T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GERÇEK ZAMANLI BİR HÜCRESEL SİNİR AĞI YAPISININ TASARIMI VE BU YAPIYLA GABOR FİLTRELERİNİN FPGA ÜZERİNDE GERÇEKLENMESİ

EVREN CESUR

DOKTORA TEZİ
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ
ANABİLİM DALI
ELEKTRONİK PROGRAMI

DANIŞMAN
PROF. DR. VEDAT TAVŞANOGLU

İSTANBUL, 2013

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

GERÇEK ZAMANLI BİR HÜCRESEL SİNİR AĞI YAPISININ TASARIMI VE BU YAPIYLA GABOR FİLTRELERİNİN FPGA ÜZERİNDE GERÇEKLENMESİ

Evren CESUR tarafından hazırlanan tez çalışması 22/11/2012 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda **Doktora Tezi** olarak kabul edilmiştir.

Tez Danışmanı
Prof. Dr. Vedat TAVŞANOĞLU
Yıldız Teknik Üniversitesi

Jüri Üyeleri
Prof. Dr. Vedat TAVŞANOĞLU
Yıldız Teknik Üniversitesi _____

Prof. Dr. Tülay YILDIRIM
Yıldız Teknik Üniversitesi _____

Prof. Dr. Sabri ARIK
Işık Üniversitesi _____

Prof. Dr. Oruç BİLGİÇ
İstanbul Kültür Üniversitesi _____

Prof. Dr. Serdar ÖZOĞUZ
İstanbul Teknik Üniversitesi _____

Bu çalışma Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) tarafından 108E023 numaralı proje kapsamında desteklenmiştir.

ÖNSÖZ

Bu çalışmam süresince her türlü desteğini ve fikirlerini benden esirgemeyen Prof.Dr. Vedat Tavşanoğlu'na, bu uzun süreçte yardımlarını esirgemeyen ve en zor anlarda destek olan çalışma arkadaşlarım Nerhun Yıldız ve Murathan Alpay'a, yaptıkları çalışmalar ve destekleri için Yard.Doç.Dr Eruğrul Saatçi, Nergis Tural Polat ve Kamer Kayaer'e, tüm sıkıntılarımı aşarken yanımda olan arkadaşım Seyhan Ağaoğlu'na ve desteklerini hissettiğim tüm arkadaşlarıma saygılarımı ve teşekkürlerimi sunarım.

Son olarak tüm eğitim hayatım boyunca manevi ve maddi desteklerini esirgemeyen annem Emine Cesur, babam Ekrem Cesur, abim Günay Cesur, dayım Metin Erdoğdu ve tüm aileme müteşekkirim.

Şubat, 2013

Evren CESUR

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

$h(x;\mu,\sigma^2)$	Gauss fonksiyonu
σ^2	Gauss fonksiyonunun varyansı
μ	Gauss fonksiyonunun ortalama değeri
$H(j\Omega)$	Gauss fonksiyonunun frekans yanıtı
$\mathscr{F}\{\}$	Fourier dönüşümü
$\mathscr{F}^{-1}\{\}$	Ters Fourier dönüşümü
$g(x)$	Bir boyutlu Gabor fonksiyonu
$g(x,y)$	İki boyutlu Gabor fonksiyonu
$g^{\mathrm{R}}(x,y)$	İki boyutlu Gabor fonksiyonunun gerçel kısmı
$g^{\mathrm{I}}(x,y)$	İki boyutlu Gabor fonksiyonunun sanal kısmı
$G(j\Omega)$	Bir boyutlu Gabor fonksiyonunun frekans yanıtı
$G(j\Omega_x, j\Omega_y)$	İki boyutlu Gabor fonksiyonunun frekans yanıtı
$\Omega_0,\ \Omega_{x0},\ \Omega_{y0}$	İki boyutlu Gabor filtresinin merkez frekansı ve x, y bileşenleri
θ_0	İki boyutlu Gabor filtresinin doğrultu açısı
$\omega_0,\ \omega_{x0},\ \omega_{y0}$	İki boyutlu ayrık uzay Gabor filtresinin merkez frekansı ve x, y bileşenleri
$X_s,\ Y_s$	İki boyutlu uzayda x ve y doğrultularındaki örnekleme periyotları
$W \times W$	İki boyutlu ayrık Gabor fonksiyonun pencerelenme boyutları
$K \times L$	Hücresel sinir ağının boyutları
$C(i,j)$	Hücresel sinir ağının bir hücresi
$S_r(i,j)$	$C(i,j)$ hücresine ait r yarıçaplı etki küresi
r	Komşuluk yarıçapı
$i,\ j$	Kartezyen koordinat düzlemindeki x ve y değerleri $i \in \{1,2,...K\}$, $j \in \{1,2,...L\}$
$x_{ij}(t)$	$C(i,j)$ hücresine ait durum değişkeni
$\dot{x}_{ij}(t)$	$C(i,j)$ hücresine ait durum değişkeninin zamandaki türevi
u_{ij}	$C(i,j)$ hücresine ait giriş
$A(i,j;k,l)$	$C(i,j)$ hücresinin geri besleme sinaptik ağırlığı
$B(i,j;k,l)$	$C(i,j)$ hücresinin ileri besleme sinaptik ağırlığı
$\boldsymbol{A}$	Uzayda değişmeyen HSA'nın geri besleme şablonu
$\boldsymbol{B}$	Uzayda değişmeyen HSA'nın ileri besleme şablonu
$\circledast$	Karşılıklı elemanların çarpılıp, sonuçların toplandığı şablon nokta çarpımı operatörü
$\boldsymbol{X}_{ij}(t)$	$C(i,j)$ hücresine ait durum değişkeninin etki alanı
$\boldsymbol{U}_{ij}$	$C(i,j)$ hücresine ait girişinin etki alanı
$\mathbf{x}(\mathbf{t})$	$KL \times 1$ boyutundaki durumların vektörü
$\dot{\mathbf{x}}(\mathbf{t})$	$KL \times 1$ boyutundaki durumların türevlerinin vektörü

$\mathbf{u}$	$(KL \times 1)$ boyutundaki giriş vektörü
$\mathbf{A}$	$(KL \times KL)$ boyutlarındaki geri besleme matrisi
$\mathbf{B}$	$(KL \times KL)$ boyutlarındaki giriş matrisi
$\{\}^{\mathrm{R}}$	$\{\}$'nın gerçel kısmı
$\{\}^{\mathrm{I}}$	$\{\}$'nın sanal kısmı
$\tilde{\boldsymbol{A}}$	Çevresel şablon (surround template)
T_s	Zamandaki örnekleme periyodu
$\alpha,\ \beta,\ b$	Gabor benzeri HSA filtresinin katsayıları
λ^2	Gabor benzeri HSA filtresinin bant genişliğini belirler
$\lceil\ \rceil$	Yukarı yuvarlama operatörü
$S(i,j,t)$	Hız ayarlı filtrenin durum değişkeni
$v_0,\ v_{x0},\ v_{y0}$	Hız ayarlı filtrenin ayarlı olduğu hız ve x, y bileşenleri
τ	Hız ayarlı filtrenin zaman sabiti
$u_{ij}(t)$	Hız ayarlı filtrenin zamanla değişen giriş görüntüsü
t_f	İki video çerçevesi arasındaki süre
I_{in}, I_{out}	Karşıtlık ayarlama bloğunun giriş ve çıkışı

KISALTMA LİSTESİ

1-B	Bir Boyutlu
2-B	İki Boyutlu
3-B	Üç Boyutlu
ASIC	Application Specific Integrated Circuits
DDR–SDRAM	Double Data–Rate Synchronous Dynamic Random–Access Memory
DSP	Digital Signal Processors
DVI	Digital Visual Interface
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
GPU	Graphical Processing Unit
HAF	Hız Ayarlı Filtre
HD	High-Definition
HDMI	High-Definition Multimedia Interface
HSA	Hücresel Sinir Ağı
PC	Personal Computer
PDF	Problability Density Function
PSNR	Peak Signal-to-Noise Ratio
RAM	Random Access Memory
RGB	Red Green Blue
ROM	Read Only Memory
RS232	Recommended Standarts 232
SNR	Signal-to-Noise Ratio
SSIM	Structural SIMilarity
SZP	Sinusoidal Zone Plate
UART	Universal Asynchronous Receiver/Transmitter
VGA	Video Graphics Array
VHDL	Very-high-speed integrated circuits (VHSIC) Hardware Description Language

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

Sayfa

ÖZET

GERÇEK ZAMANLI BİR HÜCRESEL SİNİR AĞI YAPISININ TASARIMI VE BU YAPIYLA GABOR FİLTRELERİNİN FPGA ÜZERİNDE GERÇEKLENMESİ

Evren CESUR

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Doktora Tezi

Tez danışmanı: Prof. Dr. Vedat TAVŞANOĞLU

İki boyutlu Gabor filtreleri, frekans ve yön seçici özelliğiyle insan gözüne benzeyen uzaysal band geciren filtrelerdir. Diğer yandan, Gabor filtrelerinin tat, burun ve kulak gibi bir çok gerçeklemesi vardır. Gabor filtrelerinin yapay görme gerçeklemelerindeki en zor yönü yoğun işlem gerektiren yapısıdır.

İki boyutlu sürekli uzay Gabor fonksiyonu sinüzoidal dalganın Gauss fonksiyonuyla modüle edilmesinden elde edilir. Sayısal Gabor filtresi Gabor fonksiyonunun uzayda örneklenmesiyle elde edilir ve konvolüsyon şablonu olarak kullanılır. Buna rağmen, şablon boyutları bant genişliğiyle ters orantılı olduğundan dolayı düşük bant genişlikleri için şablon boyutları oldukça büyük olur. 1998'de FIR Gabor filtrelerinin özelliklerine benzeyen bir Gabor benzeri Hücresel Sinir Ağı HSA filtresi 3×3 HSA şablonları ile gerçeklemesi önerilmiştir.

Bu tezde, yüksek çözünürlüklü (1080×1920 @60Hz) basit taramalı (progressive) video işaretini işleyebilen bir ayrık zamanlı Gabor benzeri HSA filtresinin tasarımı ve gerçeklemesi yapılmıştır. Gerçeklemedeki hesaplama iş yükünün zamanda bölünmesini temel alan bir yöntem kullanılmştır. Gerçeklenen protipte, 124.4 Mpiksel/s işlem hızında test edilen 50 euler iterasyonu gerçeklenmiştir ve saniyedeki toplam işlem gücü 119 Giga işlemdir.

Bu tezde, sabit bir doğrultuda ve hız da hareket eden bir cismin algılanmasında kullanılan hız ayarlı filtrenin (HAF) gerçeklemesi de tartışılmıştır. Bu yapının standart HSA'dan farkı olan birbirini takip eden çerçeveler arasındaki ilişki hız seçiciliğini tanımlamaktadır. HAF'ın sayısal gerçeklemesi için dört iterasyonun yeterli olduğu bir benzetim yöntemi kullanılmıştır. Buna rağmen, varsayılan DVI video arayüzlerinin çerçeveler arasındaki

bazı problemlerinden dolayı HAF gerçeklemesi için uygun değillerdir. Sonuç olarak, bu yapı PC benzetimlerin de test edilmiştir ve bir prototip olarak gerçeklemesi yapılmamıştır.

Anahtar Kelimeler: Gabor filtreleri, hücresel sinir ağı, FPGA, gerçek zamanlı görüntü ve video işleme, yüksek çözünürlük

ABSTRACT

DESIGN OF A REAL–TIME CELLULAR NEURAL NETWORK STRUCTURE AND ITS IMPLEMENTATION ON AN FPGA DEVICE FOR THE REALIZATION OF GABOR FILTERS

Evren CESUR

Department of Elektronics and Communication Engineering

Ph.D. Thesis

Supervisor: Prof. Dr. Vedat TAVŞANOĞLU

Two–dimensional Gabor filters are frequency selective filters demonstrating similar features to human retina. On the other hand, there are many implementations of Gabor filters like artificial nose, ear and taste. The most challenging aspect of an artificial vision design is its computation intensive structure.

A continuous–space 2–D Gabor function is obtained by modulating a sinusoidal wave with a Gauss function. A digital Gabor filter is obtained by sampling the Gabor function in space and using it as a convolution kernel. However, the kernel should be considerably larger for narrow–band widths, as the size of the kernel and the bandwidth are inversely proportional. In 1998, a Cellular Neural Network (CNN) based Gabor–type filter having similar properties with FIR Gabor filters, which can be implemented with 3×3 CNN templates was proposed.

In this thesis, a discrete–time CNN based Gabor–type filter is designed and implemented which is capable of processing high resolution (1080×1920 @60Hz) progressive video images in real–time. A method of dividing the computation workload in time–domain is used for the implementation. In the implemented prototype, 50 Euler iterations is realized, which is tested to process 124.4 Mpix/s, and its total computation power is 119 Giga operations per second.

In this thesis the implementation of a velocity–tuned filter (VTF) is also discussed, which is used to select an object in a video image moving in a specific orientation and velocity. The difference of this structure from a standard CNN is that some relationships between consecutive frames are defined for velocity selectivity. A simulation method of VTF is

used, where it is shown that four iterations are sufficient for the digital implementation. However, default DVI video interfaces introduce some problems between frames, hence are not suitable for an implementation of VTF. Consequently, the structure is tested on a PC simulation and the actual implementation is not realized as a prototype.

Keywords: Gabor filters, cellular neural networks, FPGA, real–time image and video processing, high resolution

BÖLÜM 1

GİRİŞ

Bu tezde yüksek çözünürlüklü video işaretlerini gerçek zamanlı Gabor filtresiyle işleyebilen sayısal bir sistem önerilmiştir. Ayrıca önerilmiş olan sistemin donanımı tasarlanmış ve FPGA üzerinde gerçeklenmiştir. Bu sistemin tasarlanması sırasında yapılan çalışmalar, ilgili literatür özeti, tezin amacı ve tezin orijinal katkısı bu bölümde verilmiştir.

1.1 Literatür Özeti

Gabor fonksiyonu, ilk olarak Dennis Gabor [1] tarafından 1946 yılında yayınlanan makalesinde frekans ile zaman (ya da uzay) boyutları arasındaki belirsizliği tanımlamak için önerilmiş bir fonksiyondur. Sinüzoidal bir dalganın Gauss fonksiyonuyla modüle edilmesiyle elde edilen Gabor fonksiyonunun iki boyutlu gösterimi ilk kez Daugman [2] tarafından yapılmıştır. İmpuls yanıtları iki boyutlu (2-B) Gabor fonksiyonu olan filtreler, bilgisayarla görme ve görüntü işleme konularındaki uygulamalarda kullanılmaktadır. Bu uygulamalara literatürdeki doku tanıma ve sınıflandırma [3], el yazısındaki karakterleri tanıma [4], doku ayrımı [5], kenar belirleme [6], görüntü sıkıştırma [7], hareket kestirme [8], nesne tanıma [9] ve şekil tanıma [10] örnek olarak verilebilir. Gabor filtresinin yön seçici özelliği kullanılarak gerçekleştirilen görsel korteks [11], bu yöntemlerin bilgisayarla görme ve sayısal görüntü işlemede kullanımını arttırmıştır. Ayrıca, ilk defa Gabor filtrelerinin memelilerin görme sistemlerini modellemek için kullanılabileceğini de [11]'de önerilmiştir. Bir görüntüdeki hareket, zaman uzayında yer değişimi gösteren bir zaman vektörü olarak ele alınabildiğinden dolayı, üç boyutlu (3-B) Gabor filtreleri cortical hücrelerinin hız ve yön hassasiyetlerinin modellenmesinde kullanılmıştır.

Gabor filtrelerinin en büyük eksikliği hesaplama yüklerinin yüksek olmasıdır. Yoğun işlem gerektiren sayısal Gabor filtrelerinin yerine Shi B. tarafından 1998 yılında iki katmanlı analog Hücresel Sinir Ağı (HSA) yapısındaki Gabor benzeri HSA filtreleri önerilmiştir [12]. Hopfield sinir ağının özel bir kümesi olan HSA, Leon O. Chua ve Lin Yang tarafından, sinir sistemimizdeki nöronların çevrelerindeki nöronlara olan bağlantılarından esinlenerek 1988 yılında önerilmiş bir yapay sinir ağı mimarisidir [13]. Ancak Gabor filtresi olarak kullanılabilecek en büyük analog HSA devresi gerçeklemesi ancak 32×32 boyundaki görüntüleri işleyebildiğinden dolayı yüksek çözünürlüklü görüntülerin bu devreyle işlenmesi neredeyse imkansızdır.

Öte yandan, Gabor filtrelerinin sayısal olarak hesaplanması, günümüzün gelişmiş kişisel bilgisayarında (Personal Computer, PC) dahi yüksek çözünürlüklü hareketli görüntüler için gerçek zamanlı olarak olarak yapılamamakta, bu tür uygulamalar ancak büyük sunucu bilgisayarlar üzerinde çalışabilmektedir. Bu nedenle Gabor filtrelerini görüntülere gerçek zamanlı olarak uygulamak için özelleştirilmiş sayısal donanımların tasarlanması gerekmiştir. Literatürdeki yüksek performanslı bu tür donanımlarda iki temel tasarım yöntemi kullanılmıştır: Sonlu süreli impuls yanıtlı (Finite Impulse Response, FIR) [14, 15, 16, 17] ve HSA [18] tabanlı. Bunlardan [14] grafik işlemci birimi (graphical processing unit, GPU), [17] uygulamaya özgü tümleşik devre (application specific integrated circuits, ASIC) ve [15, 16, 18] ise sahada proglamlanan kapı dizileri (field–programmable gate array, FPGA) üzerinde gerçeklenmiştir.

Hız ayarlı filtreler [19] ise hem uzayda hem de zamanda hesaplama yapan uzay zamansal filtrelerdir. Yani hız ayarlı filtreler sadece uzaydaki komşu pikselleri değil, komşu video çerçeveleri de ilişkilendirir. Bu nedenle, daha önce gelen video çerçevesiyle o anda gelen video çerçevesinin aynı anda işlenebilmesi için önce gelen video çerçevesinin de saklanamış olması gerekir. Hız ayarlı filtreler birçok uygulamada kullanılabilir olsalar da literatürde konuyla ilgili günümüze kadar önerilmiş bir uygulaması bulunmamaktadır.

Bu tez, Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) tarafından desteklenen 108E023 numaralı proje kapsamında yürütülen dört doktora tezinden bir tanesidir. İlk doktora tezi [20], projenin temelini oluşturan gerçek zamanlı bir HSA gerçekleme-

sidir. Bu tez ve üçüncü tezde [21] ilk tezdeki eksiklikler giderilerek, tekrar ayarlanabilir ve programlanabilir yeni bir donanım tasarlanıp gerçeklenmiştir. Ayrıca bu iki tez çalışmasında gerçeklenen sistemlerin kontrol, bellek yönetimi ve programlama arayüzü gibi blokları ortaktır ve ilgili tezlerin yazarları tarafından ekip olarak tasarlanmıştır. Kısmen bu tez kapsamında tasarlanan bu yapı tamamen bu tez kapsamında değiştirilerek gerçek zamanlı bir Gabor benzeri HSA filtresinin FPGA gerçeklemesi yapılmıştır. Buna ek olarak hız ayarlı bir filtreyle ilgili yapılmış olan çalışmalar da bu tezin kapsamındadır. TÜBİTAK projesi kapsamında yürütülen dördüncü tez [22] ise iki ve çok katlı HSA'ların aynı donanım ile gerçeklenebilmesini konu almaktadır ve bu tezle doğrudan bir ilişkisi bulunmamaktadır.

1.2 Tezin Amacı

Gabor filtreleri çoğunluka görüntü işleme uygulamalarındaki özellik çıkarma gibi işlemlerde kullanılır. Yüksek işlem yükü nedeniyle yüksek çözünürlüklü hareketli giriş görüntülerinde tüm sistemin hızının düşmemesi için Gabor filtrelerinin donanım üzerinde gerçeklenmesi yoluna gidilmiştir. Tezin amaçlarından birincisi, $1080 \times 1920@60Hz$ (1080×1920 çözünürlük, 60 Hz çerçeve hızı) gibi yüksek çözünürlüklü video işaretini gerçek zamanlı olarak işleyebilen bir Gabor filtresi yapısının önerilmesi ve önerilen donanımın FPGA üzerinde gerçeklenmesidir.

Hız ayarlı filtrelerde ise önce gelen video çerçevesinin de saklanamış olması gerektiğinden dolayı, FPGA içindeki belleklerin yanı sıra dış bellek elemanlarının da kullanılması gereklidir. Buradaki amaç sadece uzayda değil, zamanda da işlem yapabilen ve yüksek çözünürlüklü video işaretlerini işleyebilen bir hız ayarlı filtrenin tasarlanması ve gerçeklenmesidir. Günümüze kadar gerçeklenmemiş bu tür bir yapının gerçeklenebilmesi ile uygulama alanlarının da gelişmesine katkı sağlayacağı düşünülmektedir.

1.3 Orjinal Katkı

Bu tez literatüre matematiksel, benzetim ve sayısal devre gerçeklemesi olmak üzere üç alanda orijinal katkı sağlamaktadır.

Tezin matematiksel katkıları: Gabor benzeri HSA filtresinin band genişliğiyle sayısal çözümünün yakınsaması arasındaki ilişkinin net bir şekilde ortaya konması ve Gabor benzeri HSA filtre katsayılarının sabit noktalı aritmetik için dinamik bölgelerinin tespit edilmesi olarak sıralanabilir.

Tezin benzetim açısından katkıları: FIR Gabor benzeri filtrelerin şablon boyutlarıyla bant genişliği arasındaki ilişkinin deneysel olarak gösterilmesi, Gabor benzeri HSA filtresinin band genişliğiyle sayısal çözümünün iterasyon sayısı arasındaki bir ilişkinin gösterilmesi ve Gabor benzeri HSA filtre katsayıları ile HAF katsayılarının sabit noktalı airtmetik için bit genişliklerinin belirlenmesi olarak verilebilir.

Tezin sayısal devre gerçeklemesi açısından katkıları ise: Basit taramalı video işaretinin karanlık ve görünür bölgelerinin belirlenmesi için yeni kontrol sinyallerinin önerilmesi, Gabor benzeri HSA filtresi işlemcisinde kullanılan satır tampon belleklerinin tasarlanması ve gerçeklenmesi, Gabor benzeri HSA filtresi işlemcisinin aritmetik işlem bloğunun tasarlanması ve gerçeklenmesi, kontrol sinyallerinin senkronizasyonu için kullanılan geciktirme bloğunun tasarlanması ve gerçeklenmesi, filtre katsayılarının çalışma anında değiştirilebilmesini sağlayan dış seri haberleşme bloğunun ve dış mesaj formatının tasarlanması ve gerçeklenmesi, histogram hesaplama ve karşıtlık ayarlama bloklarının tasarlanması ve gerçeklenmesi, hız ayarlı filtre bloğunun tasarlanması ve tüm sistemin bilgisayar üzerinde testlerinin yapılmasını sağlayan test bloğunun tasarlanması ve gerçeklemesidir.

BÖLÜM 2

GABOR VE GABOR BENZERİ FİLTRELER

Dennis Gabor, frekans ve uzay arasında, kuantum fiziğindeki Heinsenber'in konum ve momentum arasındaki belirsizlik prensibine benzeyen bir benzerlikten yararlanarak Gabor fonksiyonunu tanımlamıştır [1]. Gabor fonksiyonu üstel bir işaretin, Gauss fonksiyonu tarafından modüle edilmesi ile elde edilmiştir. Görüntü işlemede kullanılan iki boyutlu Gabor fonksiyonu Daugman tarafından [2]'de sunulmuştur. Bu bölümde, Gabor fonksiyonu[1], Hücresel Sinir Ağı (HSA) ile gerçeklenmesi [12] incelenmiştir. Ayrıca FIR Gabor filtrelerinin şablon boyutlarının, filtrenin bant genişliğiyle ilşkisini veren bir benzetim yapılmış ve sonuçları verilmiştir.

