İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

HÜCRESEL YAPAY SİNİR AĞI İŞLEMCİSİ TASARIMI VE FPGA GERÇEKLEMESİ

YÜKSEK LİSANS TEZİ

Volkan MERİÇ

Elektronik ve Haberleşme Mühendisliği Bölümü

Elektronik Mühendisliği Programı

HAZİRAN 2016

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

HÜCRESEL YAPAY SİNİR AĞI İŞLEMCİSİ TASARIMI VE FPGA GERÇEKLEMESİ

YÜKSEK LİSANS TEZİ

Volkan MERİÇ
(504101226)

Elektronik ve Haberleşme Mühendisliği Bölümü

Elektronik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Müştak Erhan YALÇIN

HAZİRAN 2016

İTÜ, Fen Bilimleri Enstitüsü'nün 504101226 numaralı Yüksek Lisans Öğrencisi Volkan MERİÇ, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "HÜCRESEL YAPAY SİNİR AĞI İŞLEMCİSİ TASARIMI VE FPGA GERÇEKLEMESİ" başlıklı tezini aşağıdaki imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Müştak Erhan YALÇIN** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Doc. Dr. Fethullah KARABİBER** Yıldız Teknik Üniversitesi	.....
	Yrd. Doc. Dr. Mustafa ALTUN İstanbul Teknik Üniversitesi	.....
		.....

Teslim Tarihi : **2 Mayıs 2016**
Savunma Tarihi : **9 Haziran 2016**

ÖNSÖZ

Bu tez çalışmasının gerçekleşmesini sağlayan değerli hocam Prof. Dr. Müştak Erhan Yalçın'a çok teşekkür ederim. Tez çalışması boyunca verdikleri destek için iş arkadaşlarıma teşekkür ederim. Ayrıca maddi ve manevi destekleri için aileme müteşekkirim.

Haziran 2016 Volkan MERİÇ

İÇİNDEKİLER

Sayfa

KISALTMALAR

AİB : Arithmetik İşlem Birimi
ALU : Arithmetic Logic Unit
ASIC : Application Specific Integrated Circuit
BRAM : Blok RAM
CMOS : Complementary Metal Oxide Semiconductor
CNN : Celluler Neural Network
CNN-EP : Celluler Neural Network Emulator Processor
CNN-UM : CNN Universal Machine
DDR : Double Data Rate
DSP : Celluler Neural Network
DVI : Digital Signal Processor
FIFO : First In First Out
FPGA : Field Programmable Gate Array (Sahada Programlanabilir Kapı Dizileri)
FPS : Frame Per Second(saniyedeki çerçeve sayısı)
G/Ç : Giriş / Çıkış
HD : High Definition (Yüksek Çözünürlük)
HDMI : High Definition Multimedia Interface
HYSA : Hücresel Yapay Sinir Ağları
HYSA-EM : Hücresel Yapay Sinir Ağları Evrensel Makinesi
HYSA-Ei : Hücresel Yapay Sinir Ağları Emülatör İşlemcisi
I2C : Inter-Integrated Circuit
ISim : ISE Simulator
IP : Intellectual Property
LCA : Logic Cell Array
LUT : Look Up Table (Veri Tablosu)
Mbyte : Mega Byte
Mux : Multiplexer (Çoğullayıcı)
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 Random Access Memory
TSA : Tüm Sinyal Aralığı
VHDL : VHSIC (Very High Speed Integrated Circuits) Hardware Description Language
VLSI : Very Large Scale Integration (Çok Büyük Ölçekli Tümleştirme)

SEMBOLLER

A	**:** Geri Besleme Şablonu.
B	**:** Giriş/Kontrol Şablonu.
C(i,j)	**:** i. satır ve j. sütunda yer alan hücre.
f(.)	**:** Hücrenin doğrusal olmayan çıkış fonksiyonu.
I	**:** Eşik Değeri.
r	**:** Etki alanı yarıçapı.
$\mathbf{S_r(i,j)}$	**:** C(i, j) hücresinin etki alanı.
t	**:** Zaman.
$\mathbf{T_s}$	**:** Euler adım uzunluğu.
$\mathbf{x_{ij}}$	**:** C(i, j) hücresinin durum değişkeni.
$\mathbf{y_{ij}}$	**:** C(i, j) hücresinin çıkışı.

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

HÜCRESEL YAPAY SİNİR AĞI İŞLEMCİSİ TASARIMI VE FPGA GERÇEKLEMESİ

ÖZET

Gelişen CMOS teknolojisi ile kameralar giderek daha fazla günlük hayatımızın içinde yer almaya başladılar. CMOS kameralar ticari olarak hazır, kolay temin edilebilir, kullanımı kolay, ucuz ve kaliteli görüntü üretebilir hale geldiler. Bu sensörlerden daha iyi yararlanma isteği görüntü işlemenin önemini giderek arttırıyor. Giriş görüntüsünü istenilen hale getirmeye veya görüntünün içerisinden istenilen bilgileri çıkarma işlemine görüntü işleme denir. Kontrol sisteminin hızlı cevap verme gereksinimi veya işlenmiş görüntünün asıl işle beraber olma isteği gerçek zamanlı görüntü işleme ihtiyacı yaratmaktadır.

Canlılar sinir ağı sayesinde gelen verileri paralel olarak işleyebilmektedir. Sinir hücreleri düşük frekansta çalışmasına rağmen paralel işleme yeteneği sayesinde yüksek işlem kabiliyetine sahiptir ve çok hızlı karar verirler. Canlıların sinir ağından esinlenilen Hücresel Yapay Sinir Ağları (HYSA) görüntü işleme fonksiyonu için oldukça uygundur ve gerçekleştirdiği görevin basit parametreler ile değiştirilebilmesi HYSA'yı bu alanda oldukça etkili kılmaktadır. HYSA yoğun işlem gücü gerektirmesi, donanımın paralel işlem kabiliyeti nedeni ile donanım çözümlerini yazılıma göre daha uygun kılmaktadır. HYSA'nın ilk olarak analog gerçeklemeleri yapılmıştır ama bunların eksiklikleri araştırmacıları sayısal donanım çözümlerine yöneltmiştir. Prototiplemeye imkan vermesi, tasarım kolaylığı ve tekrar programlanabilirliği ile FPGA'ler günümüzde sıklıkla sayısal donanım tasarımı için kullanılmaktadır.

Bu çalışmada, ayrık zamanlı HYSA emülatörü olarak yeni bir Hücresel Yapay Sinir Ağı (HYSA) işlemcisi mimarisi sunulmuştur. Bu mimariye Hücresel Yapay Sinir Ağı Emülatör İşlemcisi (HYSA-Eİ) ismi verilmiştir. Uygulamaya yönelik bir işlemci ve bunun özel kodlanması hem hız ihtiyacını karşılama hem de esneklik sağlaması açısından oldukça kullanışlı bir çözüm sunmaktadır. HYSA bir işlemcinin aritmetik fonksiyonu olarak tasarlanmıştır. Bu fonksiyonu çağırabilen basit bir işlemci tasarlanmış ve gerçeklenmiştir. Komut n kere 3×3'lük şablonlu HYSA fonksiyonunu girişe uygular. Komutun çıkışı farklı hafıza bölümlerinde bir sonraki komutta kullanılmak üzere kayıt edilmektedir. Giriş ve başlangıç koşulu kayıtlı DDR2 bölümlerinden seçilebilir. Bu mimari işlemci çekirdeğini çevre ünitesi kontrolörleri ile birleştirmiştir.

Ayrıca, HYSA-Eİ Spartan 6 XC6SLX45 FPGA üzerinde gerçeklenmiştir. İki adet örnek program ile test edilmiş, örnek program ve test çıktıları verilmiştir. Sistemin düşük lojik kaynak tüketmektedir. Belirli bir iterasyon sayısında HYSA çekirdeği 1600×900 videoyu 15 fps'ye kadar işleyebilmektedir. Performansı iterasyon sayısının artması ile azalmaktadır. Sistem 16,6 iterasyon/s işlem hızında çalışmaktadır.

DESIGN AND FPGA IMPLEMENTATION OF CELLULAR NEURAL NETWORK PROCESSOR

SUMMARY

We see cameras more more in our daily life thanks to the developments in CMOS technology. CMOS cameras are become ready to use, easy to buy and cheap with good picture quality. Desire to benefit more from these sensors is increasing importance of image processing. Image processing phenomena used for changing image into a desired form or extracting information from it. Rapid response and online processing demands increase the real time image processing need. In order to match the computational power of real time image processing, very fast or parallel data processing units are required.

Living beings process data parallel with their Neural Networks. Even though neurons working at low frequencies they can process large amount of data with short period of a time thanks their parallel data processing capabilities. Since it was invented in 1993, Cellular Neural Network (CNN) has become very approved in visual processing. The connections in the eye and neighbourhood approach in CNN builds the biological background for the use of CNN. Bioinspired data processing method Cellular Neural Networks are suitable for image processing application and their function can change with few parameters. This make CNN more powerful at image processing domain. Apart from their use in sensory data processing or information processing element, it can also use in solving partial differential equations or information generation.

In the content of study, first of all cellular neural network has been researched. CNN's are two dimensional, space invariant non-linear processing arrays. The architecture of CNN consists of rectangular cell arrays with mxn dimensions. The minimal unit of the CNN is called "cell". These cells consist of dynamic part, sum block and activation function. The main advantage of CNN is that; many image processing algorithms can be implemented on the same architecture. Function of CNN determined by template which consist of two matrices and one constant. The templates are essential element for CNN to process the information correctly. A large collection of templates is available for 2D linear and non-linear signal processing operations.

CNN has been implemented as hardware (analog or digital) or software (computer or DSP). Parallel computation capability of hardware gives superior computation power against software. CNN's first implemented as analog arrays. These implementations are very fast and has low power consumption. However, there are certain drawbacks of analog design such as low resolution, limited cell number, high sensitivity to heat and noise, complex and expensive design. These drawbacks redirect researcher to digital design. Field Programmable Gate Arrays are programmable silicon chips which functionality determined by hardware design language (HDL). FPGA's are been used successfully at digital hardware design, because of prototyping capabilities, easy to design and reprogrammability.

In this work, we propose a new Cellular Neural Network (CNN) processing core architecture for digital emulation of discrete time CNN. This architecture is called Cellular Neural Network Emulator Processor (CNN-EP). Application specific processor and its unique coding gives practical solution for speed requirements and flexibility. CNN is design and implemented as a function of arithmetic logic unit. The proposed system is capable of executing a CNN instruction. Also a simple processor which can run CNN function is designed and implemented. System stores captured input image from camera in external memory to be processed. After processing inputs, it writes results to external memory. Also evolution of the state variables in each iteration is monitored with DVI controller in the system. The instruction operates n times 3×3 template operation on the input. Designed architecture is capable of using different initial conditions, input and iteration count. Result of the instructions can be stored at different memory segments to be used by next instructions. Input and initial images can be selected from the stored DDR2 segments. In this way set of templates which can be applied in a particular order might process an information that is cannot processed with a single template operation. Data transfer mechanism between DDR memory and CNN Core is design to have maximum throughput. CNN Emulator Processors internal controller allowed multiple templates to be performed on the same or different images stored on the external memory. Hence CNN-EP execute OUT=CNN(U,X0,Temp,n) instruction where U input image, X0 initial image, OUT result image and Temp template addresses, n is iteration count. Several templates are stored at ROM in FPGA hence CNN Emulator Processor performs several instructions. CNN-EP has also an instruction memory to perform sequential CNN instruction.

CNN-EP architecture consist of CNN processor unit, CNN control unit and instruction decoder. FPGA also contains two ROM's inside, for store instruction and store different templates. CNN processor unit calculates CNN outputs. Instruction decoders decodes instructions arbitrarily and executes them. CNN controller control CNN arithmetic unit and responsible data synchronization between block RAMs, ROM and DDR2 memory. The architecture has been combined the core unit with peripheral control units. These peripheral control units are DDR2 memory controller, camera controller and DVI monitor controller. Camera controller transfers images camera to DDR2 memory by instruction decoder command. DDR2 memory controller uses specified ports to transfers data between FPGA and DDR2 memory by commands from CNN controller. DVI controller takes output image and drive LCD monitor with HD+ (1600x900) resolution @60 Hz refresh rate.

Furthermore, FPGA implementation also is given. CNN-EP implemented at Atlyss board which contains Spartan 6 XC6SLX45 FPGA and 128 Mbyte DDR2 SDRAM. VmodCAM has been used as camera. First of all, CNN processing core is implemented and tested with ISIM simulator. Test result which compares FPGA output with MATLAB output is given. Then this core imported into real time system. System has been tested with two different sample program and test results is given. System is designed to consume minimal logic resource. CNN Processor can process upto 1600×900 video @15 fps with certain iteration count. Debug and timing analysis of system done by chipscope software and platform cable from Xilinx. One CNN iteration including reading and writing image took 5 clock cycle. 108 MHz clock frequency used for processing unit. System performance decreases with iteration count. System exactly work at 16,6 iteration/s speed.

Comparison with other CNN emulator architecture is given and its contribution explained. When the existing CNN emulators are considered, RTCNNP and CASTLE architectures doesn't use external memory so they lacking capability of taking different initial conditions and input on run time. Also iteration count cannot change without reconfiguration and limited by FPGA resources. Implemented system proven to be efficient at manner of system resource and data transfers rates. The experimental results, which generated by different instructions have shown the capabilities of the system. Finally, hardware performance of system has been given.

1. GİRİŞ

1.1 Literatür Araştırması

Hücresel Yapay Sinir Ağları (HYSA) 1988'de Chua ve Yang tarafından ortaya konulmuştur [1, 2]. YSA'ya nazaran hücrelerin yerel olarak bağlanması HYSA'yı pratik olarak gerçeklenebilir çok boyutlu işlemci dizisi haline getirmiştir. Veriler belirli bir uzaklıktaki komşu hücreler arasında paylaşılır. Komşu olmadığı hücrelere sonraki iterasyonlarda yayılma etkisi ile dolaylı olarak etki eder. HYSA'lar özellikle görüntü işlemede geniş bir uygulama alanı bulmuştur. Yol planlama algoritması ve robot yönlendirmesinde de HYSA'lar kullanılmıştır [3]. Bu dizilerin uygulanabilirliği biyolojik retina, morphogenesis sistemi ve hücresel otomatlar gibi alanlarda araştırılmıştır. Bu kategorilerin uygulama alanı silicon retina, linear ve nonlinear spatial filtreleme, kenar tespiti, hareket tespiti, sıcak nokta tespiti parmak izi iyileştirmesi, matematiksel modelleme ve dalga yayılımı [4, 5] vb. olmuştur.

HYSA-EM [6] adı altında ilk programlanabilir analog HSA dizi bilgisayarı ortaya konmuştur. Analog gerçeklemeler oldukça etkili gözükmektedir. Tasarlanan HYSA-EM yapısı analog giriş çıkışı sahip HSA yapısı ACE4k [7] ismi ile 64x64 boyutunda, ACE16k [8] ismi ile 128x128 boyutunda gerçeklenmiştir. Bu analog VLSI gerçeklemeleri yüksek hızlı ve düşük güç tüketimlidir fakat bazı eksiklikleri bulunmaktadır. Şimdiye kadar işlem çözünürlüğü dijital eşdeğeri olarak 8 bite kadar tasarlanabilmiştir ve analog gerçeklemelerde birden çok görev ardışıl olarak işlendiğinde hataya yol açmaktadır. Ayrıca sınırlı hücre sayısı, yüksek sıcaklık ve gürültü duyarlılıkları, yüksek maliyet ve tasarım zorluğu analog yapıların diğer eksiklikleridir. Bu eksikliklerin giderilmesi için HYSA-EM'lerin sayısal emülatörleri ortaya atılmıştır.

Bu sayısal emülatörler farklı araçlar ile gerçeklenmiştir. Bilgisayarda yazılım simulasyonu olarak sunulmuştur. Bilgisayar gerçeklemeleri sadece algoritma tasarım aşamasında test platformu olarak kullanılmıştır. Farklı şablon ve itetasyon sayısı

etkileri bilgisayarlar üzerinde denenmiştir. Gerçek zamanlı tasarımlar ise DSP, GPU, FPGA ve ASIC üzerinde yapılmıştır. DSP'ler ucuz, kolay programlanabilen ve düşük güç tüketimli yapılardır. Özel işlem donanımları normal işlemcilere göre daha cazip kılsada doğasında paralel olan HYSA yapılarını tamamen ardışıl işlem olarak gerçeklemek verimli değildir. ASIC gerçeklemelerinin düşük güç tüketimi ve yüksek hızlarına rağmen tasarım zorluğu ve esneksizliği tekrar programlanabilir sayısal kapı dizileri (FPGA) çözümünü daha çekici hale getirmiştir.

Roland Tetzlaff ve ekibi tarafından CESAR [9] mimarisi ortaya atılmıştır. 128x128'lik 18 bit çözünürlüklü ve programlanabilir bu mimari NERO mimarisi [10] ile daha büyük ölçekli ağlara uygulanabilir hale getirilmiştir.

320x240 çözünürlüklü görüntüyü @25 fps hızında, 12 bit çözünürlük ile 500 CNN iterasyonuna kadar çözebilen CASTLE mimarisi [11] ortaya konmuştur. Bu mimariye çok boyutluluk eklenerek, çözünürlük, şablon boyutu, hücre sayısı ve boyut sayısı anlamında esneklik getirilerek FALCON mimarisi [12] adı altında sunulmuştur.

Yıldız Teknik Üniversitesinden gerçek zamanlı HYSA işlemcisi olarak RTCNNP [13] ismi ile yeni bir mimari sunulmuştur. Bu mimarinin genel özellikleri hafızaya ihtiyaç duymaması, her bir pikselin 3 saat darbesinde hesaplanabilmesi ve iterasyon sayısının gerçeklenen işlemci birimi sayısı sağlanmasıdır. Bu mimari 640x480 çözünürlükte @60 fps hızında test edilmiştir. Daha sonra bu mimarinin kontrol birimi geliştirilerek ve sınır koşulları eklenerek RTCNNP-v2 [14] ismi ile sunulmuş ve Full HD 1080p @54 hızında test edilmiştir. Bu mimariler Steadfast-1 ve Steadfast-2 olarak da isimlendirilmiştir. Steadfast mimarileri oldukça hızlıdırlar ve düşük lojik kaynak tüketirler, fakat harici hafıza kullanmadığı için farklı giriş ve başlangıç koşulları alamamaktadır. Ayrıca HYSA'nın iterasyon sayısının değişmesi için tekrar programlanması gerekmektedir ve iterasyon sayısı FPGA kaynakları ile sınırlıdır.

1.2 Tezin Amacı

Bu tezin amacı, FPGA'de HYSA durum denkleminin tamamını gerçekleyen gerçek zamanlı çalışan ve bütün parametreleri programlanabilen ayrık zamanlı Hücresel Yapay Sinir Ağı (HYSA) gerçeklemektir. Bir diğer amacı bu mimariyi bir işlemcinin Aritmetik İşlem Birimi olarak yerleştirip HYSA'yı tüm parametreleri ile

çağırılabilen bir komut haline getirmektir. Böylelikle ancak yazılım benzetimlerinde kullanılabilen her bir parametresi programlanabilen ve HYSA'nın tüm durum denklemini gerçekleyebilen bir mimari ortaya konmuş olacaktır. Bu mimaride sistem performansı iterasyon sayısı ile düşmesine rağmen donanım verimli tasarlanarak belirli bir iterasyon sayısına kadar 1600x900 çözünürlüğünde @15 fps hızında gerçek zamanlı çalışacaktır. Bu mimari tasarımı ayrıntı olarak sunulacaktır. Tüm bu sistem gerçek bir gömülü sistem için tasarlanacak, VHDL ile kodlanacak ve performansı test edilecektir.

