portundan yazma işlemi yapılır. APU(n>1)'de ise B portundan yazma işlemi yapılırken aynı zamanda yazma işlemi yapılmayan bir BRAM'dan da sabit (g_{ij}) değeri okunur. Bu değer hem APU(n)'in 30

bit Mux'una hem de (bir sonraki) APU(n+1)'in BRAM'ına yazılır.

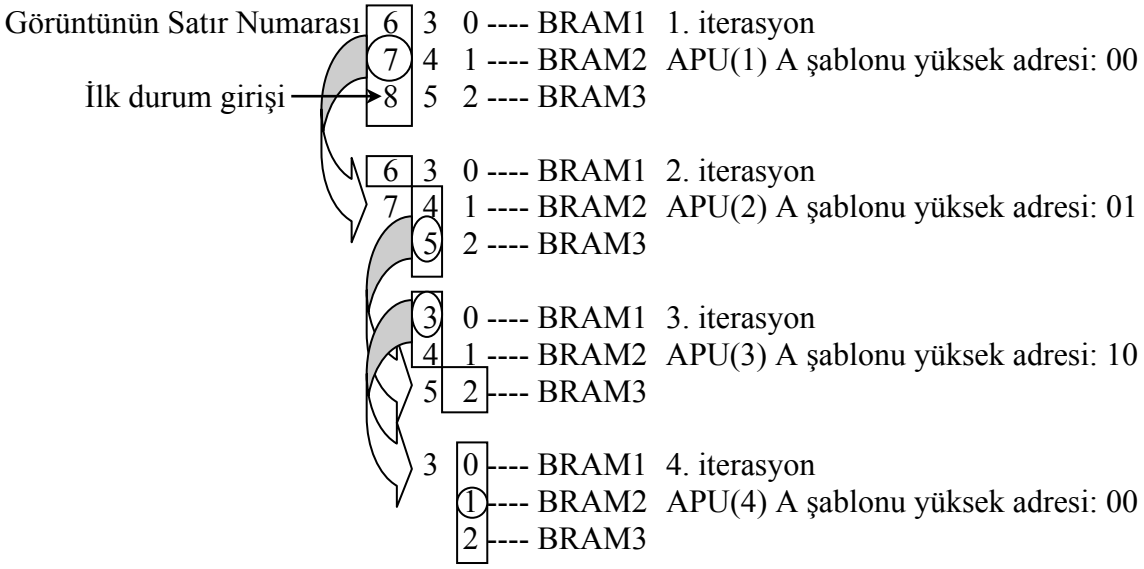


Şekil 5.5 APU(n>1) bloklarının blok diyagramı.

APU(n>1)'in BRAM'larında bulunan yazma aktif (WrEn) uçları VEYA (OR) kapılarından geçirilerek BRAM'ın B portunun "senkron init" ucuna bağlanmıştır. Bu sayede B portundan g_{ij} (18 bit) değeri okunacak BRAM dışındaki iki BRAM'ın çıkışları lojik 0 yapılır. Sonra bu çıkışlar 18 adet üç girişli VEYA kapı dizisinden geçirilerek okunması gereken sabit değer elde edilmiş olur. Eğer "senkron init" ucu kullanılmamış olsaydı, 18 adet üç girişli VEYA kapıları yerine 18 adet üç girişli Mux kullanmak gerekecekti. Bu da devrenin kapladığı alanı ve kullanılan donanım miktarını, bu iş için yaklaşık iki kat arttıracaktı.

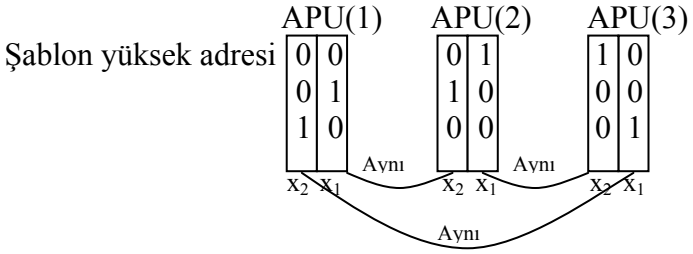
APU(n>1)'ler kendilerinden sonraki APU(n+1)'e hem sabit (14.4 bit), hem de yeni durum (9.0 bit) değerini iletirler. Durumlar ikişerli paketler halinde, sabitler ise tek başlarına, 18 bitlik bir veri yolu üzerinden iletilir. İletim zamanlamasını ayarlamak için, APU(n>1)'de 2×18 bitlik bir Mux kullanılmıştır.

APU(n) bloklarının veri işleme akışı Şekil 5.6'da görülmektedir.



Şekil 5.6 APU(n) bloklarının veri işleme akışı.

Şekil 5.6'da kare çerçeve içine alınan kısımlar o an BRAM'lar içinde bulunan durumların satır numaralarını göstermektedir. Yuvarlak içine alınan numaralar ise işlem gören satırın numarasıdır. Geniş oklarla gösterilen kısım işlenen satırın bir sonraki APU'nun hangi BRAM'ına yazılacağını göstermektedir. Yukarıdaki şekilden de görülebileceği gibi, bir çerçeve içinde yer alan satırlar APU'lar arasında hep aynı numaralı BRAM'a yazılmaktadır, ancak her APU o anda ayrı bir satırı işlediği için **A** şablonu adresinin (DiROM adresinin) yüksek iki bitlik değeri de APU'lar arasında farklı olmalıdır. "Adres ve Kontrol" bloğu tek bit şablon adresi üretir. Diğer iki şablon adresinin nasıl elde edildiği Şekil 5.7'de görülmektedir.



Şekil 5.7 DiROM adresinin yüksek iki bitinin elde edilmesi.

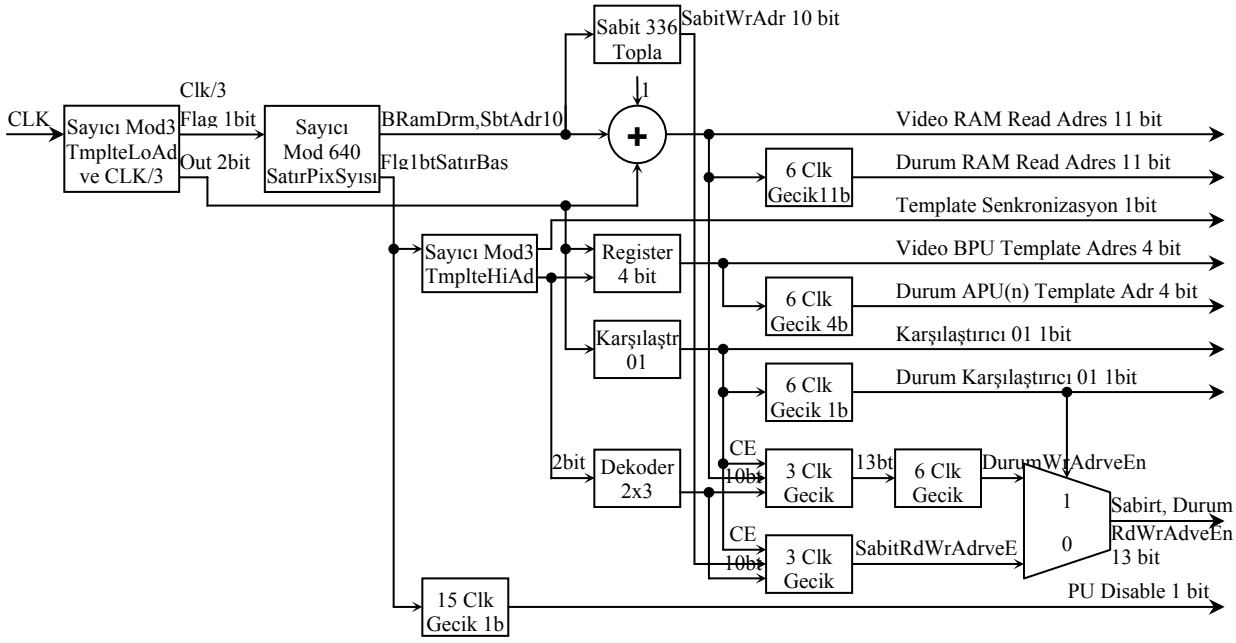
APU(1) bloğunun şablon yüksek adresi (2 bit), yani Şekil 5.7'deki ilk iki sütun (x_2 ve x_1) "Adres ve Kontrol" bloğu tarafından üretilir. Şekilden de görülebileceği gibi sadece APU(3)'ün x_2 sütunda bulunan lojik değerlerin üretilmesi yeterlidir ve bu sütun ilk iki sütunun terslerinin lojik AND işlemi yapılmış halidir ($x_2' \text{AND } x_1'$). Elde edilen bu yeni işaretle doğru bağlantılar yapılırsa, yani şablon yüksek adresi APU(1) için normal x_2 ve x_1, APU(2) için x_2 ve AND çıkışı, APU(3) için AND çıkışı ve x_1 şeklinde bağlanırsa şablon için

doğru adres değerleri elde edilmiş olur. Eklenen başka APU blokları olursa bu döngü kendini tekrar eder.

Ayrıca Şekil 5.6'de görüldüğü gibi yeni durumlar her APU'de farklı bir BRAM numarasına yazılmaktadır. Bu sorun, üç tane olan BRAM yazma aktif (WrEn) uçlarının, bir sonraki APU'ya sırası döndürülerek bağlanmasıyla çözülür. Örneğin APU(2)'daki BRAM2'ye bağlı olan WrEn ucu, APU(3)'de BRAM3'e, APU(4)'de BRAM1'e bağlıdır. Bu döngü eklenen APU'larla kendini tekrar eder.

5.4 Adres ve Kontrol Bloğunun İç Yapısı

"Adres ve Kontrol" bloğunun görevi PU bloklarının ihtiyaç duyduğu adres ve kontrol değerlerini oluşturmak, böylece PU bloklarının çalışmasını kontrol etmektir. "Adres ve Kontrol" bloğunun basitleştirilmiş blok diyagramı Şekil5.8'de görülmektedir.



Şekil 5.8 "Adres ve Kontrol" bloğunun basitleştirilmiş blok diyagramı.

"Adres ve Kontrol" bloğunun iki adet girişi vardır. Bulardan biri saat işareti, diğeri ise yukarıdaki şekilde gösterilmemiş olan ve "Video Giriş" bloğundan gelen PU aktif (enable) (PU En) işaretidir. "PU En" işareti pasifken, "Adres ve Kontrol" bloğunda hemen hemen tüm, PU bloklarında ise bütün işlemler durdurulur. "PU En" işareti "Video Giriş" bloğu tarafından aktif hale, "Adres ve Kontrol" bloğunu "PU Disable" çıkışı ile de pasif hale getirilir. "PU Disable", PU bloklarının işledikleri satırın sonuna kadar bütün işlemleri tamlanmasıyla aktif hale getirilir ve PU'ların çalışması durdurulur. PU'lar tekrar çalışmaya (bir sonraki satırın 3

adet piksel değerinin BPU'nun BRAM'ına yazılmasıyla) "Video Giriş" bloğu tarafından başlatılır. PU'ların saat frekansı, piksel frekansının üç katı civarında olmalıdır. Saat frekansı piksel frekansını üç katından yüksekse, emülatör bir pikselle ilgili hesaplamaları yeni bir piksel değeri gelmeden önce bitireceğinden PU'ların yeni piksel değerinin gelmesini beklenmesi gerekir. Bu bekleme durumunu ("Video Giriş" bloğundan çıkan) "PU En" (enable) işareti ile kontrol edilir. Yeni piksel değerinin gelmesiyle PU'lar aktif hale getirilir ve üç saat darbesi süresince çalıştırılır. Satır sonuna gelindiğinde, bütün piksel değerleri okunmuş ve BPU BRAM'larında mevcut demektir. Bu durum oluştuğunda, ardışık düzen yapısı içinde işlem gören tüm veriler işleninceye kadar, PU'lar aktif hale getirilir. Tüm işlemler bittiğinde, "Adres ve Kontrol" bloğu "PU Disable" işaretini aktif ederek "Video Giriş" bloğunun PU bloklarını durdurmasını sağlar.