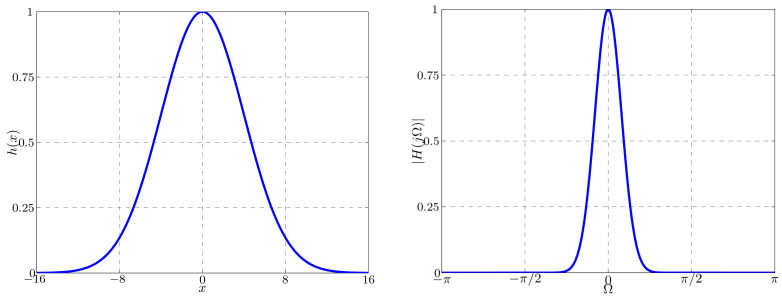
2.1 Gabor Fonksiyonları

Bir boyutlu sürekli Gauss fonksiyonu (2.1)'de verilmiştir.

$$h(x,\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.1}$$

Buradaki $x \in \mathbb{R}$, σ^2 fonksiyonun genişliğini belirleyen varyans değeri ve x eksenindeki merkez noktasını belirleyen μ değişkenleridir. Gauss fonksiyonu aynı zamanda Gauss dağılımının, olasılık yoğunluk fonksiyonu (Probability Density Function, PDF) olarak da geçmektedir. Bu dağılım ailesinin her bir üyesi sadece σ^2 ve μ parametreleri ile tam olarak tanımlanabilir.

Simetrik bir fonksiyon olan Gauss fonksiyonunun normalize edilmiş halinin x'e göre değişimi Şekil 2.1a'da verilmiştir. Fonksiyonun en büyük değeri $x = \mu$, genişliği ise σ^2 değerlerine bağlıdır. Her μ değeri için Gauss fonksiyon $x = \mu$ eksenine göre simetriktir. Gauss fonksiyonunu genişliğini σ^2 değeriyle doğru orantılıdır.



(a) (b)

Şekil 2.1 $\sigma = 4$ için bir boyutlu normalize Gauss fonksiyonunun: (a) x'e göre değişimi ve (b) Fourier dönüşümünün modülü.

Gauss fonksiyonu frekans düzleminde de yine Gauss fonksiyonudur. $\mu = 0$ için Gauss fonksiyonun Fourier dönüşümü (2.2)'de verilmiş ve genliği $|H(j\Omega)|$ Şekil 2.1b'de çizdirilmiştir.

$$H(j\Omega) = \mathscr{F}\{h(x;\mu,\sigma^2)\} = e^{-\frac{\Omega^2\sigma^2}{2}} \tag{2.2}$$

σ^2 değeri konumda Gauss fonksiyonun genişliği ile doğru orantılıyken, frekans düzleminde ters orantılıdır. Bundan dolayı konumda geniş/dar bir Gauss fonksiyonuna karşılık frekans düzleminde dar/geniş bir Gauss fonksiyonu denk gelmektedir.

Gabor fonksiyonu $g(x)$, bir alçak geçiren filtre olan Gauss filtresinin $h(x)$ impuls yanıtı olan Gauss fonksiyonunun frekansta Ω_0 (2.3) kadar ötelenmesi ile gerçeklenebilir.

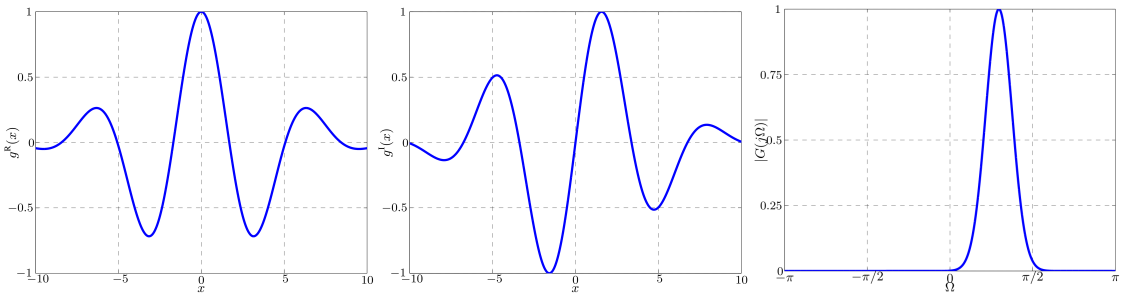
$$G(j\Omega) = \mathscr{F}\{g(x;\mu,\sigma^2)\} = e^{-\frac{\sigma^2(\Omega-\Omega_0)^2}{2}} \tag{2.3}$$

Bu öteleme işlemi ise Gauss fonksiyonu karmaşık üstel bir fonksiyon (2.4) ile çarpılmasına karşılık gelir [1]. Gauss filtresi bir alçak geçiren filtre olduğu için Gabor filtresi merkez frekansı olan bant geçiren, başka bir deyişle frekans secici bir filtredir.

$$g(x,\mu,\sigma^2) = \underbrace{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}}_{h(x)} e^{j\Omega_0 x} \tag{2.4}$$

Gabor fonksiyonunun x'e ve Fourier dönüşümünün genliğinin Ω'ya değişimi Şekil 2.2'de verilmiştir.

Bir boyutlu Gabor filtresine benzer biçimde, iki boyutlu Gabor fonksiyonu da, iki bo-



(a) (b) (c)

Şekil 2.2 $\sigma = 4$, $\Omega_0 = 0.93$ parametrelerine sahip bir boyutlu normalize Gabor fonksiyonunun: (a) gerçel parçası, (b) sanal parçası ve (c) Fourier dönüşümünün modülü.

yutlu Gauss fonksiyonunun $(\Omega_{x0}, \Omega_{y0})$ (2.6)frekansına ötelenmesi ile gerçeklenebilir[2]. Bu öteleme işlemi ise Gauss fonksiyonu karmaşık üstel bir fonksiyon ile çarpılmasına (2.5) karşılık gelir [1]. Gauss fonksiyonu bir alçak geçiren filtre olduğu için için Gabor fonksiyonu merkez frekansı $(\Omega_{x0}, \Omega_{y0})$ olan bant geçiren, başka bir deyişle frekans secici bir filtredir.

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} e^{j(\Omega_{x0}x + \Omega_{y0}y)} \tag{2.5}$$

Burada x ve y uzamsal değişken, Ω_{x0} ve Ω_{y0} açısal frekanslardır. İki boyutlu Gabor fonksiyonunun Fourier dönüşümü (2.6)'de verilmiştir.

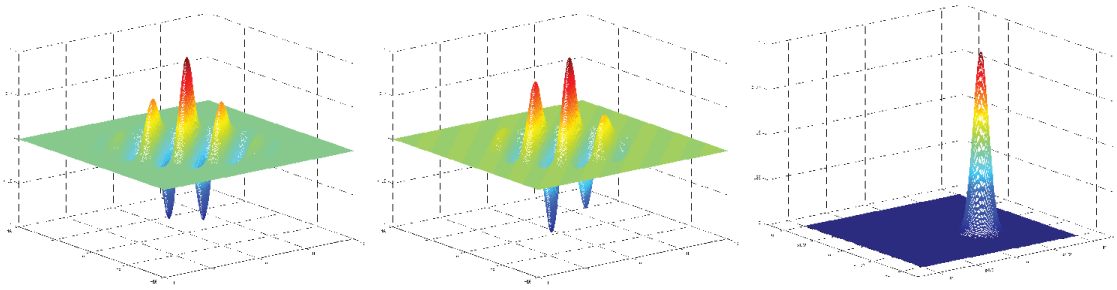
$$G(j\Omega_x, j\Omega_y) = \mathscr{F}\{g(x,y)\} = e^{-\frac{\sigma^2((\Omega_x-\Omega_{x0})^2+(\Omega_y-\Omega_{y0})^2)}{2}} \tag{2.6}$$

İki boyutlu Gabor fonksiyonunun uzamsal değişimi ve Fourier dönüşümü Şekil 2.3'de verilmiştir. Karmaşık değere sahip iki boyutlu Gabor fonksiyonu (2.5), iki parçaya ayrılabilir;

$$g^{\mathrm{R}}(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(\Omega_{x0}x + \Omega_{y0}y) \tag{2.7}$$

$$g^{\mathrm{I}}(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \sin(\Omega_{x0}x + \Omega_{y0}y) \tag{2.8}$$

R ve I indisleri sırasıyla Gabor fonksiyonun gerçel ve sanal parçası gösterirler.



(a) (b) (c)

Şekil 2.3 $\sigma = 4$, $\Omega_{x0} = 0.93$ ve $\Omega_{y0} = 0.93$ parametrelerine sahip iki boyutlu normalize Gabor fonksiyonunun: (a) gerçel parçası, (b) sanal parçası ve (c) Fourier dönüşümünün modülü.

2.2 İki boyutlu Gabor Filtreleri

İki boyutlu Gabor filtrelerinin uzaydaki yön ve frekans düzlemindeki frekans seçici özellikleri sayısal görüntü işleme uygulamaları olan kenar belirleme, doku tanıma, harf tanıma, görüntü sıkıştırma ve hareket kestirimi uygulamalarında kullanılmaktadır. İki boyutlu Gabor filtreleri frekans düzleminde belli bir frekans aralığını geçiren yani bant geçiren bir filtredir. Diğer bir deyişle belli bir yöndeki uzamsal dalgaları seçer. Sürekli uzayda tanımlanan iki boyutlu Gabor fonksiyonun sayısal görüntü işleme uygulamalarında kullanılabilmesi için örnekleme ve pencerele işlemlerinden geçirilmesi gerekir.

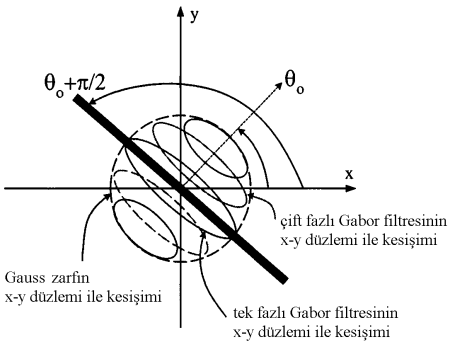
Denklem (2.5) ile verilen iki boyutlu Gabor fonksiyonu örneklendiği zaman, (2.9)'de verilen $(\omega_{x0}, \omega_{y0})$ merkez frekansına ve $2\sigma^{-1}$ bant genişliğine sahip ayrık uzay Gabor fonksiyonu elde edilir.

$$g(mX_s, nY_s) \triangleq g(m,n) = \frac{1}{2\pi\sigma^2} e^{-\frac{m^2+n^2}{2\sigma^2}} e^{j(m\omega_{x0}+n\omega_{y0})} \tag{2.9}$$

Burada $m = \frac{x}{X_s} \in \mathbb{N}$, $n = \frac{y}{Y_s} \in \mathbb{N}$, $\omega_{x0} = \Omega_{x0} X_s$ ve $\omega_{y0} = \Omega_{y0} Y_s$ dır. Denklem (2.9)'in Fourier dönüşümü alınırsa, denklem (2.10) elde edilir.

$$\mathscr{F}\{g(m,n)\} = G(e^{j\omega_x}, e^{j\omega_y}) = e^{-\frac{\sigma^2[(\omega_x-\omega_{x0})^2+(\omega_y-\omega_{y0})^2]}{2}} \tag{2.10}$$

İki boyutlu Gabor filtresi doğrultusu belli bir açı ile tanımlanmış kenarları seçmek için kullanılır [23]. Doğrultu açısı (2.11)'de tanımlanmıştır. Filtrenin seçtiği doğrultu Şekil 2.4'de



Şekil 2.4 Gabor filtresinin ayarlandığı doğrultu

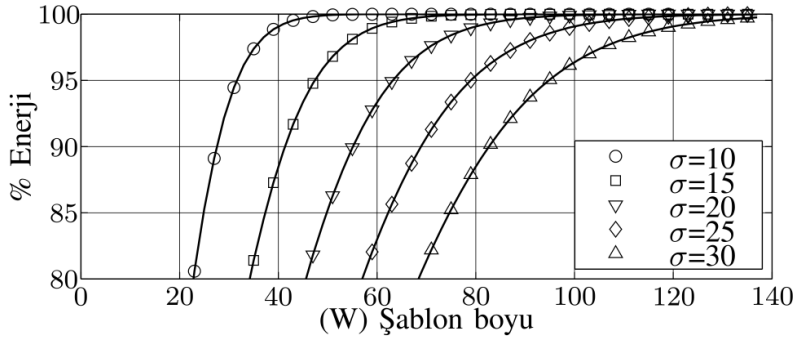
verilmiştir.

$$\theta_0 = \arctan(-\frac{\omega_{y0}}{\omega_{x0}}), \quad \theta = \theta_0 + \frac{\pi}{2} \tag{2.11}$$

Örnekleme işleminden sonra iki boyutlu sonsuz sayıda örnek elde edilir. Sonsuz sayıdaki örneğe sahip filtrenin görüntüye uygulanabilmesi için pencelereleme işlemiyle sonlu sayıdaki örnek elde edilmesi gerekir. Ayrıca Gabor fonksiyonunun bir filtre olarak kullanılabilmesi için üstel fonksiyonun zarfının içinde en az bir periyot sinüzoidal dalga olması gerekir. Filtrenin bant geniliği ise, Gauss fonksiyonun genişliğini belirleyen σ değişkeniyle ilişkilidir.

Sürekli uzay iki boyutlu Gabor fonksiyonu örneklenmesi ve pencerelenmesi sonucunda $W \times W, W \in \mathbb{N}$ boyutlarındaki Gabor filtresinin impuls yanıtı elde edilir. Gerçel değerli giriş görüntüsündeki her bir pikselin Gabor filtresiyle filtrelenmesi için W^2 karmaşık yada $2W^2$ gerçel değerli çarpma işlemi gerekir. Buradan anlaşılacağı gibi W değeri filtrenin hesap yükünü belirlemektedir. W ne kadar büyükse filtrenin çıkışının hesaplanması o kadar zor olacaktır. Bu nedenle W değerinin mümkün olduğu kadar küçük olması ve aynı zamanda pencereleme etkisinin en aza indirilebilmesi için , $W \geq \lceil 6\sigma + 1 \rceil$ koşulu ile seçilmesi gerektiği [24]'de rapor edilmiştir. Buna göre, büyük σ değerleri yani düşük bant genişlikleri için Gabor filtresinin şablon boyutu oldukça büyük olması gerekir, başka bir deyişle çarpma işlemi sayısı artar. Örneğin, $\sigma = 5$ için 31×31 boyutunda şablona ihtiyaç duyulur ve 1922 çarpma işlemi yapılması gerekir.

Şablon boyutlarının küçültmek için pencerelenmemiş Gabor fonksiyonu ile $W \times W$ boyutlarındaki şablona sahip Gabor filtresinin impuls yanıtının enerjilerini karşılaştıran bir



Şekil 2.5 Pencerelenmemiş iki boyutlu Gabor filtresi ile $W \times W$ boyutarındaki şablona sahip olan iki boyutlu FIR Gabor filtresinin impuls yanıtının enerjilerinin karşılaştırılması.

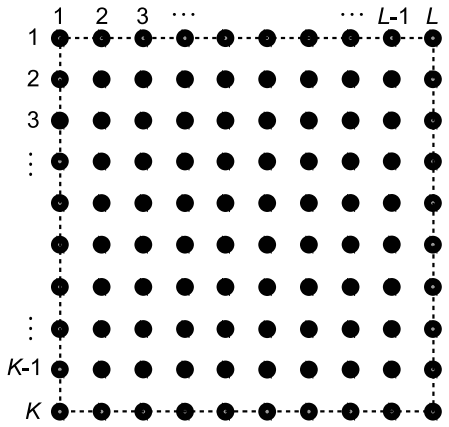
benzetim yapılmıştır. Şekil 2.5'deki benzetim sonuçlarına göre $W \geq \lceil 4\sigma + 1 \rceil = W_{min}$ koşulu için pencerelenmiş fonksiyon, orijinal fonksiyonun %99'dan daha fazla enerjisini taşıdığı görülmüştür. Buna göre şablon boyutu daha da küçültülebilir. Bulunan bu koşula göre yukarıdaki örnek tekrar hesaplanırsa, $\sigma = 5$ için şablon boyu 21×21 olur ve çıkışın hesaplanabilmesi için 882 çarpma işlemi yeterli olur.

2.3 Hücresel Sinir Ağları ve Gabor Benzeri HSA Filtreleri

Hopfield Sinir Ağı'nın özel bir kümesi olan Hücresel Sinir Ağları (HSA) Leon O. Chua ve Lin Yang sinir sistemimizdeki nöronların çevrelerindeki nöronlara olan bağlantılarına benzeyen yeni bir yapay sinir ağı mimarisini 1988 yılında ortaya koymuşlardır [13]. HSA'daki bölgesel bağlantıların sonucunda, ağın kapladığı alan azalmakta, ayrıca verilerin ağ üzerindeki taşınım yolları kısa olduğundan toplam güç harcaması düşük olmaktadır. HSA'nın bir diğer avantajı ise kararlı bir denge noktasına yakınsama süresinin ağın boyutundan bağımsız olmasıdır.

Görüntü işleme uygulamalarında HSA yapısındaki her bir hücre görüntünün bir pikseline karşı düşmektedir, dolayısıyla HSA'nın yapısal paralelliği işleme hızını oldukça arttırmaktadır. Ayrıca bu ağlar ayrık uzay, sürekli zaman, sürekli durum sistemleri olduklarından gerçek zamanlı görüntü işlemeye uygundur.

Bu alt bölümde, öncelikle doğrusal Hücresel Sinir Ağlarının matematiksel modeli, hücre durum denklemleri, komşuluk, sınır koşulları, uzayda değişmeme ve klonlanan şablonlar



Şekil 2.6 HSA'nın kartezyen koordinat sistemindeki görüntüsü

ve HSA ile ilgili son olarak tüm hücre durumlarının vektör matris formunda yazılması verilmiştir. Daha sonra ayrık zamanlı doğrusal HSA ve Gabor benzeri HSA filtrelerinin şablonları ve frekans yanıtı verilmiştir.

2.3.1 Doğrusal Hücresel Sinir Ağları

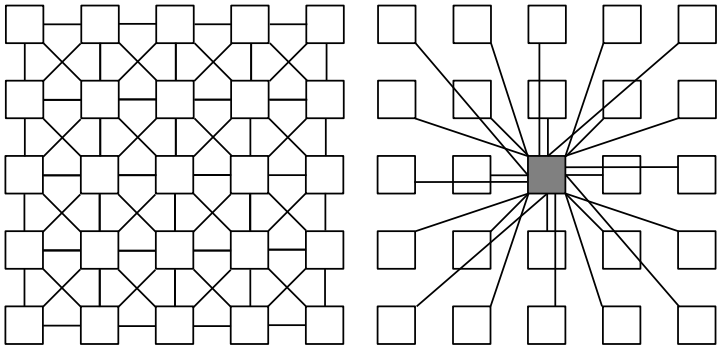
Hücresel Sinir Ağları (HSA) hücre adı verilen birbirine eşdeğer analog işlemci dizilerinden oluşmaktadır. Bu hücreler genellikle iki boyutlu olarak dizilirler. $K \times L$ boyutlarında Şekil 2.6'de verildiği gibi dizilen $C(i,j)$, $i \in \{1,2,...K\}$, $j \in \{1,2,...L\}$ hücreleri iki boyutlu HSA'yı oluşturur.

HSA'yı oluşturan $C(i,j)$ hücreleri sadece çevresindeki hücrelerle belli bir ağırlıkla bağlıdırlar. $C(i,j)$ hücresinin bağlı olduğu hücrelere komşu hücreleri denir ve bu komşuluk (2.12) denklemindeki gibi tanımlanır.

$$S_r(i,j) = \{C(k,l)|_{1\leq k\leq K, 1\leq l\leq L}, max(|k-i|,|l-j|) = r\} \tag{2.12}$$

$S_r(i,j)$ etki alanı olarak adlandırılır ve $C(i,j)$ hücresinin r komşuluğu içerisinde yer alan $C(k,l)$ hücrelerinin oluşturduğu kümeyi temsil eder. $C(i,j)$ hücresinin etki alanı içinde yer alan hücreler r komşuluk yarıçapıyla birbirine bağlanır. $r = 1(3\times 3)$ ve $r = 2(5\times 5)$ için $C(i,j)$ hücresinin $S_r(i,j)$ etki alanı Şekil 2.7'de gösterilmiştir.

$K \times L$ tane hücreden oluşan iki boyutlu doğrusal analog HSA yapısındaki hücrelere ait



(a) (b)

Şekil 2.7 (a) r=1 ve (b) r=2 için $C(i,j)$ hücresinin komşu hücrelerle olan bağlantıları.

durum denklemi (2.13)'de verilmiştir.

$$\frac{dx_{ij}(t)}{dt} = \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)x_{kl}(t) + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u_{kl} \tag{2.13}$$

Burada, $x_{ij}(t)$ $C(i,j)$ hücresinin durum değişkenini, u_{ij} hücrenin zamanla değişmeyen girişini, $A(i,j;k,l)$ ve $B(i,j;k,l)$ sırasıyla geri besleme ve ileri besleme (giriş) sinaptik ağırlıkları gösterir. $C(k,l)$ ise $C(i,j)$ hücresinin etki küresi içinde kalan komşu hücrelerini gösterir.

2.3.2 Konumdan Bağımsız HSA ve Şablonlar

Konumdan Bağımsız HSA (Space–invariant CNN) yapısında, $A(i,j;k,l)$, $B(i,j;k,l)$ sinaptik ağırlıkları konumdan bağımsızlardır ve genellikle küçük r değerleri için tanımlanır. Ağa ait herhangi bir hücre ve bağlı olduğu diğer hücrelerin bağlantı ağırlıklarını gösteren matrise klonlama şablon denir. Bu matrislerin tüm ağ üzerinde sabit olması yani aynı yapının tüm ağdaki hücreler için klonlanması anlamında kullanıldığı için klonlama şablonu ismi verilmiştir. Bir komşuluklu ($r = 1$) konumdan bağımsız bir doğrusal HSA yapısının durum denklemi (2.14)'de verilmiştir.

$$\begin{aligned}\frac{dx_{ij}(t)}{dt} &= \sum_{|k-i|\le 1}\sum_{|l-j|\le 1} A(k-i,l-j)x_{kl}(t) + \sum_{|k-i|\le 1}\sum_{|l-j|\le 1} B(k-i,l-j)u_{kl} \\ &= \sum_{k=-1}^{1}\sum_{l=-1}^{1} a_{k,l}x_{i+k,j+l}(t) + \sum_{k=-1}^{1}\sum_{l=-1}^{1} b_{k,l}u_{i+k,j+l}\end{aligned} \tag{2.14}$$

Br komşuluk için, şablonlar (3×3) boyutlarındaki matrisle ifade edilir ve (2.14) denkleminin daha basit hale getirilmesi için karşılıklı elemanları çarpılıp toplanması işlemini yapan ve (2.15)'de verilen şablon nokta çarpımı ($\circledast$) operatörü tanımlanmıştır..

$$\sum_{k=-1}^{1}\sum_{l=-1}^{1} a_{kl}x_{i+k,j+l}(t) \triangleq \begin{bmatrix} a_{-1\,-1} & a_{-10} & a_{-11} \\ a_{0\,-1} & a_{00} & a_{01} \\ a_{1\,-1} & a_{10} & a_{11} \end{bmatrix} \circledast \begin{bmatrix} x_{i-1,j-1}(t) & x_{i-1,j}(t) & x_{i-1,j+1}(t) \\ x_{i,j-1}(t) & x_{i,j}(t) & x_{i,j+1}(t) \\ x_{i+1,j-1}(t) & x_{i+1,j}(t) & x_{i+1,j+1}(t) \end{bmatrix} = \boldsymbol{A} \circledast \boldsymbol{X}_{ij}(t)$$

$$\sum_{k=-1}^{1}\sum_{l=-1}^{1} b_{kl}u_{i+k,j+l} \triangleq \begin{bmatrix} b_{-1\,-1} & b_{-10} & b_{-11} \\ b_{0\,-1} & b_{00} & b_{01} \\ b_{1\,-1} & b_{10} & b_{11} \end{bmatrix} \circledast \begin{bmatrix} u_{i-1,j-1} & u_{i-1,j} & u_{i-1,j+1} \\ u_{i,j-1} & u_{i,j} & u_{i,j+1} \\ u_{i+1,j-1} & u_{i+1,j} & u_{i+1,j+1} \end{bmatrix} = \boldsymbol{B} \circledast \boldsymbol{U}_{ij} \tag{2.15}$$

(2.14) denkleminin şablon nokta çarpımı operatörüyle yazılmış hali (2.16) denkleminde verilmiştir.

$$\dot{x}_{ij}(t) = \boldsymbol{A} \circledast \boldsymbol{X}_{ij}(t) + \boldsymbol{B} \circledast \boldsymbol{U}_{ij} \tag{2.16}$$

(2.16) denkleminde kullanılan ve (2.15)'de tanımlanan $\boldsymbol{A}$ geri besleme klonlanan şablonu, $\boldsymbol{B}$ ise ileri besleme (giriş) klonlanan şablonu olarak isimlendirilir. (2.16) denklemindeki $\boldsymbol{X}_{ij}(t)$ geri besleme klonlanan şablonuna karşı gelen durum değişkenlerini, $\boldsymbol{U}_{ij}$ ise giriş klonlanan şablonuna karşılık gelen giriş görüntüsündeki bölgesi gösterirler.Tezin bundan sonraki kısmında HSA kısaltması *konumdan bağımsız doğrusal HSA*, şablon ise *klonlanan şablon* anlamlarında kullanılmıştır.