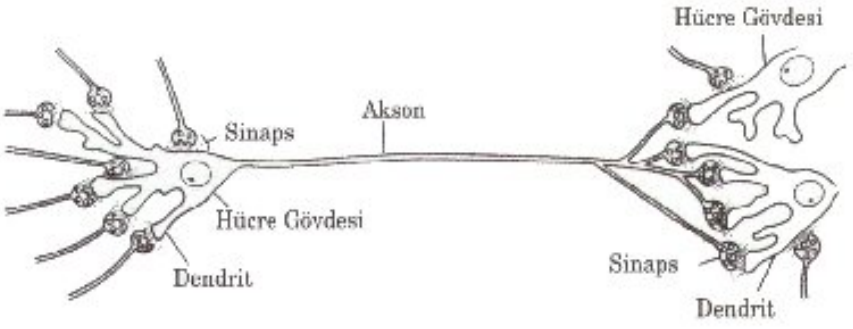
Çalışmanın içeriğinde ilk olarak Hücresel Yapay Sinir Ağları incelenmiştir. HYSA'larının ilk çıkışı, ihtiyacının ve tasarım temellerinin nereden geldiği anlatılmıştır. Ayrık zamanlı ve sürekli zamanlı HYSA'nın matematiksel denklemleri verilmiştir. Gerçeklenen yapılardaki varsayımlara değinilmiştir. Önceki HYSA gerçeklemeleri anlatılmıştır. HYSA-EM (Hücresel Yapay Sinir Ağı Evrensel Makinesi) anlatılmış ve bunun 3 farklı emülatörünün çalışma prensipleri anlatılmıştır.

Daha sonra tasarlanan sistemin mimarisi verilmiştir. Öncelikle FPGA teknolojisine değinilmiştir. Sistemin gerçeklemede kullanılan geliştirme ortamı tanıtılmıştır. HYSA-EM sayısal emülatörü olarak tasarlanan sistem, bir işlemcinin komut seti olarak gerçeklendiğinden dolayı bu sisteme Hücresel Yapay Sinir Ağı Emülatör İşlemcisi (HYSA-Eİ) ismi verilmiştir. HYSA-Eİ mimarisini komut işleme sistemi, HYSA fonksiyonu ve çevre üniteleri ayrıntılı tasarımı anlatılmıştır, FPGA gerçeklemeleri verilmiştir.

Son olarak, sonuçlar verilmiştir. HYSA-Eİ iki adet örnek program ile test edilmiş, örnek program ve test çıktıları verilmiştir. Tasarlanan sistemin performansı ölçülmüştür. Kullanılan donanım miktarı ve zaman performansı değerleri verilmiştir.

2. HÜCRESEL YAPAY SİNİR AĞLARI

Biyolojik sinir sisteminin en basit yapı taşı nöron olarak isimlendirilen sinir hücreleridir. Bir sinir hücresi dört ana bölümden oluşmaktadır; dendrit, akson, çekirdek ve bağlantılar. Basit bir sinir hücresi Şekil 2.1'de gösterilmiştir.



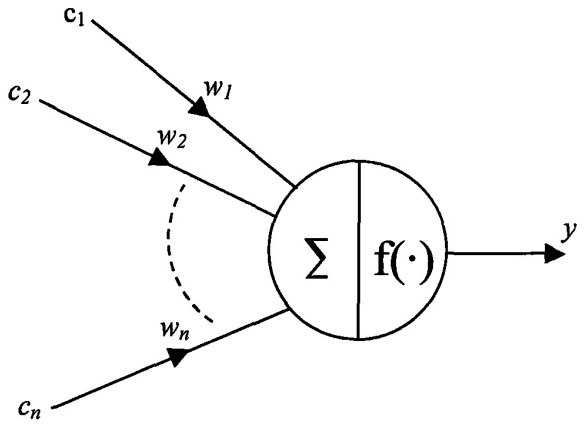
Şekil 2.1 : Basit bir sinir hücresi.

Dendritler sinir hücresinin ucunda bulunur ve görevi bağlı olduğu diğer nöronlardan veya duyu organlarından gelen sinyalleri çekirdeğe iletmektir. Toplanan sinyallerden bir çıkış üretilip üretilmeyeceği kararı verilir. Eğer üretilecekse çekirdek çıkış sinyallerini aksona iletir. Bu sinyaller akson tarafından işlenerek nöronun diğer ucunda bulunan bağlantılara yani diğer bir sinir hücresi veya terminal organına gönderilir. Bağlantılar ise yeni üretilen sinyalleri diğer nöronlara iletir.

Nöronlar arası bağlantı sinaps adı verilen dendritlerdeki geçitler ile olur. Sinapslar nöronun girişine gelen işareti belli bir ağırlık katsayısı ile çarpar. Bu şekilde bir nöronun aldığı sinyallerin ağırlıklı toplamı nöronun etkinlik seviyesini belirtir. Böyle bir yapay nöronun gösterimi Şekil 2.2'de gösterilmiştir.

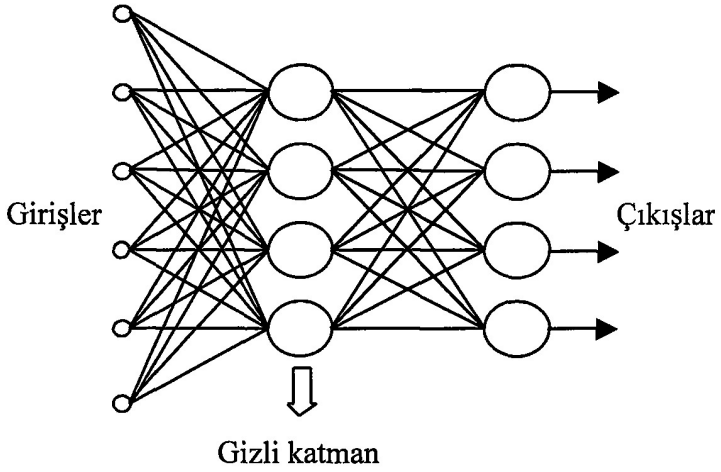
Buradaki sinir hücresinde $c_1, c_2 ... c_n$ giriş değerleri, $w_1, w_2 ... w_n$ girişin etki miktarını belirleyen, giriş ile çarpılan ağırlık katsayılarıdır. f fonksiyonu hücrenin üreteceği çıkışı belirler ve doğrusal olmayan bir fonksiyon seçilir.

Basit yapıya sahip sinir hücrelerinin bir arada etkileşim halinde oldukça karmaşık işlemleri yerine getirebilen bir yapıya dönüşebilmektedir. En genel sinir ağı modeli Şekil 2.3'te gösterilmiştir. Her nöronun birçok girişi ve tek çıkışı vardır. Bunlar



Şekil 2.2 : Basit bir sinir hücresinin matematiksel gösterimi.

birbiri ile bağlanarak sistem yayılır. Sistem içerisindeki her bir nöronun çıkışı bir diğer nöronu etkiler, böylelikle sistemin davranışını etkiler.



Şekil 2.3 : Basit bir Yapay Sinir Ağı modeli.

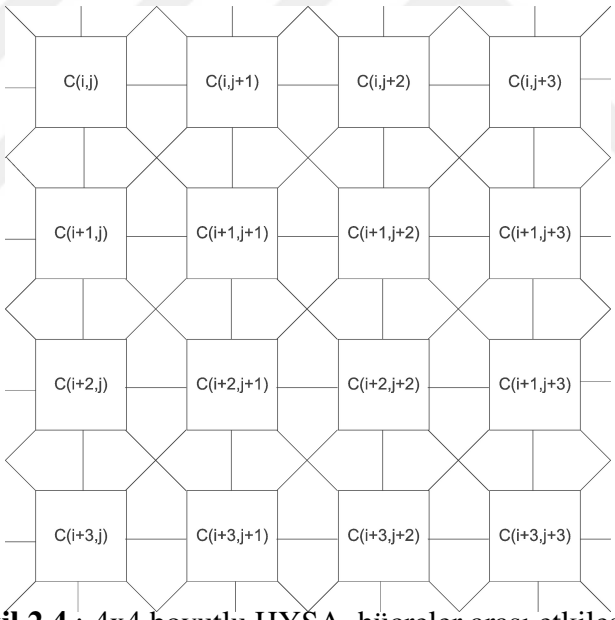
HYSA ilk olarak Yapay Sinir Ağlarının (YSA) gerçeklenebilir alternatifi olarak sunulmuştur. Hopfield sinir ağlarının özel bir kümesi olan bu bilgi işleme sistemi Hücresel Yapay Sinir Ağları (HYSA) olarak isimlendirilir. HYSA iki temele kaynağa dayanmaktadır; Sinir Ağları ve VLSI gerçeklemesi için uygun olan Hücresel Otomatlar (Celluler Automata). HYSA hücresel otomatların moore komşuluğunu kullanmaktadır. 2 boyutlu sinyal işleme özellikle görüntü işleme ve sinyal üretiminde uygulama alanı bulmuştur.

HYSA'nın en küçük birimine hücre adı verilir. Bu hücrelerde toplama, aktivasyon fonksiyonu ve geçici rejim oluşturan dinamik birim bulunur. HYSA'nın gerçekleştireceği göre iki adet matris ve bir sabit ile belirlenir. Bu değişkenlere şablon ismi verilir. Şablonun değiştirilmesi ile HYSA'nın gerçekleştirdiği görev değişir. Aktivasyon fonksiyonu olarak lineer olmayan fonksiyon kullanılır. Bütün hücreler birbirine bağlı olmamasına rağmen iterasyon ilerledikçe yani sonuca yakınsadıkça yayılma etkisi ile birbirini etkilerler.

2.1 Chua-Yang Modeli

Standart Hücresel Yapay Sinir Ağı boyutu kadar hücreler dizininden oluşur. HYSA mimarisindeki hücreler belirli komşuluk boyutunda en yakın komşularına bağlıdırlar.

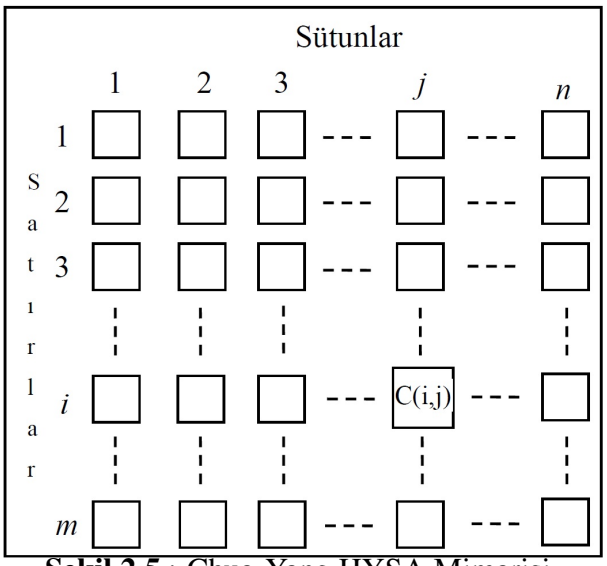
Bir 4x4 hücreli bir HYSA'da hücrelerin etkileştiği komşuları Şekil 2.4'te gösterilmiştir [1].



Şekil 2.4 : 4x4 boyutlu HYSA, hücreler arası etkileşim.

Görüntü işlemede her bir hücre bir pikseli temsil eder. i'inci satır j'inci sütundaki piksele $C(i,j)$ hücresi adı verilir. HYSA'da işlenilen görüntünün çıkışı kararlı bir duruma yakınsar. Standart HYSA mimarisi Şekil 2.5'da verilmiştir.

Bu bölümde kısaca HYSA standart modeli Chua-Yang matematiksel modeli ile anlatılacaktır. Bu modelin sürekli zaman HYSA'daki ve ayrık zaman HYSA'daki matematiksel formülleri verilecektir.



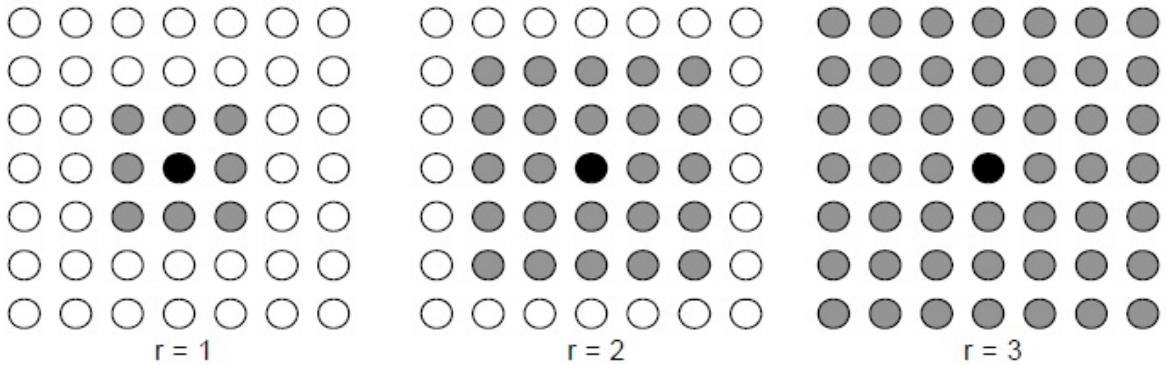
Şekil 2.5 : Chua-Yang HYSA Mimarisi.

2.1.1 Etki Küresi

Bir HYSA'da her hücre çevresindeki hücrelerle iletişim kurar, yani bir iterasyonda sadece komşuluk sınırları içerisindeki hücrelerden etkilenir. Etkileşim işlem yapılan hücre merkezde olmak üzere dışarı doğru yayılım yapar. HYSA komşuluğu aşağıda verilen denklemle tanımlanır.

$$S_r(i,j) = \{C(k,l) | max\{|k-i|, |l-j|\} \leqslant r\} \qquad 1 \leqslant k \leqslant M \ , \ 1 \leqslant l \leqslant N \tag{2.1}$$

Genellikle r=1 yarıçapı kullanılır. Bir HYSA hücresini r yarıçapı etki küresindeki komşulukları Şekil 2.6'de verilmiştir. Etki küresi içerisinde $(2r+1)^2$ hücre vardır.



Şekil 2.6 : Farklı komşuluk derecelerine göre tek bir hücrenin etki alanı.

2.1.2 Sürekli Zamanlı HYSA

Hücresel Yapay Sinir Ağları hücrelerin lineer olmayan dinamik sistemler olarak modellendiği dikdörtgen hücre dizisidir. Bu dinamik sistem durum denklemi,

çıkış denklemi, sınır koşulları ve başlangıç koşulları ile tanımlanan bir diferansiyel denklemdir.

Durum Denklemi:

r komşuluklu standart sürekli zaman Chua-Yang HYSA modeli durum denklemi:

$$\frac{dx_{i,j}(t)}{dt} = -x_{i,j}(t) + \sum_{k,l=-r}^{r} A_{i,j;k,l} y_{i+k,j+l}(t) + \sum_{k,l=-r}^{r} B_{i,j;k,l} u_{i+k,j+l} + z_{i,j} \qquad (2.2)$$

$x_{i,j}$ durum değişkeni,

$z_{i,j}$ Eşik değeri,

$u_{i,j}$ Hücrenin girişi,

$y_{i,j}$ Hücrenin çıkışı

$a_{(i,j;k,l)}$ geri besleme ağırlık katsayısı/şablonu,

$b_{(i,j;k,l)}$ giriş ağırlık katsayısı/kontrol şablonu

(i,j) hücrenin konumunu, (k,l) hücrenin komşuluklarını belirler. Bir hücreye geri besleme şablonu A durum değişkeni x ile konvolüsyonu, kontrol şablonu B giriş u ile konvolüsyonu şeklinde etki eder. Durum değişkeni x ,işlem başlamadan önce, başlangıç değeri (x_0) ile ilklendirilir. Kararlı Chua-Yang HYSA modelinin çıkışı $\{+1,-1\}$ değerine yakınsar, buna ikili çıkış özelliği denir. Yakınsama için gerekli iterasyon sayısı uygulamaya ve girişin büyüklüğüne bağlıdır. r = 1 komşuluklu A,B şablonları aşağıdaki gibi ifade edilir.

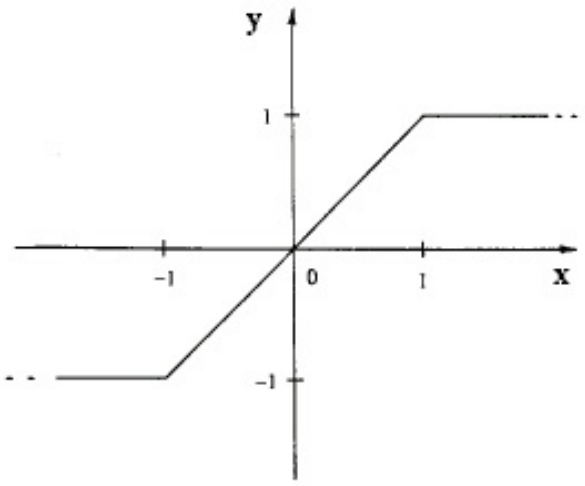
$$A = \begin{pmatrix} a_{-1,-1} a_{-1,0} a_{-1,1} \\ a_{0,-1} \quad a_{0,0} \quad a_{0,1} \\ a_{1,-1} \quad a_{1,0} \quad a_{1,1} \end{pmatrix}, B = \begin{pmatrix} b_{-1,-1} b_{-1,0} b_{-1,1} \\ b_{0,-1} \quad b_{0,0} \quad b_{0,1} \\ b_{1,-1} \quad b_{1,0} \quad b_{1,1} \end{pmatrix} \text{ve } z_{i,j}$$

Çıkış Fonksiyonu:

Çıkış fonksiyonu HYSA'da lineer olmayan tek fonksiyondur, yani yapının lineer olmamasını bu fonksiyonu sağlar. Genellik kısmi doğrusal fonksiyon kullanılır, bu fonksiyon Şekil 2.7'de verilmiştir. Kısmi doğrusal fonksiyonun matematiksel ifadesi:

$$y_{i,j}(t) = f(x_{i,j}(t)) = 0.5(|x_{i,j}(t)+1| - |x_{i,j}(t)-1|) \qquad (2.3)$$

Sınır Koşulları:



Şekil 2.7 : Çıkış eşik aktivasyon fonksiyonu.

Sınır koşulları kenarlardaki hücrelerin dizinin sınırları dışında kalan komşu hücreleridir. Bu sanal hücrelere değer atama gereksinimi ortaya çıkmıştır. Bu ihtiyaç dikdörtgen dizinin çevresine sanal hücreler ekleyerek giderilebilir. Sınır koşulları için üç genel yaklaşım bulunmaktadır.

Dirichlet Sınır Koşulu; Sanal sınır hücrelerine sabit genellikle sıfır eklenir.

Dairesel Sınır Koşulu; Sanal sınır hücrelerine dizinin diğer tarafındaki kenar hücrelerinin giriş ve durum değerleri verilir.

Neumann Sınır Koşulu; Sanal sınır hücrelerine dizinin aynı tarafındaki kenar hücrelerinin giriş ve durum değerleri verilir.

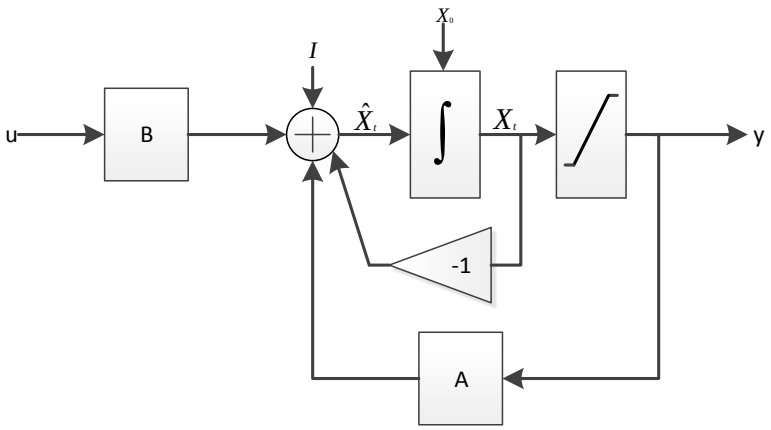
Zamandan ve Konumdan Bağımsız HYSA:

Nerdeyse bütün HYSA gerçeklemelerinde $z_{i,j}$ değerinin her bir hücrede sabit olduğu ve $a_{i,j;k,l}$, $b_{i,j;k,l}$ matrislerinin hücrenin konumundan bağımsız olduğu varsayılır. Buna zamandan ve konumdan bağımsız HYSA ismi verilir. Böylelikle $r = 1$ komşuluk için durum denklemi aşağıdaki hali alır:

$$\dot{x}_{i,j}(t) = -x_{i,j}(t) + \sum_{k,l=-1}^{1} A_{k,l} y_{i+k,j+l}(t) + \sum_{k,l=-1}^{1} B_{k,l} u_{i+k,j+l} + I \qquad (2.4)$$

Tek bir HYSA hücresi blok şeması Şekil 2.8'te verilmiştir.