Video ve durum BRAM adresleri, Mod 3 sayıcı ve Mod 640 satır piksel sayıcısının çıkışlarının toplanmasıyla oluşturulur. Şekil 5.8'de en sol tarafta görülen Mod 3 sayıcı, oluşturulan BRAM adreslerinin 1, 2, 3, 2, 3, 4, 3, 4, 5, 4, 5, 6, … şeklinde ilerlemesini sağlar. Mod 3 sayıcı her bir döngüde Mod 640 sayıcıya bir saat işareti oluşturur ve böylece Mod 640 sayıcının değeri bir artar. Mod 640 sayıcısının değeri işlenen pikselin kaçıncı piksel olduğu bilgisini içerir (piksel numarası – 1). Mod 3 sayıcı ise bu değeri üç saat darbesi içinde birer birer artırarak PU'daki BRAM'ların sırasıyla, işlenen pikselin solundaki, işlenen piksel ve işlenen pikselin sağındaki piksel değerinin her üç satır için okunmasını sağlar. Sonra Mod 640 sayıcısının değeri bir artar ve işlem bir sonraki piksele ilişkin hesaplamaların yapılması için devam eder. Yukarıda piksel değeri olarak anlatılan çalışma şekli durum değerleri için de aynıdır. Durum değerlerinin okunması için kullanılan adres değerleri, basitçe piksel değerlerinin okunması için oluşturulan adres değerlerinin 6 saat darbesi geciktirilmiş halidir. Bu gecikme BPU bloğunun APU bloklarına göre 6 saat darbesi önce çalışmaya başlamasından kaynaklanır. Bu zaman farkının nedeni, APU(1) bloğunun işlem yapabilmesi için BPU'nun hesapladığı g_{ij} değerine ihtiyaç duymasıdır.

APU(n>1)'in BRAM'larında bulunan sabitlerin (g_{ij}) saklandığı adres değerleri ise, durum\piksel değerleri için oluşturulan adres değerine 336 eklenerek elde edilir. Sabit değerleri 18 bit genişliğinde olduğundan sabitlerin adres haritasındaki adres değerleri, durum\piksel (9 bit) adres haritasındaki adres değerlerinin yarısı kadardır. Yani sabit değerlerinin saklandığı adrese yapılan 336 değerindeki ofset durum\piksel adres haritası için 336×2=672'dir. Bu alan, durum\piksel ve sınır koşulları değerlerinin saklanması için gereken 644×9 bit veri alanı için yeterlidir (672>644).

Şekil 5.8'de en soldaki Mod3 sayıcının çıkış değeri, aynı zamanda kullanılacak şablon değeri ROM (DiROM) adresinin düşük iki bitlik kısmını da oluşturur. Okunan piksel\durum değeri soldan sağa doğru kayarken kullanılan şablon değeri de soldan sağa doğru kayar. Örneğin 1, 2, 3 adreslerindeki piksel\durum değerleri için 0, 1, 2 düşük DiROM adreslerindeki şablon değerleri kullanılır, 2, 3, 4 piksel\durum adresleri için yine 0, 1, 2 DiROM adreslerindeki şablon değerleri kullanılır. Bu durum satır sonuna kadar devam eder. Yani her bir sonraki işlenen piksel\durum için aynı şablon değerleri ile çarpma yapılmaya devam eder, böylece bütün şablon görüntü üzerinde kaydırılarak her piksel\durum için işlem yapılmış olur. Satırın sonuna gelindiğinde ise Mod 640 sayıcısı, şeklin ortasındaki Mod 3 sayıcısının değerini bir arttıran bir saat darbesi üretir. Bu Mod 3 sayıcının çıkışı şablon adresinin yüksek iki bitini oluşturur. Mod 640 sayıcısı başa döndüğünde (0), yeni bir satır daha BRAM'a yazılmış demektir ve bu da şablon adresinin yüksek iki bitlik kısmının bir arttırılması ile sonuçlanır. Böylece, şablon ROM'u (DiROM) şablon matrisinin bir sonraki satırının kayıtlı olduğu adres değerine gelir. Bu işlem ortadaki Mod 3 sayıcı ile kontrol edildiği için her üç satırda bir, şablon ROM'u adresinin aynı şablon matrisi satırına geleceği aşikardır. **A** şablonu (APU'daki DiROM) adresi, **B** şablonu (BPU'daki DiROM) adresinin basitçe 6 saat darbesi gecikmiş halidir.

Şekil 5.8'in ortasındaki "Karşılaştırıcı 01" çıkışı PU bloklarındaki 30 bit Mux'ların seçme ucuna bağlı olup sabit değerlerinin Mux'a yüklenme zamanlamasını kontrol eder. APU(n)'ler için kullanılan "Karşılaştırıcı 01" işareti, BPU için kullanılan "Karşılaştırıcı 01" çıkış işaretinin basitçe 6 saat darbesi geciktirilmiş halidir. "Karşılaştırıcı 01" çıkışı aynı zamanda, şekilde görülen iki adet, üç saat darbesi geciktiricisinin saat aktif (CE) ucunu da kontrol eder.

PU bloklarındaki BRAM'lar okunurken hepsi aynı anda okunur ve hepsinin değeri birer çarpıcıya gider. Ancak BRAM'lara yazım işleminde hangi BRAM'a yazım yapılacağı belirlenmelidir. Bu iş için Şekil 5.8'in ortasında görülen 2×3 dekoder kullanılır. Bu dekoder durum ve sabitin PU'daki üç BRAM'dan hangisine yazılacağını belirler.

Durum yazma, sabit yazma\okuma adresleri basitçe daha önce piksel okuma (işlenen piksel) ve sabit (işlenen pikselin sabiti) için oluşturulan adres değerlerinin geciktirilmiş halidir. Bu adres değerleri şekilde görülen 3 saat darbesi geciktiricilerinin içerisine "Karşılaştırıcı 01" aktif iken (işlenen pikselin adres değeri oluşmuşken) alınır. "Karşılaştırıcı 01" üç saat darbesinde bir aktif olduğundan aslında üç saat darbesi geciktiricileri içlerine aldıkları veriyi 7 (3×2+1) saat darbesi geciktirirler. Bu süre piksel değerinin BPU içinde işlenerek çıkışının elde edilmesi için geçen süredir (BPU gecikme süresi). APU blokları arasında iletilen durum

(9 bit) ve sabit (18 bit) değerleri 18 bit veri yolu (bus) üzerinden iletilir. Bu yol üzerinde bazen sabit (18 bit) bazen de ikili olarak paketlenmiş durumlar (9+9=18 bit) bir sonraki APU ya iletilir. İletilecek verinin kontrolüyse Şekil 5.4 ve Şekil 5.5'in en sağında görülen Mux'lar ile sağlanır. Mux'ların seçim ucu "Karşılaştırıcı 01" in altı saat darbesi geciktirilmiş hali ile kontrol edilir. Altı saat darbesi gecikme APU bloklarının BPU bloğundan altı saat darbesi sonra çalışmaya başlamasından kaynaklanır. Bu gecikmenin nedeni BPU'nun hesapladığı değerlerin (g_{ij}) APU(1) tarafından kullanılmasıdır. "Adres ve Kontrol" ve "Video Giriş" bloklarının giriş ve çıkışlarında gözlenen bazı işaretler Ek 2'de verilmiştir.

"Video Giriş" bloğu, "Video Çıkış" bloğu ve I^2C bloğunun HSA emülasyonu ile doğrudan bir ilgisi bulunmadığından, tez kapsamında anlatılmamıştır. Bütün blokların şematik çizimleri Ek 3'de verilmiştir.

5.5 Gerçeklenen Emülatör Yapısı

Tasarlanan emülatör yapısı Celoxica firmasını RC203 kartı üzerinde bulunan Xilinx firmasının XC2V3000 -4 (Virtex-II 3000) FPGA tümdevresi üzerinde gerçeklenmiştir. Video girişi olarak bir bilgisayardan alınan VGA işaretinin sadece yeşil renk bileşeni kullanılmıştır. RC203 kartında VGA girişi olmadığından kart üzerinde bulunan 50 uçlu genişleme yuvası (50 pin expansion header) kullanılarak Texas Instruments firmasının ürettiği TVP7001 tümdevresini kullanan bir video ADC kartı dışarıdan bağlanmıştır. FPGA üzerindeki HSA emülatörü ADC kartından gelen dijitalleştirilmiş videoyu işleyerek, işlenmiş piksel verilerini Analog Devices firmasının ADV7123 video DAC tümdevresine gönderir. Video DAC'nin kırmızı, yeşil ve mavi renk girişleri aynı 9 bit çıkış verisi ile sürülerek VGA monitöründe oluşan videonun gri tonlamalı olması sağlanır. Sistem 640×480 piksel 60 fps VGA girişi ile test edilmiş ve kenar belirleme uygulamasında kullanılmıştır. Bu uygulamadaki emülatör bir adet BPU ve üç adet APU ile gerçeklenmiştir.. Kenar belirleme işlemi için kullanılan $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$ şablonları, eşik (bias) ve ilk durum değerleri (5.1) eşitliklerinde verilmiştir. Piksel verileri (9 bit) [-256 (siyah), 255 (beyaz)] arasında değer almaktadır.

$$\hat{\mathbf{A}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \hat{\mathbf{B}} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \quad \hat{I} = 68, \quad x_{ij}(0) = 0 \tag{5.1}$$

Bütün tasarım Xilinx firmasının ISE 9.1.03i yazılımının şematik editörü ve "CORE Generator" yardımcı yazılımı kullanılarak gerçeklenmiştir. Devrenin simülasyon ve hata

ayıklama (debug) işlemleri sırasıyla ISE simülatörü ve "ChipScope Pro 9.1" yazılımı ile yapılmıştır. Tasarım sayıcı, karşılaştırıcı, Mux (multiplexer), toplayıcı, LUT, kaydırmalı yazmaç (shift register) gibi basit yapılar ve çarpıcı, blok RAM gibi FPGA tümdevresine entegre edilmiş blok yapılar kullanılarak oluşturulmuştur. Bu elemanların tamamı ISE'deki "CORE Generator" yazılımı ile gerçeklenmiştir. Tasarımın bu yöntemle gerçeklenmesi tasarım üzerinde tam bir kontrole sahip olunmasını sağlamış, ancak süreci uzatmıştır.

Tasarım ISE yazılımı ile XC2V3000 (Virtex-II 3000) FPGA tümdevresi için sentezlenmiştir. Sentezleme sonucunda elde edilen emülatörün her bloğu için FPGA'de kullanılan donanım miktarı Çizelge 5.2'de verilmiştir.

Çizelge 5.2 Emülatörün her bloğun için FPGA'de kullanılan donanım miktarı (XC2V3000)

Bloklar	**Flip-floplar**	**LUT'lar**	**Blok RAM'lar**	**Çarpıcılar**
BPU	234 (%0.816)	191 (%0.666)	3 (%3.125)	3 (%3.125)
APU(1)	230 (%0.802)	191 (%0.666)	3 (%3.125)	3 (%3.125)
APU(n>1)	230 (%0.802)	212 (%0.739)	3 (%3.125)	3 (%3.125)
Adres ve Kontrol	111 (%0.387)	85 (%0.296)	0	0
Video Giriş	82 (%0.286)	83 (%0.289)	0	0
Video Çıkış	43 (%0.150)	52 (%0.181)	0	0
I^2C	34 (%0.119)	67 (%0.234)	0	0

Çizelge 5.2'den de görülebileceği gibi FPGA üzerinde gerçeklenebilecek PU sayısı, FPGA'deki çarpıcı ve blok RAM'ların sayısı ile sınırlanır. Virtex-II 3000 FPGA tümdevresinde 96 BRAM ve 96 çarpıcı bulunmasından dolayı FPGA üzerinde 32 (96/3) adet PU gerçeklenebilir. 32 adet PU ile gerçeklenen emülatör, 3×3 şablonlarla işlem yapan konumdan bağımsız bir HSA yapısının ayrık zamanlı emülasyonu 31 (ilk durumların bir sabit olması durumunda 32) Euler iterasyon ile gerçekleştirebilir.