2.3.3 HSA'nın Vektör ve Matris Formunda İfade Edilmesi

$K \times L$ Boyutundaki HSA'nın her $C(i, j)$ hücresine ait bir tane durum denklem bulunduğuna göre tüm HSA $(K \times L)$ tane durum denkleminden oluşur. Durum denklemlerinin matris halinde çözebilmek için (2.17) denklemi kullanılır. Bu nedenle giriş ve durumlar Şekil 2.8'de gösterilen paketleme yöntemiyle vektör haline getirilir.

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u} \tag{2.17}$$

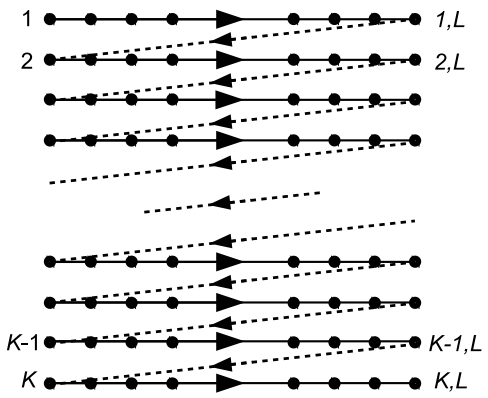
(2.17) denklemindeki $\mathbf{x}(t)$ $(KL \times 1)$ boyutundaki durum vektörü, $\mathbf{u}$ $(KL \times 1)$ boyutundaki giriş vektörü, $\mathbf{A}$ $(KL \times KL)$ boyutlarındaki geri besleme matrisi ve $\mathbf{B}$ ise $(KL \times KL)$ boyutlarındaki giriş matrisidir. Örnek olarak (3×3) boyutlarındaki bir HSA'na ait hücre durum denklemleri Şekil 2.8'deki gibi paketlenerek (2.17) denkleminde yerine konulduğunda (2.18) denklemi elde edilir.

$$
\underbrace{\begin{bmatrix} \dot{x}_{11}(t) \\ \dot{x}_{12}(t) \\ \dot{x}_{13}(t) \\ \dot{x}_{21}(t) \\ \dot{x}_{22}(t) \\ \dot{x}_{23}(t) \\ \dot{x}_{31}(t) \\ \dot{x}_{32}(t) \\ \dot{x}_{33}(t) \end{bmatrix}}_{\dot{\mathbf{x}}(t)} = \underbrace{\begin{bmatrix} a_{00} & a_{01} & 0 & a_{10} & a_{11} & 0 & 0 & 0 & 0 \\ a_{0-1} & a_{00} & a_{01} & a_{1-1} & a_{10} & a_{11} & 0 & 0 & 0 \\ 0 & a_{0-1} & a_{00} & 0 & a_{1-1} & a_{10} & 0 & 0 & 0 \\ a_{-10} & a_{-11} & 0 & a_{00} & a_{01} & 0 & a_{10} & a_{11} & 0 \\ a_{-11} & a_{-10} & a_{-11} & a_{0-1} & a_{00} & a_{01} & a_{1-1} & a_{10} & a_{11} \\ 0 & a_{-1-1} & a_{-10} & 0 & a_{0-1} & a_{00} & 0 & a_{1-1} & a_{10} \\ 0 & 0 & 0 & a_{-10} & a_{-11} & 0 & a_{00} & a_{01} & 0 \\ 0 & 0 & 0 & a_{-1-1} & a_{-10} & a_{-11} & a_{0-1} & a_{00} & a_{01} \\ 0 & 0 & 0 & 0 & a_{-1-1} & a_{-10} & 0 & a_{0-1} & a_{00} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x_{11}(t) \\ x_{12}(t) \\ x_{13}(t) \\ x_{21}(t) \\ x_{22}(t) \\ x_{23}(t) \\ x_{31}(t) \\ x_{32}(t) \\ x_{33}(t) \end{bmatrix}}_{\mathbf{x}(t)} \tag{2.18}
$$

$$
+ \underbrace{\begin{bmatrix} b_{00} & b_{01} & 0 & b_{10} & b_{11} & 0 & 0 & 0 & 0 \\ b_{0-1} & b_{00} & b_{01} & b_{1-1} & b_{10} & b_{11} & 0 & 0 & 0 \\ 0 & b_{0-1} & b_{00} & 0 & b_{1-1} & b_{10} & 0 & 0 & 0 \\ b_{-10} & b_{-11} & 0 & b_{00} & b_{01} & 0 & b_{10} & b_{11} & 0 \\ b_{-11} & b_{-10} & b_{-11} & b_{0-1} & b_{00} & b_{01} & b_{1-1} & b_{10} & b_{11} \\ 0 & b_{-1-1} & b_{-10} & 0 & b_{0-1} & b_{00} & 0 & b_{1-1} & b_{10} \\ 0 & 0 & 0 & b_{-10} & b_{-11} & 0 & b_{00} & b_{01} & 0 \\ 0 & 0 & 0 & b_{-1-1} & b_{-10} & b_{-11} & b_{0-1} & b_{00} & b_{01} \\ 0 & 0 & 0 & 0 & b_{-1-1} & b_{-10} & 0 & b_{0-1} & b_{00} \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} u_{11} \\ u_{12} \\ u_{13} \\ u_{21} \\ u_{22} \\ u_{23} \\ u_{31} \\ u_{32} \\ u_{33} \end{bmatrix}}_{\mathbf{u}}
$$

2.3.4 Ayrık Zamanlı HSA

Sürekli zaman, ayrık uzayda tanımlanan HSA yapısına ait çıkışların hesaplanabilmesi için (2.16) denklemindeki t sürekli zamanının, ayrık zamana dönüştürülmesi gerekir. Öncelikle (2.16) denklemindeki zamana bağlı terimler $t \triangleq nT_s$ şeklinde örneklenerek



Şekil 2.8 Satır Satır Paketleme

(2.19)'deki tanımlamalar elde edilir.

$$\dot{x}(t)\big]_{t\triangleq nT_s} = \dot{x}(nT_s) \triangleq \dot{x}(n)$$
$$x(t)\big]_{t\triangleq nT_s} = x(nT_s) \triangleq x(n) \tag{2.19}$$

(2.16) denklemindeki sürekli zamanda tanımlanan türev opretörü çeşitli yaklaşıklık yöntemleriyle ayrıklaştırılabilir. Basitlik, işlem azlığı ve sayısal sistemlerdeki nedensellik özelliği için (2.20) denkleminde tanımlanan Euler ileri yaklışıklığı tercih edilmiştir.

$$\dot{x}(t) \simeq \frac{x((n+1)T_s) - x(nT_s)}{T_s} \triangleq \frac{x(n+1) - x(n)}{T_s} \tag{2.20}$$

(2.16) denklemindeki türev yerine (2.20) denklemindeki yaklaşıklı ve (2.19)'deki tanımlamalar kullanılırsa (2.21) denklemi elde edilir.

$$x_{ij}(n+1) = x_{ij}(n) + T_s\left(\boldsymbol{A} \circledast \boldsymbol{X}_{ij}(n) + \boldsymbol{B} \circledast \boldsymbol{U}_{ij}\right) \tag{2.21}$$

(2.21) denkleminin basitleştirilmesi için (2.22)'deki tanımlamalar kullanılırsa (2.23) denklemi elde edilir.

$$\bar{a}_{kl} = \begin{cases} T_s a_{kl} & k,l \neq 0 \\ 1 + T_s a_{kl} & k,l = 0 \end{cases} \tag{2.22}$$
$$\bar{b}_{kl} = T_s b_{kl}$$

$$x_{ij}(n+1) = \bar{\boldsymbol{A}} \circledast \boldsymbol{X}_{ij}(n) + \bar{\boldsymbol{B}} \circledast \boldsymbol{U}_{ij} \tag{2.23}$$

(2.23) denklemi, (2.17) denklemindeki gibi vektör matris formunda yazılırsa (2.24) denklemi elde edilir.

$$\mathbf{x}(n+1) = \bar{\mathbf{A}}\mathbf{x}(n) + \bar{\mathbf{B}}\mathbf{u} \tag{2.24}$$

HSA'nın sayısal çözümü için **u**'a giriş görüntüsü ve durum değişkenlerinin ilk koşullarıyla $\mathbf{x}(0)$ (2.17) dekleminden $\mathbf{x}(1)$ hesaplanır. İlk koşulların değerleri giriş görüntüsünün kendisi yada sabit bir değer olabilir.

$$\begin{aligned} &1.iterasyon && \mathbf{x}(1) = \bar{\mathbf{A}}\mathbf{x}(0) + \bar{\mathbf{B}}\mathbf{u} \\ &2.iterasyon && \mathbf{x}(2) = \bar{\mathbf{A}}\mathbf{x}(1) + \bar{\mathbf{B}}\mathbf{u} \\ & && \vdots \\ &N.iterasyon && \mathbf{x}(N) = \bar{\mathbf{A}}\mathbf{x}(N-1) + \bar{\mathbf{B}}\mathbf{u} \end{aligned} \tag{2.25}$$

HSA kararlı ise durumlar değişkenlerinin değerleri belli değerlere yakınsar. Yani (2.25) deki iterasyonlar sonucunda $\mathbf{x}(n)$'nin değişimi durduğunda HSA'nın çıkışı bulunur.Durumların değişmesi bittiğinde yada değişim miktarı belli bir değerin altında olduğunda yineleme işlemine son verilir ve HSA'nın durumlarına ait çıkış değerleri elde edilir.

2.3.5 Gabor Benzeri HSA Filtreleri

Shi, [12] yayınında Gabor benzeri HSA filtresinin geri besleme ve giriş şablonlarını (2.26)'deki şekilde tanımlamıştır.

$$\boldsymbol{A} = \begin{bmatrix} 0 & e^{-j\omega_{y0}} & 0 \\ e^{j\omega_{x0}} & (4+\lambda^2) & e^{-j\omega_{x0}} \\ 0 & e^{j\omega_{y0}} & 0 \end{bmatrix} \boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \lambda^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.26}$$

Burada λ^2 bant genişliği ile ilgili parametre, ω_{x0} ve ω_{y0} ise iki boyutlu frekans düzlemindeki öteleme miktarlarıdır. λ ile FIR Gabor filtresinin band genişliğiyle ilgili parametresi olan σ arasında [12]'de verilen $\lambda = 0.96\sigma^{-1}$ ilişkisi vardır. $(\omega_{x0}, \omega_{y0})$ merkez frekansına ayarlı Gabor benzeri HSA filtresinin orijin noktasına olan uzaklık $\omega_0 = \sqrt{\omega_{x0}^2 + \omega_{y0}^2}$, açı ise $\theta_0 = \arctan(\omega_{y0}/\omega_{x0})$ şeklinde hesaplanır.

Filtrenin ismindeki *benzeri* kelimesi, Gabor fonksiyonun birebir aynı özelliklere sahip olmadığını fakat çok yakın özellikler gösterdiğini ima etmek için kullanılmaktadır.

Bundan sonraki kısım da Gabor benzeri HSA filtresinin transfer fonksiyonu (frekans yanıtı) hesaplanacaktır. Girişinin sabit ve sistemin kararlı olması halinde, tüm durumlar sabit değerlre gitme eğiliminde olurlar. Bunun sonucu olarak $t \to \infty$ için durumların zamana göre türevi (2.27) denkleminde gösterildiği gibi sıfıra gider.

$$\lim_{t\to\infty} \frac{dx_{ij}(t)}{dt} = 0 \tag{2.27}$$

(2.16) denklemi, (2.27) için (2.28) denklemine olur.

$$0 = \boldsymbol{A} \circledast \boldsymbol{X}_{ij}(t) + \boldsymbol{B} \circledast \boldsymbol{U}_{ij} \tag{2.28}$$

(2.28) denkleminde (2.26) şablonları yerlerine konulup, şablon nokta çarpımları yapılırsa (2.29) denklemi elde edilir.

$$\lambda^2 u_{i,j} + e^{-j\omega_{y0}} x_{i-1,j}(t) + e^{j\omega_{x0}} x_{i,j-1}(t) - (4+\lambda^2) x_{i,j}(t) + e^{-j\omega_{x0}} x_{i,j+1}(t) + e^{j\omega_{y0}} x_{i+1,j}(t) = 0 \tag{2.29}$$

elde edilir. Bu denklemin iki boyutlu Z dönüşümü alındığında (2.30) denklemiyle verilen Gabor benzer HSA filtresinin transfer fonksiyonu elde edilir.

$$H(Z_x, Z_y) = \frac{\lambda^2}{4+\lambda^2 - e^{-j\omega_{y0}} Z_x^{-1} - e^{j\omega_{x0}} Z_y^{-1} - e^{-j\omega_{x0}} Z_y - e^{j\omega_{y0}} Z_x} \tag{2.30}$$

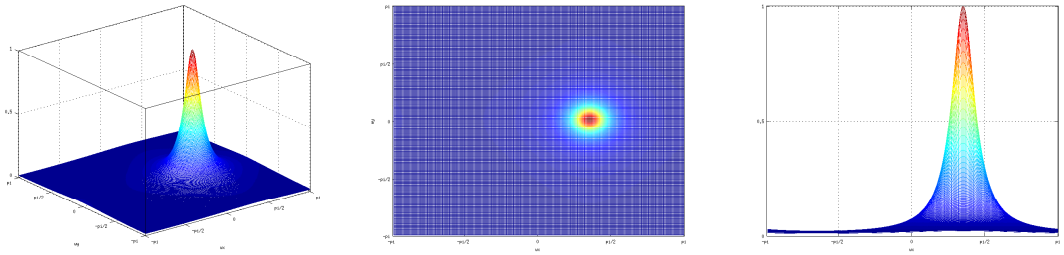
(2.30) denkleminde (Z_x, Z_y) yerine $(e^{j\omega_x}, e^{j\omega_y})$ konularak (2.31) denklemindeki Gabor benzeri HSA filtresinin frekans yanıtı elde edilir.

$$H(e^{j\omega_x}, e^{j\omega_y}) = \frac{\lambda^2}{4+\lambda^2 - 2\cos(\omega_x - \omega_{x0}) - 2\cos(\omega_y - \omega_{y0})} \tag{2.31}$$

$\cos(\omega_x - \omega_{x0})$, $\cos(\omega_y - \omega_{y0})$ terimleri Taylor serisine açılarak ilk iki terimi

$$\cos(\omega_x - \omega_{x0}) = 1 - \frac{(\omega_x - \omega_{x0})^2}{2!}, \quad \cos(\omega_y - \omega_{y0}) = 1 - \frac{(\omega_y - \omega_{y0})^2}{2!}$$

alınırsa, (2.31) denkleminde verilen Gabor benzeri HSA filtresinin frekans yanıtı, yaklaşık



(a) (b) (c)

Şekil 2.9 $\omega_0 = \pi/3$, $\theta_0 = 0$, $\lambda = 0.3$ parametrelerine sahip Gabor benzeri HSA filtresinin frekans yanıtının (a) modülü, (b) üstten görünüşü ve (c) yandan görünüşü.

olarak (2.32) denklemindeki gibi olur.

$$H(e^{j\omega_x}, e^{j\omega_y}) \simeq \frac{\lambda^2}{\lambda^2 - (\omega_x - \omega_{x0})^2 - (\omega_y - \omega_{y0})^2} \tag{2.32}$$

Gabor benzeri HSA filtresinin geçirme bandının tanımı için, bir boyutlu filtrelerde kullanılan 3 dB kesim frekansları yerine, iki boyutlu filtrelerde 6 dB kesim frekanslarının bulunması gerekir. Bunu için frekans yanıtının modülü (2.33) denklemindeki gibi $1/2$'ye eşitlenir.

$$|H(e^{j\omega_x}, e^{j\omega_y})| = \frac{\lambda^2}{\lambda^2 - (\omega_x - \omega_{x0})^2 - (\omega_y - \omega_{y0})^2} = \frac{1}{2} \tag{2.33}$$

(2.33) denkleminde gerekli işlemler yapılırsa (2.34) ile verilen çember denklemi elde edilir.

$$(\omega_x - \omega_{x0})^2 - (\omega_y - \omega_{y0})^2 = \lambda^2 \tag{2.34}$$

(2.34)'deki çember denklemi, Gabor benzeri HSA filtresinin $(\omega_{x0}, \omega_{y0})$ merkez frekansı ve λ yarı bant genişliğine sahip dairesel bir frekans yanıtına sahip olduğunu gösterir. Gabor benzeri HSA filtresinin frekans yanıtı bu $(\omega_0, \theta_0, \lambda)$, $(\pi/3, 0, .3)$ parametreleri için Gabor benzeri HSA filtresinin frekans yanıtının modülü Şekil 2.9'de verilmiştir.

2.4 Sonuç

Bu bölümde, ilk olarak bir ve iki boyutlu Gabor fonksiyonlarının Gauss fonksiyonlarından elde edilmesi ve frekans yanıtları verilmiştir. Sürekli uzayda tanımlanan iki boyutlu

Gabor fonksiyonlarının, ayrık uzaydaki görüntülere uygulanabilmesi için ilk olarak örneklenmesi ve sonrada sonlu sayıda katsayı ile ifade edilebilmesi için pencerelenmesi gerekir. Bu yöntemle elde edilen filtrenin boyutlarıyla, filtrenin bant genişliği arasındaki ilişkiyi veren bir benzetim yapılmıştır ve bu benzetim sonuçlarına göre $W \times W, W \in \mathbb{N}$ boyutlarındaki filtre ve $W \geq \lceil 4\sigma + 1 \rceil = W_{min}$ koşulu için Gabor filtresi, Gabor fonksiyonun %99 enerjisini içermektedir.

Son olarak, hücresel sinir ağlarıyla gerçeklenen Gabor filtresinin özellikleri verilmiştir.

BÖLÜM 3

GABOR BENZERİ HSA FİLTRELERİNİN DEVRE GERÇEKLEMELERİ

Bu bölümde, ilk olarak Gabor benzeri HSA filtresinin analog devre gerçeklemesi [12] incelenmiştir, daha sonra ayrık zamanlı Gabor benzeri HSA filtresinin sayısal devre gerçeklemesi için yapılan çalışmalar [25] ve kanıtı yapılan iki tane teorem verilmiştir [26].

3.1 Gabor Benzeri HSA Filtresinin Analog Devre Gerçeklemesi

(2.26)'deki şablonlar, karmaşık değerli hücre durum değişkenine sahip (2.16)'deki denklemde yerine konulup gerekli düzenlemeler yapıldığında (3.1) elde edilir.

$$\dot{x}^{\mathrm{R}}_{ij}(t) + j\dot{x}^{\mathrm{I}}_{ij}(t) = \boldsymbol{A}^{\mathrm{R}} \circledast \boldsymbol{X}^{\mathrm{R}}_{ij}(t) - \boldsymbol{A}^{\mathrm{I}} \circledast \boldsymbol{X}^{\mathrm{I}}_{ij}(t) + \boldsymbol{B} \circledast \boldsymbol{U}_{ij} + j\{\boldsymbol{A}^{\mathrm{R}} \circledast \boldsymbol{X}^{\mathrm{I}}_{ij}(t) + \boldsymbol{A}^{\mathrm{I}} \circledast \boldsymbol{X}^{\mathrm{R}}_{ij}(t)\} \tag{3.1}$$

(3.1)'deki denklem gerçel ve sanal kısımlarına ayrılırsa, (3.2)'deki denklem takımı elde edilir.

$$\dot{x}^{\mathrm{R}}_{ij}(t) = \boldsymbol{A}^{\mathrm{R}} \circledast \boldsymbol{X}^{\mathrm{R}}_{ij}(t) - \boldsymbol{A}^{\mathrm{I}} \circledast \boldsymbol{X}^{\mathrm{I}}_{ij}(t) + \boldsymbol{B} \circledast \boldsymbol{U}_{ij} \tag{3.2a}$$

$$\dot{x}^{\mathrm{I}}_{ij}(t) = \boldsymbol{A}^{\mathrm{R}} \circledast \boldsymbol{X}^{\mathrm{I}}_{ij}(t) + \boldsymbol{A}^{\mathrm{I}} \circledast \boldsymbol{X}^{\mathrm{R}}_{ij}(t) \tag{3.2b}$$

(3.2)'deki $\boldsymbol{A}^{\mathrm{R}}$ ve $\boldsymbol{A}^{\mathrm{I}}$ matrisleri, Gabor benzeri HSA filtresinin geri besleme şablonu $\boldsymbol{A}$'nın sırasıyla gerçel ve sanal kısımlarıdır. Ve $\boldsymbol{A}^{\mathrm{R}}, \boldsymbol{A}^{\mathrm{I}}$ matrislerinin değerleri (3.3)'de verilmiştir.