2.1.3 Ayrık Zamanlı HYSA



Şekil 2.8 : Bir hücrenin Chua-Yang HYSA Modeli.

Durum değişkenlerinin türev içermesinden dolayı ağ denklemleri diferansiyel denklemleridir. Diferansiyel denklemlerin sayısal çözümü yaklaşıklık yöntemleridir. HYSA'da kolay gerçeklenebilirliği ve yakın sonuçlar vermesi nedeni ile genellikle ileri Euler yaklaşıklığı kullanılır.

(2.2) ve (2.3) $t = nT_s$ ile örneklenerek ileri Euler yaklaşıklık formülü:

$$\dot{x_(t)} \backsimeq \frac{x((n+1)T_s) - x(nT_s)}{T_s} = \frac{x((n+1) - x(n))}{T_s} \tag{2.5}$$

$$\frac{x((n+1)T_s) - x(nT_s))}{T_s} = -x_{i,j}(n) + \sum_{k,l=-1}^{1} A_{k,l} y_{i+k,j+l}(n) + \sum_{k,l=-1}^{1} B_{k,l} u_{i+k,j+l}(n) + I \tag{2.6}$$

$$x_{i,j}(n+1) = (1-T_s)x_{i,j}(n) + T_s \left(\sum_{k,l=-1}^{1} a_{k,l} y_{i+k,j+l}(n) + \sum_{k,l=-1}^{1} b_{k,l} u_{i+k,j+l}(n) + I \right) \tag{2.7}$$

Bu denklemi basitleştirmenin iki yolu vardır; birinci Tüm Sinyal Aralığı (TSA) ismi ile adlandırılan yöntemdir. Diğeri örnekleme aralığı $T_s = 1$ alınarak elde edilir. TSA modelinde durum değişkeni x_{ij} doyuma ulaştığında çıkış y_{ij}'de doyuma ulaşacağından çıkmıştır. Böylelikle çıkış durumlara eşit alınırsa ağ denklemlerinde lineerlik sağlanır.

$$y_{ij} = f(x_{ij}) = x_{ij} \tag{2.8}$$

Böylelikle denklem

$$x_{ij}(n+1) = \sum_{k,l=-1}^{1} A^*_{k,l} x_{i+k,j+l}(n) + \sum_{k,l=-1}^{1} B^*_{k,l} u_{i+k,j+l}(n) + I^*) \tag{2.9}$$

$$A^* = \begin{pmatrix} T_s a_{-1,-1} & T_s a_{-1,0} & T_s a_{-1,1} \\ T_s a_{0,-1} & 1 - T_s(1 - a_{0,0}) & T_s a_{0,1} \\ T_s a_{1,-1} & T_s a_{1,0} & T_s a_{1,1} \end{pmatrix}, B^* = T_s \cdot B \text{ ve } I^* = T_s \cdot I$$

(2.9) denklemi 2 parçalı, olarak yazılabilir. (2.10) denklemindeki g_{ij} her piksel için bir kere hesaplanması yeterlidir. (2.11) denklemindeki $v_{ij}(n)$ her bir iterasyonda hesaplanır.

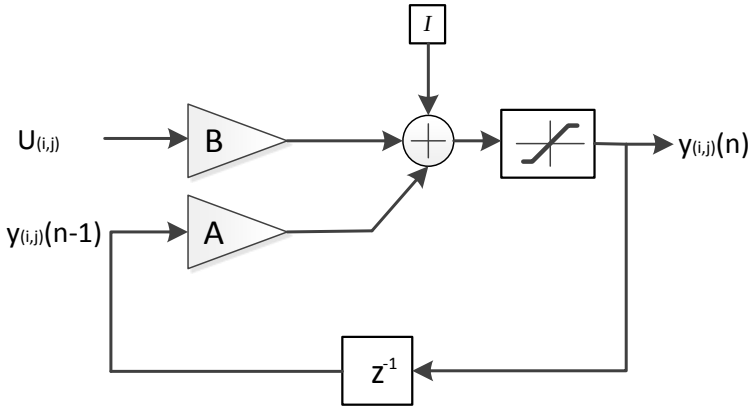
$$g_{ij} = \sum_{k,l=-1}^{1} B^*_{k,l} u_{i+k,j+l}(n) + I^* \tag{2.10}$$

$$v_{ij}(n) = \sum_{k,l=-1}^{1} A^*_{k,l} x_{i+k,j+l}(n) + g_{ij} \tag{2.11}$$

Önerilen sistemde durum değişkeni ve çıkış kayıt edilerek her iki yöntem de kullanılabilir olarak tasarlanmıştır. Bu tezde durum değişkeni ve çıkışın ikisi de kayıt edilerek her iki durum denklemini gerçekleme olanağı verilmiştir. Yapılan denemelerde $T_s = 1$ alınarak HYSA durum denkleminin ayrık zamanlı hali sadeleştirilmiştir:

$$x_{i,j}(n+1) = \sum_{k,l=-1}^{1} A_{k,l} y_{i+k,j+l}(n) + \sum_{k,l=-1}^{1} B_{k,l} u_{i+k,j+l}(n) + I \tag{2.12}$$

Ayrık zamanlı HYSA modeli 2.9 verilmiştir.

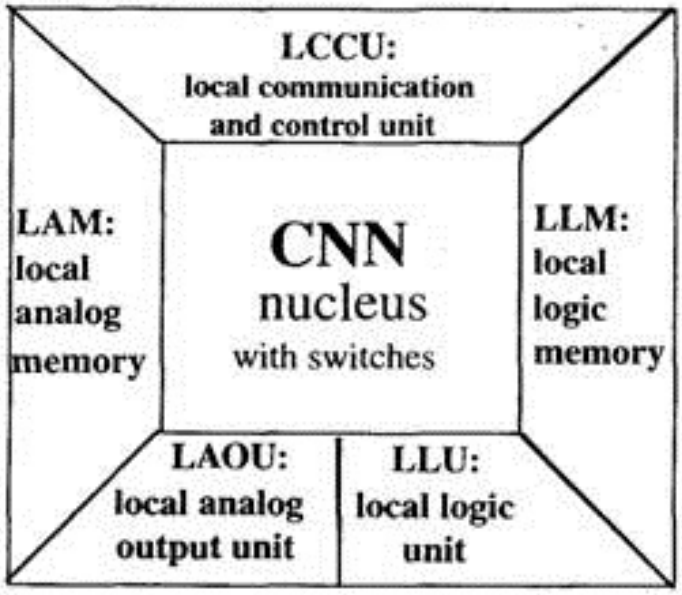


Şekil 2.9 : Ayrık zamanlı HYSA modeli.

Girişin kontrol şablonu ile konvolüsyonu, bir önceki sonuç geri besleme ile konvolüsyonu, eşik değeri toplanarak durum x hesaplanır. İlk çevrimde bir önceki çıkış yerine başlangıç koşulu kullanılır. Durum aktivasyon enerjisi fonksiyonundan geçirilerek sonuç elde edilir. İstenilen sonuç elde edilene kadar veya belirli bir iterasyon sayısı bu işlem tekrar edilerek sonuç elde edilir.

2.2 HYSA-EM ve Emülatörleri

HYSA Evrensel Makinesi (HYSA-EM) (CNN Universal Machine (CNN-UM)) ilk tek ongada gerçeklenmiş gerçek zamanlı algoritmik programlanabilir analog dizi işlemcisidir [6]. Analog ve lojik işlemlerin (analojik) bir arada yer aldığı programlanabilir bir işlemci yapısıdır. Görüntü işleme analog kısım tarafından, kontrol ve programlama işlemleri lojik kısım tarafından yönetilir. Mimari yapısı Şekil 2.10'te verilmiştir.



Şekil 2.10 : HYSA-EM mimarisi.

HYSA-EM mimarisinin bir hücresi, HYSA işlem çekirdeği etrafında aşağıdaki birimleri barındırır.

* CNN: HYSA hesaplamalarının gerçekleştiği birim.
* LCCU: Yerel iletişim ve kontrol birimi.
* LLM: Yerel sayısal hafıza.
* LLU: Yerel lojik birim.

* LAOU: Yerel analog çıkış birimi.

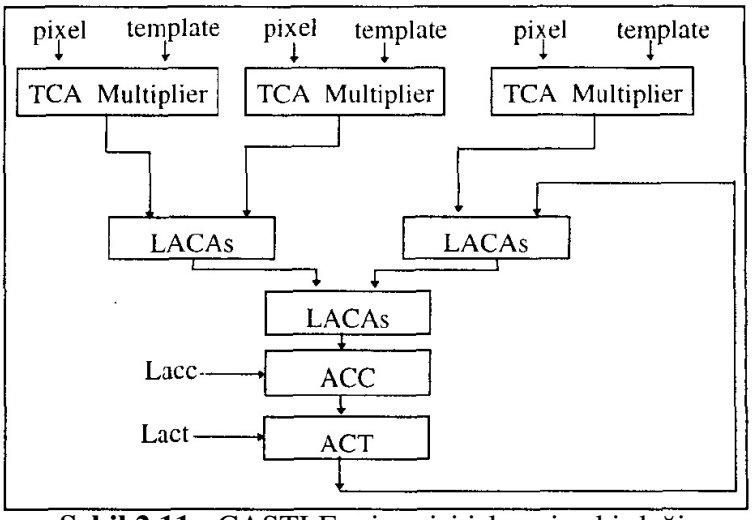
* LAM: Yerel analog hafıza.

Bu mimariden ACE4k [7] ve ACE16k [8] isimli analojik tümdevreler geliştirilmiştir. ACE16k 128x128 boyutundaki girişi 8 μs'de, 7-8 bit çözünürlükte çalışır. Dijital veri yolu ile 640x480 çözünürlükte gri tonlu görüntüyü 100 fps hızda işleyebilir. 4W güç tüketmektedir.

2.2.1 CASTLE ve FALCON Mimarileri

Macar bilimler akademisinden Peter Szolgay ve ekibi tarafından 1998 yılında HYSA-EM emülatörü olarak CASTLE mimarisi tanıtılmıştır [11]. Çip 70 mm^2 alanda 0.35 μm teknolojisi ile 24 HYSA işlemcisi barındırır. Orta ölçekli görüntü akışlarında gerçek zamanlı çalışabilmektedir. Örneğin 240x320 çözünürülükte 25 fps hızında 500 HYSA iterasyonunu 12 bit çözünürlük ile işleyebilir.

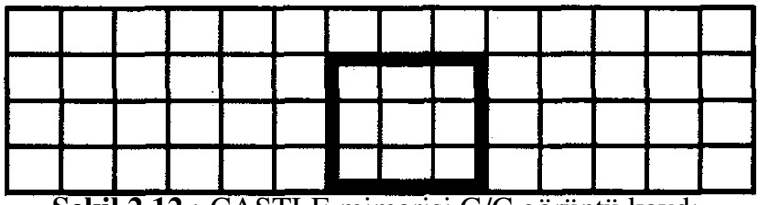
Bu mimari HYSA modeli olarak Chua-Yang modelinin değiştirilmiş hali Tüm Sinyal Aralığı (TSA) modelini kullanmaktadır. Uyarlanmış şablonlar sayesinde denklem 3x3 konvolüsyon, toplama ve seçim fonksiyonu ile gerçeklenebilir hale gelmiştir. İşlemci çekirdeği mimarisi Şekil 2.12'te gösterilmiştir.



Şekil 2.11 : CASTLE mimarisi işlemci çekirdeği.

Bu işlemci çekirdeği 3 çarpıcı, 3 toplayıcı ve harici akümülatörler barındırır. Bu tasarımda 3x3'lük konvolüsyon 3 saat darbesinde tamamlanır. Hesaplama yaklaşık olarak 24 ns sürer. Tek bir yongada bunun gibi 24 adet gerçeklenebilir, buda 1ns/HYSA işlemi demektir.

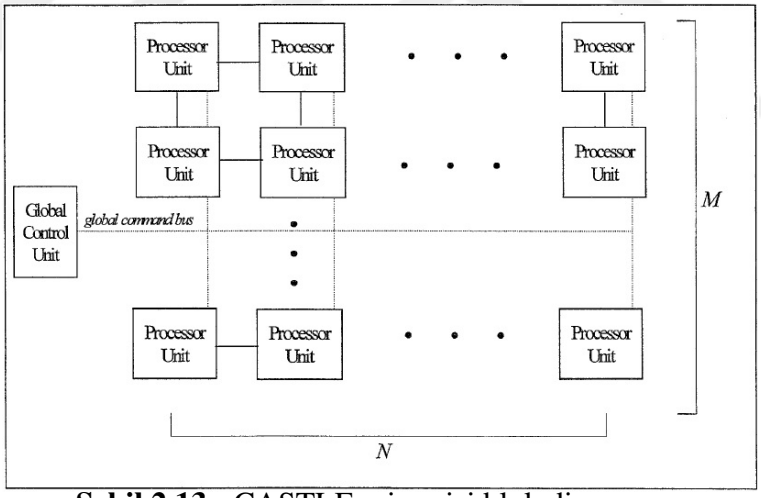
İşlem hızı olarak yukarıdaki kısım doğru olsa da G/Ç problemlerini de hesaba katmak gerekir. Bu mimari resmin tamamını barındıracak kadar büyük hafıza içermemektedir. 3x3'lük konvolüsyon içinde en az 3 satırlık görüntü gerekir. Bu yapı Şekil 2.12'te gösterilmiştir.



Şekil 2.12 : CASTLE mimarisi G/Ç görüntü kaydı.

Koyu renkli kısım konvolüsyonun gerçekleştirildiği yeri gösterir. Ayrıca şablon değerleri yonga üzerinde saklanır.

İşlemci dizisi ise global kontrol birimi ve MxN'lik işlemci dizisi barındırır. Bu yapı 2.13'te gösterilmiştir. İşlenilecek görüntü bunlar üzerinde yatay olarak ayrıştırılır. Her bir işlemci dizisi komşu işlemcisi ile konuşabilmektedir ve bunlar global kontrol yolu aracılığı ile kontrol edilir.



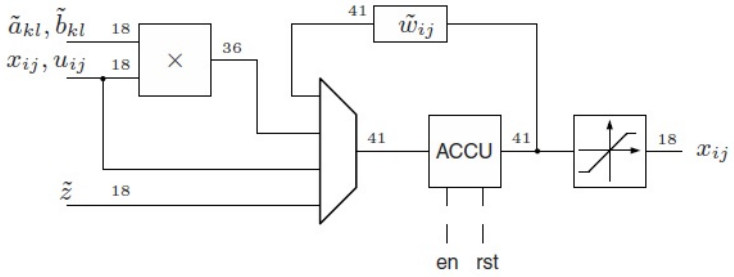
Şekil 2.13 : CASTLE mimarisi blok diyagramı.

FALCON mimarisi ise CASTLE mimarisinin yeniden tasarlanıp çok katmanlı HYSA gerçeklemesine uygun hale getirilmesi ve bit çözünürlüğü, dizi büyüklüğü ve şablon büyüklüğü ayarlanabilirliği eklenmesi ile oluşturulmuştur. Bu mimari 24 bit çözünürlükte, 340x480 piksel görüntüyü 178 iterasyon/s hızda işleyebilmektedir.

2.2.2 CESAR ve NERO Mimarileri

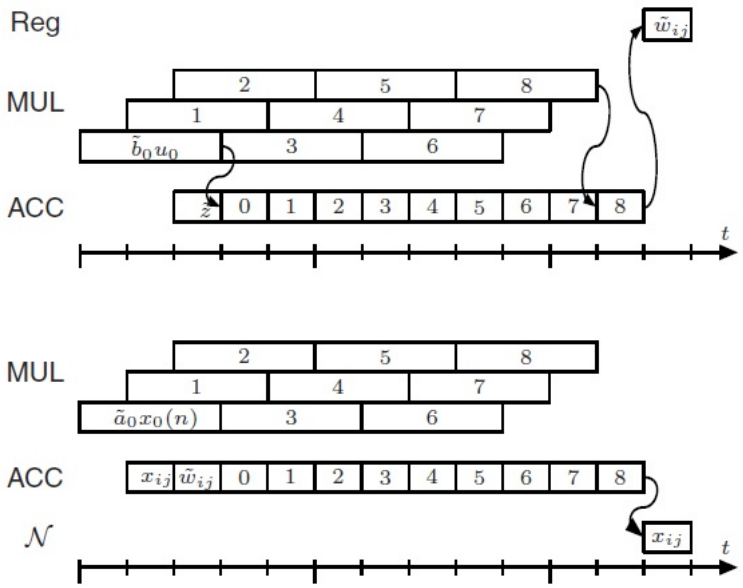
HYSA-EM emülatörü olarak geliştirilen CESAR mimarisi 128x128 hücre, 3x3 komşuluk ve 18 bit çözünürlüğüne sahiptir. Bu mimari Tüm Sinyal Aralığı (TSA)

modeli kullanmaktadır. CESAR mimarisinin bir işlem ünitesi blok diyagramı Şekil 2.14'te gösterilmiştir.



Şekil 2.14 : CESAR mimarisi işlem ünitesi.

Giriş ve durum hesaplamaları aynı işlem ünitesi içerinde yapılır. 3x3'lük konvolüsyon için hesaplama bloğu verimliliğini arttırmak üzere çarpıcılar 3 aşamalı ardışık düzende yer alır. Bu yapı Şekil 2.15'te görülmektedir.



Şekil 2.15 : CESAR mimarisi 3x3'lük işlem için zaman çizelgesi.

İlk olarak *w* olarak isimlendirilen *b* şablonu ve *u* girişi konvolüsyonu gerçekleştirilir. Daha sonra *a* şablonu ve *x* eski durum konvolüsyonu önceki değere eklenerek ilerler ve durum hesaplanır.

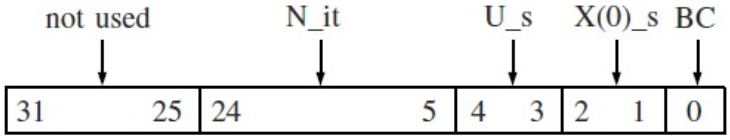
CESAR mimari ardışıl HYSA operasyonlarının işletilmesine olanak sağlar. Bu operasyonları mikrokomutlar şeklinde yazılmıştır. Sınır koşulu "NEUMANN" veya "DIRICHLET" seçimlidir. Giriş ve durum için 3 seçenek sunulmuştur:

* Hafızadan oku.

* Önceki işlemden kopyala.

* Sıfıra eşitle.

Bir HYSA komutunun bit yerleşimi Şekil 2.16'de verilmiştir.

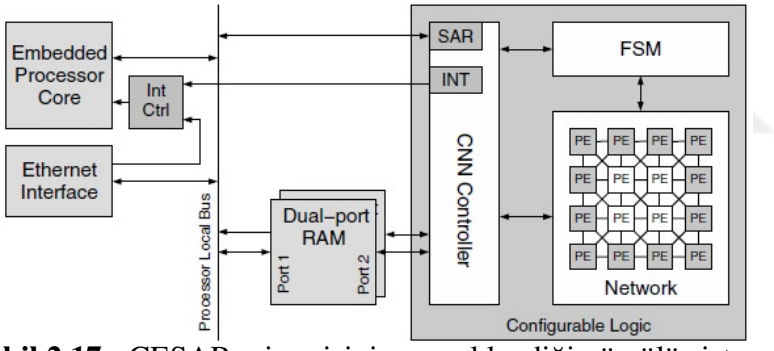


Şekil 2.16 : CESAR mimarisi mikrokomutu.