ISE'nin sentez raporuna göre emülatörün Virtex-II 3000 FPGA'i üzerinde çalışabileceği en yüksek saat frekansı yaklaşık 136 MHz'dir. Saat frekansı çarpıcıların maksimum çalışma hızı ile sınırlanmıştır. Bu hız FPGA'in üretim teknolojisi ile ilgili olup emülatörün kontrol yapısı üzerinde yapılacak bir değişiklikle çalışma hızı daha fazla arttırılamaz.

ISE'nin "Place and Route" raporuna göre gerçeklenen sistemin çalışabileceği maksimum çalışma frekansı yaklaşık 123 MHz'dir. Bundan sonraki hesaplamalarda emülatörün çalışabileceği maksimum saat frekansı 123 MHz olarak alınacaktır.

Video çerçevesinin her bir pikseli için yapılacak hesaplama üç saat darbesi gerektirdiğinden, emülatör saniyede 41 Mega piksel (123/3) işleyebilmektedir. Bir satırdaki bütün piksel verileri emülatöre iletildiğinde (satır sonunda) emülatörün yeni satırı işlemeye hazır olması için (peline yapısı ve "Adres ve Kontrol" bloğundaki gecikmeler nedeniyle) 8 piksel periyodu zaman geçmesi gerekmektedir. Bu nedenle, emülatörün işleyebileceği video hızını hesaplarken bir satırdaki piksel sayısına 8 eklemek gereklidir. Örneğin, 640×480 piksel boyutundaki bir video çerçevesinde hesaplanacak 307200 (640×480) adet piksel varken, bu hesaplamayı yapabilmek için 311040 (648×480) piksel periyodu zamana ihtiyaç vardır. Emülatör en yüksek hızda çalıştırıldığında (123 MHz) 131 fps (41 Mega/311040) 640×480 piksel video görüntüsünü 3×3 HSA şablonları ve 31 Euler iterasyonu ile işleyebilir. Burada piksel değerleri 9 bit (gri tonlamalı), şablon değerleri ise 18 bit ile ifade edilmektedir.

Bu hesaplama şekli VGA sinyalleri için doğru sonuç vermez, çünkü 640×480 piksel 60 fps VGA işaretinin bir çerçevesinde senkronizasyon işaretleri ile birlikte 800×525 piksel periyodu mevcuttur. Emülatörün bir saniyede işleyebileceği video çerçevesi sayısı (fps) (5.2) denklemi ile hesaplanabilir.

$$\text{fps} = \frac{F_{piksel}}{(H+8)\times V} \tag{5.2}$$

Burada H video görüntüsünün (aktif bölge) satır piksel sayısı, V ise video standardının senkronizasyon işaretleri dahil gönderdiği satır sayısıdır. Örneğin, 640×480 piksel video görüntüsü işlenecek olsun ve bir çerçeve periyodu da 800×525 piksel periyodu olsun. Bu durumda emülatör saniyede 120 çerçeveye (41 Mega/(648×525)) kadar veri işleyebilir.

6. SONUÇLAR

Gerçek zamanlı video işleyen yeni bir Hücresel Sinir Ağı (HSA) emülatörü yapısı önerilmiş ve Celoxica RC203 kartı üzerinde bulunan Xilinx Virtex-II 3000 FPGA tümdevresi ile gerçeklenmiştir. Gerçeklenen emülatör, bir kenar belirleme uygulamasında 640×480 piksel 60 fps monokrom VGA video girişiyle test edilmiş ve çıkış videosu VGA monitöründe gözlenmiştir.

Emülatörün çalışabileceği maksimum saat frekansı (Virtex-II 3000 için) ISE yazılımının sentezleyici raporuna göre 136 MHz, "Place and Route" raporuna göre ise 123 MHz olarak belirlenmiştir. Emülatörün saat frekansı FPGA'deki çarpıcıların hızı ile sınırlanmaktadır. Bu sınır FPGA'in üretim teknolojisinden kaynaklanmakta olup, emülatörün kontrol yapısını değiştirerek çalışabileceği maksimum saat frekansını yükseltmek mümkün değildir.

Bir pikselin çıkış değerinin hesaplanması üç saat darbesi zaman alır ve bu değerin hesaplanması için yapılacak Euler iterasyonu sayısı gerçeklenen işlem birimleri'nin sayısına bağlıdır. İşlem birimlerinin çalışabileceği maksimum saat frekansı 123 MHz olduğu ve her bir çıkış piksel değerinin hesaplanması için üç saat darbesi gerektiğinden, emülatör saniyede 41 Mega piksel (123/3) işleyebilmektedir. Virtex-II 3000 FPGA tümdevresinde 96 çarpıcı ve 96 blok RAM bulunduğundan 32 adet işlem birimi gerçeklenebilir. Bu 32 işlem birimiyle çıkış değerlerinin hesaplanması için 31 Euler iterasyonu ((96/3)–1) gerçekleştirilebilir. Önerilen emülatör yapısının temel özellikleri şunlardır:

- 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).

Gerçeklenen emülatör yapısı FPGA dışında bir bellek elemanı kullanmamakta ve video görüntülerini gerçek zamanlı olarak işlemektedir. Literatürde şu ana dek gerçeklenmiş ve bu iki özelliği de bünyesinde bulunduran yalnızca bir çalışma göze çarpmaktadır (Martínez vd. 2007). Martínez vd. (2007) çalışmasındaki Virtex-4 üzerinde gerçeklenen yapının bu tezde gerçeklenen yapıya göre bazı iyi ve kötü yanları mevcuttur. Bu tezde gerçeklenen yapı giriş görüntüsünden g_{ij} sabitlerini hesaplar ve saklar, ancak giriş görüntüsünü saklamaz. Bu durum, işlenmiş görüntü ile giriş görüntüsü arasında bir işlem yapılarak çıkış görüntüsünün oluşturulmasını gerektiren uygulamaların gerçeklenmesini zorlaştırır. Martínez vd. (2007) çalışmasındaki yapıda ise giriş görüntüsü kaybedilmez ve g_{ij} sabitleri her kullanımdan önce

tekrar hesaplanır. Bu durum, her işlem birimi için FPGA üzerinde kullanılan donanım miktarını iki kat arttırmaktadır. Martínez vd. (2007) çalışmasında gerçeklenen yapıdaki her işlem birimi iki adet blok RAM ve iki adet DSP48 bloğu içerir (Çizelge 3.1) ve çalışma frekansı piksel frekansının en az 17 katı olmalıdır. Çalışma frekansının çok yüksek olması bu yapının Virtex-II ve Spartan-3 aileleri gibi, Virtex-4'e göre daha yavaş ve dolayısıyla düşük maliyetli olan FPGA aileleri için uygun olmaması sonucunu doğurur. Bu tezde gerçeklenen yapıdaki her işlem birimi üç adet blok RAM ve üç adet çarpıcı içerir ve çalışma frekansı piksel frekansının en az üç katı olmalıdır. Bu iki yapı karşılaştırıldığında, bu tezde gerçeklenen yapının Martínez vd. (2007) yapısına göre kabaca 1.5 kat fazla donanımı kullanarak 5.6 kat hızlı çalıştığı söylenebilir. Bu hız artışıyla, yapının Virtex-II FPGA tümdevresinde gerçek zamanlı video işlemesine olanak sağlanmıştır. Eğer yapı giriş görüntüsünü de saklayacak şekilde değiştirilirse, kullandığı donanım miktarı Martínez vd. (2007) yapısının üç katı olacaktır. Bu durumda, üç kat fazla donanım kullanılarak sistemin 5.6 kat hızlı çalışması sağlanmış olur. Bu hız artışı, Virtex-4 FGPA'leri kullanıldığında, 1280×1024 piksel SXGA video işaretlerinin (piksel frekansı 60 fps için 108 MHz, 75 fps için 135 MHz) gerçek zamanlı olarak işlenmesini mümkün kılar.

Tez'de gerçeklenen yapıdaki A şablonu işle birimleri (APU(n)) doğrusal dizilmiştir. İleride bu yapı geliştirilerek işlem birimlerinin düzlemsel (matrisel) dizilimine olanak sağlanabilir. Böylece, işlem birimlerinde bir blok RAM ve bir çarpıcı kullanılarak daha küçük işlem birimleri oluşturulabilir. Bu sayede, işlem birimi sayısı istenen hız ve işlenecek piksel sayısına göre daha verimli ve esnek bir şekilde ayarlanabilir.

Bu tez çalışması çerçevesinde "A New Approach to Emulate CNN on FPGAs for Real Time Video Processing" adlı bildiri "Proceedings of the 11th International Workshop on Cellular Neural Networks and Their Applications (CNNA 2008)"da yayımlanmış ve aynı adlı toplantıda sunum ve demonstrasyonu yapılmıştır. Ayrıca, "Gerçek Zamanlı Video İşleyen Yeni Bir Hücresel Sinir Ağları Emülatörü" adlı bildiri özeti "Gömülü Sistemler ve Uygulamaları Sempozyumu (GömSis 2008) Bildiri Özetleri" kitapçığında yayımlanmış ve aynı adlı toplantıda poster sunumu yapılmıştır (Kayaer ve Tavşanoğlu, 2008a; 2008b).

Tez çalışmasında gerçeklenen yapı "CPU Turkey 2008" adlı yarışmanın "Akademik Yenilikçi Gömülü Sistem Tasarımı" kategorisine "RTCNNP" (Real-Time CNN Processor) adılı projeyle katılmış ve birincilik ödülü almıştır. Ayrıca, yarışma dahilinde "CeBIT Eurasia Bilişim" fuarında, 7-12 Ekim 2008 tarihleri arasında gerçeklenen yapının demonstrasyonu yapılmıştır.

KAYNAKLAR

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., (1988a), "Cellular Neural Networks: Theory", IEEE Transactions on Circuits and Systems, 35: 1257-1272.

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

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

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.

Feher, B., Szolgay, P., Roska, T., Radványi, A. G., Szirányi, T., Csapodi, M., László, K., Nemes, L., Szatmári, I., Tóth, G. ve Venetianer, P. L., (1996), "ACE a digital floating point CNN emulator engine", Fourth IEEE International Workshop on Cellular Neural Networks and Their Applications (CNNA'96), 24-26 June 1996, Seville / Spain.

Hidvégi, T., Keresztes, P. ve Szolgay, P., (2002), "An accelerated digital CNN-UM (CASTLE) Architecture by using the Pipe-Line Technique", 7th IEEE International Workshop on 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.

Kayaer, K. ve Tavsanoglu, V., (2008a), "A New Approach to Emulate CNN on FPGAs for Real Time Video Processing", 11th IEEE International Workshop on Cellular Neural Networks and Their Applications (CNNA'2008), 14-16 July 2008, Santiago de Compostela / Spain.

Kayaer, K. ve Tavşanoğlu, V., (2008b), "Gerçek Zamanlı Video İşleyen Yeni Bir Hücresel Sinir Ağları Emülatörü", Gömülü Sistemler ve Uygulamaları (GömSis'2008), 3-5 Kasım, İstanbul.

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.

Kincses, Z., Nagy, Z. ve Szolgay, P., (2006), "Implementation of nonlinear template runner emulated digital CNN-UM on FPGA", 10th IEEE International Workshop on Cellular Neural Networks and Their Applications (CNNA'2006), 28-30 August 2006, Istanbul / Turkey.