(3.3), (3.2)'de yerine konursa (3.4) elde edilir.

$$\boldsymbol{A}^{\mathrm{R}} = \begin{bmatrix} 0 & \cos\omega_{y0} & 0 \\ \cos\omega_{x0} & -(4+\lambda^2) & \cos\omega_{x0} \\ 0 & \cos\omega_{y0} & 0 \end{bmatrix}, \ \boldsymbol{A}^{\mathrm{I}} = \begin{bmatrix} 0 & -\sin\omega_{y0} & 0 \\ \sin\omega_{x0} & 0 & -\sin\omega_{x0} \\ 0 & \sin\omega_{y0} & 0 \end{bmatrix} \tag{3.3}$$

$$\begin{aligned} \dot{x}^{\mathrm{R}}_{ij}(t) = & \begin{bmatrix} 0 & \cos\omega_{y0} & 0 \\ \cos\omega_{x0} & -(4+\lambda^2) & \cos\omega_{x0} \\ 0 & \cos\omega_{y0} & 0 \end{bmatrix} \circledast \begin{bmatrix} x^{\mathrm{R}}_{i-1,j-1}(t) & x^{\mathrm{R}}_{i-1,j}(t) & x^{\mathrm{R}}_{i-1,j+1}(t) \\ x^{\mathrm{R}}_{i,j-1}(t) & x^{\mathrm{R}}_{i,j}(t) & x^{\mathrm{R}}_{i,j+1}(t) \\ x^{\mathrm{R}}_{i+1,j-1}(t) & x^{\mathrm{R}}_{i+1,j}(t) & x^{\mathrm{R}}_{i+1,j+1}(t) \end{bmatrix} \\ & - \begin{bmatrix} 0 & -\sin\omega_{y0} & 0 \\ \sin\omega_{x0} & 0 & -\sin\omega_{x0} \\ 0 & \sin\omega_{y0} & 0 \end{bmatrix} \circledast \begin{bmatrix} x^{\mathrm{I}}_{i-1,j-1}(t) & x^{\mathrm{I}}_{i-1,j}(t) & x^{\mathrm{I}}_{i-1,j+1}(t) \\ x^{\mathrm{I}}_{i,j-1}(t) & x^{\mathrm{I}}_{i,j}(t) & x^{\mathrm{I}}_{i,j+1}(t) \\ x^{\mathrm{I}}_{i+1,j-1}(t) & x^{\mathrm{I}}_{i+1,j}(t) & x^{\mathrm{I}}_{i+1,j+1}(t) \end{bmatrix} + \lambda^2 u_{i,j} \end{aligned} \tag{3.4a}$$

$$\begin{aligned} \dot{x}^{\mathrm{I}}_{ij}(t) = & \begin{bmatrix} 0 & \cos\omega_{y0} & 0 \\ \cos\omega_{x0} & -(4+\lambda^2) & \cos\omega_{x0} \\ 0 & \cos\omega_{y0} & 0 \end{bmatrix} \circledast \begin{bmatrix} x^{\mathrm{I}}_{i-1,j-1}(t) & x^{\mathrm{I}}_{i-1,j}(t) & x^{\mathrm{I}}_{i-1,j+1}(t) \\ x^{\mathrm{I}}_{i,j-1}(t) & x^{\mathrm{I}}_{i,j}(t) & x^{\mathrm{I}}_{i,j+1}(t) \\ x^{\mathrm{I}}_{i+1,j-1}(t) & x^{\mathrm{I}}_{i+1,j}(t) & x^{\mathrm{I}}_{i+1,j+1}(t) \end{bmatrix} \\ & + \begin{bmatrix} 0 & -\sin\omega_{y0} & 0 \\ \sin\omega_{x0} & 0 & -\sin\omega_{x0} \\ 0 & \sin\omega_{y0} & 0 \end{bmatrix} \circledast \begin{bmatrix} x^{\mathrm{R}}_{i-1,j-1}(t) & x^{\mathrm{R}}_{i-1,j}(t) & x^{\mathrm{R}}_{i-1,j+1}(t) \\ x^{\mathrm{R}}_{i,j-1}(t) & x^{\mathrm{R}}_{i,j}(t) & x^{\mathrm{R}}_{i,j+1}(t) \\ x^{\mathrm{R}}_{i+1,j-1}(t) & x^{\mathrm{R}}_{i+1,j}(t) & x^{\mathrm{R}}_{i+1,j+1}(t) \end{bmatrix} \end{aligned} \tag{3.4b}$$

(3.4)'deki şablon nokta çarpım işlemleri sonucunda (3.5) elde edilir.

$$\begin{aligned} \dot{x}^{\mathrm{R}}_{ij}(t) = & \cos\omega_{x0}\ x^{\mathrm{R}}_{i,j-1}(t) + \cos\omega_{x0}\ x^{\mathrm{R}}_{i,j+1}(t) + \cos\omega_{y0}\ x^{\mathrm{R}}_{i-1,j}(t) + \cos\omega_{y0}\ x^{\mathrm{R}}_{i+1,j}(t) \\ & - \sin\omega_{x0}\ x^{\mathrm{I}}_{i,j-1}(t) + \sin\omega_{x0}\ x^{\mathrm{I}}_{i,j+1}(t) + \sin\omega_{y0}\ x^{\mathrm{I}}_{i-1,j}(t) - \sin\omega_{y0}\ x^{\mathrm{I}}_{i+1,j}(t) \\ & - (4+\lambda^2)\ x^{\mathrm{R}}_{i,j}(t) + \lambda^2\ u_{i,j} \end{aligned} \tag{3.5a}$$

$$\begin{aligned} \dot{x}^{\mathrm{I}}_{ij}(t) = & \sin\omega_{x0}\ x^{\mathrm{R}}_{i,j-1}(t) - \sin\omega_{x0}\ x^{\mathrm{R}}_{i,j+1}(t) - \sin\omega_{y0}\ x^{\mathrm{R}}_{i-1,j}(t) + \sin\omega_{y0}\ x^{\mathrm{R}}_{i+1,j}(t) \\ & + \cos\omega_{x0}\ x^{\mathrm{I}}_{i,j-1}(t) + \cos\omega_{x0}\ x^{\mathrm{I}}_{i,j+1}(t) + \cos\omega_{y0}\ x^{\mathrm{I}}_{i-1,j}(t) + \cos\omega_{y0}\ x^{\mathrm{I}}_{i+1,j}(t) \\ & - (4+\lambda^2)\ x^{\mathrm{I}}_{i,j}(t) \end{aligned} \tag{3.5b}$$

(3.6) incelendiğinde, $\cos\omega_{x0}[x^{\mathrm{R}}_{i,j-1}(t) - x^{\mathrm{R}}_{i,j}(t)]$ gibi pozitif katsayılı olan denklem parçaları iki düğüm arasında iletkenliği $\cos\omega_{x0}$ olan bir direnç elemanıyla, $-\sin\omega_{x0}[x^{\mathrm{I}}_{i,j-1}(t) - x^{\mathrm{R}}_{i,j}(t)]$ gibi negatif katsayılı olan denklem parçaları iki düğüm arasına değeri $-\sin\omega_{x0}$

olacak bir OTA (Operational Transconductanse Amplifier) elemanıyla, $\lambda^2 u_{i,j}$ denklem parçası $x^{\mathrm{R}}_{i,j}$ düğümüne bağlanan bir akım kaynağı elemanıyla, $-(4+\lambda^2+2\cos\omega_{x0}+\cos\omega_{y0}+\sin\omega_{x0}+\sin\omega_{y0})x^{\mathrm{I}}_{i,j}(t)$ denklem parçası $x^{\mathrm{R}}_{i,j}$ düğümüyle toprak arasına bağlı iletkenliği $(4+\lambda^2+2\cos\omega_{x0}+\cos\omega_{y0}+\sin\omega_{x0}+\sin\omega_{y0})$ olan bir direç elemanıyla ve $\dot{x}^{\mathrm{R}}_{ij}(t)$ ise toprakla $x^{\mathrm{R}}_{i,j}$ düğümü arasına konacak bir kapasite elemanıyla analog olarak gerçeklenebilir.

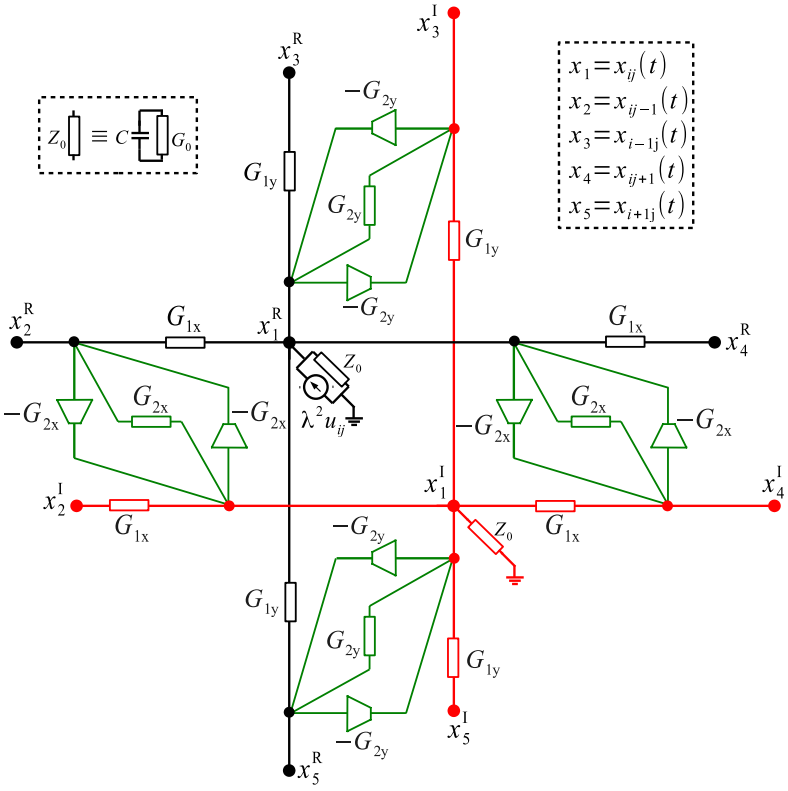
$$\begin{aligned}\dot{x}^{\mathrm{R}}_{ij}(t) &= \cos\omega_{x0}[x^{\mathrm{R}}_{i,j-1}(t)-x^{\mathrm{R}}_{i,j}(t)]+\cos\omega_{x0}[x^{\mathrm{R}}_{i,j+1}(t)-x^{\mathrm{R}}_{i,j}(t)]+\cos\omega_{y0}[x^{\mathrm{R}}_{i-1,j}(t)-x^{\mathrm{R}}_{i,j}(t)]\\ &+\cos\omega_{y0}[x^{\mathrm{R}}_{i+1,j}(t)-x^{\mathrm{R}}_{i,j}(t)]-\sin\omega_{x0}[x^{\mathrm{I}}_{i,j-1}(t)-x^{\mathrm{R}}_{i,j}(t)]+\sin\omega_{x0}[x^{\mathrm{I}}_{i,j+1}(t)-x^{\mathrm{R}}_{i,j}(t)]\\ &+\sin\omega_{y0}[x^{\mathrm{I}}_{i-1,j}(t)-x^{\mathrm{R}}_{i,j}(t)]-\sin\omega_{y0}[x^{\mathrm{I}}_{i+1,j}(t)-x^{\mathrm{R}}_{i,j}(t)]+\lambda^2 u_{i,j}\\ &-(4+\lambda^2+2\cos\omega_{x0}+\cos\omega_{y0}+\sin\omega_{x0}+\sin\omega_{y0})x^{\mathrm{R}}_{i,j}(t) \qquad (3.6a)\\ \dot{x}^{\mathrm{I}}_{ij}(t) &= \sin\omega_{x0}[x^{\mathrm{R}}_{i,j-1}(t)-x^{\mathrm{I}}_{i,j}(t)]-\sin\omega_{x0}[x^{\mathrm{R}}_{i,j+1}(t)-x^{\mathrm{I}}_{i,j}(t)]-\sin\omega_{y0}[x^{\mathrm{R}}_{i-1,j}(t)-x^{\mathrm{I}}_{i,j}(t)]\\ &+\sin\omega_{y0}[x^{\mathrm{R}}_{i+1,j}(t)-x^{\mathrm{I}}_{i,j}(t)]+\cos\omega_{x0}[x^{\mathrm{I}}_{i,j-1}(t)-x^{\mathrm{I}}_{i,j}(t)]+\cos\omega_{x0}[x^{\mathrm{I}}_{i,j+1}(t)-x^{\mathrm{I}}_{i,j}(t)]\\ &+\cos\omega_{y0}[x^{\mathrm{I}}_{i-1,j}(t)-x^{\mathrm{I}}_{i,j}(t)]+\cos\omega_{y0}[x^{\mathrm{I}}_{i+1,j}(t)-x^{\mathrm{I}}_{i,j}(t)]\\ &-(4+\lambda^2+2\cos\omega_{x0}+\cos\omega_{y0}+\sin\omega_{x0}+\sin\omega_{y0})x^{\mathrm{I}}_{i,j}(t) \qquad (3.6b)\end{aligned}$$

$$\begin{aligned}&G_{1x}=\cos\omega_{x0},\ G_{1y}=\cos\omega_{y0},\ G_{2x}=\sin\omega_{x0},\ G_{2y}=\sin\omega_{y0},\ b=\lambda^2,\ C=1\\ &G_0=4+\lambda^2+2\cos\omega_{x0}+2\cos\omega_{y0}+2\sin\omega_{x0}+2\sin\omega_{y0}\end{aligned} \qquad (3.7)$$

(3.6)'de (3.7) değerleri yerlerine konursa (3.8) elde edilir.

$$\begin{aligned}C\dot{x}^{\mathrm{R}}_{ij}(t) &= G_{1x}[x^{\mathrm{R}}_{i,j-1}(t)-x^{\mathrm{R}}_{i,j}(t)]+G_{1x}[x^{\mathrm{R}}_{i,j+1}(t)-x^{\mathrm{R}}_{i,j}(t)]+G_{1y}[x^{\mathrm{R}}_{i-1,j}(t)-x^{\mathrm{R}}_{i,j}(t)]\\ &+G_{1y}[x^{\mathrm{R}}_{i+1,j}(t)-x^{\mathrm{R}}_{i,j}(t)]-G_{2x}[x^{\mathrm{I}}_{i,j-1}(t)-x^{\mathrm{R}}_{i,j}(t)]+G_{2x}[x^{\mathrm{I}}_{i,j+1}(t)-x^{\mathrm{R}}_{i,j}(t)]\\ &+G_{2y}[x^{\mathrm{I}}_{i-1,j}(t)-x^{\mathrm{R}}_{i,j}(t)]-G_{2y}[x^{\mathrm{I}}_{i+1,j}(t)-x^{\mathrm{R}}_{i,j}(t)]+-G_0 x^{\mathrm{R}}_{i,j}(t)\lambda^2 u_{i,j} \qquad (3.8a)\\ C\dot{x}^{\mathrm{I}}_{ij}(t) &= G_{2x}[x^{\mathrm{R}}_{i,j-1}(t)-x^{\mathrm{I}}_{i,j}(t)]-G_{2x}[x^{\mathrm{R}}_{i,j+1}(t)-x^{\mathrm{I}}_{i,j}(t)]-G_{2y}[x^{\mathrm{R}}_{i-1,j}(t)-x^{\mathrm{I}}_{i,j}(t)]\\ &+G_{2y}[x^{\mathrm{R}}_{i+1,j}(t)-x^{\mathrm{I}}_{i,j}(t)]+G_{1x}[x^{\mathrm{I}}_{i,j-1}(t)-x^{\mathrm{I}}_{i,j}(t)]+G_{1x}[x^{\mathrm{I}}_{i,j+1}(t)-x^{\mathrm{I}}_{i,j}(t)]\\ &+G_{1y}[x^{\mathrm{I}}_{i-1,j}(t)-x^{\mathrm{I}}_{i,j}(t)]+G_{1y}[x^{\mathrm{I}}_{i+1,j}(t)-x^{\mathrm{I}}_{i,j}(t)]-G_0 x^{\mathrm{I}}_{i,j}(t) \qquad (3.8b)\end{aligned}$$

(3.8)'deki denklemler Şekil 3.1'deki [12] iki katmanlı analog devre ile gerçeklenir. Şe-



Şekil 3.1 İki boyutlu Gabor benzeri HSA filtresi analog devre gerçeklemesi

kil 3.1'de siyah renkteki devre elemanları Gabor benzeri HSA filtresinin gerçel kısmını, kırmızı renkteki devre elemanlarıysa sanal kısmını gerçeklemektedir. Yeşil renkteki devre elemanları iki katman arasındaki ilişkiyi gerçeklemektedir.

3.2 Gabor Benzeri HSA Filtresinin Sayısal Devre Gerçeklemesi

Gabor benzeri HSA filtresinin hücre durum denklemi (3.2)'de verilmiştir. Bu denklemlerin sayısal olarak gerçeklenebilmesi için zamanda örneklenmesi (2.20)'deki Euler ileri yaklaşıklığıyla örneklenirse (3.9) elde edilir.

$$x_{ij}^{\mathrm{R}}(n+1) = x_{ij}^{\mathrm{R}}(n) + T_s\{\boldsymbol{A}^{\mathrm{R}} \circledast \boldsymbol{X}_{ij}^{\mathrm{R}}(n) - \boldsymbol{A}^{\mathrm{I}} \circledast \boldsymbol{X}_{ij}^{\mathrm{I}}(n) + \boldsymbol{B} \circledast \boldsymbol{U}_{ij}\} \tag{3.9a}$$

$$x_{ij}^{\mathrm{I}}(n+1) = x_{ij}^{\mathrm{I}}(n) + T_s\{\boldsymbol{A}^{\mathrm{R}} \circledast \boldsymbol{X}_{ij}^{\mathrm{I}}(n) + \boldsymbol{A}^{\mathrm{I}} \circledast \boldsymbol{X}_{ij}^{\mathrm{R}}(n)\} \tag{3.9b}$$

(3.9) denklemlerindeki adım aralığı T_s'in değeri (3.10)'de tanımlanan optimum adım aralığı [27] olarak seçildiği zaman (3.11)'de verilen denklemler elde edilir.

$$T_s = \frac{1}{|a_{00}|} = \frac{1}{4+\lambda^2} \tag{3.10}$$

(3.11) denklemlerindeki merkez elemanı sıfır olan $\tilde{\boldsymbol{A}}^{\mathrm{R}}$ çevresel şablonun (Surround template) değerleri (3.12) verilmiştir[28].

$$x_{ij}^{\mathrm{R}}(n+1) = T_s\{\tilde{\boldsymbol{A}}^{\mathrm{R}} \circledast \boldsymbol{X}_{ij}^{\mathrm{R}}(n) - \boldsymbol{A}^{\mathrm{I}} \circledast \boldsymbol{X}_{ij}^{\mathrm{I}}(n) + \boldsymbol{B} \circledast \boldsymbol{U}_{ij}\} \tag{3.11a}$$

$$x_{ij}^{\mathrm{I}}(n+1) = T_s\{\tilde{\boldsymbol{A}}^{\mathrm{R}} \circledast \boldsymbol{X}_{ij}^{\mathrm{I}}(n) + \boldsymbol{A}^{\mathrm{I}} \circledast \boldsymbol{X}_{ij}^{\mathrm{R}}(n)\} \tag{3.11b}$$

(3.12)'deki $\tilde{a}_{00}^{\mathrm{R}}$ değeri (3.13)'de hesaplanmıştır.

$$\tilde{\boldsymbol{A}}^{\mathrm{R}} = \begin{bmatrix} 0 & \cos\omega_{y0} & 0 \\ \cos\omega_{x0} & 0 & \cos\omega_{x0} \\ 0 & \cos\omega_{y0} & 0 \end{bmatrix} \tag{3.12}$$

$$\tilde{a}_{00}^{\mathrm{R}} = a_{00}^{\mathrm{R}} T_s + 1 = -(4+\lambda^2)\frac{1}{(4+\lambda^2)} + 1 = 0 \tag{3.13}$$

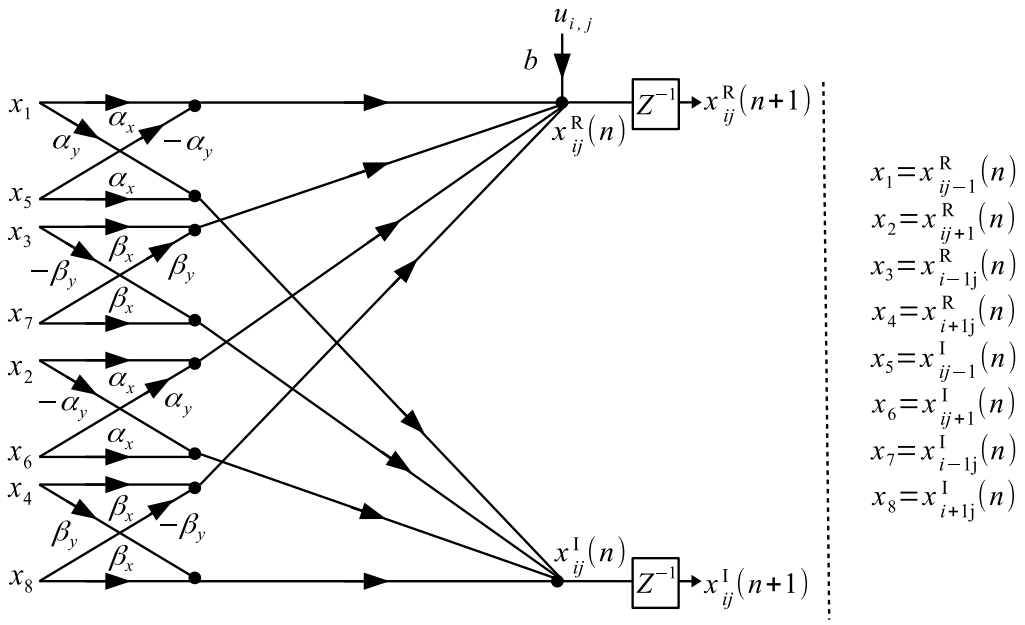
(3.11) durum denklemlerindeki şablon nokta çarpımları yapılırsa (3.14)'deki denklemler elde edilir.

$$\begin{aligned} x_{i,j}^{\mathrm{R}}(n+1) =& \frac{\cos\omega_{y0}}{4+\lambda^2} x_{i-1,j}^{\mathrm{R}}(n) + \frac{\cos\omega_{y0}}{4+\lambda^2} x_{i+1,j}^{\mathrm{R}}(n) + \frac{\cos\omega_{x0}}{4+\lambda^2} x_{i,j+1}^{\mathrm{R}}(n) + \frac{\cos\omega_{x0}}{4+\lambda^2} x_{i,j-1}^{\mathrm{R}}(n) \\ &+ \frac{\sin\omega_{y0}}{4+\lambda^2} x_{i-1,j}^{\mathrm{I}}(n) - \frac{\sin\omega_{y0}}{4+\lambda^2} x_{i+1,j}^{\mathrm{I}}(n) + \frac{\sin\omega_{x0}}{4+\lambda^2} x_{i,j+1}^{\mathrm{I}}(n) - \frac{\sin\omega_{x0}}{4+\lambda^2} x_{i,j-1}^{\mathrm{I}}(n) \\ &+ \lambda^2 u_{i,j} \end{aligned} \tag{3.14a}$$

$$\begin{aligned} x_{i,j}^{\mathrm{I}}(n+1) =& \frac{\cos\omega_{y0}}{4+\lambda^2} x_{i-1,j}^{\mathrm{I}}(n) + \frac{\cos\omega_{y0}}{4+\lambda^2} x_{i+1,j}^{\mathrm{I}}(n) + \frac{\cos\omega_{x0}}{4+\lambda^2} x_{i,j+1}^{\mathrm{I}}(n) + \frac{\cos\omega_{x0}}{4+\lambda^2} x_{i,j-1}^{\mathrm{I}}(n) \\ &- \frac{\sin\omega_{y0}}{4+\lambda^2} x_{i-1,j}^{\mathrm{R}}(n) + \frac{\sin\omega_{y0}}{4+\lambda^2} x_{i+1,j}^{\mathrm{R}}(n) - \frac{\sin\omega_{x0}}{4+\lambda^2} x_{i,j+1}^{\mathrm{R}}(n) + \frac{\sin\omega_{x0}}{4+\lambda^2} x_{i,j-1}^{\mathrm{R}}(n) \end{aligned} \tag{3.14b}$$

(3.14)'de verilen denklemlerdeki katsayılar (3.15)'deki gibi tanımlanırsa (3.16)'de verilen denklemler elde edilir.

$$\alpha_x = \frac{\cos\omega_{x0}}{4+\lambda^2},\ \beta_x = \frac{\cos\omega_{y0}}{4+\lambda^2},\ \alpha_y = \frac{\sin\omega_{x0}}{4+\lambda^2},\ \beta_y = \frac{\sin\omega_{y0}}{4+\lambda^2},\ b = \frac{\lambda^2}{4+\lambda^2} \tag{3.15}$$



Şekil 3.2 Gabor benzeri HSA filtresi kelebek yapıdaki akış diyagramı

(3.16)'deki denklemlerle gerçeklenen iki boyutlu Gabor benzeri HSA filtresinin sayısal devresine ait işaret akış diyagramı Şekil 3.2'de verilmiştir.