32 bitlik bu komutta ayrılmış blokların konumları ve anlamları:

* (31-25) kullanılmıyor.
* (24-5) N_it: İterasyon sayısı.
* (4-3) U_s: Giriş işlem seçimi.
* (2-1) X(0)_s: Başlangıç durumu işlem seçimi.
* (0) BC Sınır koşulu seçimi.

Bir gömülü sistemin parçası olarak CESAR mimarisi Xilinx Virtex 5 FPGA üzerinde gerçeklenmiştir. Bu gömülü sistemin blok diyagramı Şekil 2.17'de verilmiştir.

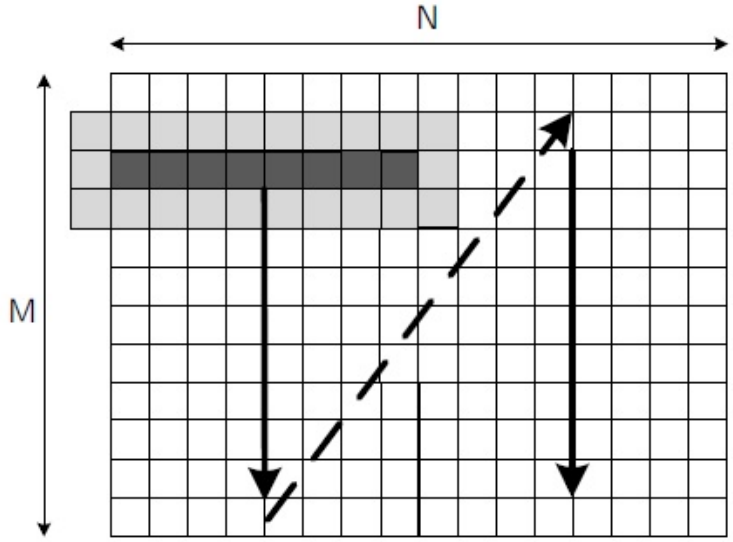


Şekil 2.17 : CESAR mimarisinin gerçeklendiği gömülü sistem.

İşlem ünitesinin ana fonksiyonu çarpma, FPGA üzerindeki adanmış DSP'ler tarafından gerçekleştirilir. PowerPC yanişlemcisi harici hafıza ve ana ünite arasında veri aktarımını ethernet arayüzü ile sağlar. Program kodunu, giriş verisini, başlangıç koşullarını ve şablonları saklayan Çift Taraflı RAM'e hem yardımcı işlemci hem de HYSA kontrolörü ulaşabilir. Yardımcı işlemci ardışıl HYSA işlemlerini işler. HYSA kontrolcüsü yazılım ile ilklendirilir ve işlemin bittiğini kesme ile bildirir.

Bu gerçekleme 100 Mhz sistem saat frekansı hızında, 18 bit çözünürlüklü ve 128 hücreli çalışmaktadır. 15 saat darbesinde bir sonraki durum hesaplayabilir. 20 iterasyonlu bir işlem yaklaşık 3 μs sürmektedir.

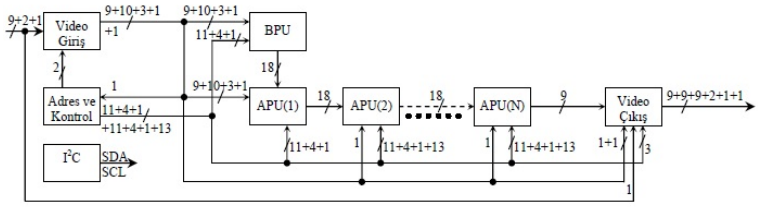
NERO mimarisinde büyük boyutlu 2D ağları işleyebilmek için CESAR mimarisindeki hücre işlemci eşleşmesi üzerinde değişiklik yapılmıştır [10]. Bu işlemde giriş görüntüsü düşey parçalara bölünmüştür ve satırda ilerleyerek işlenir. Bu işleme şeması Şekil 2.18'de verilmiştir.



Şekil 2.18 : NERO mimarisi işleme şeması.

2.2.3 Steadfast-1 ve Steadfast-2 Mimarileri

K. Kayaer ve V. Tavşanoğlu RTCNNP isminde değiştirilmiş bir ayrık zamanlı HYSA yapısı sundular ve daha sonra bu yapı Steadfast-1 olarak isimlendirildi. Steadfast-1 hafıza olarak sadece FPGA içerisindeki BRAM'leri kullanır. N iterasyonun gerçeklendiği FPGA içerisindeki sistem blok diyagramı Şekil 2.19'de gösterilmiştir.



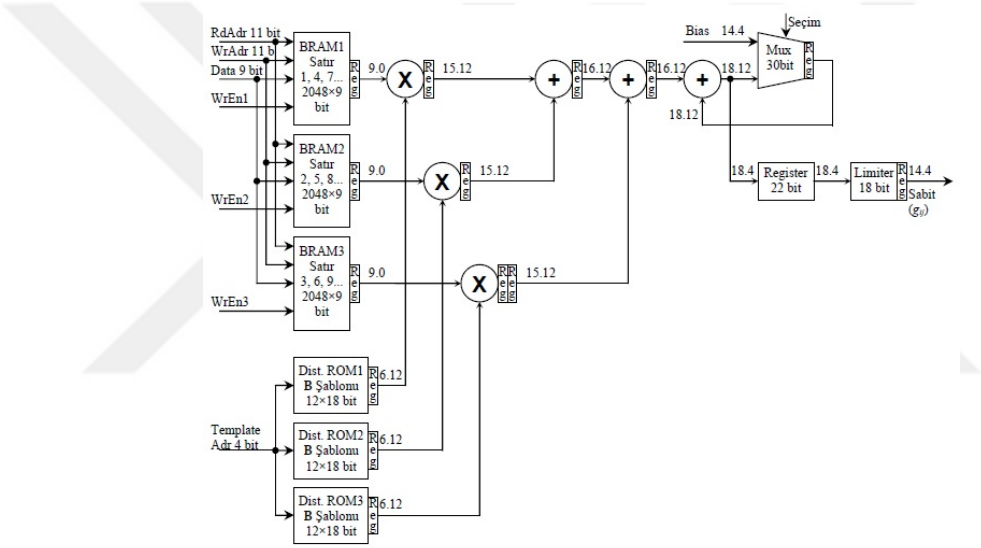
Şekil 2.19 : Steadfast-1 mimarisi sistem şeması.

Görüntü kamera ve bilgisayardan alınarak FPGA içerisinde işleniyor ve çıkış monitör aracılığı ile gözlemlenebiliyor. Mimari A şablonu işlem birimi (APU) ve B şablonu işlem birimi (BPU) isimli iki ana işlem ünitesi barındırır. "Video Giriş" görüntü pikseli verisini BPU'ya aktarır, ayrıca "Adres ve Kontrol" bloğunu kontrol eder ve "Video Çıkış" bloğu senkronizasyonunda kullanılır. "Adres ve Kontrol" bloğu işlem ünitelerinin çalışması kontrol eder, BRAM adreslerini üretir ve uygun olan BRAM'leri

aktif hale getirir. "Video Çıkış" bloğu işlenmiş görüntüyü 640x480 çözünürlükle 60 fps hız ile VGA formatında monitöre aktarır. I2C bloğu I2C seri veri aktarım protokolü ile Video ADC tümdevresinin çalışma parametreleri içindeki yazmaçlara yazar.

Bu yapı Tüm Sinyal Aralığı (TSA) modelini kullanır. Burada (2.10), (2.11) denklemlerini çözer. g_{ij} sabit olduğundan girişin B şablonu ile konvolüsyonun sadece bir kere hesaplanması yeterlidir. BPU bloğundan bu işlemin başında bir adet yer almaktadır. APU bloğu önceki durum ile A şablonu konvolüsyonu uygulayarak durumu hesaplar. Bu sonuç bir sonraki APU bloğunun girişini oluşturur, böylelikle durum iterasyonunu gerçekleştirir. Bu bloktan iterasyon sayısı kadar ard arda bağlanır.

BPU bloğunun blok diyagramı Şekil 2.20'de verilmiştir.



Şekil 2.20 : Steadfast-1 BPU blok diyagramı.

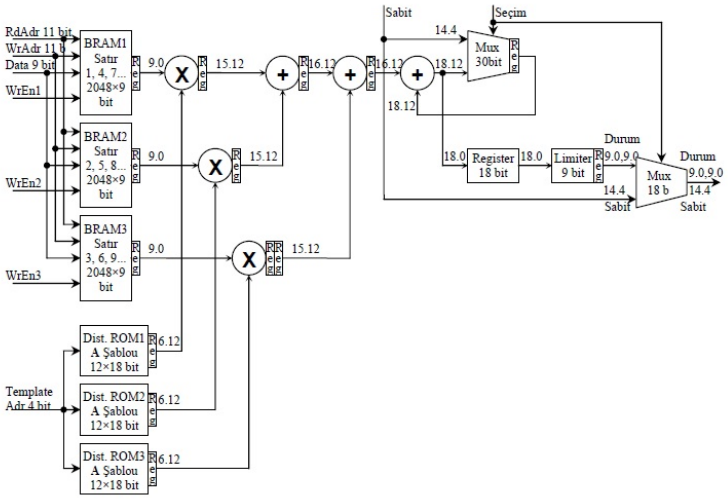
3 adet blok RAM giriş görüntüsünün 3 satırını saklamaktadır. Birinci satır BRAM1'e, ikinci satır BRAM2'ye ve üçüncü satır BRAM3'e yazılır. İlk 3 piksel BRAM'e yazıldıktan sonra işlem birimi veri almaya devam eder. BRAM'ler çift yönlüdür, aynı anda hem yazma hem okuma yapabilir. Üçüncü satır yazma işlemi bittikten sonra birinci satıra tekrar yazmaya başlar. 3 adet ROM'da ise B şablonu değerleri saklanır. Bu şablon değerlerinin DiROM içerisindeki dağılımı 2.21'te verilmiştir.

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

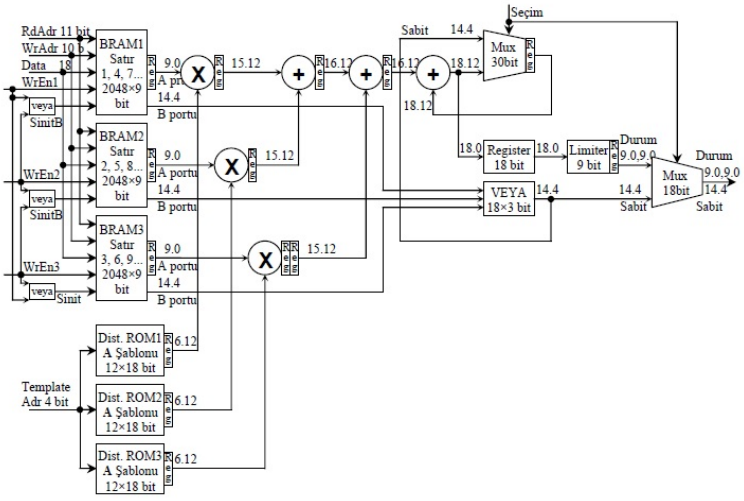
Şekil 2.21 : Steadfast-1 B şablon değerlerinin DiROM yerleşimi.

Bunlar giriş değerleriyle paralel okunur ve çarpma işlemi gerçekleştirilir. Bunların toplanması ile elde edilen değere eşik değeri eklenerek g_{ij} sabit değeri elde edilir.

APU(1) bloğunun blok diyagramı Şekil 2.22'de verilmiştir. APU(n>1) için blok diyagramı 2.23'de verilmiştir.



Şekil 2.22 : Steadfast-1 APU(1) blok diyagramı.



Şekil 2.23 : Steadfast-1 APU(n>1) blok diyagramı.

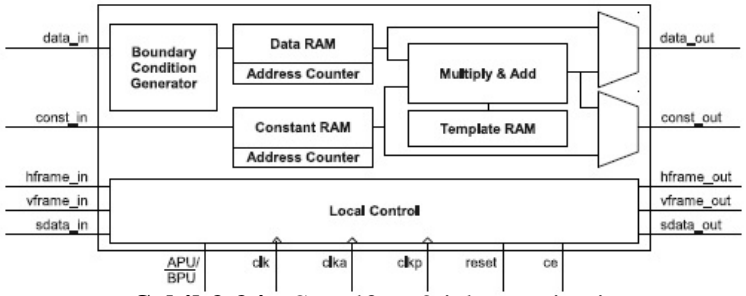
APU bloğu BPU ile benzer yapıya sahiptir. Giriş olarak durum alarak bir sonraki durum değerlerini hesaplar. ROM'larda ise A şablonu değerleri saklar. Başlangıç durumu giriş değeri veya sabit bir değer seçilebilir. Buradaki sabit giriş eşik değeri yerine, BPU'da hesaplanan değerdir. İlk iterasyonda sonraki APU ünitelerinin BRAM yazma aktif uçları VEYA kapısından geçirilerek BRAM'in B portunun "senkron init" ucuna bağlanmıştır. Böylelikle değeri okunacak BRAM dışındakilerin lojik 0 olması sağlanır.

Steadfast-1 mimarisi Xilinx Virtex-II 300 FPGA üzerinde gerçeklenmiştir. 3x3 HYSA şablonları ile 640x480 boyutundaki görüntüyü 60 fps hız ile gerçek zamanlı işleyebilmektedir. Mimarinin başlıca özellikleri aşağıdaki gibidir.

* Harici hafızaya ihtiyaç duymaz.
* Her bir pikselin hesaplanması 3 saat frekansı sürer.
* Gerçeklenen işlem ünitesi sayısı iterasyon sayısını belirler.

Daha sonra N. Yıldız ve V. Tavşanoğlu tarafından bu mimari geliştirilerek Steadfast-2 adı altında sunulmuştur. Steadfast-1 mimarisi ile çalışma prensibi aynıdır, bu mimarinin hafıza yapısı optimize edilmiş, sınır koşulları eklenmiş ve işlem ünitende değişiklikler yapılmıştır.

Steadfast-2 işlem ünitesi blok diyagramı Şekil 2.24'de verilmiştir.



Şekil 2.24 : Steadfast-2 işlem ünitesi.

Steadfast-2 mimarisi 1920x1080 boyutundaki görüntüyü 54 fps hız ile gerçek zamanlı işleyebilmektedir.

3. SİSTEM MİMARİSİ ve FPGA GERÇEKLEMESİ

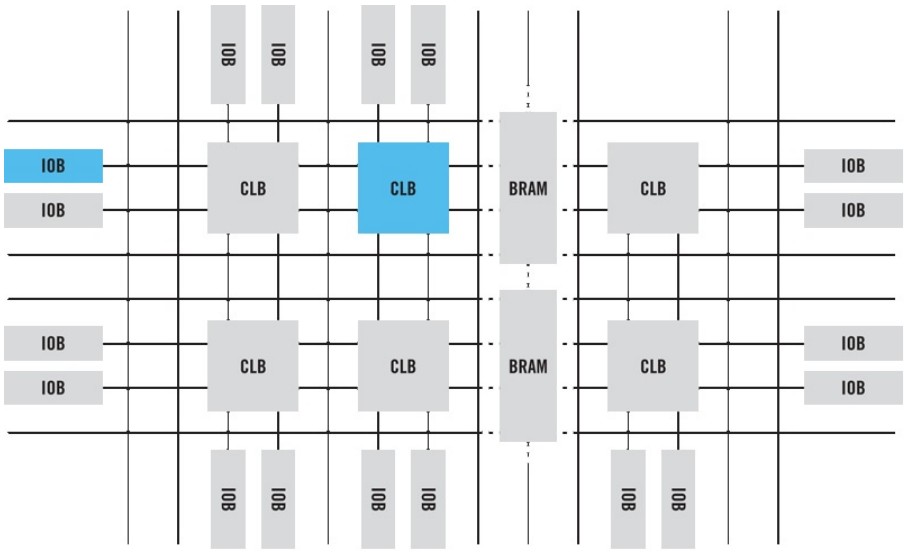
3.1 FPGA Teknolojisi

FPGA (Field Programmable Gate Array, Sahada Programlanabilir Kapı Dizileri), programlanabilir mantık blokları ve bu bloklar arası bağlantıların olduğu sayısal tümdevrelerdir. FPGA teknolojisinin temeli 1985 yılında Xilinx firması tarafından Logic Cell Array (LCA) ismi adı altında tanıtıldı. O zamandan beri bu mimari zamanla yapılanmaya ve gelişmeye devam ediyor. Bu mimaride sayısal devreler, basit işlevsel sayısal blokların programlanabilir anahtarlarla birbirine bağlanması ile oluşur. FPGA gerçek avantajı tasarım aşamaları sırasında hatta tasarımdan sonra sahada tekrar programlanabilirliğidir. FPGA mimarileri lojik kapı sayıları, yönlendirme metotları ve özel işlevli bloklarına göre isimlendirilir.

Mimarilerinden bağımsız olarak FPGA'lar harici cihazlarla haberleşmek için giriş çıkış (IOB, G/Ç) bloklarına sahiptir. Bu bloklar birçok G/Ç standardını barındırır, ayrıca giriş, çıkış veya çift yönlü olarak programlanabilirler. FPGA'ler sayısal devre sentezi için içerisinde konfigüre edilebilir mantık blokları (CLB) barındır. Mantık blokları LUT (Look Up Table), flip flop, MUX gibi lojik elemanlardan oluşmaktadır. Bu CLB'ler arası bağlantılarda programlanabilirdir. Şekil 3.1'de FPGA'nın 3 ana yapı taşı IOB (G/Ç), CLB ve ara bağlantılar görülmektedir.

Ayrıca geçici hafıza ihtiyacını karşılamak için FPGA'lar içerisinde belirli boyutlarda dahili RAM barındırırlar, bunlara blok RAM (BRAM) ismi verilir. BRAM'ler klasik RAM'ler ile aynı özelliktedirler, çift olarak kullanılabilirler yani aynı hafızaya iki port aynı anda erişebilir. Yazma ve okuma aynı anda yapılabilir. Xilinx firmasının özel blokların içerisinde 18x18 çarpma ve toplama (MAC) işlemini gerçekleştirebilen XtremeDSP blokları da vardır. FIFO blokları ilk giren verinin ilk verildiği özel hafızalardır. BRAM veya lojik kapılarla gerçeklenebilirler.

FPGA içerisinde sayısal devre donanım tasarım dilleri ile veya devre şematiği ile oluşturulabilir. Tezde VHDL dili kullanılmıştır. Tezde kullanılan FPGA Xilinx



Şekil 3.1 : FPGA iç yapısı.

firmasının bir modelidir. Xilinx birçok kullanışlı yazılım ve donanım tasarım aracı sunmaktadır. Sayısal devreyi sentezlemek için ISE aracı kullanılmıştır. Çarpıcı, BRAM, saat üreteci veya fir filtre gibi standart kullanılan donanımların Xilinx firması tarafından hazır blokları (IP Core) sağlanmaktadır "Ip core generator" eklentisi IP Core'ları tasarımımıza eklememizi sağlamaktadır. Hata ayıklamak için "Chipscope" programı ile birlikte "Platform Cable" donanım arayüzü ile kullanılmıştır. HYSA algoritması tasarım aşamasında algoritmanın sonuçlarını karşılaştırmak için VHDL test kodu yazılmış ve ISE simülatörü (ISim) kullanılmıştır.