Liñán, G., Foldesy, P., Espejo, S., Domínguez-Castro, R. ve Rodríguez-Vázquez, A., (1999), "A 0.5μm CMOS 10^6 Transistors Analog Programmable Array Processor for Real-Time Image Processing", 25th European Solid-State Circuits Conference (ESCIRC'99), 21-23 Sept. 1999, Duisburg / Germany.

Liñán, G, Domínguez-Castro, R., Espejo, S. ve Rodríguez-Vázquez, A., (2001), "ACE16k: A Programmable Focal Plane Vision Processor with 128 x 128 Resolution", 15th European Conference on Circuit Theory and Design, 28-31 August 2001, Espoo / Finland.

Liñán, G., Espejo, S., Domínguez-Castro, R. ve Rodríguez-Vázquez, A., (2002a), "ACE4k: An Analog I/O 64 × 64 Visual Microprocessor Chip with 7-bit Analog Accuracy", International Journal of Circuit Theory and Applications, 30 (2/3): 89-116.

Liñán, G. , Rodríguez-Vázquez, A., Espejo, S. ve Domínguez-Castro, R., (2002b): "ACE16k: A 128x128 Focal Plane Analog Processor with Digital I/O", 7th IEEE International Workshop on Cellular Neural Networks and Their Applications (CNNA'2002), 22-24 July 2002, Frankfurt / Germany.

Martínez-Alvarez, J. J., Garrigós-Guerrero, F. J., Toledo-Moreo, F. J. ve Ferrández-Vicente, J. M., (2007), "High Performance Implementation of an FPGA-Based Sequential DT-CNN", 2nd International Work-Conference on the Interplay between Natural and Artificial Computation (IWINAC'2007), 18-21 June 2007, Murcia / Spain.

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 July 2002, Frankfurt / Germany.

Nagy, Z. ve Szolgay, P., (2003), "Configurable Multi-Layer CNN-UM Emulator on FPGA", IEEE Transactions on Circuits and Systems-I, 50 (6): 774-778.

Rodríguez-Vázquez, A., Liñán-Cembrano, G., Carranza, L., Roca-Moreno, E., Carmona-Galán, R., Jiménez-Garrido, F., Domínguez-Castro, R. ve Meana, S. E., (2004), "ACE16k: The Third Generation of Mixed-Signal SIMD-CNN ACE Chips Toward VSoCs", IEEE Transactions on Circuits and Systems-I: Regular Papers, 51 (5): 851-863.

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.

Saatçi, E. (2003), Image Processing Using Cellular Neural Networks, Doktora Tezi, London South Bank University.

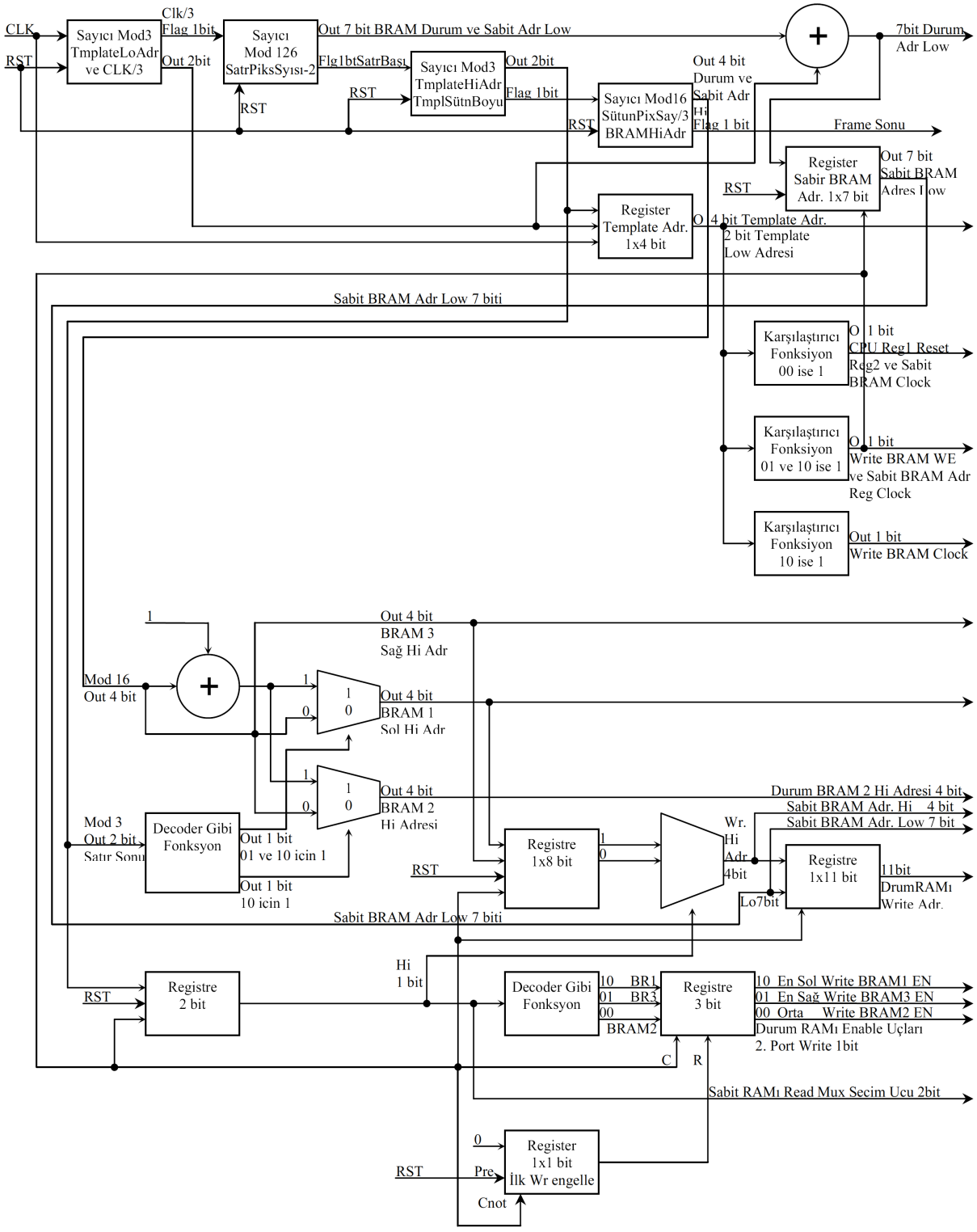
Toledo, F. J., Martínez, J. J., Garrigós, F. J. ve Ferrández, J. M., (2005a), "Image Processing with CNN in a FPGA-Based Augmented Reality System for Visually Impaired People", 8th International Work-Conference on Artificial Neural Networks (IWANN'2005), 8-10 June 2005, Barcelona / Spain.

Toledo, F. J., Martínez, J. J., Garrigós, F. J. ve Ferrández, J. M., (2005b), "An Augmented Reality Visual Prothesis for People Affected by Tunneling Vision", 1st International Work-Conference on the Interplay between Natural and Artificial Computation (IWINAC'2005), 15-18 June 2005, Canary Islands / Spain.

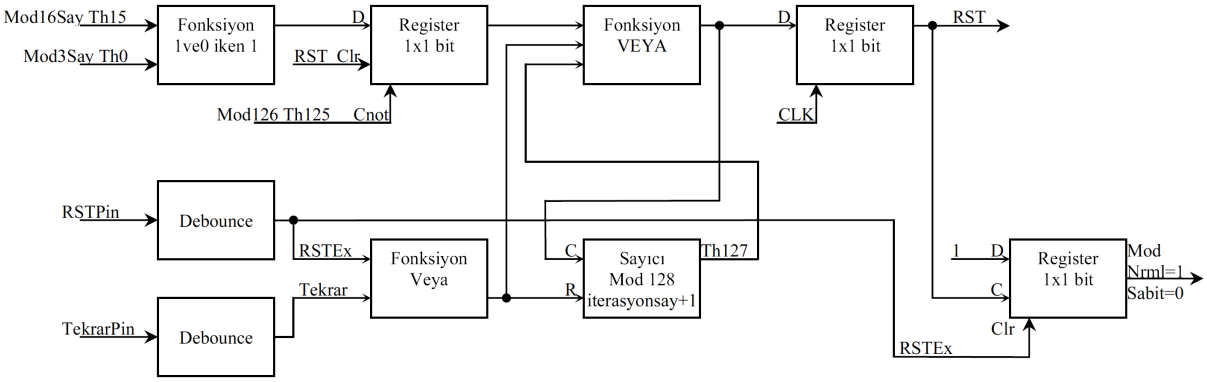
Vörösházi, Z., Nagy, Z., Kiss, A. ve Szolgay, P., (2006), "An Embedded CNN-UM Global Analogic Programming Unit implementation on FPGA", 10th IEEE International Workshop on Cellular Neural Networks and Their Applications (CNNA'2006), 28-30 August 2006, Istanbul / Turkey.

EKLER

Ek 1 Tez dahilinde gerçeklenen ilk HSA emülatöründeki “Adres ve Kontrol” bloğunun blok diyagramı



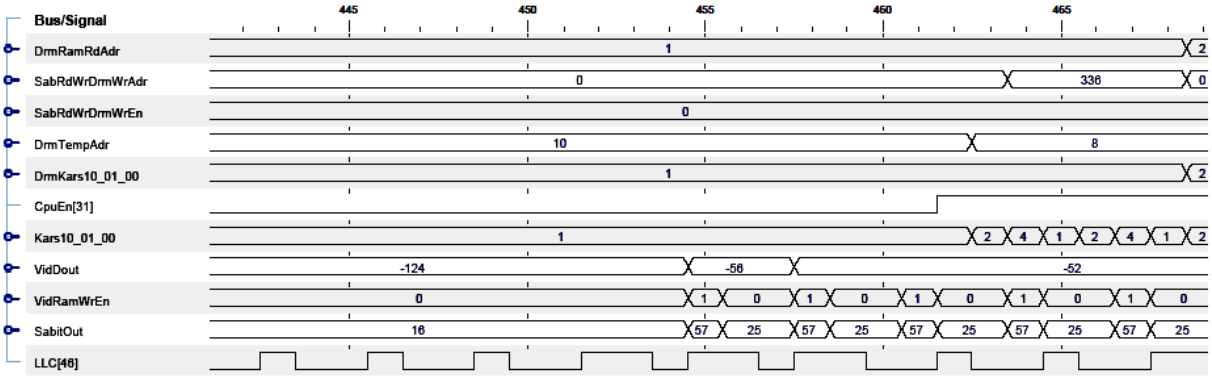
Şekil Ek 1.1.a Adres ve Kontrol bloğunun blok diyagramı.



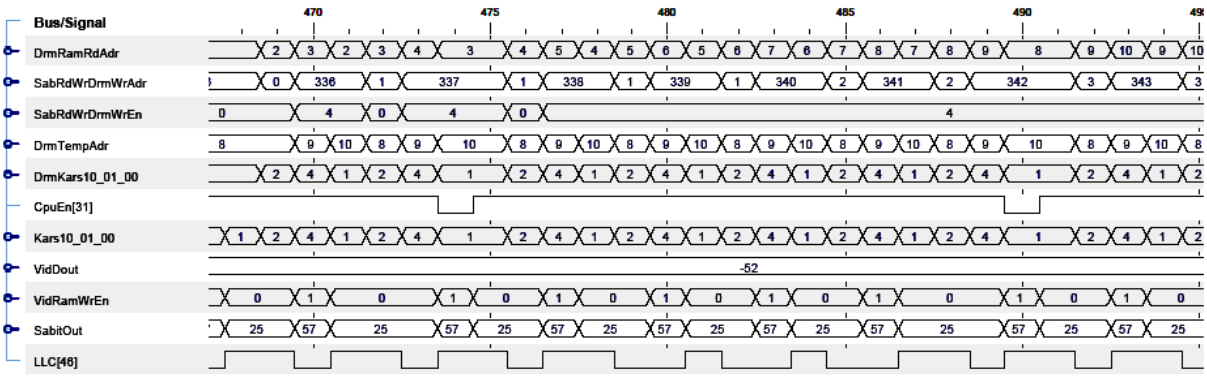
Şekil Ek 1.1.b Adres ve Kontrol bloğunun blok diyagramı (devamı).