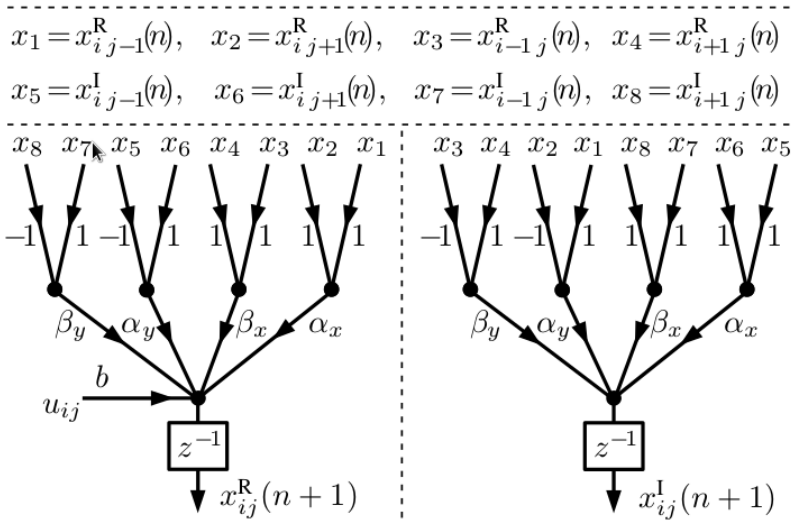
$$\begin{aligned} x_{i,j}^{R}(n+1) =& \beta_x x_{i-1,j}^{R}(n) + \beta_x x_{i+1,j}^{R}(n) + \alpha_x x_{i,j+1}^{R}(n) + \alpha_x x_{i,j-1}^{R}(n) \\ &+ \beta_y x_{i-1,j}^{I}(n) - \beta_y x_{i+1,j}^{I}(n) + \alpha_x x_{i,j+1}^{I}(n) - \alpha_x x_{i,j-1}^{I}(n) \\ &+ b u_{i,j} \end{aligned} \tag{3.16a}$$

$$\begin{aligned} x_{i,j}^{I}(n+1) =& \beta_x x_{i-1,j}^{I}(n) + \beta_x x_{i+1,j}^{I}(n) + \alpha_x x_{i,j+1}^{I}(n) + \alpha_x x_{i,j-1}^{I}(n) \\ &- \beta_y x_{i-1,j}^{R}(n) + \beta_y x_{i+1,j}^{R}(n) - \alpha_x x_{i,j+1}^{R}(n) + \alpha_x x_{i,j-1}^{R}(n) \end{aligned} \tag{3.16b}$$

(3.16)'deki denklemlerde aynı katsayılar ortak paranteze alınırsa (3.17)'deki denklemler elde edilir. Bu denklemler Şekil 3.3'de verilen işaret akış diyagramlarıyla gerçeklenir.

$$\begin{aligned} x_{i,j}^{R}(n+1) =& \beta_x \left[x_{i-1,j}^{R}(n) + x_{i+1,j}^{R}(n) \right] + \alpha_x \left[x_{i,j+1}^{R}(n) + x_{i,j-1}^{R}(n) \right] \\ &+ \beta_y \left[x_{i-1,j}^{I}(n) - x_{i+1,j}^{I}(n) \right] + \alpha_x \left[x_{i,j+1}^{I}(n) - x_{i,j-1}^{I}(n) \right] \\ &+ b u_{i,j} \end{aligned} \tag{3.17a}$$

$$\begin{aligned} x_{i,j}^{I}(n+1) =& \beta_x \left[x_{i-1,j}^{I}(n) + x_{i+1,j}^{I}(n) \right] + \alpha_x \left[x_{i,j+1}^{I}(n) + x_{i,j-1}^{I}(n) \right] \\ &- \beta_y \left[x_{i-1,j}^{R}(n) + x_{i+1,j}^{R}(n) \right] - \alpha_x \left[x_{i,j+1}^{R}(n) + x_{i,j-1}^{R}(n) \right] \end{aligned} \tag{3.17b}$$



Şekil 3.3 Kaynak iyileştirilmesi yapılmış Gabor benzeri HSA filtresinin akış diyagramı

Şekil 3.2'deki işaret akış diyagramında 16 tane iki girişli çarpma işlemi, 15 tane iki girişli toplama işlemi ve 2 tane bellek elemanı bulunurken, Şekil 3.3'de işaret akış diyagramında ise 8 tane iki girişli çarpma işlemi, 15 tane iki girişli toplama işlemi ve 2 tane bellek elemanı bulunmaktadır. Sayısal devre gerçeklemelerinde önemli bir eleman olan çarpıcı yapısı bu şekilde yarı yarıya düşürülmüştür.

3.2.1 Dinamik Bölge ve Yakınsaklık Analizi

Gabor benzeri HSA filtresinin sayısal devre gerçeklemesinde kullanılan sabit noktalı aritmetik sırasında meydana gelecek taşma işlemlerinden kaçınmak için, dinamik bölgenin belirlenerek tasarım sırasında dikkat edilmesi gerekmektedir. Diğer taraftan, gerçeklemedeki Euler iterasyon sayısı da belirlenmelidir.Bu bölümde, dinamik bölge ve yakınsaklık analizi ile ilgili iki teorem verilmiştir.

Teorem 1 *Giriş ve hücre durumlarının ilk değerlerinin Euclidean normları* $[-1,1]$ *aralığında ise, tüm hücre durumlarına ait değerler hesaplamalar sırasında aynı aralıkta kalır. Başka bir ifadeyle,* $\forall i,j,n$ *için, eğer* $|u_{ij}| \leq 1$ *ve* $\|\mathbf{x}_{ij}(0)\| \leq 1$ *için*

$$|x^R_{ij}(n)| \leq 1,\ \ |x^I_{ij}(n)| \leq 1. \tag{3.18}$$

elde edilir. Buradaki $\mathbf{x}_{ij}(n) = \left[x^R_{ij}(n)\ x^I_{ij}(n)\right]^T$*'dir ve* $\|\cdot\|$ *Euclidean normu gösterir.*

Kanıt 1 (3.16) *hücre fark denklemleri* (3.19)*'de verildiği gibi matris formunda yazalım.*

$$\mathbf{x}_{ij}(n+1) = T_s\left\{\mathbf{G}_x\mathbf{x}_{i\,j-1}(n) + \mathbf{G}_x^t\mathbf{x}_{i\,j+1}(n) + \mathbf{G}_y^t\mathbf{x}_{i-1\,j}(n) + \mathbf{G}_y\mathbf{x}_{i+1\,j}(n) + \lambda^2\mathbf{u}_{ij}\right\} \quad (3.19)$$

(3.19)*'deki* $\mathbf{G}_\kappa$, $\mathbf{x}_{ij}(n)$ *ve* $\mathbf{u}_{ij}$ *değerleri* $\kappa \in \{x,y\}$ *ile* (3.20)*'de verilmiştir.*

$$\mathbf{G}_\kappa = \begin{bmatrix} \cos\omega_{\kappa 0} & -\sin\omega_{\kappa 0} \\ \sin\omega_{\kappa 0} & \cos\omega_{\kappa 0} \end{bmatrix},\ \mathbf{x}_{ij}(n) = \begin{bmatrix} x_{ij}^R(n) \\ x_{ij}^I(n) \end{bmatrix},\ \mathbf{u}_{ij} = \begin{bmatrix} u_{ij} \\ 0 \end{bmatrix}, \quad (3.20)$$

(3.19)*'den* (3.21) *elde edelir.*

$$\begin{aligned}\|\mathbf{x}_{ij}(n+1)\| &= (4+\lambda^2)^{-1}\left\|\mathbf{G}_x\mathbf{x}_{i\,j-1}(n)+\mathbf{G}_x^t\mathbf{x}_{i\,j+1}(n)+\mathbf{G}_y^t\mathbf{x}_{i-1\,j}(n)+\mathbf{G}_y\mathbf{x}_{i+1\,j}(n)+\lambda^2\mathbf{u}_{ij}\right\| \\ &\le (4+\lambda^2)^{-1}\left\{\|\mathbf{G}_x\mathbf{x}_{i\,j-1}(n)\|+\|\mathbf{G}_x^t\mathbf{x}_{i\,j+1}(n)\|+\|\mathbf{G}_y^t\mathbf{x}_{i-1\,j}(n)\|+\|\mathbf{G}_y\mathbf{x}_{i+1\,j}(n)\|\right. \\ &\qquad \left.+\lambda^2\|\mathbf{u}_{ij}\|\right\}. \quad (3.21)\end{aligned}$$

$\mathbf{G}_\kappa$ *Givens dönüşümü olduğu için* $\|\mathbf{G}_\kappa\mathbf{x}_{ija}(n)\| = \|\mathbf{x}_{ij}(n)\|$, $\mathbf{u}_{ij}$ *vektörünün bir elemanı sıfır olduğu için* $\|\mathbf{u}_{ij}\| = |\mathbf{u}_{ij}|$ *olur ve* $|u_{ij}| \le 1$ *and* $\|\mathbf{x}_{ij}(0)\| \le 1$ *oldukları varsayılırsa* (3.21)*'den* (3.22) *elde edilir.*

$$\begin{aligned}\|\mathbf{x}_{ij}(n+1)\| &\le (4+\lambda^2)^{-1}\left\{\|\mathbf{x}_{i\,j-1}(n)\| + \|\mathbf{x}_{i\,j+1}(n) + \|\mathbf{x}_{i-1\,j}(n)\| + \|\mathbf{x}_{i+1\,j}(n)\| + \lambda^2|u_{ij}|\right\} \\ &\le (4+\lambda^2)^{-1}(4+\lambda^2) = 1 \quad (3.22)\end{aligned}$$

(3.22)*'e göre,* $\forall i,j,n$, $|x_{ij}^R(n)| \le 1$, $|x_{ij}^I(n)| \le 1$ *dir.* ■

Teorem 2 *Ayrık zamanlı Gabor benzeri HSA filtresinin hücre durumlarının kararlı değerlere ulaşması için gerekli Euler iterasyonu sayısı filtrenin bant genişliğiyle ters orantılıdır.*

Kanıt 2 (2.17)*'deki doğrusal HSA'nın vektör matris formu gibi ayrık zamanlı Gabor benzeri HSA filtresinin hücre durum denklemleri vektör matris formunda yazılır.* $\mathbf{A}$ *geri besleme matrisini sıfırdan farklı elemanları* $\tilde{\mathbf{A}}$ *şablonunun* $\tilde{a}_{0\,-1}$, $\tilde{a}_{01}$, $\tilde{a}_{-10}$, $\tilde{a}_{10}$ *elemanlarının* T_s *ile çarpımlarıdır ve bu değerler matrisin her satırında sadece bir defa bulunmaktadır. Sonuç olarak,* $\mathbf{A}$ *matrisinin her satırındaki elemanların mutlak değlerinin toplamının en büyüğü* $\mathbf{A}$ *is* $\|\mathbf{A}\|_\infty = \max_\xi \sum_{\zeta=1}^{KL} |a_{\xi\zeta}| = 4/(4+\lambda^2)$ *dir. Gershgorin teoremine*

[ddd] göre $|\sigma_\rho|_{max} \leq 4/(4+\lambda^2)$ *dir. Buradaki* $|\sigma_\rho|_{max}$, $\mathbf{A}$ *matrisinin özdeğerlerinin* σ_ρ, $\rho \in \{1,2,...KL\}$ *en büyüğü olan spektral yarıçapıdır.*

Yukarıdan anlaşılacağı gibi, büyük bant genişlikleri λ *için spektral yarıçap küçük olur ve bundan dolayı hücre durumları kararlı hale daha hızlı yakınsarlar. Küçük bant genişliği içinde tersi geçerlidir.* ■

3.2.2 Benzetim Sonuçları

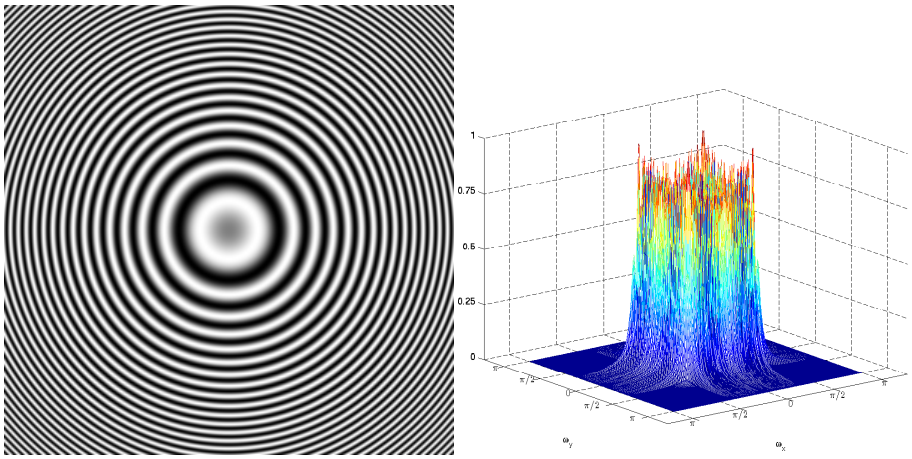
Bu bölümde, ilk olarak ayrık zamanlı Gabor benzeri HSA filtresinin gerçeklenmesi için gerekli iterasyon sayısının tespiti için yapılan benzetim ve sonuçları verilmiştir. Sayısal devre gerçeklemelerinde kaynak kullanımının azaltılması için sabit noktalı aritmektik kullanılmaktadır, bu nedenle filtre katsayıları ve durum değişkenlerinin kaç bit ile gerçekleneceğiyle ilgili yapılan analiz, benzetim ve sonuçlar verilmiştir.

3.2.2.1 İterasyon Sayısının Tespiti

Bölüm 2.3.4'de ayrık zamanlı HSA'nın sonucunun yani ağın kararlı durumunun hesaplanabilmesi için hücre fark denklemlerinin iterasyonla çözülmesi gerektiği gösterilmişti. Ayrık zamanlı Gabor benzeri HSA filtresinin sonucunun hesaplanabilmesi için gerekli iterasyon sayısının filtrenin bant genişliğiyle ters orantılı olduğu Teorem 2'de kanıtlanmıştır. İterasyon sayısı filtrenin bant genişliği dışında HSA hücre durumlarının ilk koşullarına ve giriş görüntüsüne de bağlıdır. Burada sonuçları ve detayları verilen benzetimde filtre katsayılarıyla iterasyon sayısı arasındaki ilişki tanımlanmıştır.

İlk olarak benzetimin algoritması ve daha sonra benzetim sırasında kullanılan giriş görüntüsü, filtre parametreleri, benzerlik kriteri ve filtre sonucunun kararlı hal çıkışının hesaplanmasıyla ilgili detaylar verilmiştir. Bu detaylardan sonra benzetim sonuçları ve tartışma verilmiştir.

Benzetimin algoritması şu şekildedir: belirlenen filtre parametreleri ve giriş görüntüsüne göre kararlı haldeki çıkış hesaplanır ve kayıt edilir, belirlenen filtre parametleri, giriş görüntüsü ve ilk koşula göre hücre durumlarının çıkışları hesaplanır ve kayıt edilir, her



(a) (b)

Şekil 3.4 (a) 8 bit parlaklık değerine sahip sinusoidal zone plate ve (b) iki boyutlu ayrık Fourier dönüşümü.

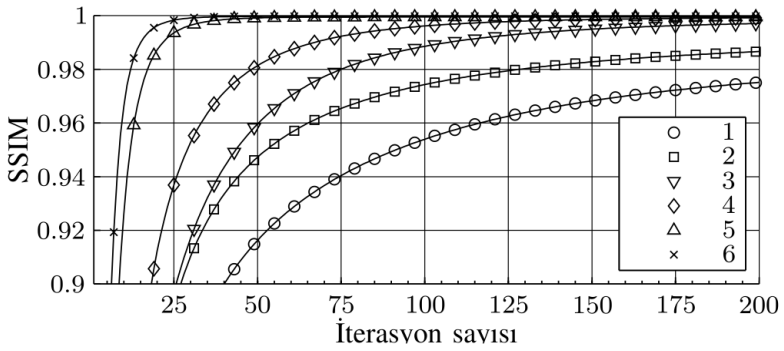
iterasyon sonucunda kararı hal çıkışıyla iterasyon çıkışı gerekli kritere gelip gelmediği kontrol edilir. Bu şekilde seçilen tüm filtre parametreleri için benzetim tekrarlanır.

Benzetim sırasında giriş görüntüsü olarak 256 gri seviyeli 1080×1920 boyutlarında Sinusoidal Zone Plate (SZP) [29] kullanılmıştır. Şekil 3.4 verilen giriş görüntüsü, içerdiği her yönde ve her frekanstaki bileşenlerden dolayı seçilmiştir. Filtre parametreleri (λ, ω_0) çifti için $(0.3, \pi/3)$, $(0.3, \pi/9)$, $(0.1, \pi/3)$, $(0.1, \pi/9)$, $(0.02, \pi/3)$ ve $(0.02, \pi/9)$ olarak seçilmiştir. İlk koşullar, hücre durumlarının gerçel kısmı için giriş görüntüsünün aynısı, sanal kısmı için sıfır olarak alınmıştır.

Hücre durumlarına ait kararlı hal çıkış değerleri (3.23) denkleminde verildiği hesaplanmıştır [28]. (3.23)'deki $\mathscr{F}$ ve $\mathscr{F}^{-1}$ sırasıyla ayrık Fourier dönüşümü ve ters dönüşümünü gösterir.

$$x_{ij}(\infty) = \mathscr{F}^{-1}\left\{-\frac{\mathscr{F}\{b_{ij}\}}{\mathscr{F}\{a_{ij}\}}\mathscr{F}\{u_{ij}\}\right\} \tag{3.23}$$

Karşılaştırma kriteri olarak, iki görüntü arasında görsel açıdan bir benzerlik sunan SIMilarity (SSIM) indeks [24] kullanılmıştır. SSIM, PSNR, SNR v.b. kriterlere göre filtrenin çıkışının görsel olarak daha iyi belirlemektedir. Benzetim sonuçları Şekil 3.5'de $(0.3, \pi/3)$, $(0.3, \pi/9)$, $(0.1, \pi/3)$, $(0.1, \pi/9)$, $(0.02, \pi/3)$ ve $(0.02, \pi/9)$ değerleri için 1'den 6'ya kadar sırayla verilmiştir. Bu onuçlarına göre SSIM değeri 0.96 için iterasyon sonucu ile



Şekil 3.5 Sinusoidal zone plate

kararlı hal arasındaki benzerliğin yeterli olduğu gözlenmiştir. Örnek olarak $(0.02, \frac{\pi}{9})$ çifti için çizilen eğri 0.96 SSIM değerini kestiği noktadaki iterasyon değeri 120'dir. Demek ki $(0.02, \frac{\pi}{9})$ parametrelerine sahip Gabor benzeri HSA filtresinin sayısal olarak gerçeklenebilmesi için 120 iterasyon gerekmektedir. Aynı şekilde $(0.3, \frac{\pi}{3})$ parametreleri için 10 iterasyon yeterlidir.

3.2.2.2 Bit Genişliklerinin Tespiti

Sayısal devre gerçeklemeleride kullanılan sabit noktalı aritmetikte bit genişliklerine bağlı olarak meydana gelen yuvarlama hatalarının tespiti için bu benzetim yapılmıştır. Ayrıca daha bit genişliği kaynak kullanımını da etkilemektedir. Teorem 1'de tüm katsayıların ve durumların tüm işlemler sırasında $[-1, 1]$ aralığında kalacağı kanıtlanmıştı. Buna göre, tüm katsayılar ve durumlar bir bit işaret olmak üzere 1.X formatında ifade edilebilir.

İlk olarak benzetimin algoritması ve daha sonra benzetim sırasında kullanılan giriş görüntüsü, filtre parametreleri, iterasyon sayısı ve benzerlik kriteri ilgili detaylar verilmiştir. Bu detaylardan sonra benzetim sonuçları ve tartışma verilmiştir.

Benzetimin algoritması şu şekildedir: belirlenen filtre parametreleri ve giriş görüntüsüne göre benzetim kayan noktalı olarak yapılır ve sonuç kayıt edilir, b katsayısı, α ve β katsayıları, $x_{ij}^{\mathrm{R}}(n)$ ve $x_{ij}^{\mathrm{R}}(n)$ durumların değerleri, bu_{ij} çarpım değeri ve u_{ij} giriş değerlerinden her seferinde bir tanesinin bit genişliği 3'ten 32'ye kadar benzetim tekrarlanırken diğer değerler kayan noktalı olarak tutulur ve sonuçlar kayıt edilir, bu işlemler her değişken için tekrar edilir ve sonuçlar kayıt edilir. Kayan noktalı sonuçlarla sabit noktalı sonuçlar

karşılaştırılarak grafik üzerinde çizdirilir.

Benzetim Şekil 3.3'deki işaret akış diyagramı, $\theta = \frac{\pi}{4}$, iterasyon sayısı 50, $\omega_0 = \frac{\pi}{4}$ ve $\lambda = (0.02, 0.1, 0.3)$ değerleriyle yapılmıştır. Kriter olarak kullan PSNR değeri 60 dB yeterli olarak kullanılmıştır.

Şekil 3.6'de b katsayısın bit genişliğine göre Gabor benzeri HSA filtresindeki yuvarlama hatalarına göre meydana gelen bozulma verilmiştir. Sonuçlar incelendiğinde en kötü durum da b katsayısının 60 dB PSNR değeri için 18 bit genişliğinde olması gerektiği görünmektedir.

Şekil 3.7'de α, β katsayılarının bit genişlişliklerine göre Gabor benzeri HSA filtresindeki yuvarlama hatalarına göre meydana gelen bozulma verilmiştir. Sonuçlar incelendiğinde en kötü durumlarda α, β katsayılarının 60 dB PSNR değeri için 14 bit genişliğinde olması gerektiği görünmektedir.

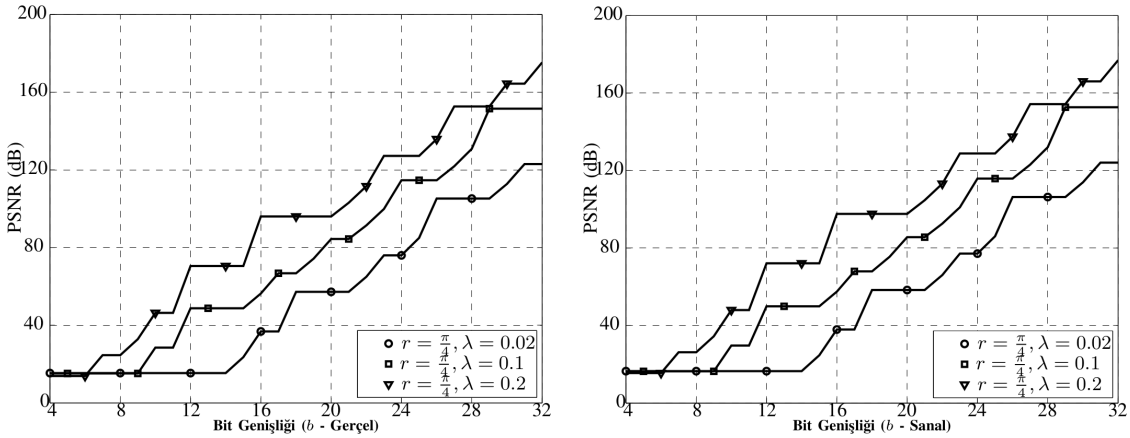
Şekil 3.8'de $x_{ij}^{\mathrm{R}}(n)$ ve $x_{ij}^{\mathrm{R}}(n)$ durumların değerlerinin bit genişlişliklerine göre Gabor benzeri HSA filtresindeki yuvarlama hatalarına göre meydana gelen bozulma verilmiştir. Sonuçlar incelendiğinde en kötü durumlarda $x_{ij}^{\mathrm{R}}(n)$ ve $x_{ij}^{\mathrm{R}}(n)$ durumların değerlerinin 60 dB PSNR değeri için 14 bit genişliğinde olması gerektiği görünmektedir.