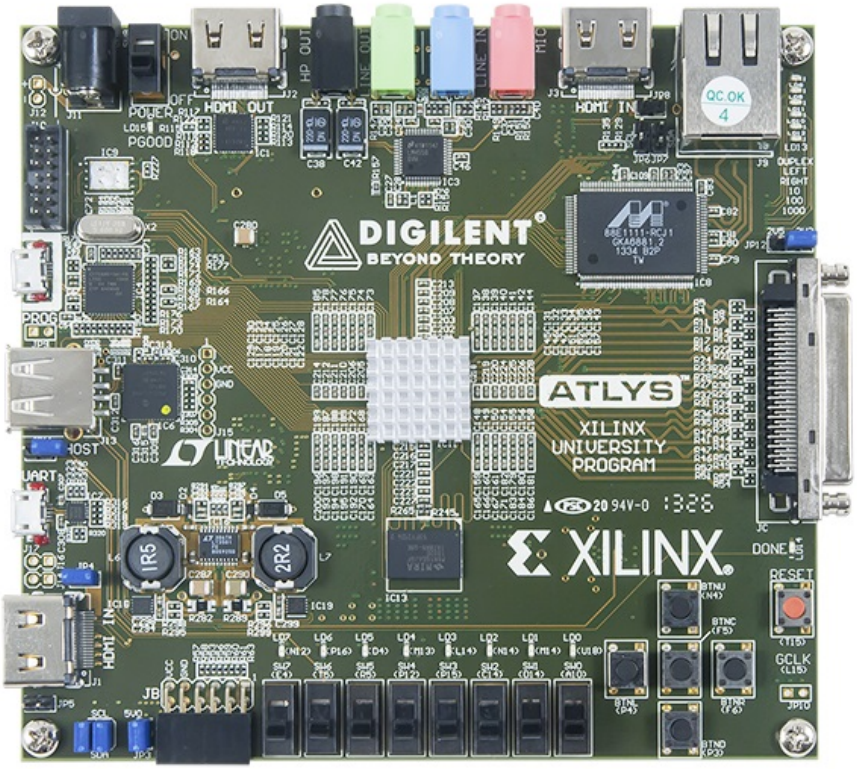
3.2 Geliştirme Ortamı

Ayrık zamanlı HYSA benzetimini Aritmetik İşlem Biriminde barındıran HYSA-Eİ mimarisi, üzerinde Xilinx Spartan 6 FPGA ve 128 Mbyte DDR2 SDRAM bulunan Digilent firmasının "Atlys Board" isimli gömülü sistemi üzerinde gerçeklenmiştir.

Şekil 3.2 kullanılan geliştirme kartı gösterilmiştir.

Atlys geliştirme kartı üzerinde bulunan Xilinx Spartan 6 LX45 FPGA bazı temel özellikleri:

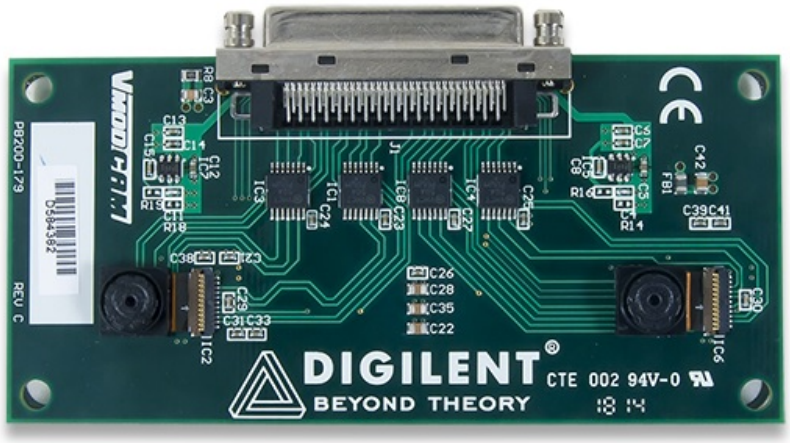
* 6,822 slice, her birinde 6 LUTs 8 FlipFlop
* 2.1Mbits Blok RAM
* 8 DCM 4 PLL saat işareti üreteci
* 58 DSP slice



Şekil 3.2 : Spartan 6 Atlys Geliştirme Kartı.

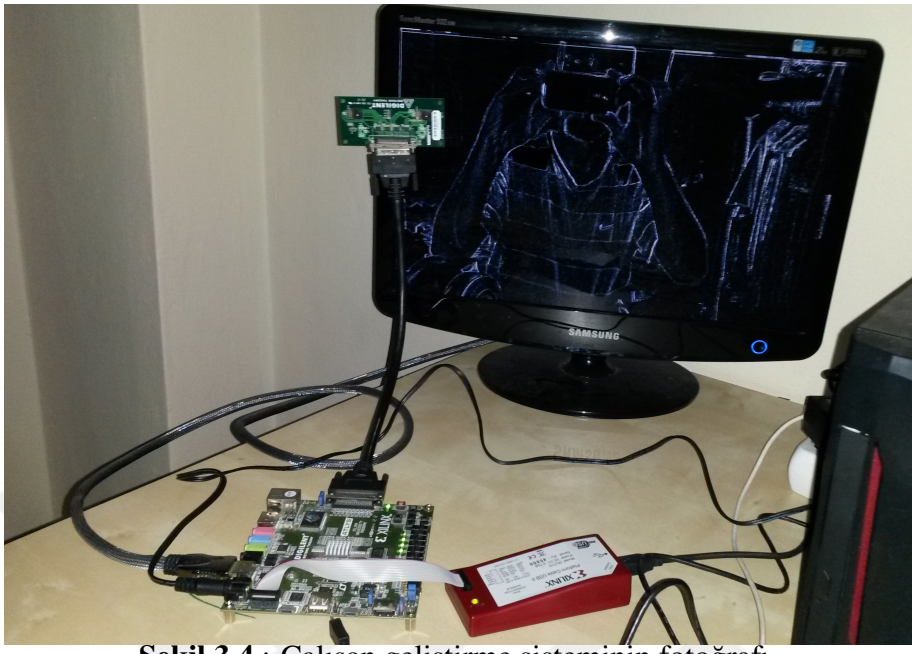
Görüntü almak için bu kartın üzerinde bulunan VHDCI arayüzü ile bağlanan "VmodCAM" isimli çift CMOS kameranın bir tanesi kullanılmıştır.

Şekil 3.3 kullanılan geliştirme kartı gösterilmiştir.



Şekil 3.3 : VmodCAM Çift CMOS Kamera.

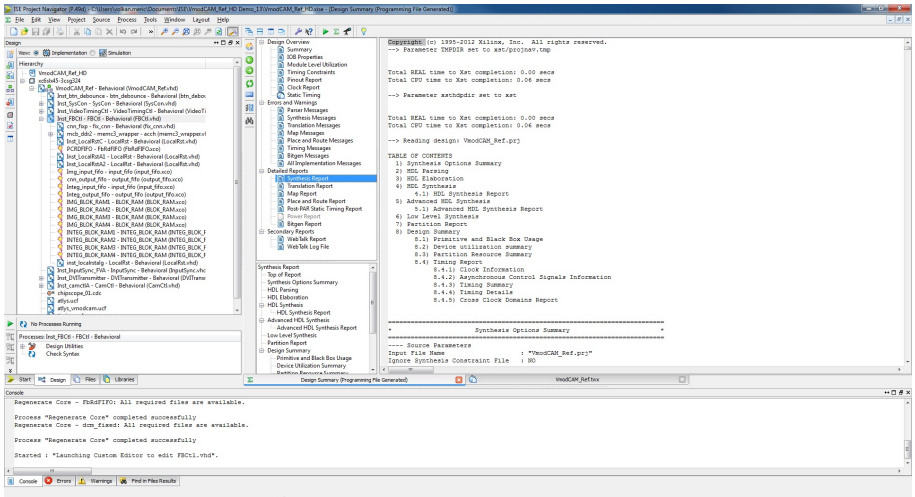
Üzerinde gerçek zamanlı kenar tespiti şablonu ile HYSA çalışan geliştirme sisteminin fotoğrafı Şekil 3.4'te verilmiştir.



Şekil 3.4 : Çalışan geliştirme sisteminin fotoğrafı.

3.3 FPGA Gerçeklemesi

FPGA kodu ISE 14.4 geliştirme ortamında gerçeklenmiştir. Saat işareti üreteci, BRAM, FIFO ve çarpma bloğu için "IP CORE" kullanılmıştır. Kamera, DVI ve DDR2 arayüzüleri digilent firmasının örnek kodlarından alınmış ve proje uygun olacak şekilde modifiye edilmiştir. HYSA-Eİ ve bunları yöneten tepe modülü VHDL dili ile yazılmıştır. Donanım geliştirme ortamı ve proje dosyası görüntüsü Şekil 3.5'te verilmiştir.

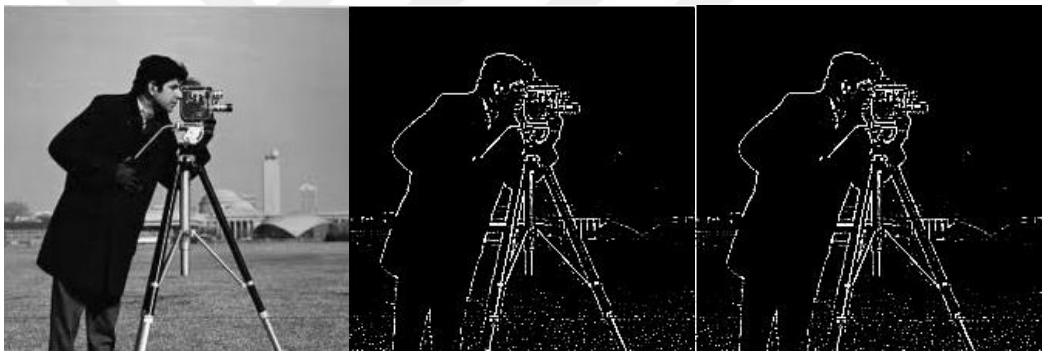


Şekil 3.5 : ISE Proje dosyasi.

İlk olarak bir HYSA hücresi daha sonra iterasyonu MATLAB ortamında gerçeklenmiştir. Ayrıca VHDL dilinde FPGA'da sabit şablonlar ile çalışan HYSA işlem birimi gerçeklenmiştir. FPGA HYSA gerçeklemesini test etmek için giriş dosyadan okuyan, çıkışı dosyaya yazan bir VHDL test kodu yazılmıştır. Bu test kodu ISIM test ortamında koşturulmuştur. MATLAB'ta standart bir kaç görüntü dosyası oluşturulmuş ve MATLAB sonuçları ile FPGA sonuçları karşılaştırılmıştır. Görüntü işlemede standart test görüntülerinden biri olan kameraman ile test edilmiştir. Aşağıdaki kenar tespiti şablonu ile iki iterasyon koşturulmuştur.

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, B = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \text{ ve } I = -1$$

Kameraman giriş görüntüsünü gri seviyeli hali, MATLAB sonucu ve FPGA sonucu Şekil 3.6'te sırası ile gösterilmiştir.



(a) (b) (c)

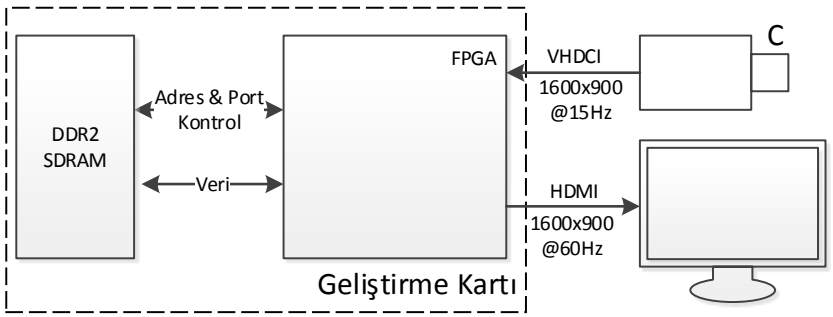
Şekil 3.6 : (a) Gri seviyeli giriş görüntüsü, (b) MATLAB çıkış görüntüsü (c) FPGA çıkış görüntüsü.

Bir HYSA iterayonunun çok düşük hata oranı (16 bit çözünürlük hatası) ile çalıştığını gördükten sonra gerçek zamanlı Hücresel Sinir Ağı Benzetim İşlemcisi tasarım ve kodlanmasına geçilmiştir.

3.3.1 HYSA-Eİ Gerçeklemesi

İşlem basamakları paralel olan HYSA doğası itibari ile FPGA gerçeklemesine çok uygundur. HYSA Evrensel Makinesi sayısal emülatörü olarak tasarlanan sistem, bir işlemcinin komut seti olarak gerçeklendiğinden dolayı bu sisteme Hücresel Yapay Sinir Ağı Emülatör İşlemcisi (HYSA-Eİ) ismi verilmiştir. Tasarlanan HYSA-Eİ gerçeklendiği sistem blok diyagramı Şekil 3.7'te gösterilmiştir. Sistem kamera ile görüntüyü yakalamakta ve FPGA üzerinde HYSA-Eİ işlemlerini gerçekleştirmektedir.

Harici DDR2 SDRAM, HYSA iterasyon sonuçlarını, kameradan alınan giriş görüntüsü ve çıkış görüntüsünü saklamak için kullanılmaktadır. Durum değişkenlerinin her bir iterasyondaki gelişimi, sisteme DVI ile bağlanan bir monitör aracılığı ile gözlemlenebilir. Sistemin blok diyagramı Şekil 3.7'te verilmiştir.



Şekil 3.7 : Genel sistem blok diyagramı.

Hücresel Yapay Sinir Ağının integral işlemini gerçekleştirmek için ileri euler iterasyonu kullanılmıştır. Buda ayrık zamanda gecikmeye denk düşmektedir. Buradan doğan hafıza gereksinimi DDR2 SDRAM harici hafıza birimi ile karşılanmıştır. Her bir iterasyonda iterasyon ara değişkeni hafızaya tekrar yazılmıştır.

HYSA emülatör işlemcisinin dahili kontrolörü, harici hafızada saklanan görüntülere birden fazla ve farklı HYSA şablonunun işlenmesine olanak vermektedir. HYSA-Eİ `Y=HYSA(U,X0,Sab,n)` fonksiyonu (komutunu) işleyebilmektedir.

u giriş görüntüsü,

X_0 başlangıç görüntüsü,

Y çıkış görüntüsü,

Sab kullanılacak şablonun adresi,

n iterasyon sayısı

Sab şablonu aşağıdaki değişkenlerden oluşmaktadır.

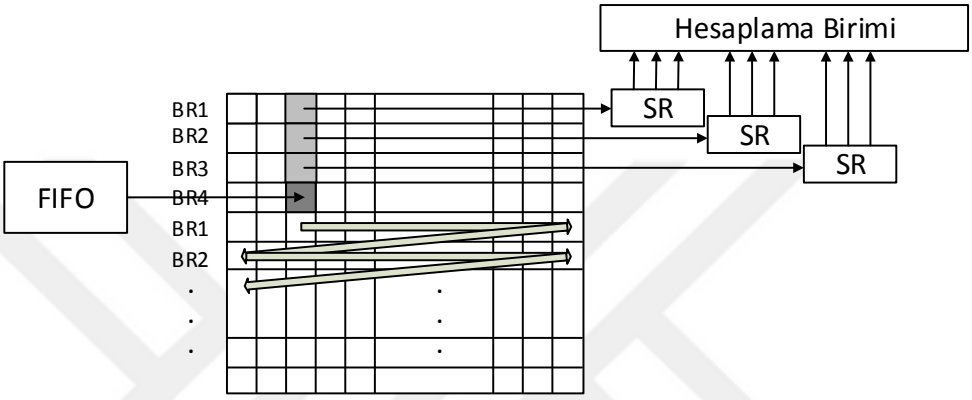
$$A = \begin{pmatrix} a_{-1,-1} a_{-1,0} a_{-1,1} \\ a_{0,-1} \; a_{0,0} \; a_{0,1} \\ a_{1,-1} \; a_{1,0} \; a_{1,1} \end{pmatrix}, B = \begin{pmatrix} b_{-1,-1} b_{-1,0} b_{-1,1} \\ b_{0,-1} \; b_{0,0} \; b_{0,1} \\ b_{1,-1} \; b_{1,0} \; b_{1,1} \end{pmatrix} \text{ ve } I$$

`A` geri besleme şablonu,

`B` giriş(kontrol) şablonu,

`I` eşik değeri

Bütün iki boyutlu dizi (görüntü) üzerindeki HYSA gerçeklemesi zaman paylaşımlı yöntemle yapılmaktadır. Aritmetik İşlem Birimi tek bir hücrenin sonucunu hesaplamaktadır. Dizinin tamamı öncelikle satırda (yatay), satır bittikçe sütunda (düşey) ilerleyerek yani 3x1'lük pencereleri görüntü üzerinde gezdirerek tamamlanır. Bunlar kaydırılarak 3x3'lük pencere elde edilir. Bu şema Şekil 3.8'de gösterilmiştir. Görüntünün satırlarının yer aldığı BRAM isimleri yanlarına yazılmıştır. Yeni bir satıra geçildiğinde üstüne yazılır.



Şekil 3.8 : Görüntünün BRAM'lerde yerleşimi ve okunması.

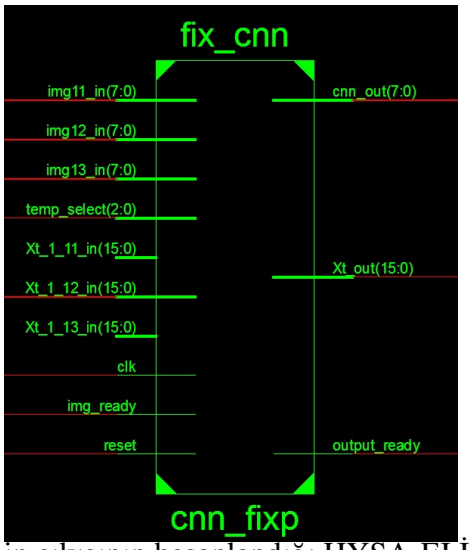
Birden çok farklı görüntü işleme fonksiyonlarını gerçekleştiren farklı şablonlar FPGA dahili ROM'unda saklanmaktadır. HYSA-Eİ işlemci komutlarının saklandığı komut hafızasını dahili ROM'dan okumaktadır. Ayrıca sistem çıkış değerini farklı DDR2 hafıza bölümlerinde saklayarak, bu sonuçların daha sonraki HYSA komutları tarafından giriş veya başlangıç koşulu olarak kullanılabilmesine olanak vermektedir.

3.3.1.1 HYSA İşlem Birimi

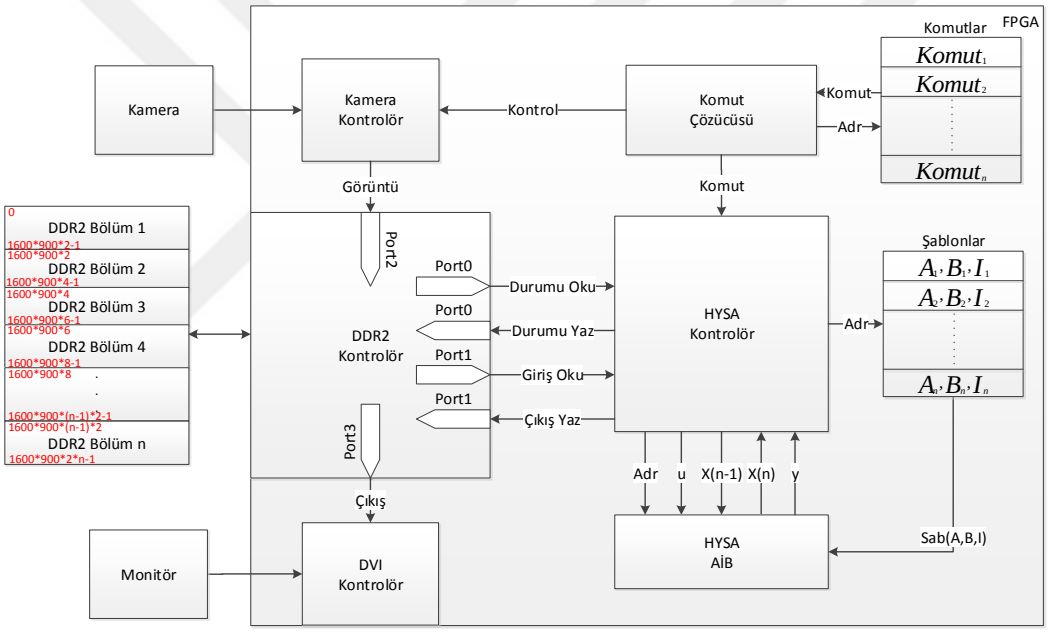
Şekil 3.9'de gerçeklenen HYSA-Eİ işlem bloğunun FPGA şematiği verilmiştir.