Ek 2 Uygulamada kullanılan HSA emülatötünüdeki “Adres ve Kontrol” ve “Video Giriş” bloklarının giriş ve çıkışlarında gözlenen bazı işaretler

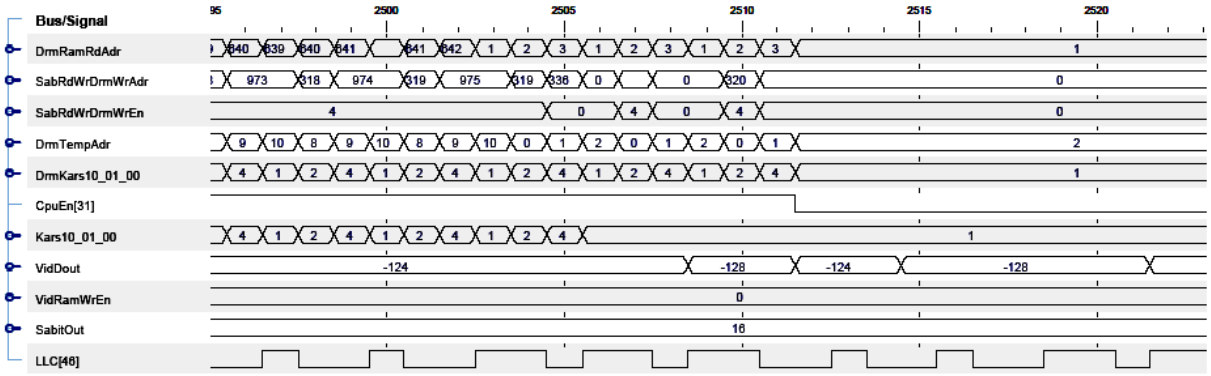
Uygulamada kullanılan HSA Emülatörünün “Adres ve Kontrol” ve “Video Giriş” bloklarının giriş ve çıkışlarında gözlenen bazı işaretler aşağıdaki şekillerde verilmiştir. Bu şekiller ChipScope Pro programı ile oluşturulan lojik analizör bloğunun devre çalışırken yaptığı kayıtlardan elde edilmiştir.



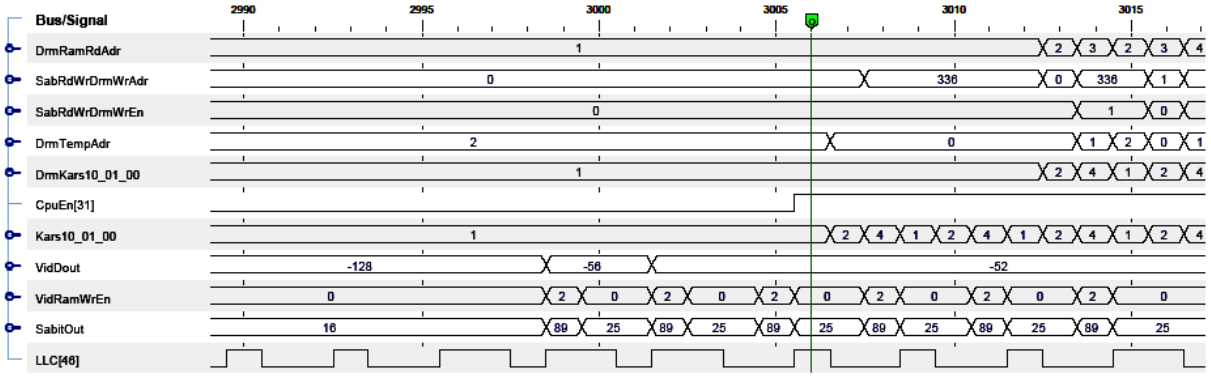
Şekil Ek 2.1 Yeni bir satırı işlenmeye başlarken işaretlerin durumu.



Şekil Ek 2.2 Satır işlenirken işaretlerin durumu.

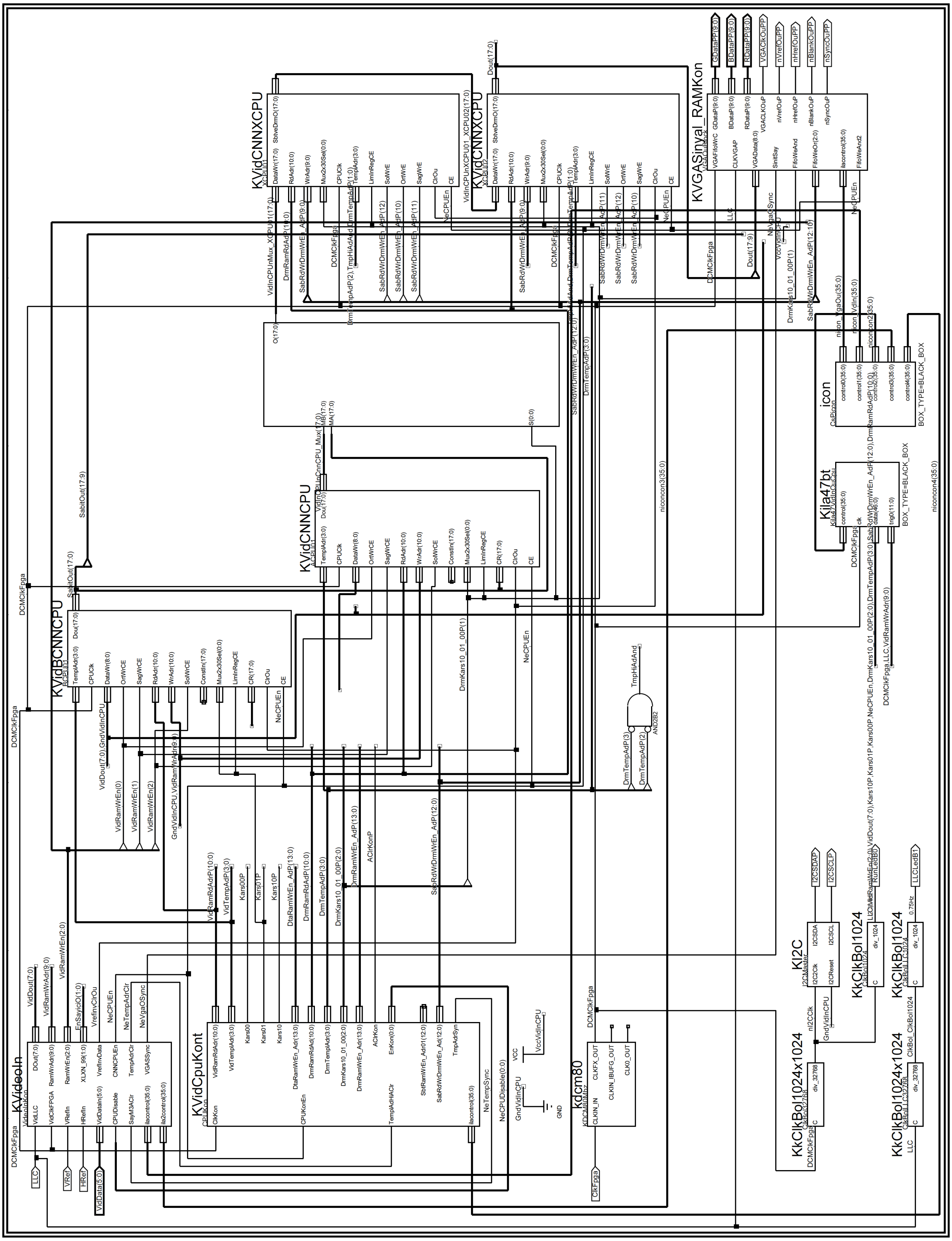


Şekil Ek 2.3 Satır sonuna gelindiğinde işaretlerin durumu.

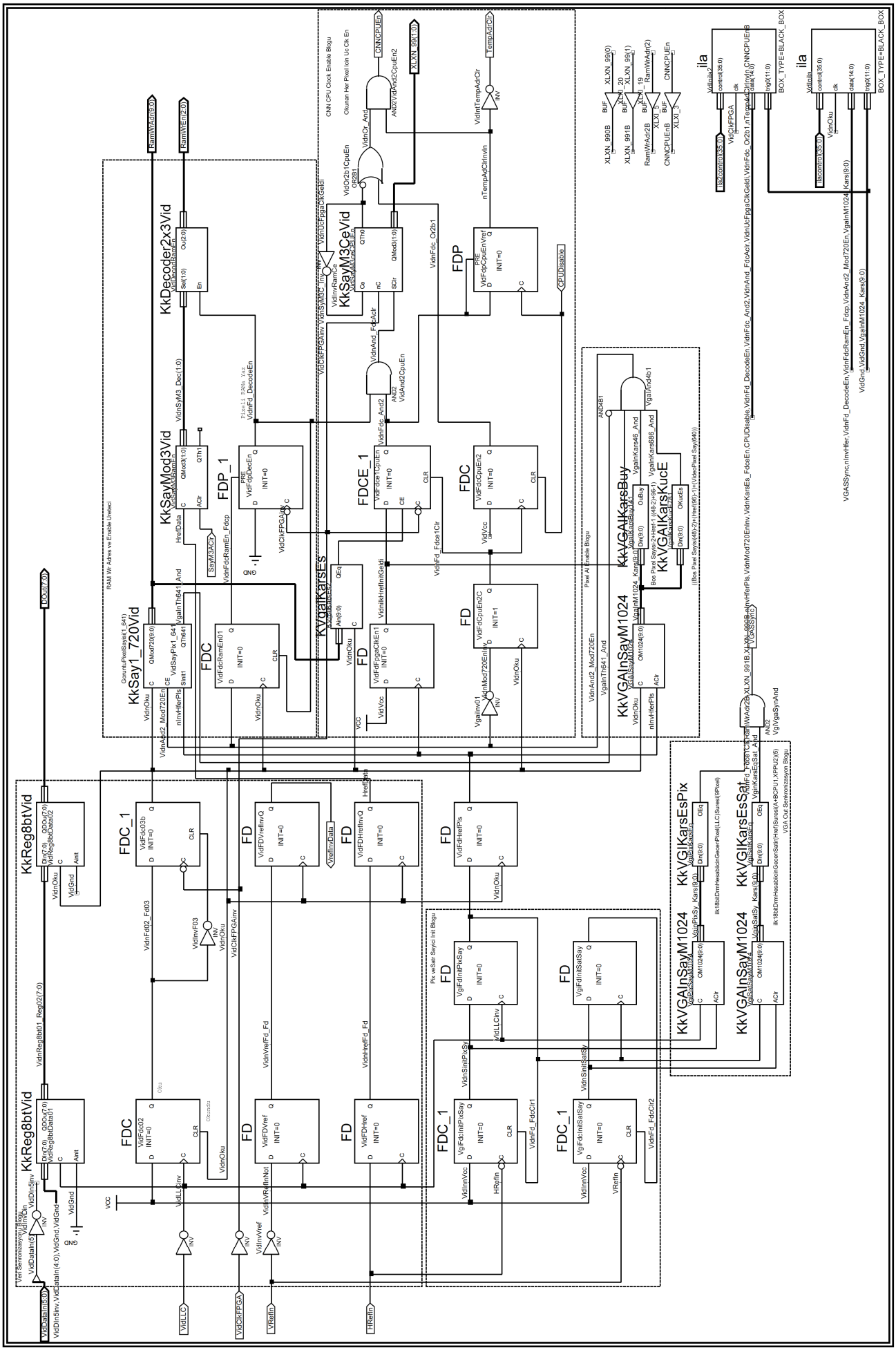


Şekil Ek 2.4 Bir sonraki satıra başlarken işaretlerin durumu.