Şekil 3.9'de bu_{ij} çarpım değerinin bit genişlişiğine göre Gabor benzeri HSA filtresindeki yuvarlama hatalarına göre meydana gelen bozulma verilmiştir. Sonuçlar incelendiğinde en kötü durumlarda bu_{ij} çarpım değerinin 60 dB PSNR değeri için 14 bit genişliğinde olması gerektiği görünmektedir.

Şekil 3.10'de u_{ij} giriş görüntüsünün bit genişlişiğine göre Gabor benzeri HSA filtresindeki yuvarlama hatalarına göre meydana gelen bozulma verilmiştir. Sonuçlar incelendiğinde en kötü durumlarda u_{ij} giriş görüntüsünün 60 dB PSNR değeri için 8 bit genişliğinde olması gerektiği görünmektedir.

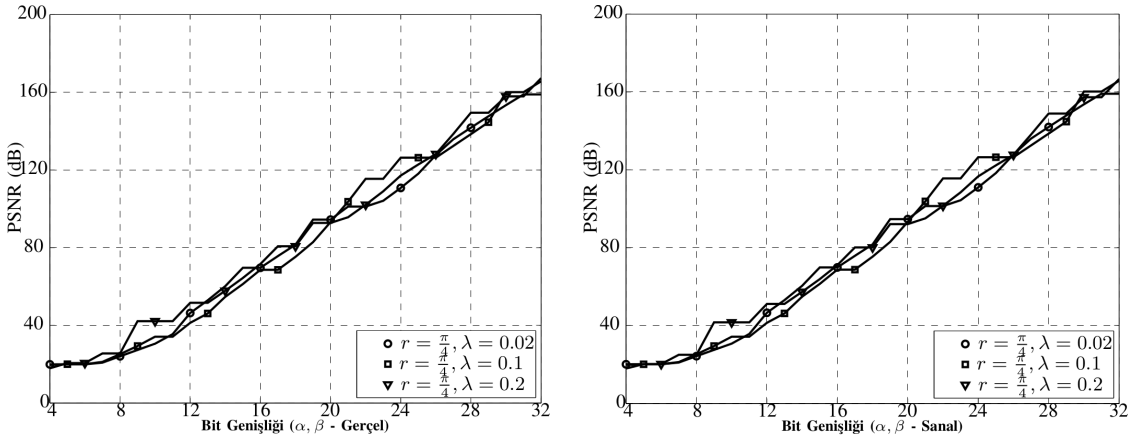
Benzetim sonuçlarına göre 60 dB PSNR değeri için b katsayısının en az 18 bit, α, β katsayılarının en az 14 bit ve $x_{ij}^{\mathrm{R}}(n)$ ve $x_{ij}^{\mathrm{R}}(n)$ durumlarının ise en az 14'er bit olması gerekir.

Son olarak Şekil 3.11'de ilk olarak bu_{ij} çarpımının bit genişliği 24 bit, α, β ve b katsayılarınnın bit genişlikleri 16 bit seçilerek, $x_{ij}^{\mathrm{R}}(n)$ ve $x_{ij}^{\mathrm{I}}(n)$'lerin bit genişlikleri 3–32 bit



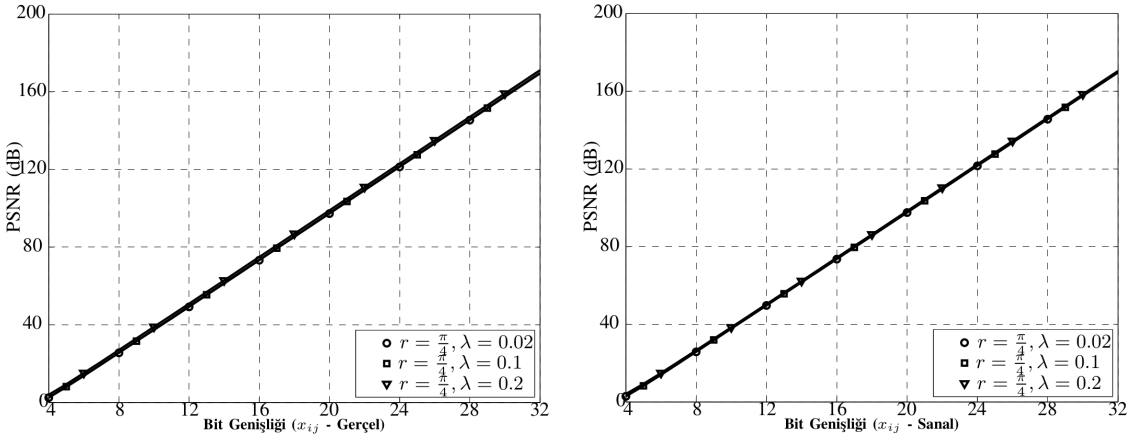
(a) (b)

Şekil 3.6 *b* katsayısının bit genişliği.



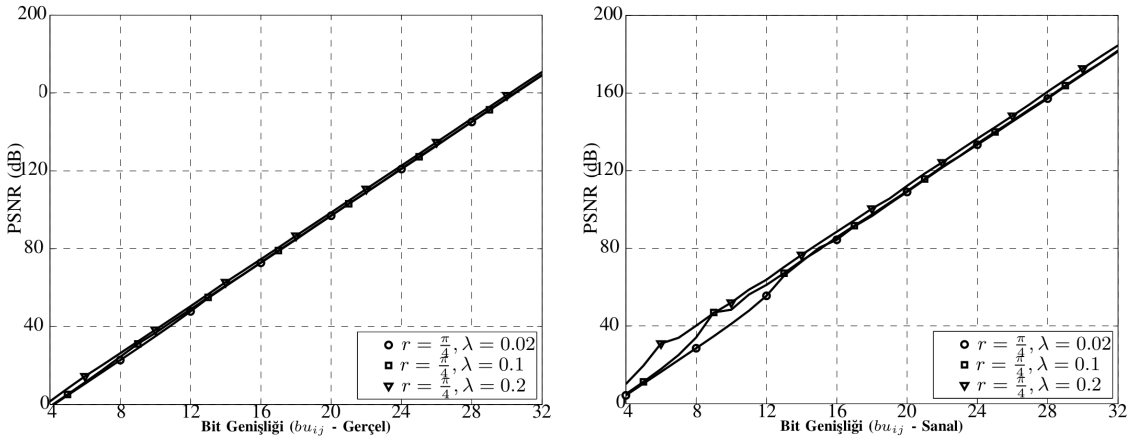
(a) (b)

Şekil 3.7 Katsayıların bit genişliği.



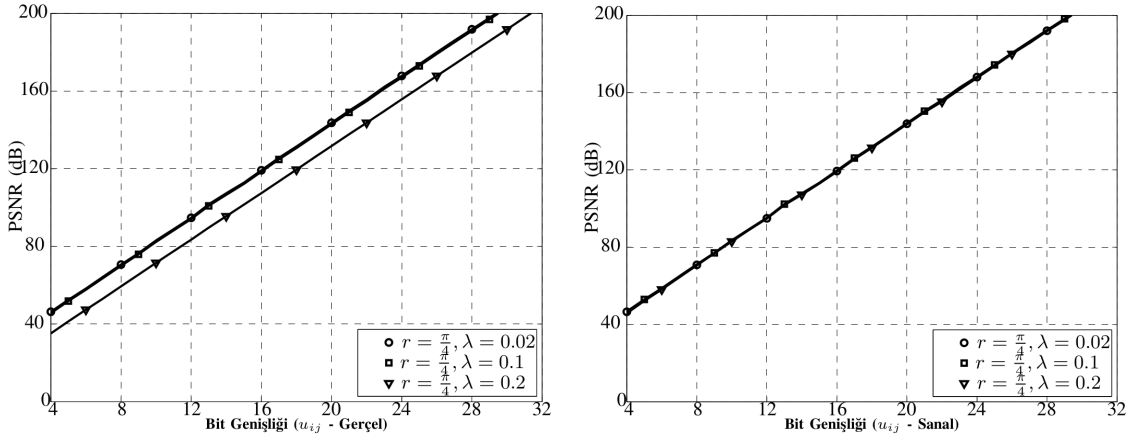
(a) (b)

Şekil 3.8 Durumların bit genişliği.



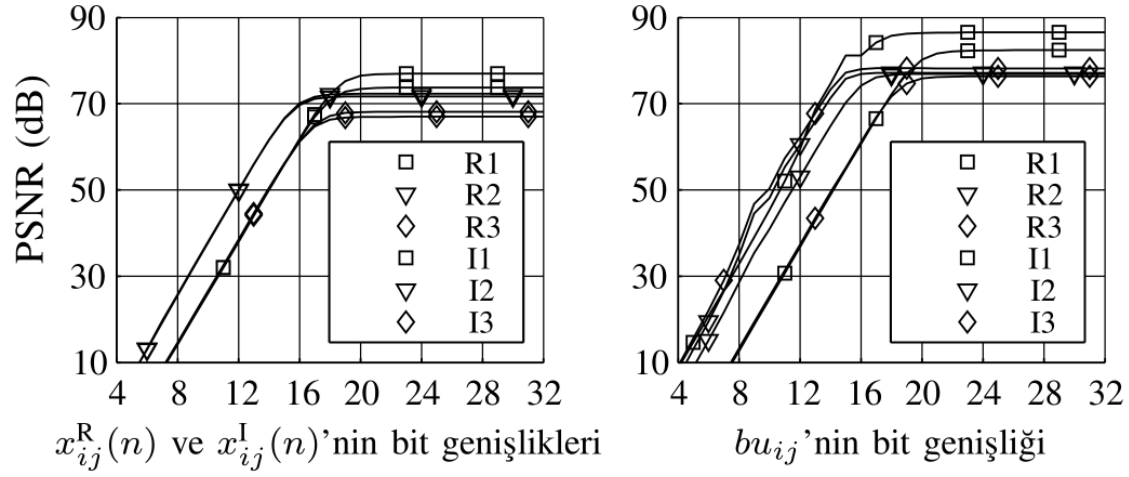
(a) (b)

Şekil 3.9 bu_{ij} çarpımının bit genişliği.



(a) (b)

Şekil 3.10 Girişin bit genişliği.



Şekil 3.11 $x_{ij}^{\mathrm{R}}(n)$, $x_{ij}^{\mathrm{I}}(n)$ ve bu_{ij}'nin bit genişliklerinin tespiti.

aralığında değiştirilmiştir, daha sonraise $x_{ij}^{\mathrm{R}}(n)$ ve $x_{ij}^{\mathrm{I}}(n)$ lerin bit genişlikleri 24 bit sabit tutularak bu_{ij}'nin bit genişliği 3–32 bit arasında değiştirilmiştir. Şekil 3.11'de 1'den 3'e kadar olan eğriler (ω_0, λ) çifti nin $(\pi/9, 0.02)$, $(\pi/3, , 0.02)$ ve $(\pi/9, 0.1)$ değerleri için çizdirilmiştir. Benzetim sonuçlarına göre durumların ve bu_{ij}'nin bit genişlikleri sırasıyla 12 bit ve 24 bit seçildiği zaman PSNR değeri 60 dB'nin üzerinde olduğu görünmektedir. Bu sonuçlara göre α, β ve b katsayılarının bit genişliği 16 bit, durumların bit genişliği 12 bit ve bu_{ij} çarpımının bit genişliği ise 24 bit olması sonucuna varılmıştır.

3.3 Sonuç

Bu bölümde, ilk olarak [12]'de verilen Gabor benzeri HSA filtresini gerçekleyen analog devrenin detayları incelenmiştir. Daha sonra tez kapsamında gerçeklemesi yapılan ayrık zamanlı Gabor benzeri HSA filtresinin detayları ve iki tane işaret akış diyagramı verilmiştir.

Ayrık zamanlı Gabor benzeri HSA filtresinin dinamik bölge ve yakısaklık analiziyle ilgili iki teorem verilmiş kanıtları yapılmıştır. Bu teoremlere göre filtrenin tüm katsayıları ve durum değişkenlerinin değerleri işlemler sırasında $[-1, 1]$ aralığı içinde kaldığı kanıtlanmıştır. Ayrıca filtre sonlu sayıda iterasyon sonucunda kararı bölgeye gideceği ve iterasyon sayısının bant genişliğiyle ters orantılı olduğu kanıtlanmıştır.

Son olarak sayısal devre gerçekleme öncesi Gabor benzeri HSA filtresinin, iterasyon sayısının ve bit genişliklerinin tespiti için benzetimler yapılmış ve sonuçları detaylı olarak incelenmiştir. Bu sonuçlara göre α, β ve b katsayılarının bit genişliği 16 bit, durumların bit genişliği 12 bit ve bu_{ij} çarpımının bit genişliği ise 24 bit olması gerekir.

BÖLÜM 4

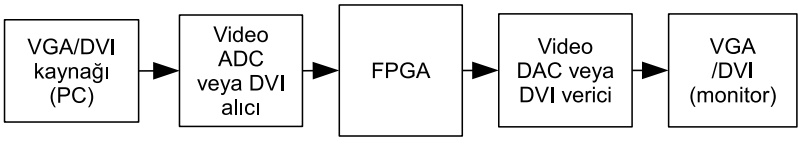
TASARLANAN GERÇEK ZAMANLI BİR HSA GERÇEKLEMESİ

Bu bölümde sırasıyla TÜBİTAK projesi kapsamında tasarlanan gerçek zamanlı bir HSA gerçeklemesinin blok diyagramı, video giriş/çıkış blokları, bellek organizasyonu, işlemci dizileri ve yerel kontrol [30, 31] mantığı, işlemci bloğu, gecikme bloğu, seri programlama bloğu, histogram hesaplama ve karşıtlık ayarlama bloğu ve PC testlerinde kullanılan PC test ortamının detayları verilmiştir.

4.1 Gerçek Zamanlı HSA Gerçeklemesinin Blok Diyagramı ve Mimarisi

Sistemin basitleştirilmiş blok diyagramı Şekil 4.1'de verilmiştir. Video kaynağından gelen sıralı görüntü HSA işlemci dizisinin üzerinde gerçeklendiği FPGA tümdevresinde işlendikten sonra, işlenmiş görüntü monitör/TV görünmektedir. Sistemin çalışmasının daha iyi anlaşılabilmesi için ilk olarak video görüntüsünün detaylarının incelenmesi gerekir. Video giriş tümdevresi, sayısal/analog video kaynağından gelen video görüntüsünü satır–satır gelecek şekilde 6 farklı sayısal sinyale dönüştürmektedir. Bu sinyaller görevleri bakımından kontrol ve veri sinyalleri olmak üzere ikiye ayrılırlar.

- Kontrol Sinyalleri:
  - *pclk* : Her bir yükselen/düşen kenarında yeni bir pikselin geldiğini belirler. Tüm kontrol ve veri sinyalleri bu sinyal ile senkron olarak çalışır.



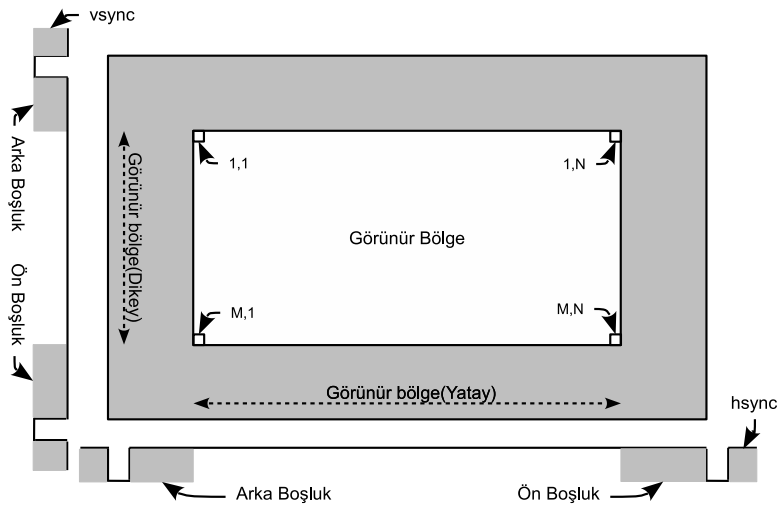
Şekil 4.1 Sistemin basitleştirilmiş blok diyagramı

  - *hsync*: Yeni bir satırın başladığını belirtir. Yani görüntünün bir satırının başlangıç ve bitiş noktalarını işaretler. Her iki *hsync* darbesi arasında görüntünün bir satırı işaretlenir.
  - *vsync*: Yeni bir video çerçevesinin başladığını belirtir. Yani bir video çerçevesinin başalangıç ve bitiş noktalarını işaretler. Her iki *vsync* darbesi arasında videonun bir çerçevesi işaretlenir.

- Veri Sinyalleri:
  - *RED (R)* : Her bir piksel parlaklık değerinin kırmızı bileşenidir. Genellikle 8/10–bit (256/1024 seviye) olarak kullanılır.
  - *Green (G)* : Her bir piksel parlaklık değerinin yeşil bileşenidir. Genellikle 8/10–bit (256/1024 seviye) olarak kullanılır.
  - *Blue (B)* : Her bir piksel parlaklık değerinin mavi bileşenidir. Genellikle 8/10–bit (256/1024 seviye) olarak kullanılır.

Şekil 4.2'de görüldüğü gibi video çerçevesi görünür ve görünmeyen (karanlık) iki bölümden oluşmaktadır. Görünür bölge işlenecek olan görüntünün bulundupu kısımdır. Karanlık bölge ise sistemlerin senkronizasyonu için kullanılmaktadır. Örneğin karanlık bölge gerçek zamanlı HSA gerçeklemesinde blokların resetlenmesi ve bellek elemanlarının ilk değerlerinin yüklenmesi için gerekli süreyi sağlamaktadır. *hsync*, *vsync* video kontrol sinyalleri görünür bölgenin sınırlarını sırasıyla yatay ve düşeyde belirler. *pclk* sinyalinin her bir darbesine karşılık *RGB* kanalında bir piksel değeri gelir.

Video kaynağına yada video giriş tümdevresine göre *hsync*, *vsync* kontrol sinyalleri farklı polaritelerde olabilir, bu nedenle standart kontrol sinyallerine ihtiyaç duyulmuştur. Kontrol sinyalleri videodaki karanlık bölgeleri işaretlerken, aynı zamanda daha kolay üretilmelidir ve işlemciler içinde senkronizasyon için basit geciktirme işlemi yapılabilmelidir. Bu nedenlerden dolayı Şekil 4.3'de karanlık ve görünen bölgeleri işaretleyen *hframe* ve *vframe* isimli yeni kontrol sinyalleri kullanılmıştır. *hframe* ve *vframe* kontrol sinyallerinin değerlerine göre görünür bölgenin sağı, solu, üstü ve altı belirlenebilmektedir. Görünür bölgenin sınırlarının belirlenmesi, sınır koşullarının üretilmesinde, filtre bloğu içindeki iç



Şekil 4.2 Video kontrol sinyalleri ve görünür bölge

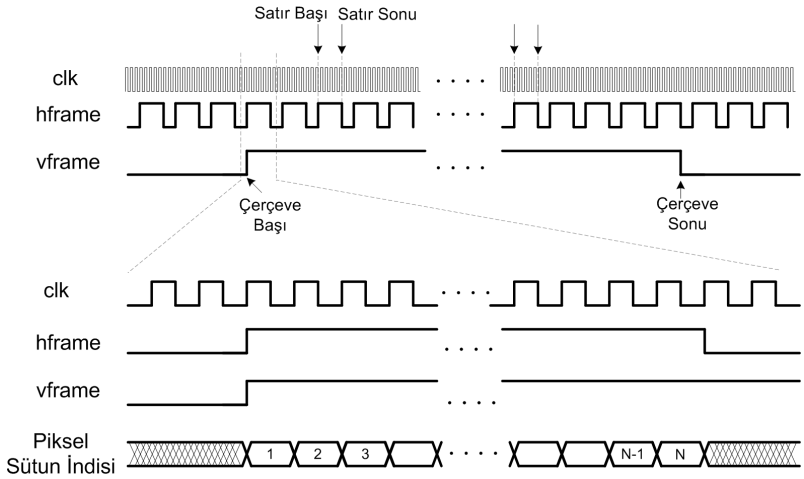
hframe = 0 vframe = 0	hframe = 1 vframe = 0	hframe = 0 vframe = 0
hframe = 0 vframe = 1	hframe = 1 vframe = 1 (görünür bölge)	hframe = 0 vframe = 1
hframe = 0 vframe = 0	hframe = 1 vframe = 0	hframe = 0 vframe = 0

Şekil 4.3 *hframe*, *vframe* sinyalleri ve görünür bölgenin işaretlenmesi.

blokların resetlenmesinde, bellek elemanlarının ilk değerlerini almasında kullanılmaktadır.

Yeni kontrol sinyallerinin detayları Şekil 4.4'de verilmiştir. *vframe* sinyalinin yükselen kenarı yeni bir video çerçevesinin başlangıcını, düşen kenarı ise bitişini gösterir. *hframe* sinyalinin yükselen kenarı video satırının başlangıcını, düşen kenarı ise bitişini gösterir. *pclk* sinyalinin yükselen kenarında yeni bir piksel gelir. Veri, *hframe* ve *vframe* sinyalleri *pclk* sinyaline senkrondur.

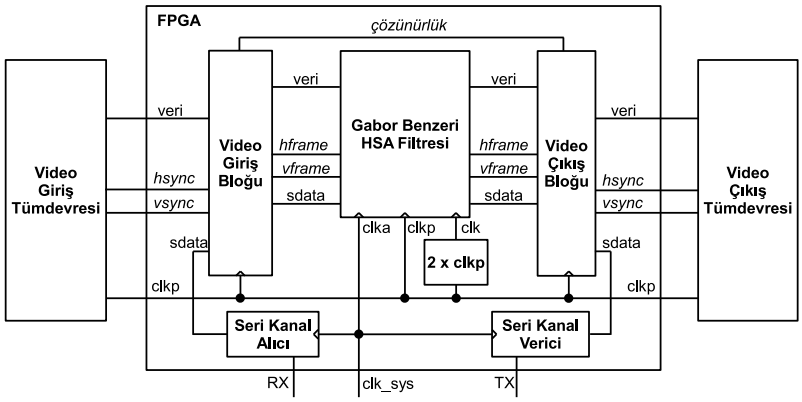
Şekil 4.1'de verilen sistemde FPGA tümdevresi içinde gerçeklenen sistemin blok diyagramı Şekil 4.5'de verilmiştir. FPGA tümdevresi yanında DVI alıcı ve DVI verici tümdevreleri de bulunmaktadır. FPGA tümdevresi içinde, video giriş bloğu, HSA işlemci dizisi, video çıkış bloğu, seri programlama alıcı ve verici blokları bulunmaktadır. Bütün blok-



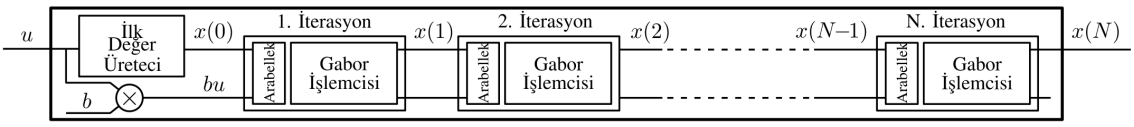
Şekil 4.4 *hframe*, *vframe* sinyallerinin zamandaki değişimi.

ların saat, reset ve kontrol sinyalleri bulunmaktadır. Saat sinyali dışındaki sinyaller blok diyagramının basitleştirilmesi için diyagrama konulmamıştır.