Gerçeklenen HYSA-Eİ blok şeması Şekil 3.10'da görülmektedir.

Kamera kontrolörü komut çözücüsünden gelen komut ile görüntüyü kameradan DDR2 hafızaya aktarır. Komut çözücüsü FPGA'in dahili ROM'unda saklı olan komutları sırayla okur ve bunları çözerek işletir. HYSA kontrolörü HYSA Aritmetik İşlem Birimini (AİB) yönetir ve blok RAM, ROM ve DDR2 arasındaki veri transferi senkronizasyonundan sorumludur. DDR2 kontrolörü belirlenmiş portları kullanarak FGPA ve DDR2 arasındaki veri transferini komut çözücüsünden gelen komutlar



Şekil 3.9 : Bir hücrenin çıkışının hesaplandığı HYSA-EI İşlem Birimini FPGA şeması.



Şekil 3.10 : HYSA-Eİ Mimarisi.

çerçevesinde gerçekleştir. DVI kontrolörü çıkış görüntüsünü HDMI port aracılığı ile LCD ekrana HD+(1600×900) çözünürlüğünde @60 hızında aktarır.

Kullanılan DDR2 RAM veri yolu 16 bit uzunluğundadır. Kameradan 16 bitlik çözünürlüğünde renkli görüntü alınıyor. Bu da üç bantlı (RGB) görüntü demektir. Bu projede RGB görüntüden gri ölçekli görüntüye geçilerek HYSA işlemi için bu görüntü Blok RAM'lere aktarılıyor. Gri ölçekli görüntü (I) basitçe kırmızı (R), yeşil (G) ve mavi (B) bantlarının değerlerinin ağırlıklı toplamıdır. Dönüşüm formülü 3.1'te verilmiştir:

$$I(x;y) = 0.2989R(x;y) + 0.5870G(x;y) + 0.1140B(x;y) \quad (3.1)$$

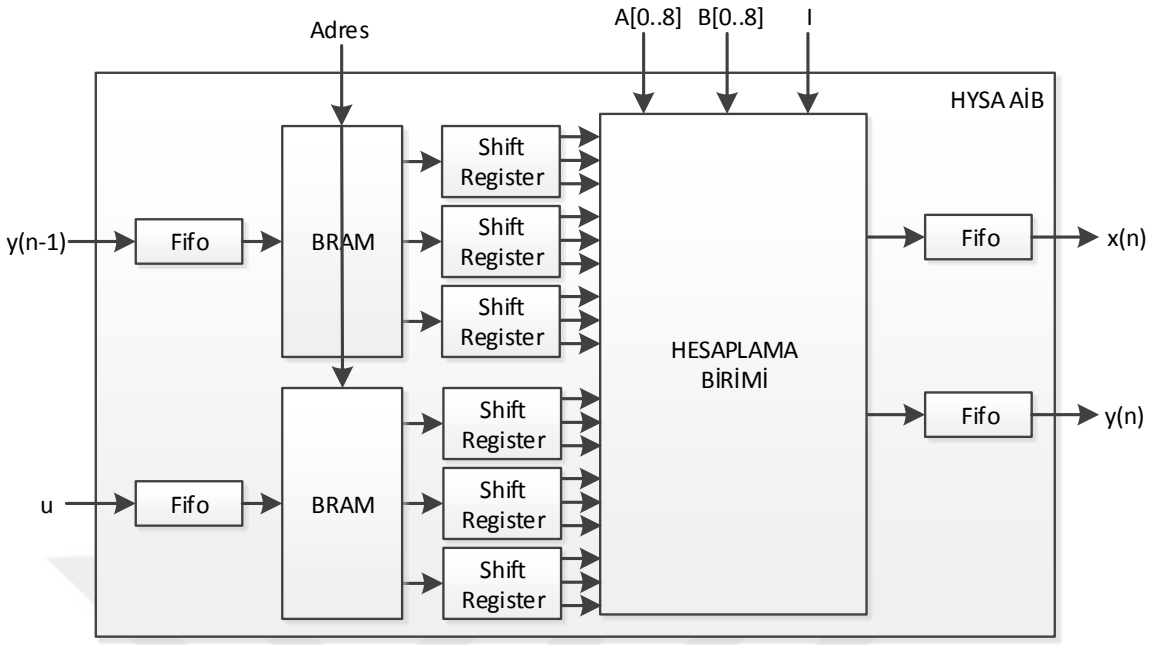
BRAM'ler yatay bir görüntü çizgisini veya durum değerini saklayabilecek (1600x2 byte) büyüklüğündeler.

Ardışıl bir işlemin zaman kullanımını minimize etmek için HYSA ve hafızalar arasın veri iletimi işlemleri kendi içlerinde olabildiğince verimli yani paralel gerçeklenmiştir. Birbirinden bağımsız çalışabilen her bir birim paralel gerçeklenerek minimum bir iterasyon tamamlanma süresi elde edilmeye çalışılmıştır. Veri aktarma üniteleri, çevre elemanı kontrol üniteleri ve HYSA aritmetik işlem ünitesi işlenebilen veri büyüklüğünü aktarmak için paralel gerçeklenmiştir, kontrol sinyalleri ile aktif hale getirilmektedir.

HYSA AİB'nin çalışması Şekil 3.11'da gösterilmiştir. Bir hücrenin hesaplanması için (3×3)'lük giriş ve durum gereklidir. DDR2 hafızada görüntünün tamamının bulunması rağmen parçalı veri okumanın verimsizliği nedeni ile 3 satır görüntü ve durum geçici olarak BRAM'lerde tutulmaktadır. DDR2 hafızalar belirli bir boyutta veriyi, bizim projemizde 128 byte, bir kerede gönderdiği için veri senkronizasyonu amacı ile fifo'lar kullanılmıştır. Görüntü ve durum ara değişenini tutmak için 8 adet blok RAM kullanılmıştır. 3 adet blok RAM her biri bir yatay görüntü çizgisini barındırmak suretiyle HYSA AİB'nin giriş değerlerini sağlamaktadır. Bu değerler kaydıran yazmaçlar (shift register) ile kaydırılarak (3×3)'lük görüntü penceresi oluşturulmaktadır. Aynı anda bir adet blok RAM fifo'dan gelen giriş değerini almak için kullanılmaktadır. Aynı şekilde 3 adet blok RAM durum değeri girişi ve 1 adet blok RAM fifo'dan gelen veri için kullanılmaktadır. Bir yatay çizgi bittiğinden bu BRAM'ler aralarında kaydırılarak yeni bir yatay çizgi alımına hazır hale gelmektedir. Bu yöntemle HYSA işlemi için gerekli olan (3×3)'lük görüntü ve (3×3)'lük durum değeri elde edilmiş olur. Kullanılacak şablon değerleri ROM'dan okunur. Hesaplanan çıkış görüntüsü ve yeni durum değeri fifo'lar aracılığı ile DDR2 RAM'e aktarılır.

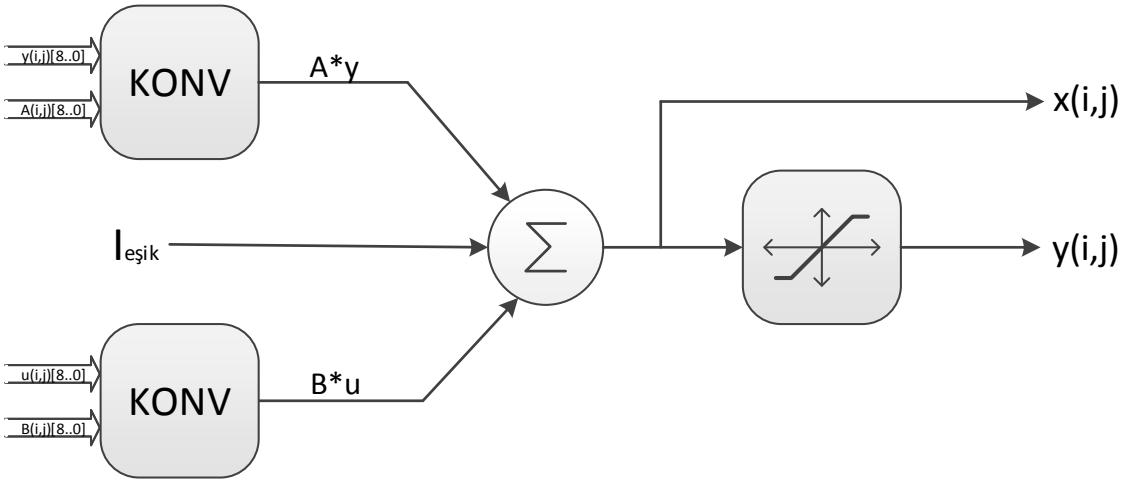
Hesaplama Birimi:

Aritmetik İşlem Birimi içerisinde birçok fonksiyon gerçeklenebilecek yapıda kurulmuştur, fakat şimdilik hesaplama ünitesinde sadece Hücresel Yapay Sinir Ağı



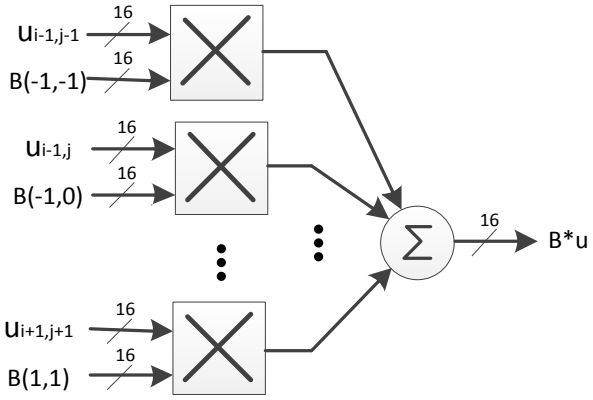
Şekil 3.11 : HYSA Aritmetik İşlem Birimi.

gerçeklenmiştir. Bu HYSA bir hücresinde çıkışı ve durum ara değişkeni (2.3) ve (2.12) denklemlerine göre hesaplanır. Bu yapının blok diyagramı 3.12 verilmiştir.



Şekil 3.12 : HYSA Hesaplama Birimi.

Bir hücre içerisinde 2 adet 3x3 konvolüsyon işlemi (18 adet çarpma + 2 toplama), bir adet toplama işlemi ve kısmi doğrusal fonksiyon vardır. Çarpma işlemi sabit noktalı olarak gerçeklenmiştir. Konvolüsyon işleminin blok diyagramı 3.13 verilmiştir.



Şekil 3.13 : Konvolüsyon işlemi.

Kullanılan lojik kapı sayısını azaltmak için çarpma işlemi özel donanım tabanlı çarpıcılar (XtremeDSP) ile yapılmıştır.

Çıkış fonksiyonu kısmi lineer fonksiyon (3.2) karşılaştırıcılar ile gerçeklenmiştir.

$$f(x) = \begin{cases} -1, & \text{eğer } x < -1 \\ x, & \text{eğer } -1 \leqslant x \leqslant 1 \\ 1, & \text{eğer } x > 1 \end{cases} \tag{3.2}$$

3.3.1.2 HYSA Kontrol Birimi

HYSA kontrol birimi ana görevi HYSA AİB'ne gerekli parametreleri, komut çözücüden aldığı komutlar çerçevesinde iletmektedir. Bu işlemi gerçekleştirebilmek için DDR2 kontrolörü aracılığı ile DDR2 SDRAM'in belirli portlarından veri alış verişinde bulunur. Bu veri transferinin senkronizasyonu yine bu birim içerisindeki durum makineleri ve sayıcılar ile sağlanır.

Bu ünitenin başlıca gerçekleştirdiği görevler aşağıdaki gibi sıralanabilir:

* Komut çözücüden gelen HYSA komutun işlenmesini sağlar.
* Komut çözücüden gelen parametrelere dayanarak HYSA AİB'ne giriş görüntüsünü, durum değişkeninin önceki değerini, hesaplanan değerini ve şablon değerleri adresinin gönderir. Çıkış görüntüsü ve yeni durum değerini alır.
* DDR2 kontrolörün sıfırıncı portu ile durum değişkeninin önceki değerini okur ve hesaplanan değeri yazar.
* DDR2 kontrolörün birinci portu ile giriş görüntüsünü okur ve çıkış görüntüsünü

yazar.

* DDR2 kontrolörün ikinci portu ile çıkış görüntüsünü yazar.

* İşlem yapılan hücrenin konumunu tutar.

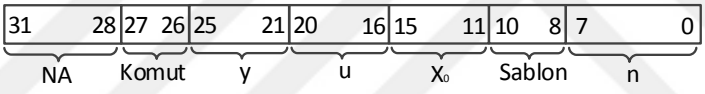
* Aritmetik işlem birimi içerisindeki BRAM'lerin adreslerini kontrol eder.

* Bir satır bittiğinde BRAM'leri kaydırarak işlemi bir satır alta taşır.

3.3.1.3 Komut Çözme Birimi

Komut Çözme Birimi; HYSA-Eİ'nin komutlarını program hafızasından yani FPGA ROM'undan okur, bunları çözer ve HYSA kontrolörüne aktarır. Olabildiğince basit bir komut çözücü gerçeklenmiştir. Her bir komut sonrasında komut sayıcısını bir artırarak bir sonraki işlemci komutunu okur. "İşlem Bitti" komutunu gördüğünde başa dönerek aynı işlemleri tekrar eder.

İşlenilecek olan kod 32 bit'liktir ve bunun içerisinde statik olarak komut tipi ve fonksiyon değişkenleri yerleştirilmiştir. Şekil 3.14 bu format verilmiştir.



Şekil 3.14 : Komut bit yerleşimi.

Proje kapsamında DDR2 bölme seçimi için 5 bitlik yer bırakılmıştır; 32 farklı bölme seçilebilir. 8 farklı şablon tanımlanmıştır, 3 bit Şablon seçimi için ayrılmıştır. 3 adet komut belirlendiğinden, komut seçimi için 2 bit ayrılmıştır. Bir komut içerisinde fonksiyon parametrelerinin anlamları ve bit yerleşimi:

(31-28) *NA*: Anlam atanmamış, daha sonrası için bırakılmış

(27-26) *Komut*: Koşturulacak komutun binary değeri

(25-21) *y*: Çıkışın yazılacağı DDR2 RAM bölmesi seçimi

(20-16) *u*: Girişin okunacağı DDR2 RAM bölmesi seçimi

(15-11) X_0: Durum değişkeni başlangıç değeri DDR2 RAM bölmesi seçimi, 0 yazılırsa sıfır alınır.

(10-8) *Sablon*: İşletilecek olan şablon matrislerinin yer aldığı ROM bölmesi seçimi

(7-0) *n*: İterasyon sayısı

Bu tez kapsamında komut setinde 3 adet "Komut" gerçeklenmiştir.

Görüntü Yakala: Kamera kontrolörüne yeni bir görüntüyü alıp DDR2 RAM'in belirtilen hafızasına kayıt etmesi emrini iletir. Binary komut karşılığı "01" dir.

Örnek Komut: Görüntü yakala ve bu görüntüyü DDR2 RAM'in birinci bölmesine kaydet

Örnek Komutun hexadecimal karşılığı: (0x5xxx) **HYSA işlemi gerçekleştir:** Parametreleri komutun içerisinde verilen bir HYSA ağını bir görüntü için belirlenen şablonlar ile belirlenen iterasyon sayısınca çalıştırır ve sonucu kayıt eder. Binary komut karşılığı "10" dır.

Örnek Komut: Birinci DDR bölmesindeki görüntüye 2 numaralı şablon ile 5 iterasyon çevrimi uygula, başlangıç koşulu sıfır al ve sonucu üçüncü DDR2 bölmesine kayıt et.

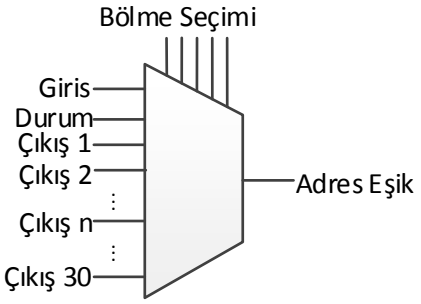
Örnek Komutun hexadecimal karşılığı: (0x9000)

İşlem Bitti: İşlenecek olan komutların bittiğini belirtir. İşlemci ilk komuta geri döner. Binary komut karşılığı "11" dir. Örnek Komutun hexadecimal karşılığı: (0xFxxx)

Yazılacak veya okunacak DDR2 bölme seçimi DDR2 adres değerine tanımlanan eşik değeri eklenmesi ile gerçekleştirilir. Gerçeklenen yapılın sayısal devre karşılığı 32 bitlik seçicidir. 5 bitlik DDR2 bölme seçimi sayısal devresi Şekil 3.15'da gösterilmiştir.

1. Bölme Eşik (Giris): 0
2. Bölme Eşik (Durum): 1600x900x2
3. Bölme Eşik (Çıkış 1): 1600x900x4
4. Bölme Eşik (Çıkış 2): 1600x900x6

n+2. Bölme Eşik (Çıkış n): 1600x900x(n+1)x2



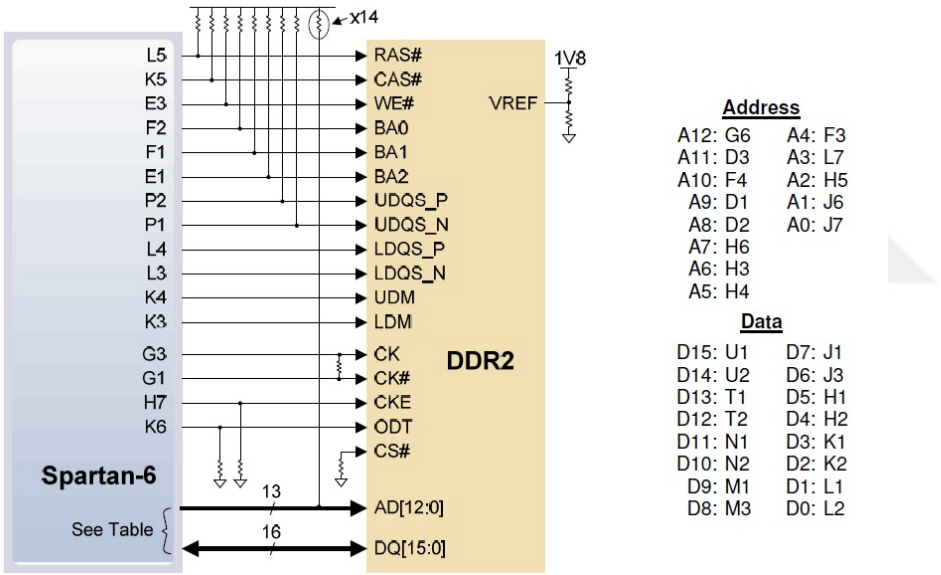
Şekil 3.15 : DDR2 adresine eşik değeri ataması.

3.3.2 Çevre Üniteleri

3.3.2.1 DDR2 Arayüzü

Son yıllarda birçok geliştirme platformu artan geçici hafıza ihtiyacını karşılamak için Double Data Rate (DDR) SDRAM kullanmaktadır. Saat işaretin hem yükselen hem de düşen kenarında veri transferi yaptıklarından dolayı çift veri hızlı olarak isimlendirilmişlerdir. DDR2 standardı buna ek olarak hafızaya karşılık veri yolu hızını iki katına çıkararak bir hafıza hücresi çevriminde 4 veri kelimesinin aktarılmasını sağlamıştır.

Geliştirme kartı üzerinde Micron MT47H64M16-25E 16 bit veri yolu uzunluklu, 128 Mbyte büyüklüklü DDR2 hafıza bulunmaktadır. 800 Mhz veri hızına kadar çalışabilmektedir. Kullanılan DDR2 hafızanın FPGA bağlantı şeması Şekil 3.16'te verilmiştir.



Şekil 3.16 : DDR2 ve FPGA arası bağlantı şeması.