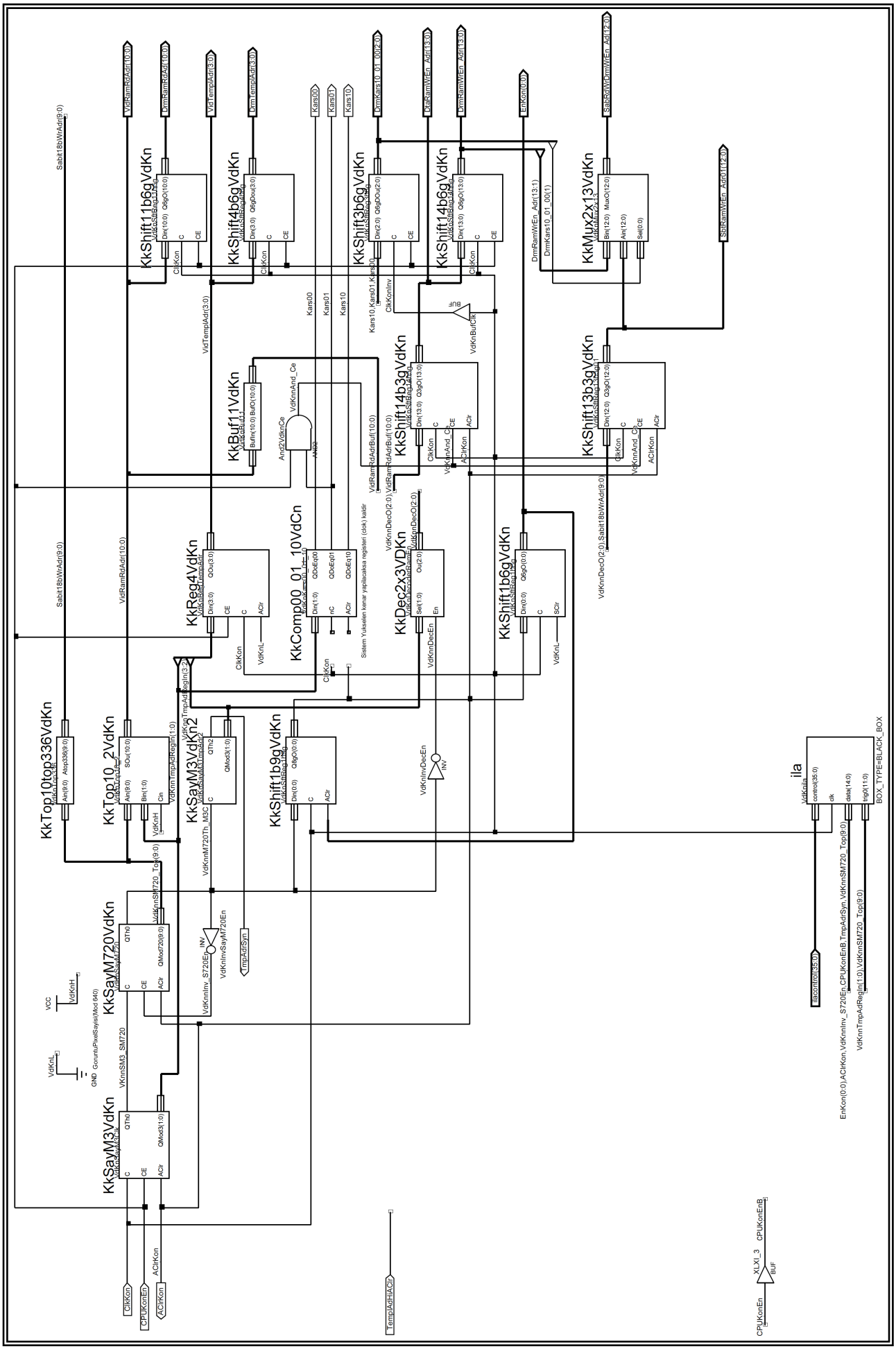
Ek 3 Uygulamada kullanılan HSA emülatörünün şematik çizimleri