Video giriş bloğu, DVI alıcı tümdevresinden gelen veri, *hsync*, *vsync* kontrol sinyalleri ve piksel saat sinyalini kullanarak DVI alıcı tümdevresinden gelen videonun çözünürlüğünü belirler ve video çıkış bloğuna iletir. Ayrıca *hsync* ve *vsync* kontrol sinyallerinden daha önce açıklanan *hframe* ve *vframe* kontrol sinyallerini üretir. Video giriş bloğunun bir diğer görevi ise sisteme güç verildiğinde ve giriş video sinyalindeki herhangi bir bozukluk anında tüm sistemin resetlemesidir. Video çıkış bloğu, video giriş bloğundan gelen çözünürlük bilgisi, HSA işlemci dizisinden gelen veri, *hframe* ve *vframe* kontrol sinyallerinden DVI verici tümdevresi için gerekli veri, *hsync*, *vsync* sinyallerini üretir. HSA işlemci dizisi, HSA'nı gerçeleyen işlemci dizisini içerir. Video giriş bloğunda gelen *RGB* veri ve *hframe*, *vframe* kontrol sinyalleri *pclk* sinyali ise senkron olarak HSA işlemci dizisine girer, HSA işlemci dizisi içinde gerekli hesaplamalar yapılırken senkronizasyon bozulmayacak şekilde *hframe*, *vframe* kontrol sinyalleri geciktirilir ve video çıkış bloğuna aktarılır. HSA işlemci dizisinin toplam gecikmesi N saat darbesi kadar ise, *hframe* ve *vframe* kontrol sinyalleri senkronizasyonun bozulmaması için blok içinde N saat darbesi kadar geciktirilir. Seri programlama blokları, sistemin bir PC ile kontrol edilmesi için tasarlanamıştır. PC ile UART standartıyla haberleşen bloklar, sistemdeki diğer bloklarla saat ve veri hattı olmak üzere senkron iki hat ile haberleşmektedir. HSA şablon katsayıları, sınır koşulları, ilk koşulların değeri, video giriş bloğunun ürettiği çözünürlük bilgilerinin okunması için



Şekil 4.5 Gerçek zamanlı bir HSA gerçeklemsinin temel blokları



Şekil 4.6 HSA gerçeklemesinin işlemci dizisinin basitleştirilmiş blok diyagramı. Tamamen iş hattı formunda tasarlanan her işlemci bir iterasyona karşılık gelmektedir.

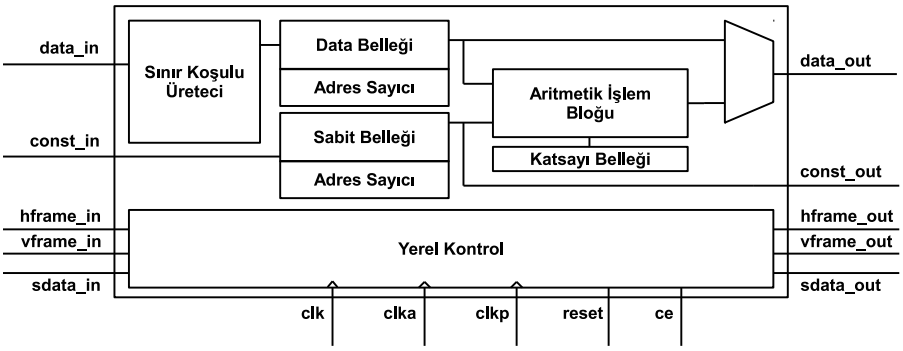
kullanılmaktadır.

HSA'nı gerçekleyen işlemci dizisinin basitleştirilmiş blok diyagramı Şekil 4.6'de verilmiştir. İşlemci dizisindeki her bir işlemci (2.25)'deki bir iterasyona karşılık gelmektedir. Giriş görüntüsü u_{ij} zamanla değişmediğinden dolayı her iterasyonda bu_{ij} çarpımı sabit kalacaktır. Bu nedenle çarpım bir kere hesaplanır ve tüm işlemcilere kontrol sinyallerine senkron şekilde aktarılır. İlk değer üreteci bloğu, ilk işlemcinin hesaplayacağı ilk iterasyon sonucu için HSA'nın durumlarına ait ilk değerleri üretir. İlk değerlere giriş görüntüsünün kendisi, sıfır veya sabit bir değer verilebilir. İşlemci dizisini oluşturan işlemcilerin detayları HSA işlemcisi başlığında verilmiştir.

4.2 HSA İşlemci Dizisi

HSA işlemcisi bir biri ardına dizilerek HSA işlemci dizisini ve sonuç olarak ayrık HSA'nın bir gerçeklemesini oluşturur. Uç bağlantıları ve iç yapısı Şekil 4.7'de verilen HSA işlemcisi, sınır koşulu üretici, veri belleği, sabit belleği, katsayı belleği, yerel kontrol ve aritmetik işlem bloklarına sahiptir.

Sınır koşulu üreteci bloğu sınır koşullarını üretir. Veri belleği HSA durumlarına ait değer-



Şekil 4.7 HSA işlemcisinin uç bağlantıları ve iç yapısı.

leri, sabit belleği işlemci dizisinden önce hesaplanan ve sonra bütün işlemcilere aktarılan bu_{ij} değerlerini ve katsayı belleği ise HSA'na ait şablon değerlerini saklarlar. Veri belleğinde saklanan bir önceki iterasyona ait çıkışlar aritmetik işlem bloğuna hem de bir çoğullayıcıyla direk diğer bloğa aktarılır. Sabit belleğinde saklanan sabit değerler gerekli işlemler için aritmetik işlem bloğuna ve diğer işlemciye iletilmek için sabit çıkışana bağlıdır. Yerel kontrol bloğu, kontrol sinyallerini kullanarak işlemciye ait blokların tüm kontrol sinyallerini üretir. Ayrıca *hframe* ve *vframe* sinyallerini işlemci gecikmesi kadar geciktirerek veri ile senkronizasyonunu sağlar. Gabor işlemcisine ait uç bağlantılarının türleri ve açıklamaları Çizelge 4.1'de verilmiştir. Blokların detayları kendi başlıkları altında verilmiştir.

4.2.1 Bellek Organizasyonu

Bu alt bölümde HSA işlemcisine ait veri belleği, sabit belleği ve katsayı belleğine ait tasarım detayları bulunmaktadır. $C(i,j)$ hücresine ait $n+1$. iterasyon sonucunun hesaplanabilmesi için $x_{i-1\,j}(n)$, $x_{i+1\,j}(n)$, $x_{ij-1}(n)$, $x_{ij+1}(n)$ değerlerine ve bu_{ij} değerine ihtiyaç vardır. Seri olarak gelen video verisinden gerekli hücre değerlerinin elde edilmesi Şekil 4.8'deki bellek organizasyonunda verilmiştir. Bu indislere ait bir önceki iterasyona ait durumların değerlerini kullanabilmek için, $K \times L$ boyutundaki ağ için $2 \times L + 3$ tane durum değerini bellekte saklamak gereklidir. Bellekte saklanması gereken değerler Şekil 4.8'de gri olarak gösterilmiştir. Siyah piksel o andaki işlem sonucunun hesaplanacağı pikseli, gri pikseller işlem yapılan bölgeyi, kırmızı olan pikseller işlenmiş, yeşil renkte olanlar ise hafızada saklanan ve işlenecek piksel değerlerini gösterir. HSA işlemcisinde

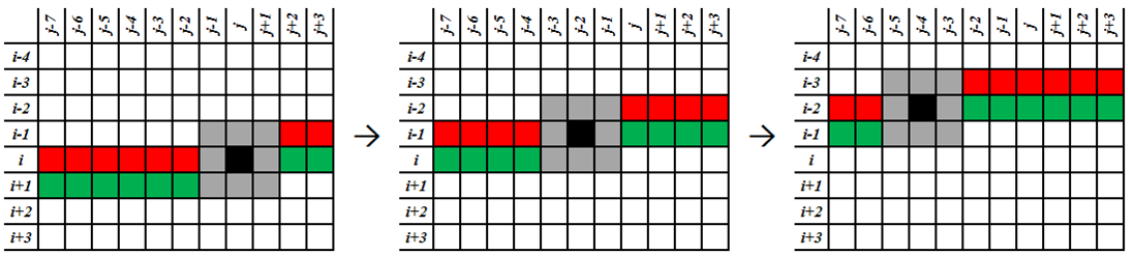
Çizelge 4.1 HSA işlemcisine ait uç bağlantıları ve açıklamaları

Sinyal Adı	G/Ç	Türü	Açıklama
data_in	Giriş	Vektör	Önceki işlemcinin hesapladığı durumlara ait değerler.
const_in	Giriş	Vektör	Önceki işlemciden gelen bu_{ij} değerleri.
data_out	Çıkış	Vektör	İşlemcinin hesapladığı durumlara ait değerler.
const_out	Çıkış	Vektör	Sonraki işlemciye giden bu_{ij} değerleri.
hframe_in	Giriş	Bit	Önceki işlemciden gelen kontrol sinyali
vframe_in	Giriş	Bit	Önceki işlemciden gelen kontrol sinyali
hframe_out	Çıkış	Bit	İşlemci gecikmesi kadar geciktirilip sonraki işlemciye iletilen kontrol sinyali
vframe_out	Çıkış	Bit	İşlemci gecikmesi kadar geciktirilip sonraki işlemciye iletilen kontrol sinyali
sdata_in	Giriş	Bit	Önceki işlemciden gelen seri programlama veri sinyali
sdata_out	Çıkış	Bit	Sonraki işlemciye giden seri programlama veri sinyali
clkp	Giriş	Bit	Piksel saat sinyali
clka	Giriş	Bit	Seri programlama için yardımcı saat sinyali (50 MHz)
clk	Giriş	Bit	İşlemci çalışma hızını belirleyen saat sinyali, piksel saat sinyalinin iki katı hızındadır.
reset	Giriş	Bit	Lojik '1' olduğunda işlemciyi resetler, çalışma durumunda lojik '0' olması gerekir.
ce	Giriş	Bit	Lojik '1' olduğunda işlemci çalışır, lojik '0' ise işlemci durur.

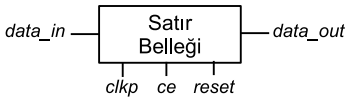
bulunan belleklerin görevleri şunlardır:

- Veri belleği, bir önceki işlemcinin hesapladığı durumlara ait değerleri saklar.
- Sabit belleği, işlemci dizisinden önce bir kere hesaplanan bu_{ij} değerini durum değişkenlerine senkron etmek için saklar.
- Katsayı belleği, aritmetik işlem bloğunun ihtiyaç duyduğu şablon değerlerini saklar.

HSA işlemcisinde $(i-1, j)$, $(i+1, j)$, $(i, j-1)$ ve $(i, j+1)$ indisli durumlara ve (i, j) indisli sabit değere ihtiyaç vardır. Bu nedenden dolayı $K \times L$ boyutundaki ağ için durum



Şekil 4.8 Bellek organizasyonu.

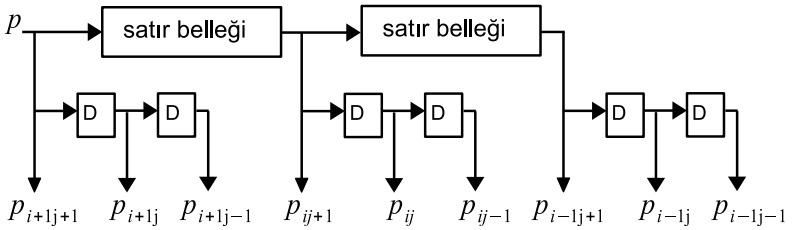


Şekil 4.9 Satır tampon belleği bloğu ve uç bağlantıları.

Çizelge 4.2 Satır tampon belleğine ait uç bağlantıları ve açıklamaları

Sinyal Adı	G/Ç	Türü	Açıklama
data_in	Giriş	Vektör	Bellekte saklanacak veri girişi.
data_out	Çıkış	Vektör	Bellekte saklanan veri çıkışı.
clkp	Giriş	Bit	Belleğin saat girişi, piksel saat sinyalinin hızındadır.
reset	Giriş	Bit	Lojik '1' olduğunda bellek adresini sıfırlar, çalışma durumunda lojik '0' olması gerekir.
ce	Giriş	Bit	Lojik '1' olduğunda bellek girişindeki değeri hafızaya kayıt ederken aynı adresteki değeri çıkışna verir, lojik '0' durumun da ise bellek çalışmasını durdurur.

belleği $2 \times L + 3$ boyutunda, sabit belleği ise L boyutundadır. Bir satıra ait değerlerin bellekte saklanabilmesi için görüntü işleme uygulamalarında kullanılan, ilk giren ilk çıkar (Fist In First Out, FIFO) şeklinde ve herhangi bir kontrole ihtiyaç duymadan çalışan satır tampon belleği kullanılmıştır. Sabitler için bir tane, veriler için ise 3×3 şablon boyutu için iki satır tampon belleği kullanılmıştır. Gerçeklenen satır tampon belleği blok diyagramı ve uç bağlantıları Şekil 4.9'de verilmiştir. Satır tampon belleğine ait uç bağlantılarını türleri ve açıklamaları Çizelge 4.2'de verilmiştir. Satır tampon belleği, FPGA tümdevresi içinde bulunan blok RAM'ler kullanılarak gerçeklenmiştir. Blok RAM'ler yazma ve okuma olmak üzere iki portlu çalışacak şekilde ayarlanmış ve yazma ve okuma portlarına ait adres uçları bir satırdaki toplam piksel sayısına kadar sayan bir sayıcıya bağlanmıştır. İlk olarak sayıcının gösterdiği adresteki veri okunur ve çıkışa aktarılır, sonra aynı adrese girişten gelen yeni veri yazılır ve sayıcı değeri bir arttırılır. Bu işlemler tek saat darbesinde gerçekleştirilir. Okuma ve yazma işlemleri *ce* girişi lojik '1' olduğu sürece devam eder. HSA işlemcisinde kullanılan *hframe* kontrol sinyali bu girişe bağlanarak geçerli veri olduğu sürece satır tampon belleğinin çalışması sağlanır. Satır tampon belleğinin boyu ve bit genişliği tasarım aşamasında (sentezleme öncesi) VHDL dilinin jenerikleri kullanılarak değiştirilebilmektedir.



Şekil 4.10 İşlemci bloğuna ait girişlerin video sinyalinden satır–ara–belleği ile elde edilmesi

Gabor benzeri HSA filtresinin her bir hücresine ait çıkışın hesaplanabilmesi için gerekli girişler Şekil 3.2'de verilmiştir. Bu girişlerin Gabor işlemcisinin çarp ve topla bloğunun girişi olarak elde etmek için durumları 2 tane satır belleği ve 6 yazmaca, sabitin ise bir tane satır belleği gerekir. Şekil 4.10'da verilen tasarımda birinci satır belleğinin girişi, çıkışı ve ikinci satır belleğinin çıkışına ikişer tane yazmaç bağlanmıştır. Bu yazmaçların çıkışlarında ağın (i, j). durum değişkenine ait etki alanına ait değerler vardır. Ayrıca $bu_{i,j}$ sabit değerinin senkronizasyonu için bir tane satır belleği bulunur.

Şekil 4.11'de satır tampon belleğinin giriş, çıkış ve sakladığı verilerin saat darbeleri ile değişimi verilmiştir. İşlem sıra numarası reset durumundan sonraki saat darbelerini sayısını göstermektedir. İlk durumda blok RAM adresi 1 ve P_{11} değeri satır tampon belleğinin girişindedir. İlk saat darbesi ile 1 adresindeki X verisi çıkışa aktarılır ve aynı adrese P_{11} değeri yazılır ve sonraki saat darbelerin de adres arttırılır, veri çıkışa aktarılır ve girişteki veri yeni adrese yazılır. Satır sonunda ise hafızadaki veri çıkışa aktarılır, o satıra ait son veri son adrese yazılır ve blok RAM adresi 1 değerine getirilir. İkinci satır geldiğinde bu işlemler tekrar devam eder. Buradaki en büyük sorun blok RAM'in içinde eski video çerçevesine ait bilginin kalmasıdır, bu sorun videonun karanlık bölgesinde gelmeye devam eden *hframe* kontrol sinyali sayesinde blok RAM içeriği sıfırlanarak çözülür.

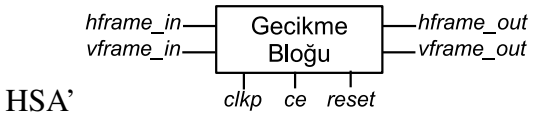
Bir işlemcide kullanılan RAM miktarın için sayısal bir örnek vermek gerekirse 1080×1920 çözünürlüklü bir görüntüde her satır için 1920 pikselin sığabileceği 2048 $(2 \times 1K)$ uzunluklu RAM rezerve edilmelidir. Örnek olarak durumların gerçel ve sanal kısımları 12 bit, sabitlerin ise 24 bit uzunluklu olduğu kabul edilirse: sabit için yalnızca bir tane satır tampon belleği yeterliyken durumlar için iki tane satır tampon belleği gerekir. Dolayısıyla gereken RAM miktarı 2048x12x2x2 + 2048x24x1 = 18 Kbayt olur.

İşlem Sıra No	Blok RAM Adresi	Satır Belleği Girişi	Blok RAM'in Son Durumu									Satır Belleği Çıkışı
			B1	B2	B3	B4	B5	B6	B7	···	BL	
1	1	P11	P11	X	X	X	X	X	X	···	X	X
2	2	P12	P11	P12	X	X	X	X	X	···	X	X
3	3	P13	P11	P12	P13	X	X	X	X	···	X	X
4	4	P14	P11	P12	P13	P14	X	X	X	···	X	X
5	5	P15	P11	P12	P13	P14	P15	X	X	···	X	X
6	6	P16	P11	P12	P13	P14	P15	P16	X	···	X	X
7	7	P17	P11	P12	P13	P14	P15	P16	P17	···	X	X
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮	
L	L	P1L	P11	P12	P13	P14	P15	P16	P17	···	P1L	X
L+1	1	P21	P21	P12	P13	P14	P15	P16	P17	···	P1M	P11
L+2	2	P22	P21	P22	P13	P14	P15	P16	P17	···	P1M	P12
L+3	3	P23	P21	P22	P23	P14	P15	P16	P17	···	P1M	P13
L+4	4	P24	P21	P22	P23	P24	P15	P16	P17	···	P1M	P14
L+5	5	P25	P21	P22	P23	P24	P25	P16	P17	···	P1M	P15
L+6	6	P26	P21	P22	P23	P24	P25	P26	P17	···	P1M	P16
L+7	7	P27	P21	P22	P23	P24	P25	P26	P27	···	P1M	P17
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮	
2L	L	P2L	P21	P22	P23	P24	P25	P26	P27	···	P2M	P1L

Şekil 4.11 Satır tampon belleğinin giriş, çıkış ve iç verilerinin saat darbeleri ile değişimi.

4.2.2 Gecikme Bloğu

Video giriş bloğundan gelen görünen bölgeyi belirleyen *hframe* ve *vframe* kontrol sinyallerinin işlenen veriyle senkron olarak bir sonraki işlemci bloğuna aktarılabilmesi için, bu sinyallerin geciktirilmesi gerekmektedir. Gabor işlemcisinin toplam gecikmesi $\times L+3+3$ tür. Bu nedenle *hframe* ve *vframe* kontrol sinyalleri verinin gecikmesi olan 2 satır, 6 piksel kadar geciktirilmesi gerekir. Piksel frekansında çalışan bu blok sentezleme aşamasında girilen jenerik değerlere göre *hframe* ve *vframe* sinyallerini Gabor işlemcisinin toplam gecikmesi kadar geciktirmektedir. Tasarım aşamasında bloğun kaç satır ve kaç piksel gecikme yapacağı bilgisi tasarımcı tarafından girilmelidir. Bloğun tek sınırlaması en az 3 piksel süresi kadar gecikme yapabilmesidir. Bunun dışında istenen satır ve piksel süresi kadar gecikme yapabilmektedir. Gecikme bloğunun tasarımı *hframe* kontrol sinyalinin yapısından dolayı, sadece iki sayıcı ile tasarlanmıştır. İlk sayıcı piksel gecikmesini, ikinci sayıcı ise satır gecikmesini saymaktadır. Bloğun uç bağlantılarının gösterildiği blok diyagramı Şekil 4.12'de verilmiştir. Gecikme bloğuna ait uç bağlantılarını türleri ve açıklamaları Çizelge 4.3'de verilmiştir.



Şekil 4.12 Kontrol şinyallerinin işlenen veriler ile senkronizasyonunu sağlayan gecikme bloğunun uç diyagramı

Çizelge 4.3 Gecikme bloğuna ait uç bağlantıları ve açıklamaları

Sinyal Adı	G/Ç	Türü	Açıklama
hframe_in	Giriş	Bit	Geciktirilecek kontrol sinyali.
vframe_in	Giriş	Bit	Geciktirilecek kontrol sinyali.
hframe_out	Çıkış	Bit	Geciktirimiş kontrol sinyali.
vframe_out	Çıkış	Bit	Geciktirimiş kontrol sinyali.
clkp	Giriş	Bit	Gecikme bloğunun saat girişi, piksel saat sinyalinin hızındadır.
reset	Giriş	Bit	Lojik '1' olduğunda sayıcılar sıfırlanır, çalışma durumunda lojik '0' olması gerekir.
ce	Giriş	Bit	Lojik '1' normal çalışmasına devam eder, lojik '0' durumun da ise bellek çalışmasını durdurur.

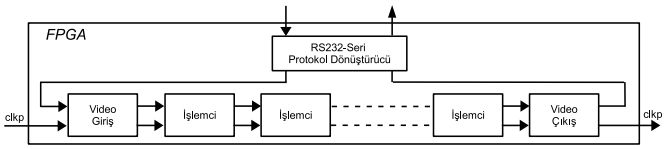
4.3 Diğer Bloklar

Gerçek zamanlı HSA'nın tasarımı ve gerçeklemesi sırasında tasarlanan ve kullanılan diğer bloklar bu bölümde verilmiştir. Çalışma anında ağın ve sistemin parametrelerinin değiştirilmesi için kullanılan seri programlama bloğu, ağ çıkışının karşıtlığını ayarlamakta kullanın histogram hesaplama ve karşıtlık ayarlama blokları ve tasarımı yapılan HSA işlemci ve dizisinin PC ortamında test edilmesini sağlayan PC test ortamıdır.

4.3.1 Seri Programlama Arayüzü

Şablon değerleri, sınır koşulları, vb. işlemci katsayıları ve özelliklerinin çalışma esnasında değiştirilebilmesi ve okunması için seri bir programlanma arayüzü tasarlanmıştır. *hframe* ve *vframe* kontrol sinyallerinden esinlenerek Şekil 4.13'teki haberleşme hattı topolojisi önerilmiştir. Her işlemci seri protokol ile kendisine ulaşan veriyi alır, veriyi bir saat sinyali geciktirerek (tamponlayarak) bir sonraki işlemciye iletir, kendisi ile ilgiliyse veriyi çözümler ve gerekiyorsa bir sonraki bloğa aktarırken veri paketinin üzerinde değişiklik yapar.

Seri paketin ilk işlemciye girmesi ile son işlemciden çıkması arasında işlemci sayısı ka-



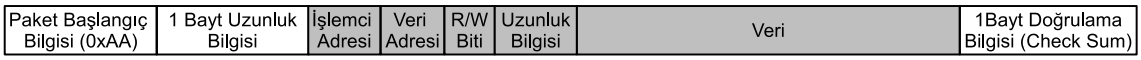
Şekil 4.13 Seri haberleşme topolojisi

dar saat çevrimi gecikme vardır. Bu gecikme ilk bakışta haberleşme hızını azaltacak gibi görünsede seri haberleşmenin çalışma hızından (50 MHz) dolayı herhangi bir sorun oluşturmamaktadır. Hızı daha da somutlaştırmak gerekirse RS232 portundan 115200 Baud hızında bir bitlik bir bilgi gelene kadar seri haberleşme sinyali 347. işlemciye ulaşmış olur.