3.3.2.2 VmodCam Arayüzü

Giriş görüntüsünü almak için VmodCAM çift CMOS kamera FPGA kartına VHDCI portu aracılığı ile bağlanmaktadır. VmodCAm'in üzerinde 2 adet Aptina MT9D112 2-Megapiksel CMOS dijital görüntü sensörü bulunmaktadır. Bu kameranın bazı özellikleri:

* Üzerinde bulunan görüntü akış işlemcisi sayesinde çıkış formatları (YCrCb, RGB or BAYER)seçilebilmektedir.
* 1600x1200 maksimum çözünürlüğü 15 fps hızda verebilmektedir.

* I2C arayüzü ile CMOS sensörler kontrol edilmektedir.

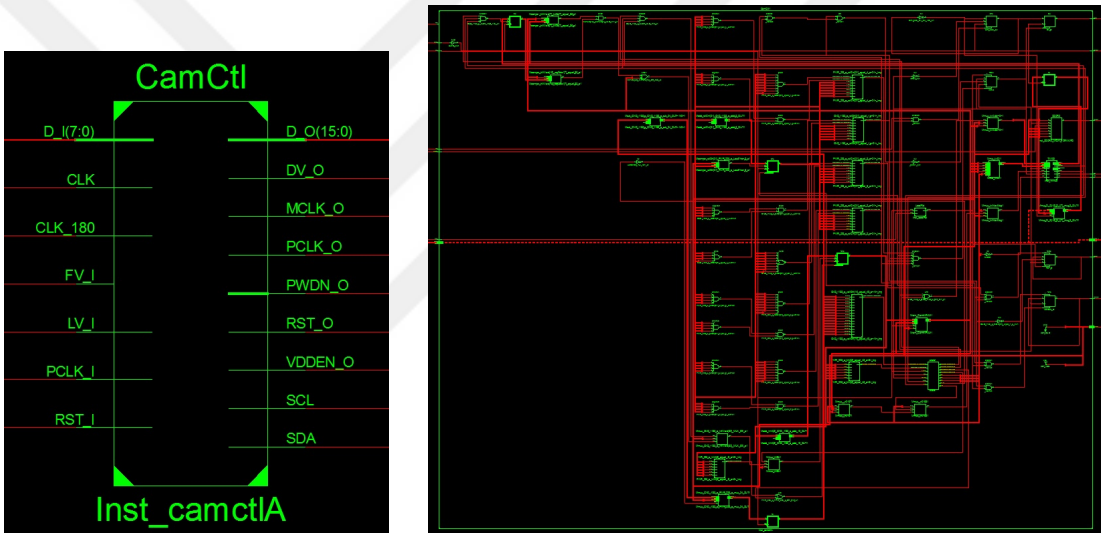
* Otomatik kazanç ve beyaz dengesi.

* Görüntü düzeltme algoritmaları.

* Görüntü orantılama ve kesme.

Kameraların birbirinden bağımsız arayüzleri vardır. Biz projemizde CMOS sensörlerin birini kullanmaktayız. 15 fps'ta 1600x1200 görüntüyü standart HDMI görüntü oranlarına uyması için 1600x900'e kırparak alıyoruz. Görüntü kalitesinin iyileştirmek için sensörü otomatik beyaz dengelemesi, kazanç ve poz açıklığı açık olarak kullanıyoruz. Görüntüyü RGB565 formatında alıyoruz.

Şekil 4.5'de kamerayı kontrol eden bloğunun FPGA şematiği verilmiştir.



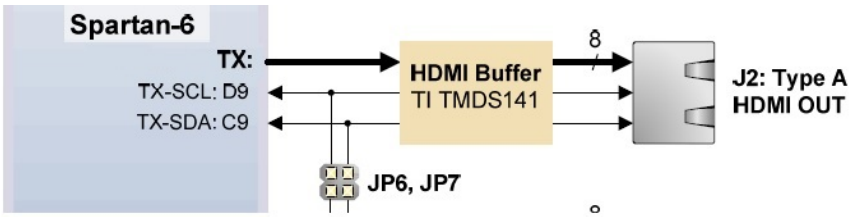
(a) (b)

Şekil 3.17 : FPGA içerisindeki Kamera Kontrol bloğu.

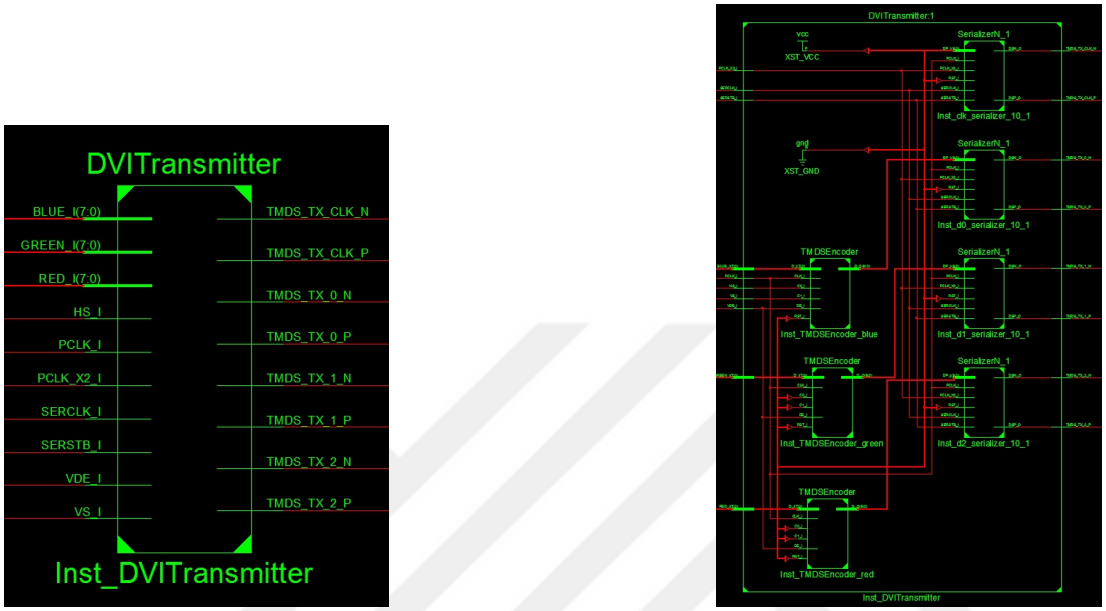
3.3.2.3 HDMI Arayüzü

Görüntü FPGA'den monitöre HDMI arayüzü ile aktarılmaktadır. Sinyal seviyesini korumak için geliştirme kartı üzerinde TI firmasının TMDS141 HDMI tekrarlayıcı (repeater) bulunuyor. 4 adet farksal veri yolu TDMS sinyali ve I2C portu bu entegre ile tamponlanıp HDMI konnektörüne iletiliyor. HDMI standartında bulunan diğer sinyaller kullanılmamaktadır. Kullanılan HDMI çıkışının FPGA bağlantı şeması Şekil 3.18'de verilmiştir.

Şekil 3.19'te HDMI çıkışa görüntüyü ileten DVI Kontrolör bloğu FPGA şematiği ve içeriği verilmiştir.



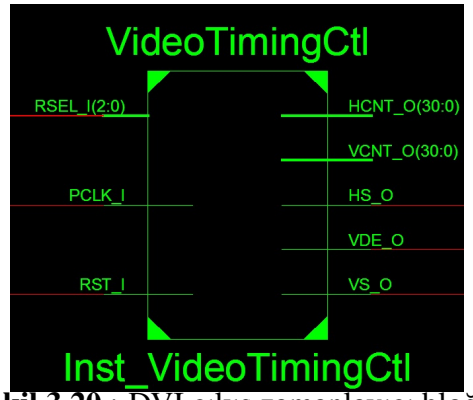
Şekil 3.18 : HDMI tampon ve FPGA arası bağlantı şeması.



(a) (b)

Şekil 3.19 : FPGA içerisindeki DVI çıkış bloğu.

DVI kontrol ünitesine her renk için 8, toplamda 24 bitlik RGB formatındaki giriş verisini TDMS kodlayıcı ve paralel-seri çeviriciler ile HDMI tamponuna gönderiyor. Buradan da HDMI konnektöre bağlanan monitör ile çıkış görüntüsü gözlemleniyor. Şekil 3.20'de gösterilen ayrı bir "Video Zamanlayıcı" bloğu Yatay Eşleşme (HS), Dikey Eşleşme (VS) ve veri hazır (VDE) sinyallerini saat sinyali, (CLK) ve sayıcıları yardımı ile istenilen çözünürlüğe göre üretmektedir.



Şekil 3.20 : DVI çıkış zamanlayıcı bloğu.

4. SONUÇLAR ve ÖNERİLER

Bu sistem digilent firmasının Atylss kartında gerçeklendi. Atlys kartı Xilinx Spartan 6 XC6SLX45 FPGA barındırmaktır. Kamera olarak VmodCam kamera kullanılmıştır. VHDCI arayüzü ile kamera FPGA kartına bağlanmıştır. Gerçeklenen sistem HD+ (1600x900) çözünürlükteki görüntüyü @15 Hz hıza kadar işleyebilmektedir. Sistem performansı iterasyon sayısının artması ile düşmektedir. Bir iterasyon görüntüyü alma, aktarma ve DDR2 RAM'e yazma dahil 50 ms'den az sürmektedir. FPGA 108 Mhz saat frekansında çalıştırılmıştır.

Bir piksel 2 byte ile ifade edilmektedir, yani görüntü boyutu:

$$\frac{1600 \cdot 900 \cdot 2}{1024 \cdot 1024} \approx 2,75 \text{ Mbyte} \tag{4.1}$$

Gerçeklenen sistemde 128 Mbyte'lık DDR2 RAM bulunmaktadır. Buna göre gerçeklenebilecek DDR2 RAM bölme sayısı:

$$\frac{128}{2,75} \approx 46 \text{ DDR2 RAM Bölmesi} \tag{4.2}$$

4.1 HYSA-Eİ Testi

HYSA-Eİ "Görüntü Yakala","HYSA" işlemi gerçekleştir ve "bitti, Başa dön" komutlarını işleyebilmektedir.

HYSA komutu `Y=HYSA(U,X0,Sab,n)` fonksiyonu gerçeklemektedir. Bu fonksiyondaki değişkenlerin anlamları:

y: Çıkışın yazılacağı DDR2 RAM bölmesi seçimi

u: Girişin okunacağı DDR2 RAM bölmesi seçimi

X_0: Durum değişkeni başlangıç değeri DDR2 RAM bölmesi seçimi, 0 yazılırsa sıfır alınır.

Sab: Şablon matrisi ROM bölmesi seçimi

n: İterasyon sayısı

HYSA-Eİ kabiliyetlerini göstermek için, komut hafızasına HYSA işlemci kodları yazıldı.

4.1.1 Örnek İşlemci Kodları ve Çıktıları

HYSA-Eİ kabiliyetlerini göstermek için, komut hafızasına HYSA işlemci kodları yazıldı. Kayıt alabilmek için sistem bir sonraki komuta geçmeden bekler hale getirildi. Sistemin çıkışı HDMI'dan kayıt cihazı ile kayıt edildi. Görünürlüğü arttırmak için çıkış görüntülerinin renkleri ters çevrildi.

Dikey Hariç Kenar Tespiti: Bu algoritma 4 adet komuttan oluşmaktadır.

* 1. komut giriş görüntüsünü birinci DDR2 RAM bölmesine yazmaktadır.

* 2. komut kenar tespiti komutudur. Giriş görüntüsünü birinci DDR2 RAM bölmesinden okur, birinci DDR2 bölmesindeki giriş görüntüsünü başlangıç durumunu alarak 5 iterasyon birinci ROM bölmesindeki kenar tespiti şablonu ile HYSA'yı koşturur ve çıkışı üçüncü DDR2 RAM bölmesine yazar.

* 3. komut kenar tespiti yapılmış görüntüden dikey kenarları siler. Giriş görüntüsünü üçüncü DDR2 RAM bölmesinden okur, başlangıç durumunu sıfır alarak 1 iterasyon ikinci ROM bölmesindeki dikey kenar silme şablonu ile HYSA'yı koşturur ve çıkışı dördüncü DDR2 RAM bölmesine yazar.

* 4. komut algoritmanın tamamlandığı yani işlemin bittiğini belirtir. İşlemci birinci komuta geri döner.

Bu kodlar ve sonuçlarını aşağıda gösterilmektedir.

```
1: procedure DIKEY HARIÇ KENAR TESPITI
2:     DDRBolum1=Görüntü Yakala,
3:     DDRBolum3=HYSA(DDRBolum1,DDRBolum1,Sablon1,n=5),
4:     DDRBolum4=HYSA(DDRBolum3,0,Sablon2,n=1 ,
5:     bitti, Başa Dön
6: end procedure
```

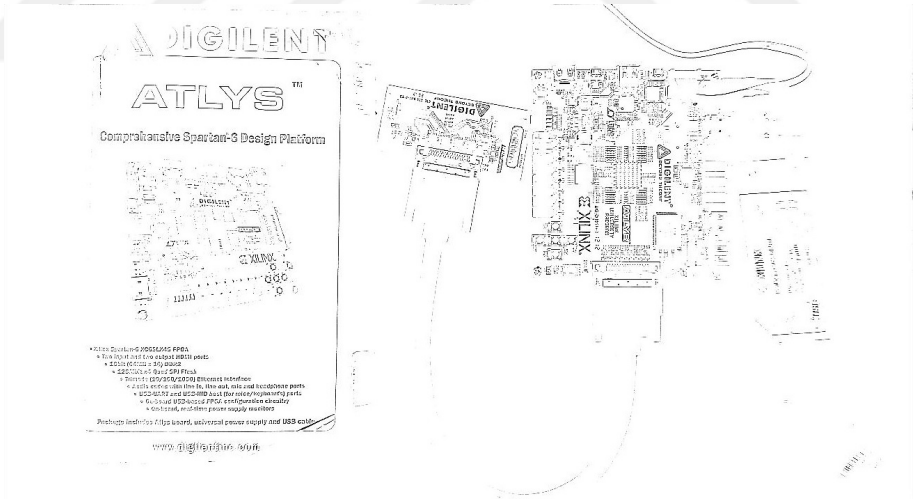
Sablon1 kenar tespiti şablonu [15]:

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, B = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \text{ ve } I = -1$$

Sablon2 dikey kenar silme şablonu [15]:

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{pmatrix} \text{ ve } I = -2$$

Şekil 4.1 : Örnek giriş görüntüsü.



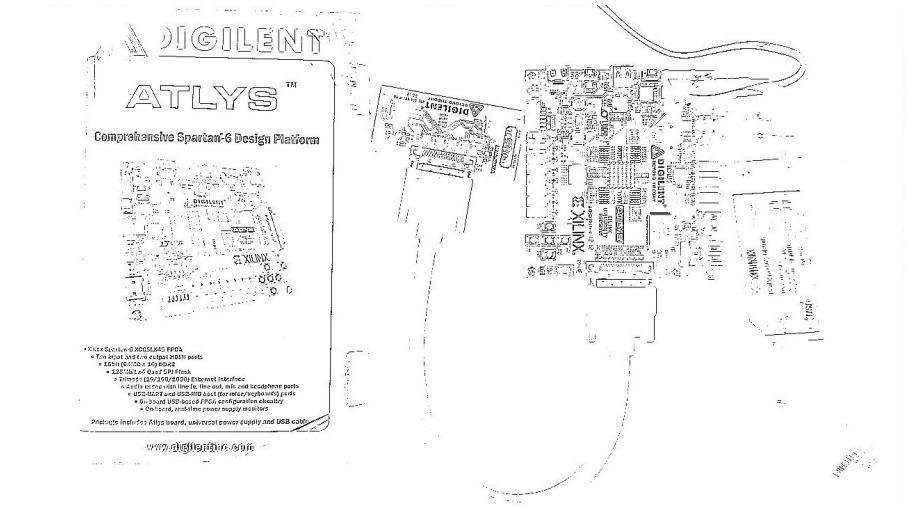
Şekil 4.2 : 1. HYSA Komutu: Kenar tespiti şablonu 1. iterasyon sonucu.

Kenar tespiti komutunun başarılı bir şekilde çalıştığını görüyoruz. Dikey kenar silme komutunun üst üste gelmiş siyah çizgileri sildiği görülmektedir.

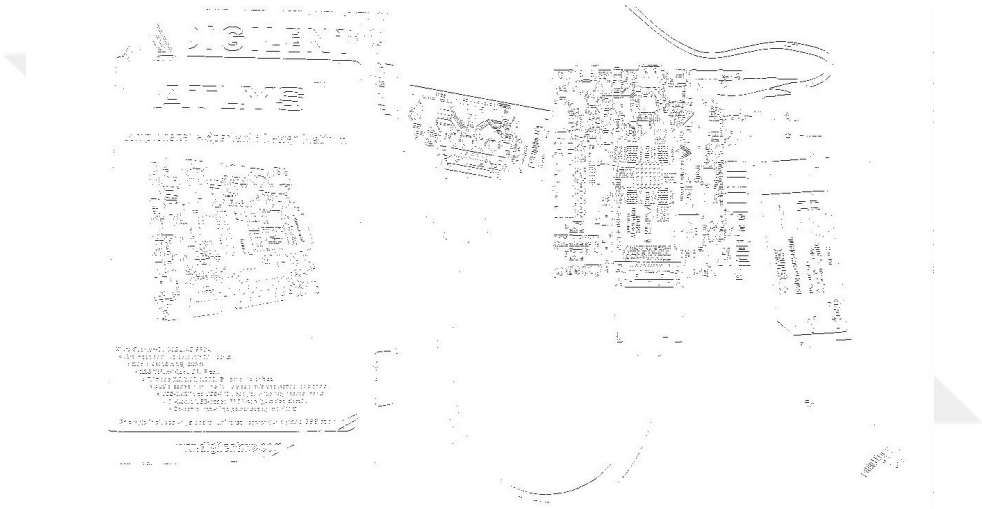
Binary Hareketli Cisim Tespiti: Bu algoritma 6 adet komuttan oluşmaktadır.

* 1. komut giriş görüntüsünü birinci DDR2 RAM bölmesine yazmaktadır.

* 2. komut kenar tespiti komutudur. Giriş görüntüsünü birinci DDR2 RAM bölmesinden okur, başlangıç durumunu sıfır alarak 3 iterasyon birinci ROM



Şekil 4.3 : 1. HYSA Komutu: Kenar tespiti şablonu 5. iterasyon sonucu.



Şekil 4.4 : 2. HYSA Komutu: Dikey çizgi silme şablonu sonucu.

bölmesindeki kenar tespiti şablonu ile HYSA'yı koşturur ve çıkışı üçüncü DDR2 RAM bölmesine yazar.

* 3. komut yeni giriş görüntüsünü dördüncü DDR2 RAM bölmesine yazmaktadır.

* 4. komut kenar tespiti komutudur. Giriş görüntüsünü dördüncü DDR2 RAM bölmesinden okur, başlangıç durumunu sıfır alarak 3 iterasyon birinci ROM bölmesindeki kenar tespiti şablonu ile HYSA'yı koşturur ve çıkışı beşinci DDR2 RAM bölmesine yazar.

* 5. komut hareket algılama (iki görüntünün farkını alma) komutudur. Giriş görüntüsünü beşinci DDR2 RAM bölmesinden okur, başlangıç durumunu üçüncü DDR2 RAM bölmesinden alarak 1 iterasyon üçüncü ROM bölmesindeki hareket tespiti şablonu ile HYSA'yı koşturur ve çıkışı altıncı DDR2 RAM bölmesine yazar.

* 6. komut ortalama alma (binary çıkış ile yumuşatma) komutudur. Giriş görüntüsünü altıncı DDR2 RAM bölmesinden okur, başlangıç durumunu altıncı DDR2 RAM bölmesinden alarak 1 iterasyon dördüncü ROM bölmesindeki ortalama alma şablonu ile HYSA'yı koşturur ve çıkışı yedinci DDR2 RAM bölmesine yazar.

* 7. komut algoritmanın tamamlandığı yani işlemin bittiğini belirtir. İşlemci birinci komuta geri döner.

Bu kodlar ve sonuçlarını aşağıda gösterilmektedir.