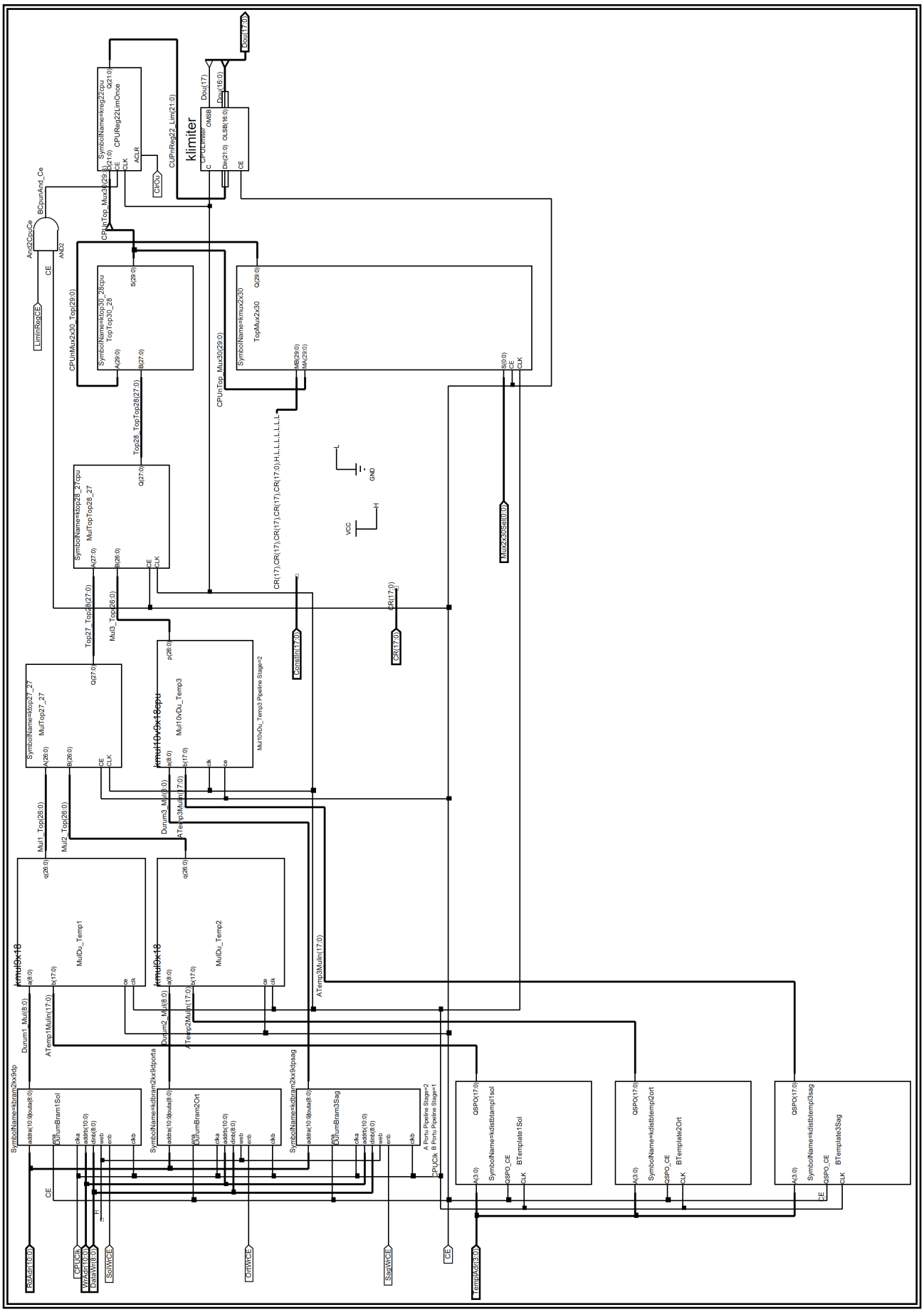
Şekil Ek 3.1 Emülatörün genel şeması



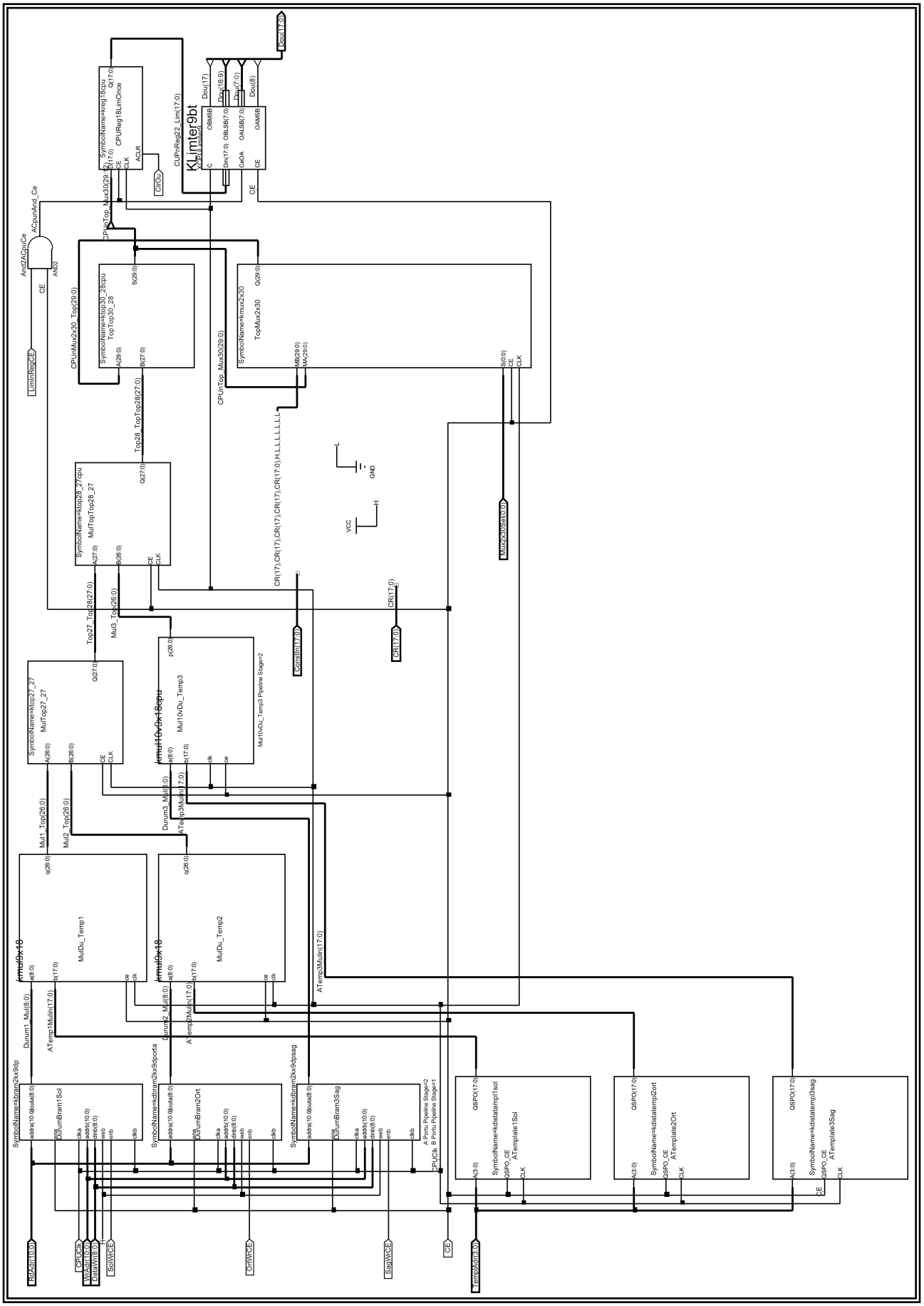
Şekil Ek 3.2 “Video Giriş” bloğu



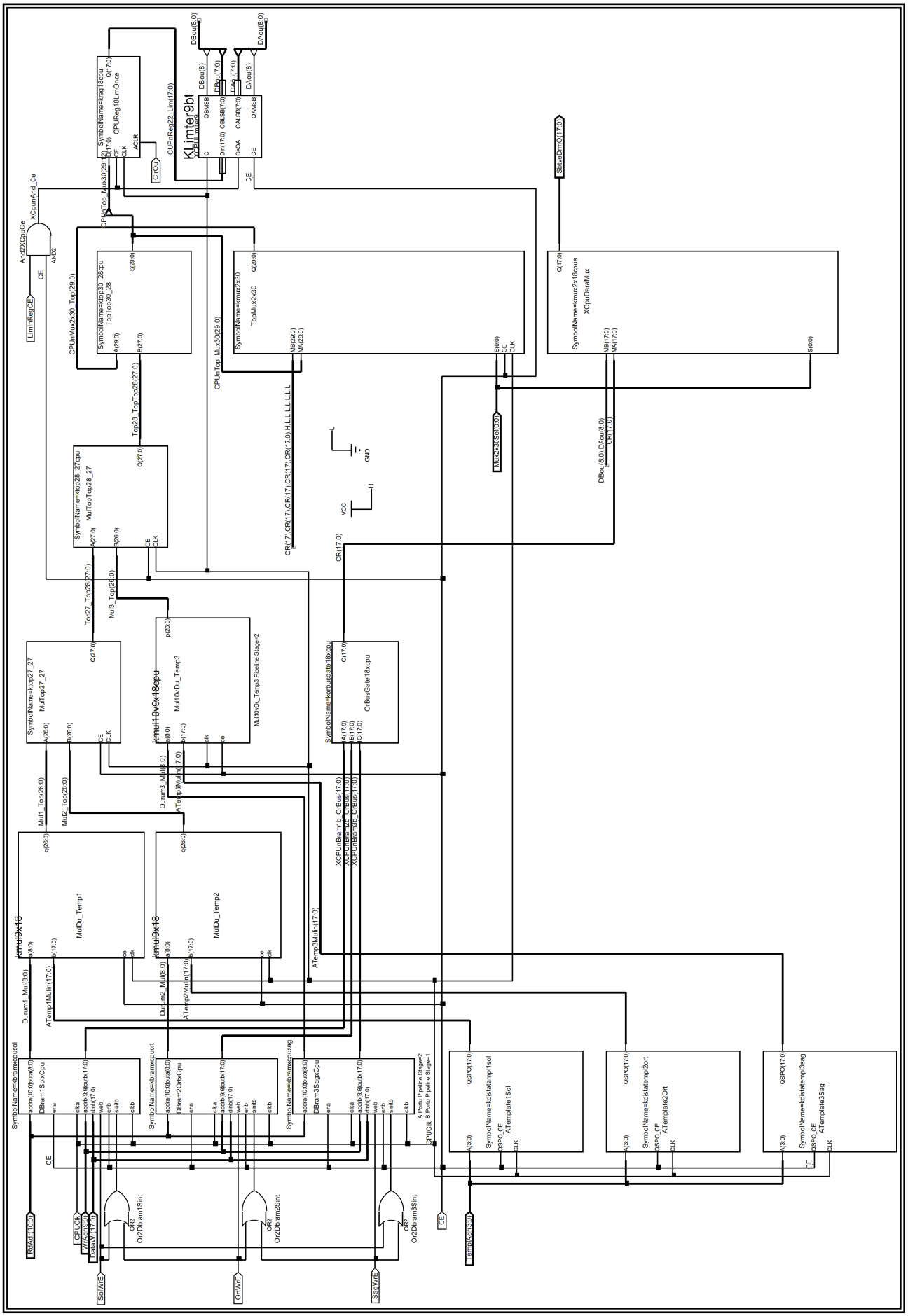
Şekil Ek 3.3 “Adres ve Kontrol” bloğu



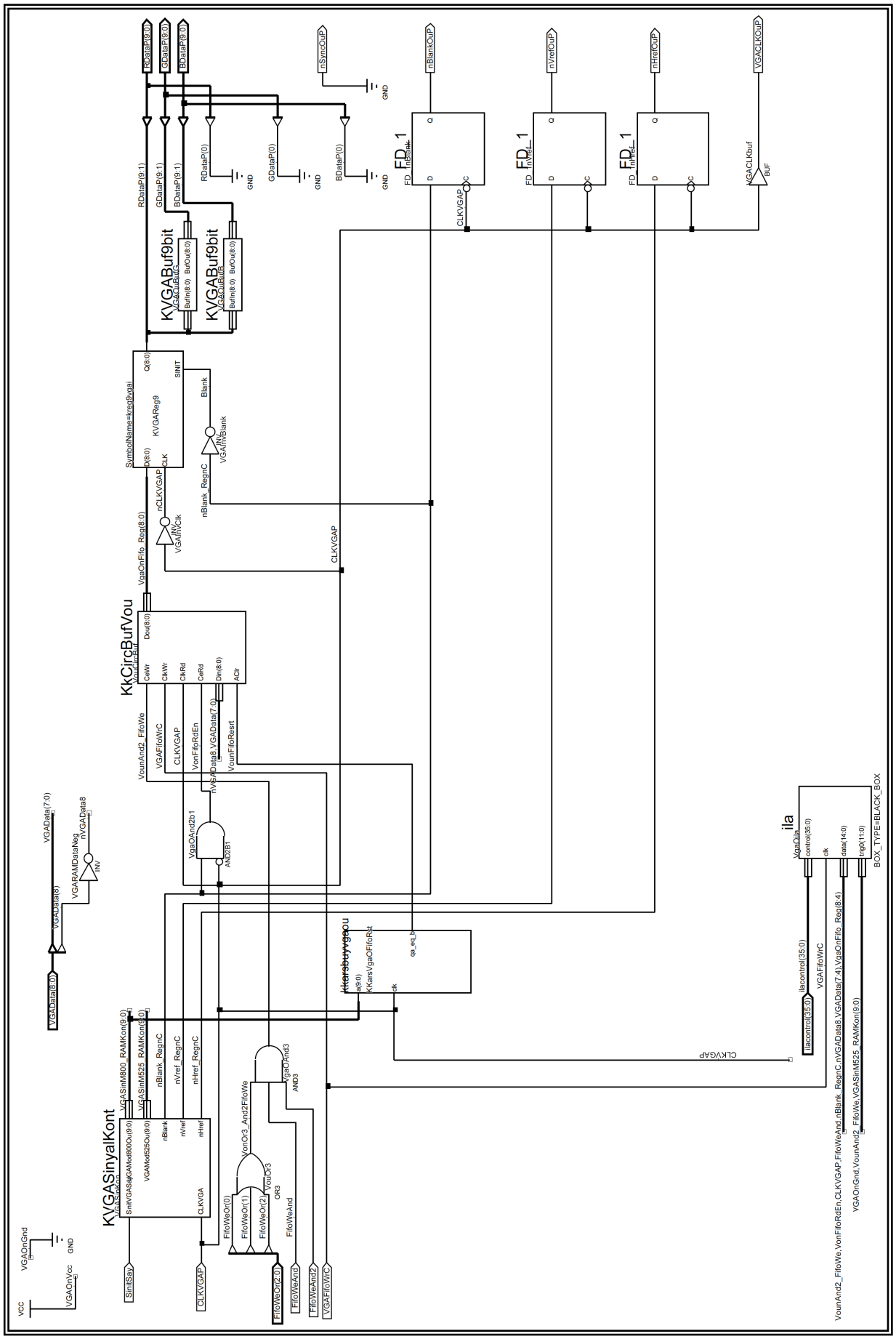
Şekil Ek 3.4 BPU bloğu



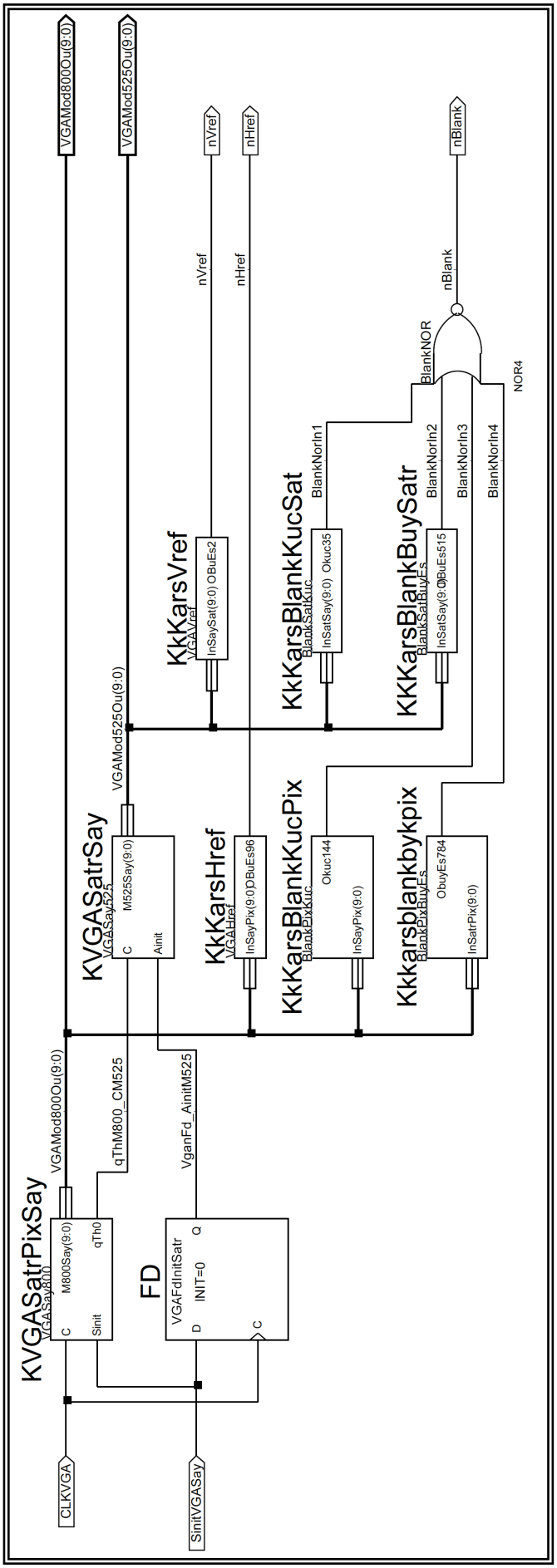
Şekil Ek 3.5 APU(1) bloğu



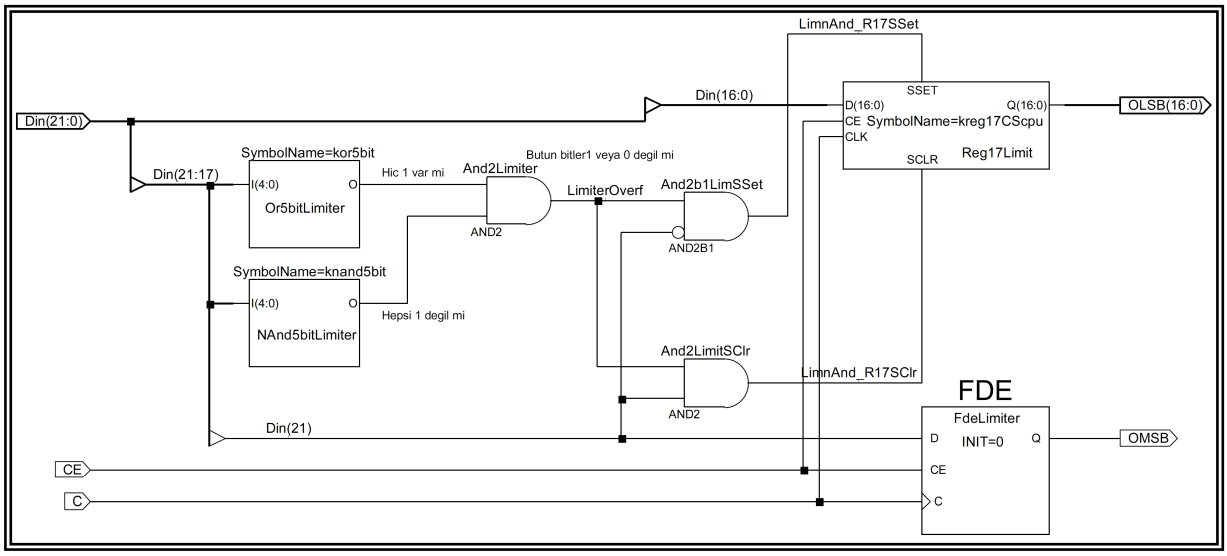
Şekil Ek 3.6 APU(n>1) blokları



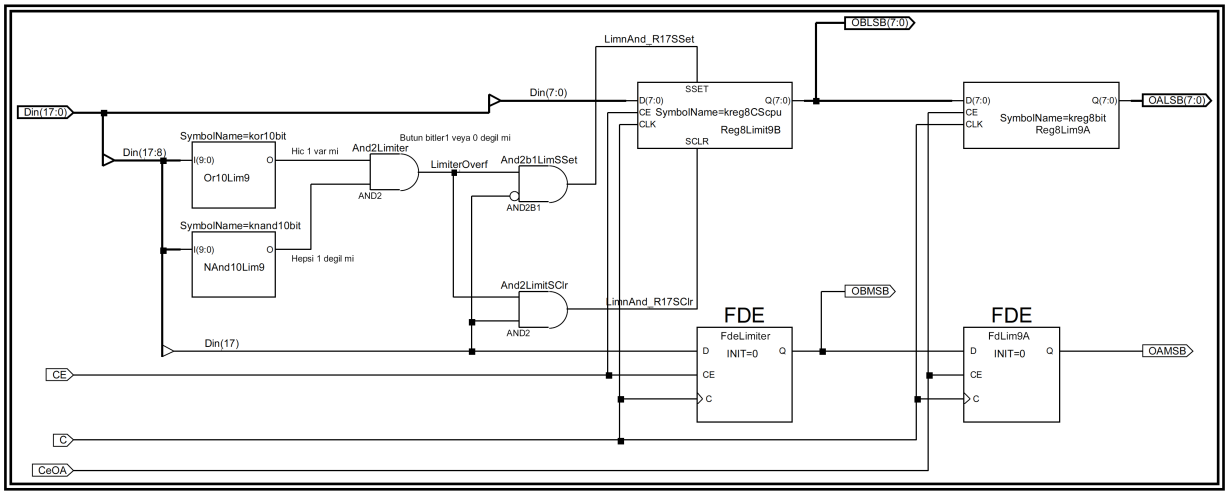
Şekil Ek 3.7 “Video Çıkış” bloğu



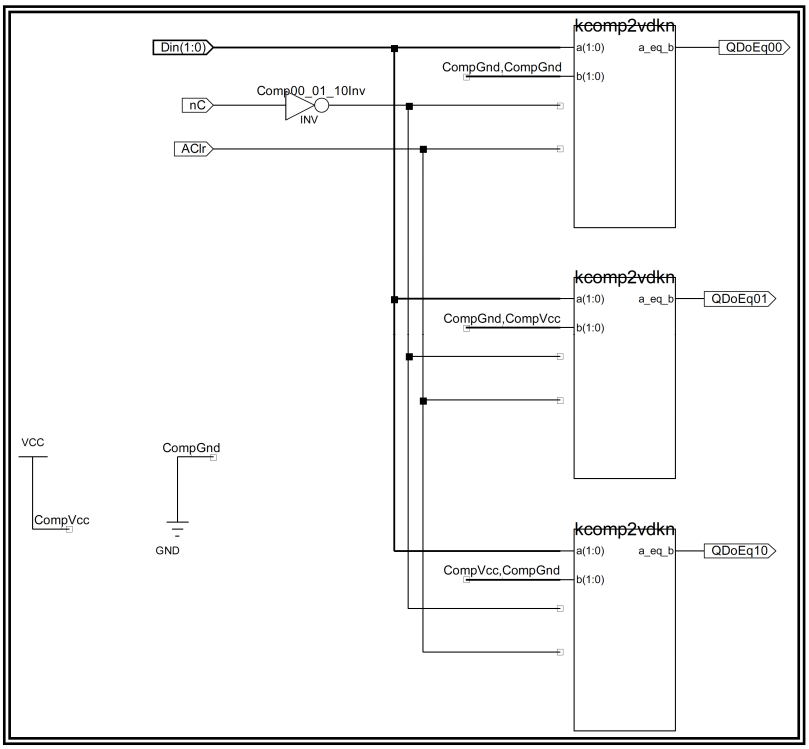
Şekil Ek 3.8 KVGASinyalKont bloğu



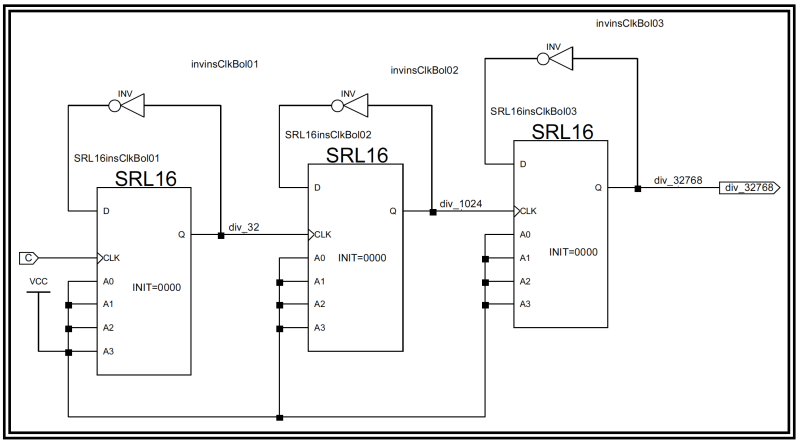
Şekil Ek 3.9 klimiter bloğu



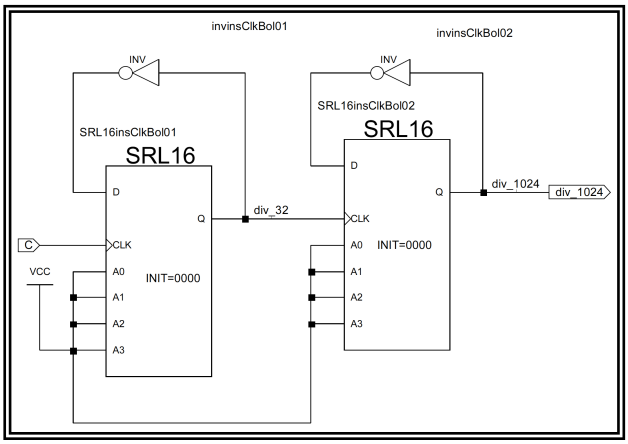
Şekil Ek 3.10 klimiter9bt bloğu



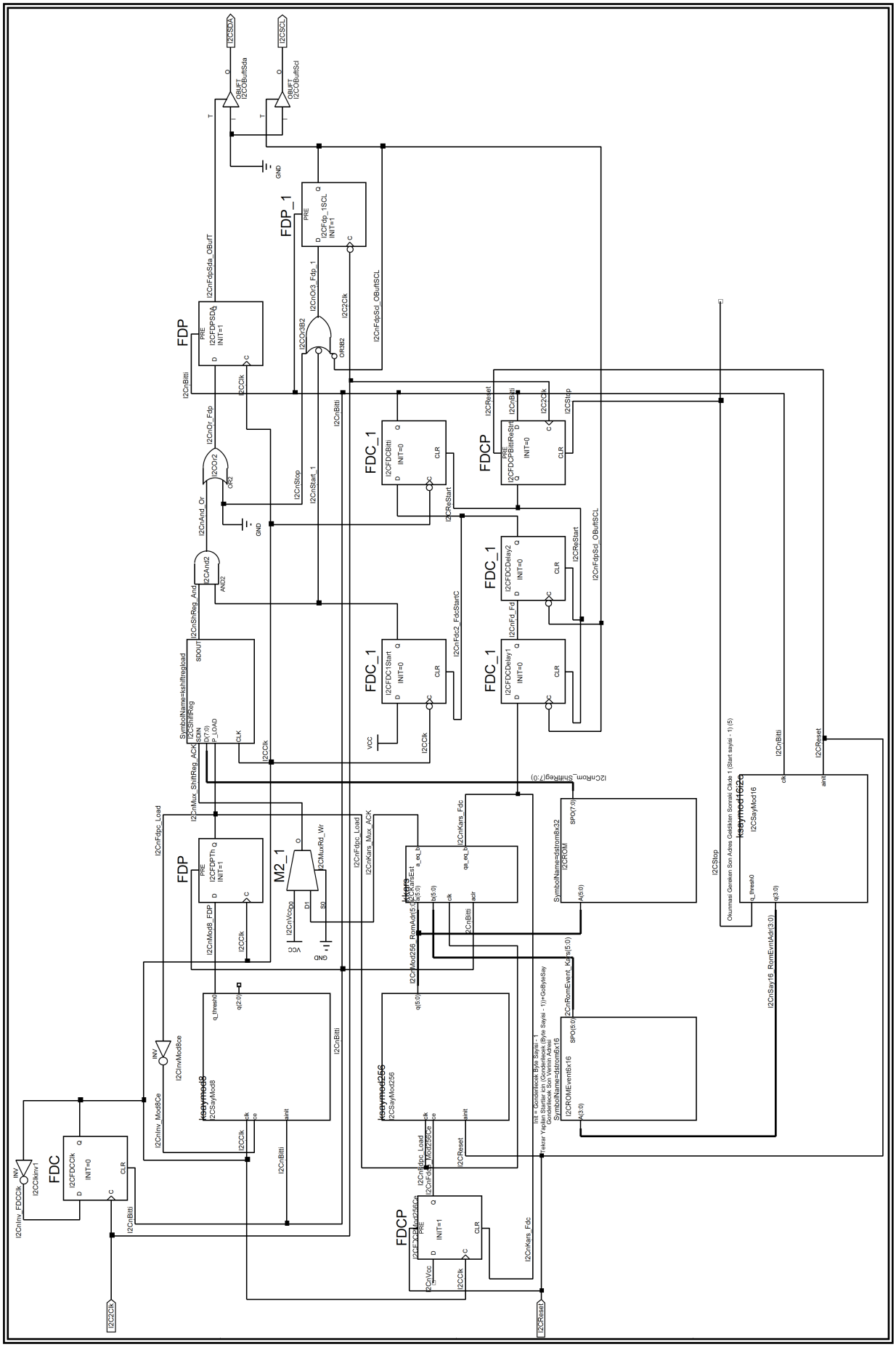
Şekil Ek 3.11 KkComp00_01_10VdCn bloğu



Şekil Ek 3.12 KkClkBol1024x1024 bloğu



Şekil Ek 3.13 KkClkBol1024 bloğu



Şekil Ek 3.14 I2C bloğu

ÖZGEÇMİŞ

Doğum tarihi	04.06.1977	
Doğum yeri	İstanbul	
Lise	1992 - 1995	Özel Üsküdar Fen Lisesi ve Bilfen Lisesi
Lisans	1995 - 1999	Yıldız Üniversitesi Elektrik-Elektronik Fak. Elektronik ve Haberleşme Müh. Bölümü
Yüksek Lisans	1999 - 2002	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Müh. Anabilim Dalı Elektronik Programı
Doktora	2002 - 2008	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Müh. Anabilim Dalı Elektronik Programı

Çalıştığı kurumlar

Ağustos - Eylül 1997	ONTROL A.Ş. AR-GE Bölümünde Genel Staj
Ağustos - Eylül 1998	ISIKO Ltd. Şti. Teknik Servis Bölümünde Mesleki Staj
Kasım 2000 - Aralık 2005	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Araştırma Görevlisi