1: **procedure** BINARY HAREKETLI CISIM TESPITI
2: DDRBolum1=Görüntü Yakala,
3: DDRBolum3=HYSA(DDRBolum1,0,Sablon1,n=3),
4: DDRBolum4=Görüntü Yakala,
5: DDRBolum5=HYSA(DDRBolum4,0,Sablon1,n=3),
6: DDRBolum6=HYSA(DDRBolum5,DDRBolum3,Sablon3,n=1),
7: DDRBolum7=HYSA(DDRBolum6,DDRBolum6,Sablon4,n=1),
8: bitti, Başa Dön
9: **end procedure**

Sablon3 hareketli cisim tespiti şablonu:

$$A = \begin{pmatrix} 0\,0\,0 \\ 0\,1\,0 \\ 0\,0\,0 \end{pmatrix}, B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ ve } I = -1$$

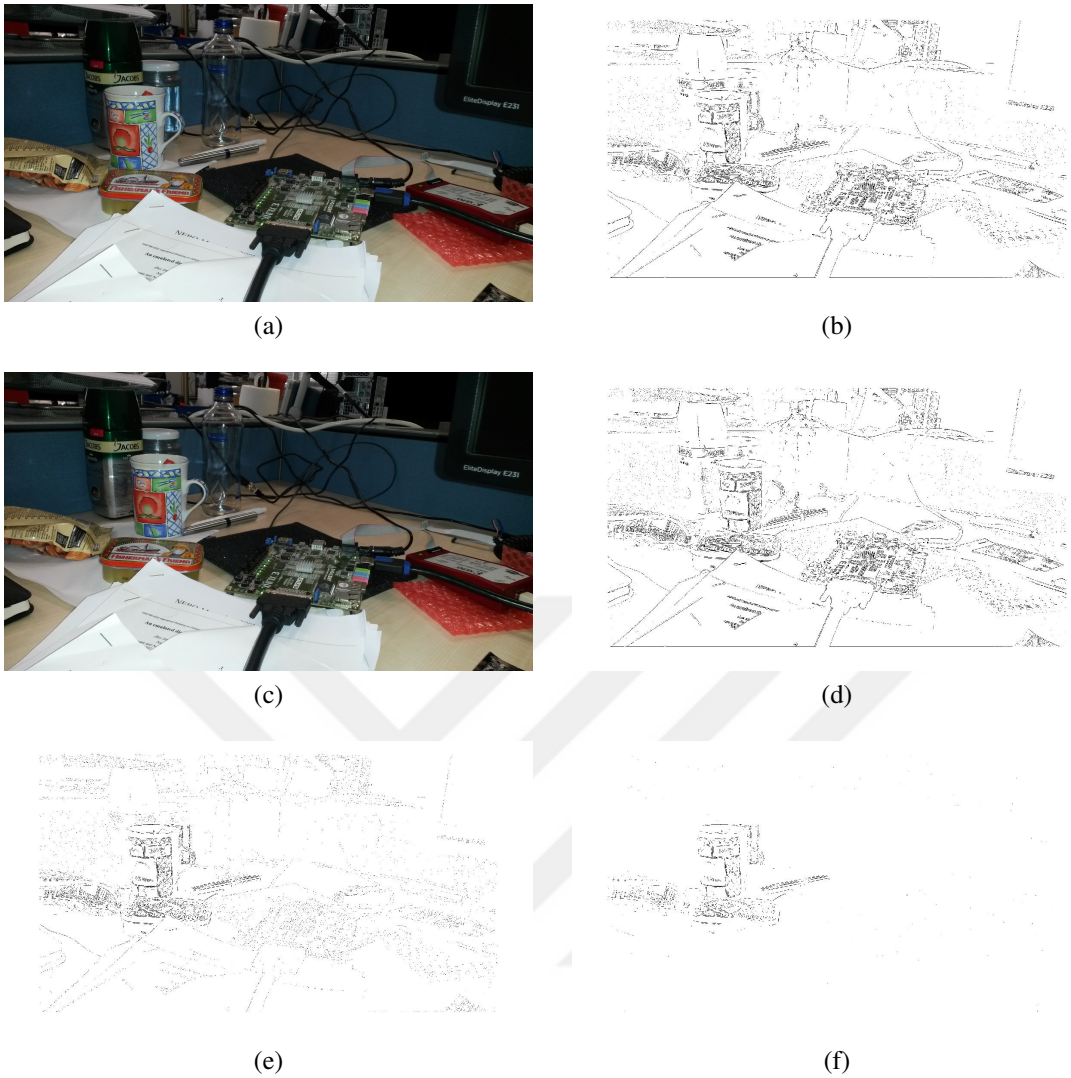
Sablon4 ortalama alma şablonu:

$$A = \begin{pmatrix} 0\,1\,0 \\ 1\,2\,1 \\ 0\,1\,0 \end{pmatrix}, B = \begin{pmatrix} 0\,0\,0 \\ 0\,0\,0 \\ 0\,0\,0 \end{pmatrix} \text{ ve } I = 0$$

Ortadaki bardak ve etrafı sadece hareket ettirilmiştir. Bunun tespit edilebildiği görülmektedir. Fakat hareket tespitini sadece fark alma işlemi ile yaptığından, kenar tespitindeki küçük farklılıklarda gürültü olarak yansımaktadır. Bu gürültüyü gidermek için ortalama alma işleme yapılmıştır. Başarılı bir şekilde çalıştığı gözlemlenmiştir.

4.2 Sonuçlar

Bu tez çalışmasında yeni bir HYSA emülatör işlemcisi mimarisi ortaya konmuştur ve bu FPGA üzerinde gerçeklenmiştir. Bu işlemci Ayrık Zamanlı HYSA işleminin gerçekleyebilmektedir ve değişkenlerinin hepsi programlanabilirdir. Üretilen sonuçlar



(a) (b)

(c) (d)

(e) (f)

Şekil 4.5 : (a) Birinci görüntü. (b) Birinci görüntünün kenar tespiti sonucu. (c) Bardağın sağa hareket ettiği ikinci görüntü. (d) İkinci görüntünün kenar tespiti sonucu (e) Hareketli cisim tespiti sonucu. (f) Hareket tespiti sonucunun ortalaması

harici hafızada kayıt edilerek bir sonraki komut tarafından kullanılabilir. Böylelikle ardışıl HYSA fonksiyonları gerçeklenebilir. Tasarlanan sistemin FPGA kaynak kullanımı bakımından verimli, işlem gücü yüksek yani hızlı olmasına özen gösterilmiştir. Farklı şablonlar ve HYSA komutları ile yapılan deneylerin sonuçları, bu işlemcisinin kabiliyetlerini göstermektedir. En son olarak sistemin performans değerleri verilmiştir.

4.2.1 Lojik Kaynak Kullanımı

FPGA donanım kullanım tablosu Çizelge 4.1'de verilmiştir.

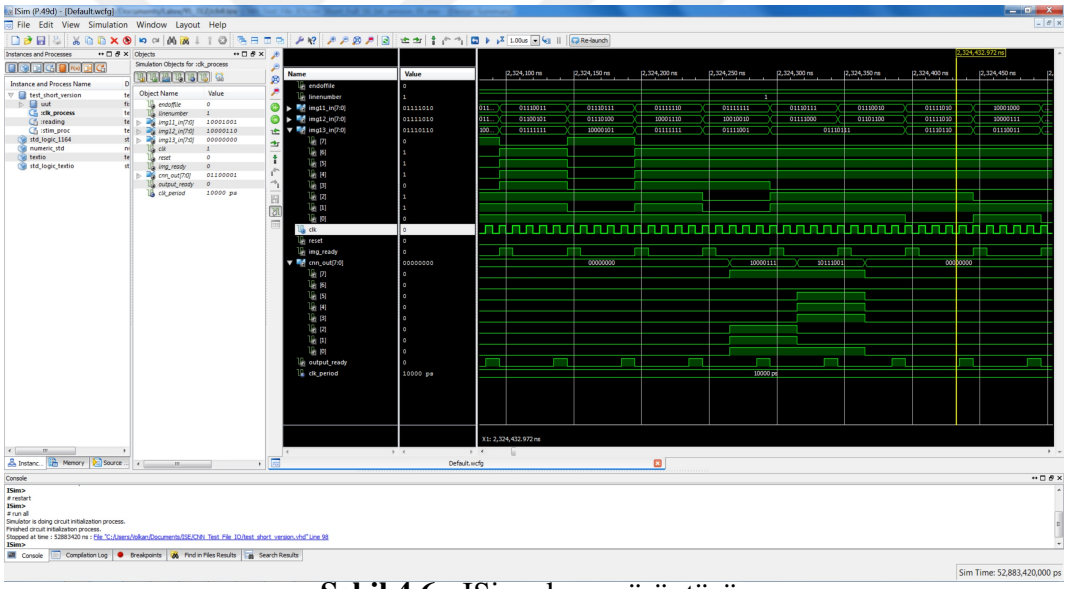
Çizelge 4.1 : Donanım kullanım tablosu.

Lojik Kaynak	Kullanılan	Uygun	Yüzde
Slice'lar	831	6822	%12
FlipFlop'lar	891	2427	%36
RAMB16BWER'ler	20	116	%17
DSP48A1'lar	18	58	%31

4.2.2 Zaman Analizi

ISE'nin sentez zamanlama raporuna göre çalışabileceği maksimum frekans 122 Mhz'dir. Sistemde kamera kontrol bloğu 80 Mhz'te, HYSA ve DVI çıkış bloğu 108 Mhz'te çalıştırılmıştır.

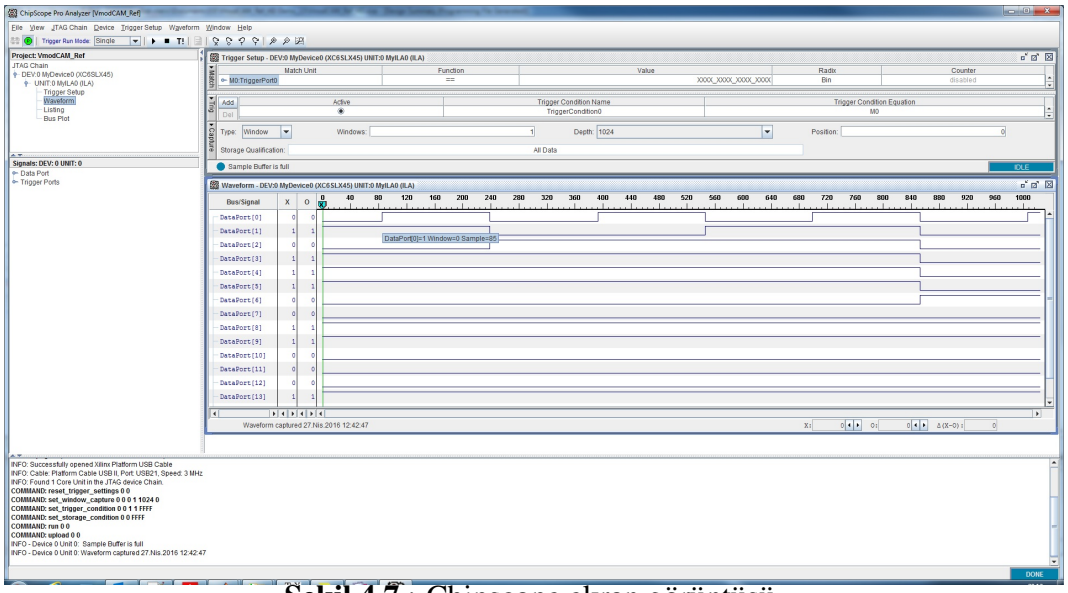
Dosyadan giriş çıkış ile test edilen HYSA algoritmasından ISim test sonucu ekran görüntüsü Şekil 4.6'te verilmiştir. Buradan görülebileceği üzere giriş hazır sinyalinden sonra tek bir HYSA hücresinin hesaplanması 3 saat darbesi sürmektedir.



Şekil 4.6 : ISim ekran görüntüsü.

Sistemin hız kısıtını yaratan DDR2 SDRAM yazma ve okumadır. DDR2 SDRAM'e yazma ve okumadan dolayı sistemin hız kestirimi sentezlenen devreye bakılarak yapılamamaktadır. Sistem performansı sistem çalışırken Xilinx firmasının hata ayıklama aracı "Platform Cable" ve programı "Chipscope" aracılığı ile ölçülmüştür. Durum değişkeni yazan DDR2 portu adres değişiminin "Chipscope" program görüntüsü Şekil 4.7'te verilmiştir.

Adres sayacı 32 kelime yazıldığından değişmektedir. 160 saat darbesi sürdüğüne göre tek bir pikselin iterasyonu 5 saat darbesi sürmektedir. Buradan sistemin performansı:



Şekil 4.7 : Chipscope ekran görüntüsü.

$$\frac{1600 \cdot 900 \cdot 5}{108 \cdot 10^6} \approx 16,6 \tag{4.3}$$

Bu işlemci 16,6 iterasyon/s hızda işlem yapabilmektedir.

4.2.3 Literatür ile Karşılaştırması

Literatürde yer alan mimarilerin bazıları ile önerilen HYSA-Bİ mimarisinin performans başarımları Çizelge 4.2'de verilmiştir.

Çizelge 4.2 : Mimarilerin performans karşılaştırması.

	FALCON	CESAR	Steadfast	HYSA-Bİ
Saat frekansı	133 MHz	100 MHz	123 MHz	108 MHz
Boyut	640x480	640x480	640x480	1600x900
Bit çözünürlüğü	64	18	8	16
3x3 konv. süresi	1 ns	150 ns	30 ns	50 ns
iteration/s	20,3	14,7	31	16,6

Karşılaştırma tablosu incelendiğinde FALCON mimarisinin bulundurduğu paralel işlemciler nedeni ile en hızlı olduğu görülmektedir, ayrıca en yüksek bit çözünürlüğüne sahiptir. Steadfast mimarilerisindeki hafıza bulundurmayan yapı bu mimarinin iterasyon/s bazında en hızlı olmasını sağlamaktadır, ayrıca hızını iterasyon sayısından bağımsız kılmaktadır. Önerilen HYSA-Bİ mimarisinin harici hafıza yazıp okuma işlemleri hızını kısıtlayan temel faktördür. Paralel yazılan hafıza okuma, yazma ve HYSA işlemleri bu dezavantajı azaltmıştır, fakat yinede iterasyon sayısı ile hızı düşmektedir.

Daha önceki mimariler ile önerilen HYSA-Bİ mimarasi karşılaştırıldığında en büyük avantajı esnekliğidir. CASTLE ve Steadfast mimarilerinde sadece şablon ve iterasyon sayısı değiştirilebiliyor. FALCON mimarisinde bunlara ek olarak bit çözünürlüğü ve şablon büyüklüğü de değiştirilebiliyor. Sadece CESAR mimarisi ardışıl HYSA işlem koşturma kabiliyetine sahipti, fakat sadece bir önceki değeri kullanabiliyordu.

Tasarlanan mimarinin kaynak kullanımı ve zaman başarımının önceki mimarilere yakın olduğu görülmektedir. Literatüre katkısı ise kayıt edilen sonuçların daha sonraki HYSA işlemlerinde kullabilen ilk mimaridir. Böylelikle tek bir HYSA işlemi ile gerçeklenemeyen ardışıl işlemler bu mimari ile gerçeklenebilmektedir.

4.3 Öneriler

Gelecekteki çalışmalarda, HYSA iterasyonları ardışık düzende yazılarak sistemin gerçek zamanlı performansı iyileştirilebilir. Aritmetik İşlem Birimine HYSA harici fonksiyonlar (komutlar) eklenebilir. Eklenilecek öğrenme algoritması bloğu ile kameradan alınan görüntüler istenilen çıkış ile karşılaştırılıp, şablon tespiti yapılabilir. Ayrıca sistemde yer alan çift kameranın diğerini de kullanılıp çift görünümden yararlanıp işlemci kabiliyetleri arttırılabilir. Çift görüşten yararlanarak görüntüye derinlik eklenmesi ile üç boyutlu görüntü işlenebilir.

KAYNAKLAR

[1] **Chua, L. ve Yang, L.** (1988). Cellular Neural Networks: Theory, *IEEE Transactions On Circuits and Systems*.

[2] **Chua, L. ve Yang, L.** (1988). Cellular Neural Networks: Applications, *IEEE Transactions On Circuits and Systems*.

[3] **Yeniçeri, R. ve Yalçın, M.E.** (2012). A new CNN based path planning algorithm improved by the Doppler effect, *2012 13th International Workshop on Cellular Nanoscale Networks and their Applications*, s.1–5.

[4] **Yeniceri, R. ve Yalcin, M.** (2008). An implementation of 2D locally coupled relaxation oscillators on an FPGA for real-time autowave generation, *The 11th International Workshop on Cellular Nanoscale Networks and their Applications*.

[5] **Yeniceri, R. ve Yalcin, M.** (2009). An Emulated Digital Wave Computer Core Implementation, *Proc. of European Conference on Circuit Theory and Design 2009 (ECCTD'09)*.

[6] **Roska, T. ve Chua, L.O.** (1993). The CNN universal machine: an analogic array computer, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, *40*(3), 163–173.

[7] **Linan, G.**, **Espejo, S.**, **Dominguez-Castro, R. ve Rodriguez-Vazquen, A.** (2000). The CNNUC3: an analog I/O 64x64 CNN universal machine chip prototype with 7-bit analog accuracy, *Cellular Neural Networks and Their Applications, 2000. (CNNA 2000). Proceedings of the 2000 6th IEEE International Workshop on*, s.201–206.

[8] **Linan, G.**, **Dominguez-Castro, R.**, **Espejo, S. ve Rodriguez-Vazquez, A.** (2001). ACE16K: An advanced focal-plane analog programmable array processor, *Solid-State Circuits Conference, 2001. ESSCIRC 2001. Proceedings of the 27th European*, s.201–204.

[9] **Müller, J.**, **Becker, R.**, **Müller, J. ve Tetzlaff, R.** (2012). CESAR: Emulating Cellular Networks on FPGA, *The 13th International Workshop on Cellular Nanoscale Networks and their Applications*.

[10] **Braunschweig, R.**, **Müller, J.**, **Müller, J. ve Tetzlaff, R.** (2013). NERO mastering 300k CNN cells, *European Conference on Circuit Theory and Design (ECCTD)*.

[11] **A. Zarandy, P. Keresztes, T.R. ve Szolgay, P.** (1998). "CASTLE: an emulated digital CNN architecture; design issues, new results, *IEEE International Conference on Electronics Circuits and Systems*.

[12] **Nagy, Z. ve Szolgay, P.** (2003). Configurable multilayer CNN-UM emulator on FPGA, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*.

[13] **N. Yildiz, E.C. ve Tavsanoglu, V.** (2008). A New Approach to Emulate CNN on FPGAs for Real Time Video Processing, *The 11th International Workshop on Cellular Nanoscale Networks and their Applications*.

[14] **N. Yildiz, E.C. ve Tavsanoglu, V.** (2012). Demonstration of the Second Generation Real-Time Cellular Neural Network Processor: RTCNNP-v2, *The 13th International Workshop on Cellular Nanoscale Networks and their Applications*.

[15] **L. Kek, K.K. ve Roska, T.**, (2007). Cellular Wave Computing Library, Hungarian Academy of Science, Cellular sensory wave Computers Laboratory, BUDAPEST, 2.1 sürüm.

ÖZGEÇMİŞ

Ad Soyad: Volkan MERİÇ

Doğum Yeri ve Tarihi: Muradiye 08.10.1987

Adres:Tübitak MAM Yerleşkesi BTE Binası Gebze/KOCAELİ

E-Posta:volkan.mrc@gmail.com

Lisans:İstanbul Teknik Üniversitesi, Elektronik Mühendisliği

Mesleki Deneyim ve Ödüller: Araştırmacı, TUBİTAK BİLGEM, Kasım 2010-

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- **Meriç V.** and Yalçın M.E., 2016: A New Implementation of Fully Programmable Complete CNN Processor Core on FPGA *The 15th International Workshop on Cellular Nanoscale Networks and their Applications*, August 23-25, 2016 Dresden, GERMANY.