Şekil 4.13'teki RS232–Seri protokol dönüştürücü bloğunun görevi ise dış dünyadan gönderilen verileri seri protokole, içerden gelen seri protokoldeki verileri ise tekrar RS232'ye dönüştürmektir. PC tarafından gönderilen paket yapısı Şekil 4.14'de verilmiştir. Başla ve bitir bitleri hariç seri protokolüne ait paket ortada gri renkte gösterilmiştir. Dış paketin başlangıç bilgisi $0xAA$ baytıdır. Başla baytını seri protokoldeki toplam uzunluk bilgisinin 8 bite genişletilmiş bir kopyası takip eder. Tekrarın nedeni paketin çözülmesinin basitleştirmektir. Paketin sonunda ise bir baytlık bir doğrulama bilgisi yer alır. Hesabının basit olması için doğrulama bilgisi tüm paket baytlarının eldesiz toplamı olarak seçilmiştir.

RS232 arayüzünden (PC'den) gelen verinin doğrulaması yapıldıktan sonra bu bilgi Şekil 4.13'te gösterildiği üzere video girişine, oradan ilk işlemciye, ikincisine, vb. şekilde video çıkışına kadar yayılır. Yol üzerinde paket adresi ile kendi adresi eşleşen işlemciler paketi değerlendirir. Eğer yapılacak olan işlem işlemciye yazmak ise ilgili işlemci yazma işlemini gerçekleştirir ve mesajı aynen iletir. Mesaj hat boyunca değişmeden tekrar RS232 arayüzüne ulaşır, buradan da tekrar paketlenerek PC'ye gönderilir. Verinin değiştirilmemesinin avantajı, Gabor işlemcileri gibi aynı katsayılara ve konfigürasyona sahip olan işlemcilere aynı işlemci adresinin atanarak gönderilmesi gereken mesaj sayısının azaltılabilmesidir. Aynı mesaj, son olarak PC'e ulaşır ve PC gönderdiği/aldığı verileri eşleyerek verinin doğru iletilmiş olduğunu onaylar.

Eğer yapılacak olan işlem okuma işlemi ise PC seri protokol mesajının veri kısmına keyfi bir değer atayarak FPGA'e iletir. FPGA içinde ilerleyen mesaj ilgili işlemciye ulaştığında



Şekil 4.14 RS232 haberleşem arayüzünün protokolü

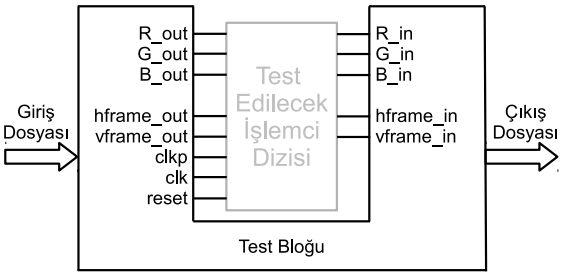
işlemci mesajın veri alanına gerçek veriyi yerleştirerek mesajı bir sonraki bloğa iletir. Yani ilgili bloktan önce rastgele veriye sahip olan mesaj, sonrasında doğru veri ile yüklenmiş olarak yoluna devam eder. RS232 arayüzü yeni mesajı paketleyerek PC'e aktarır. PC okunan paketin doğrulama bilgisini sınadıktan sonra okunan verinin doğru olduğuna kanaat getirir.

Eğer PC'den FPGA'e ulaşan bilgide hata söz konusu olursa RS232–Seri protokol arayüzü çelişen bir doğrulama bilgisi hesaplar. Hata algılayan sistem seri protokol mesajının işlemci adresi kısmını onaltı tabanında $0xFFF$ değeri ile değiştirerek işlemcilere gönderir. $0xFFF$ işlemci adresi rezervedir ve işlemcilere "Dikkat! Hatalı bir mesaj geldi. Dikkate almayın." anlamını taşır. Dolayısıyla hiçbir bloğa bu adres verilmemelidir. Hat boyunca RS232 arayüzüne ulaşan mesaj tekrar paketlenerek PC'ye iletilir. PC okuduğu mesajın işlemci adres kısmının $0xFFF$ olduğunu algıladığında son gönderdiği mesajın FPGA tarafından değerlendirilmediğini algılayarak ilgili mesajı tekrar göndermelidir. Eğer FPGA'den PC'e gönderilen veri bozulursa PC doğrulama yapamaz ve dolayısıyla okuduğu bilgide hata olduğunu algılar. PC'nin vereceği en doğru karar son mesajı tekrar göndermektir, mesajın yazma veya okuma olması bu kararı etkilemez ve herhangi bir probleme yol açmaz.

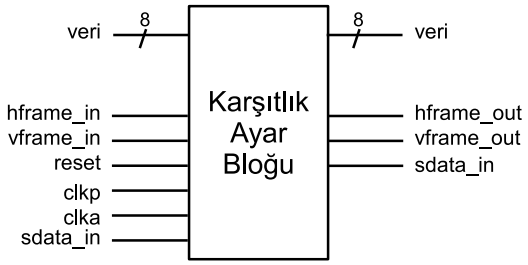
RS232–Seri protokol arayüzü farklı hızlarda çalışan iki sistemin haberleşmesi için gerekli iki tane tampon bellek, RS232 ve seri protokol standartlarını gercekleyen kontrol bloğudan oluşmaktadır. Eğer PC ile FPGA'in ya da başka bir cihazla FPGA'in haberleşmesi daha hızlı olması istenirse UART yerine farklı bir haberleşme arayüzü konularak daha hızlı bir haberleşmeye geçilebilir.

4.3.2 PC Test Bloğu

Şekil 4.15 ile verilen test bloğu teknik olarak bir donanım olmasa da bu blok kuşkusuz projenin en önemli yan bloğudur. FPGA üzerinde çalışacak donanımın tasarımı ve VHDL



Şekil 4.15 PC test bloğunun uç bağlantıları



Şekil 4.16 Karşıtlık ayarlaşam bloğu uç bağlantıları

diliyle yazılmasından sonraki ilk aşama VHDL kodunun Modelsim ile test edilmesidir. Görevleri Matlab tarafından oluşturulan görüntülerin okunması, video giriş bloğunu taklit ederek okunan görüntünün *hframe* ve *vframe* sinyalleriyle beraber video işlemci dizisine verilmesi, işlemci dizisinin sonucunun alınması ve sonucun bir çıkış dosyasına yazılmasıdır. Herhangi bir bloğun çalışma onayı alabilmesi için bu test bloğu ile test edilmesi ve test çıkışıyla Matlab sonucunun en önemsiz bitlerine kadar bire bir aynı olmaları gerekmektedir.

4.3.3 Histogram Hesaplama ve Karşıtlık Ayarlama Bloğu

HSA'nın çıkışına insan gözü tarafından daha iyi algılanabilmesi için tasarımı ve gerçeklemesi yapılmıştır. Seri olarak programlanabilen ve dolayısıyla kendi işlemci adresine sahip olan bu bloğun görevleri video çerçevesinin histogramını çıkartmak ve parametrelerine göre görüntünün karşıtlığını ayarlamaktır. Bloğun uç bağlantıları Şekil 4.16'da, yaptıkları görevler ise Çizelge 4.4'de verilmiştir.

Karşıtlık ayarlamak için girişteki minimum piksel değeri 0'a, maksimum piksel değeri ise 255'e denk gelecek şekilde doğrusal bir aralık açma yöntemi kullanılırken minimumdan küçük değerler 0, maksimumdan büyük değerler ise 255'e atanır. Bloğun çalışma

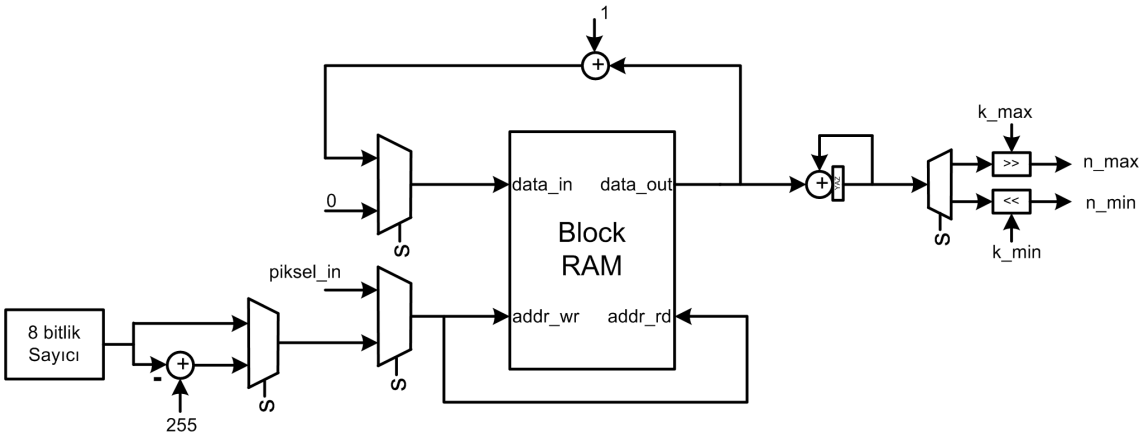
Çizelge 4.4 Karşıtlık ayarlama bloğuna ait uç bağlantıları ve açıklamaları

Sinyal Adı	G/Ç	Türü	Açıklama
veri	Giriş	Vektör	Piksek parlaklık değeri.
veri	Çıkış	Vektör	Karşıtlık ayarlaması yapılmış piksel parlaklık değeri.
hframe_in	Giriş	Bit	Histogram ve karşıtlık hesaplamalama bloklarını kotrol eder.
vframe_in	Giriş	Bit	Histogram ve karşıtlık hesaplamalama bloklarını kotrol eder.
hframe_out	Çıkış	Bit	Kontrol sinyalinin blok kadar geciktirilmiş hali.
vframe_out	Çıkış	Bit	Kontrol sinyalinin blok kadar geciktirilmiş hali.
clkp	Giriş	Bit	Bloğunun saat girişi, piksel saat sinyalinin hızındadır.
clka	Giriş	Bit	Seri haberleşme için yardıöcı saat sinyali.
sdata_in	Giriş	Bit	Seri haberleşme veri girişi.
sdata_out	Çıkış	Bit	Seri haberleşme veri çıkışı.
reset	Giriş	Bit	Lojik '1' olduğunda sayıcılar sıfırlanır, çalışma durumunda lojik '0' olması gerekir.

esnasında programlanabilen iki çalışma modundan birinde sabit minimum ve maksimum piksel değerleri kullanılırken diğer modda bu değerler histogramdan otomatik olarak hesaplanır. Sabit minimum/maksimum modu seçildiğinde minimum ve maksimum değerleri de çalışma esnasında programlanabilir.

Histogram hesaplandıktan sonra karşıtlık ayarlaması;

- 0 parlaklık değerlerinden başlayarak piksel sayıları kümülatif toplamı hesaplanır,
- Bu toplam sonucu görüntüdeki toplam piksel sayısının %10'unu geçtiği andaki piksel parlaklık değeri bulunur (k_{min}),
- $2^8 - 1 = 255$ (8 bit ile ifade edilen görüntüler için) parlaklık değerinden başlayarak geriye doğru piksel sayılarının kümülatif toplam hesaplanır,
- Bu toplam sonucu görüntüdeki toplam piksel sayısının %10'unu geçtiği andaki piksel parlaklık değeri bulunur (k_{max}).
- Bu değerler hesaplandıktan sonra eski görüntüdeki (I) piksel parlaklık değerlerinden yeni görüntüdeki (I^N) piksel parlaklık değerleri (4.1) denklemindeki gibi hesaplanır.

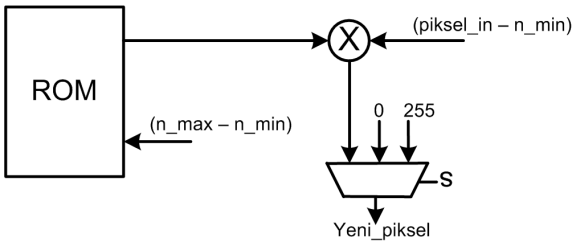


Şekil 4.17 Histogram hesaplama bloğunun sayısal devresi.

$$I_{out} = \begin{cases} 0, & I_{in} < k_{min} \\ 255 * \dfrac{I_{in} - k_{min}}{k_{max} - k_{min}}, & k_{min} < I_{in} < k_{max} \\ 255, & I_{in} > k_{max} \end{cases} \tag{4.1}$$

Histogram ve karşıtlık iyileştirme işlemlerinin matematik ve algoritmasından sonra bu bloğun sayısal devre olarak gerçeklenmesi verilecektir. Blok temel olarak iki kısımdan oluşmaktadır. Birinci kısım görüntünün histogramını hesaplar, ikinci kısım ise hesaplanan histogramdan yararlanarak görüntünün karşıtlığını ayarlar. Bu blok video üzerinde çalıştığı için görüntünün karşıtlığının ayarlanması için gereken histogram bir önceki video çerçevesinden elde edilmektedir.

Şekil 4.17'de histogram hesaplama bloğunun blok diyagramı verilmiştir İki portlu RAM'in adres girişlerine görüntüden gelen piksel parlaklık değeri, veri çıkışı portu ise toplayıcı üzerinden veri girişi portuna bağlanmıştır. RAM okuma öncelikli olarak çalışmaktadır. Bu sayede tek bir saat darbesinde aynı adresten verinin okunması, okunan değere 1 eklenmesi ve sonucun tekrar aynı adrese yazılması işlemleri yapılabilmektedir. Histogram hesaplama bloğunun çalışmaya başlayacağı zaman, hesaplamanın biteceği zaman ve çoğullayıcılar*hframe*ve*vframe*kontrol sinyalleri tarafından kontrol edilirler. Veri girişindeki çoğullayıcının görevi histogram hesaplaması bittiğinde RAM'in içeriğinin sıfırlanması için veri girişine sıfır değerinin ya da histogram hesaplaması sırasında veri çıkışından gelen değerin yönlendirilmesidir. Adres girişlerindeki çoğullayıcının görevi ise histogram hesaplaması bittiğinde RAM'in içeriğinin sıfırlanabilmesi için gerekli adreslerin yönlen-



Şekil 4.18 Karşıtlık iyileştirme hesaplamasını gerçekleyen sayısal devre

dirilmesi, histogram hesaplanırken görüntü piksel parlaklık değerinin yönlendirilmesi ve son olarak da görüntünün karşıtlığının iyileştirilmesinde kullanılacak minimum ve maksimum değerlerin bulunması sırasında sayıcıdan gelen adres değerlerinin yönlendirilmesidir.

Sayıcı, toplayıcı ve çoğullayıcıdan oluşan bloğun görevi RAM'in içeriğinin temizlenmesi, minimum ve maksimum değerlerin histogram üzerinden hesaplanabilmesi için gerekli adreslerin üretilmesidir. RAM'in veri çıkışında bulunan bloğun görevi ise histogram hesaplaması bittikten sonra veri çıkışındaki değerlerin kümülatif toplamını bularak belli bir değer ile karşılatırmasını yapmaktır. Bu sayede minimum ve maksimum değerler bulunur.

Şekil 4.18'de verilen sayısal devre karşıtlık ayarlama matematiğini gerçeklemektedir. Sabit bir sayının değişken bir sayıya bölme işleminin sayısal devrelerle gerçeklemesi zor olduğundan 255/x işlemi önceden hesaplanarak çalışma anında değiştirilemeyen ROM içine konulmuştur. Bu bölme işleminin sonucu ROM içinden okunur, bu tasarım yöntemine taramalı tablo (look-up table) denir. ROM adres girişine $k_{max} - k_{min}$ değeri bağlanmıştır, bu sayede $\frac{255}{k_{max}-k_{min}}$ sonucu veri çıkışından alınır ve çarpma işlemi sonucu çıkış pikseli elde edilir.

4.4 Sonuç

Bu bölümde, TÜBİTAK 108E023 numaralı proje kapsamında tasarımı yapılan bir gerçek zamanlı HSA gerçeklemesinin kontrol mantığı, giriş çıkış blokları, bellek organizasyonu, işlemci bloğu, gecikme bloğu, seri programlama arayüzü, histogram hesaplama ve karşıtlık ayarlama bloğu ile donanım gerçeklemesi olmayan PC test ortamının detayları verilmiştir.

Kayaer tarafından [20] ve [32]'de önerilen HSA gerçeklemesindeki her iterasyonun ayrı işlemci tarafından zamanda bölünerek gerçeklenmesi mantığı kullanılmıştır. [20]'de bulunan merkezi kontrol yerine yerel kontrol mantığı kullanılmıştır. Yerel kontrol sinyallerini kolay gerçeklenmesi ve kontrolünün kolaylığı için yeni bir tasarım önerilmiş ve detayları verilmiştir. Yerel kontrolün kullanıldığı bu gerçeklemede [20]'deki iş hattı mantığı ile tasarlanmış ve gerçeklenmiştir.

Video kaynağından gelen kontrol sinyalleri görüntünün görünen bölgesinin daha rahat işaretlenebilmesi için, görünür bölgede değerleri lojik '1' olan $hframe$ ve $vframe$ kontrol sinyallerinin tasarımı yapılmış ve kullanılmıştır. Her bloğun girişinde ve çıkışında bu sinyaller bulunmaktadır. $hframe$ ve $vframe$ kontrol sinyalleri gerçeklemedeki her bloğun çalışmasını, durmasını ve ilk değerlerine dönmesini kontrol etmektedir. Bu sayede merkezi kontrolün getirdiği zorlukların üstesinden gelinmiştir.

İşlemcinin bellek organizasyonu satır tampon bellekleriyle yapılmıştır. Önceki tasarımda 3×3 şablon boyutlarına sahip HSA için 3 satıra ait değerler saklanırken yeni tasarımda 2 satır yeterli olmaktadır. Bu sayede tasarımın en kritik noktası olan bellek ihtiyacından $1/3$ oranında tasarruf edilmiştir. Tasarımı ve gerçeklemesi yapılan satır tampon bellekleri kendi yazma ve okuma adreslerini ürettikleri için sadece satır başı ve sonunu gösteren bir kontrol sinyali ile kontrol edilebilirler. Durumlar ve sabitlerin kontrol sinyalleri ile senkronizasyonun bozulmaması için, işlemcinin toplam gecikmesi kadar kontrol sinyallerinin de gecitirilmesi gerekmektedir. Bu nedenle kontrol sinyallerini istenen değer kadar geciktiren gecikme bloğu tasarımı yapılmıştır.

HSA'nın şablon değerleri, ilk koşulların değerleri, sınır koşulları ve diğer blokların çalışma parametreleri çalışma anında değiştirilebilmesi için seri bir programlama arayüzü tasarımı yapılmıştır. Bu arayüz sayesinde sistemdeki herhangi bir değişiklik için tekrar sentezleme gibi vakit kaybına yol açacak işin yapılmasına gerek kalmamıştır. Aynı zaman da seri programlama arayüzü ile çalışma anında sistemin durumu izlenerek herhangi bir sorun olup olmadığına bakılabilmektedir. Bu sayede sistem sadece PC ortamın da değil donanım üzerinde de test edibilmektedir.

HSA'nın çıkışı karşıtlığı insan gözünün daha rahat alguılayabilmesi için karşıtlık ayarlama (Contrast Streching) yöntemi kullanılarak iyileştirme yapılmıştır.

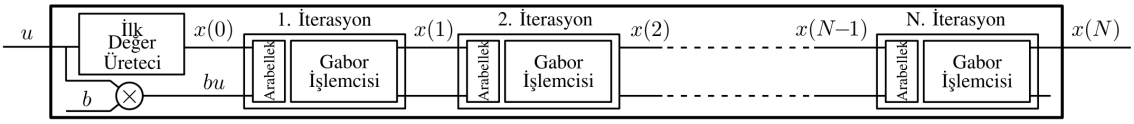
BÖLÜM 5

GABOR BENZERİ HSA FİLTRELERİNİN GERÇEKLENMESİ VE PROTOTİP SİSTEMLER

İki boyutlu Gabor filtreleri doku tanıma ve sınıflama [3], el yazısı tanıma [4], doku ayrımı [5], kenar belirleme [6], görüntü sıkıştırma [7], hareket kestirimi [8], nesne tanıma [9] ve şekil tanıma [10] gibi görüntü işleme uygulamalarında yaygın olarak kullanılmaktadır. Bu uygulamalar günümüzün gelişmiş bilgisayarlarına rağmen yüksek çözünürlüklü (YÇ) hareketli video görüntülerini işleyecek hızlara çıkamazlar. YÇ video görüntülerini işleyebilmek için yazılımsal çözümler yerine donanımsal çözümlerin geliştirilmesi gerekmiştir. Donanımsal çözümler de temel olarak iki temel yöntem kullanılmıştır. Bunlar FIR [14, 15, 16, 17] ve HSA [18] tabanlıdır. Bu gerçeklemelerden [14] GPU, [17] ASIC ve [15, 16, 18] ise FPGA üzerinde geliştirilmişlerdir.

Önceki bölümde verilen gerçek zamanlı HSA gerçeklemesindeki video giriş/çıkış blokları, bellek organizasyonu, işlemci dizileri, yerel kontrol mantığı, işlemci bloğu, gecikme bloğu, seri programlama bloğu, histogram hesaplama ve karşıtlık ayarlama bloğu Gabor benzeri HSA filtresi gerçeklemesinde de kullanılmışlardır. Gabor benzeri HSA filtresinin gerçeklemesi sırasında HSA işlemcisinin aritmetik işlem bloğu, Gabor benzeri HSA filtresini hesaplayacak şekilde değiştirilmiştir.

Bu bölümde sırasıyla Gerçek Zamanlı HSA Gabor Benzeri Filtrelerin blok diyagramı, Gabor benzeri HSA filtresinin işlemci dizisi, aritmetik işlem bloğunun detayları, Gabor benzeri HSA filtresinin kaynak kullanmı, prototip sistemler ve diğer sistemlerle karşılaştırmalar verilmiştir.



Şekil 5.1 Gabor benzeri HSA filtresi işlemci dizisinin basitleştirilmiş blok diyagramı. Tamamen iş hattı formunda tasarlanan her işlemci bir iterasyona karşılık gelemektedir.

5.1 Gerçek Zamanlı Gabor Benzeri HSA Filtresinin Blok Diyagramı ve Mimarisi

Gerçek zamanlı sistemin basitleştirilmiş blok diyagramı Şekil 4.1'de verilmiştir. Video kaynağından gelen sıralı görüntü Gabor benzeri HSA filtresinin üzerinde gerçeklendiği FPGA tümdevresinde işlendikten sonra, monitör/TV ye aktarılmaktadır.

Şekil 4.1'de verilen sistemde FPGA tümdevresi içinde gerçeklenen gerçek zamanlı sistemin blok diyagramı Şekil 4.5'de verilmişti. HSA işlemci dizisi yerine Gabor benzeri HSA filtresinin işlemci dizisi konularak gerçek zamanlı Gabor filtresi elde edilmiştir. HSA işlemcisiyle Gabor işlemcisi arasındaki fark, HSA işlemcisi sadece tek katmanlı (gerçel) hesaplamaları yaparken, Gabor işlemcisi iki katmanlı (karmaşık değerli) hesaplamaları yapabilmektedir. Burada gerçeklenen Gabor işlemcisi tüm karmaşık değerli hesaplamları değil, sadece Gabor şablonlarına özel karmaşık değerli hesaplamaları yapar.

Gabor benzeri HSA filtrelerini gerçekleyen işlemci dizisinin basitleştirilmiş blok diyagramı Şekil 5.1'de verilmiştir. İşlemci dizisindeki her bir işlemci (2.25)'deki bir iterasyona karşılık gelmektedir. Giriş görüntüsü u_{ij} zamanla değişmediğinden dolayı her iterasyonda bu_{ij} çarpımı sabit kalacaktır. Bu nedenle çarpım bir kere hesaplanır ve tüm işlemcilere kontrol sinyallerine senkron şekilde aktarılır. İlk değer üreteci bloğu, ilk işlemcinin hesaplayacağı iterasyon için HSA'nın durumlarına ait ilk değerleri üretir. İlk değerler giriş görüntüsünün kendisi, sıfır veya sabit bir değer alınabilir. Gabor benzeri HSA filtresi işlemci dizisini oluşturan işlemcilerin detayları Gabor işlemcisi başlığında verilmiştir.

5.2 Gabor İşlemcisi

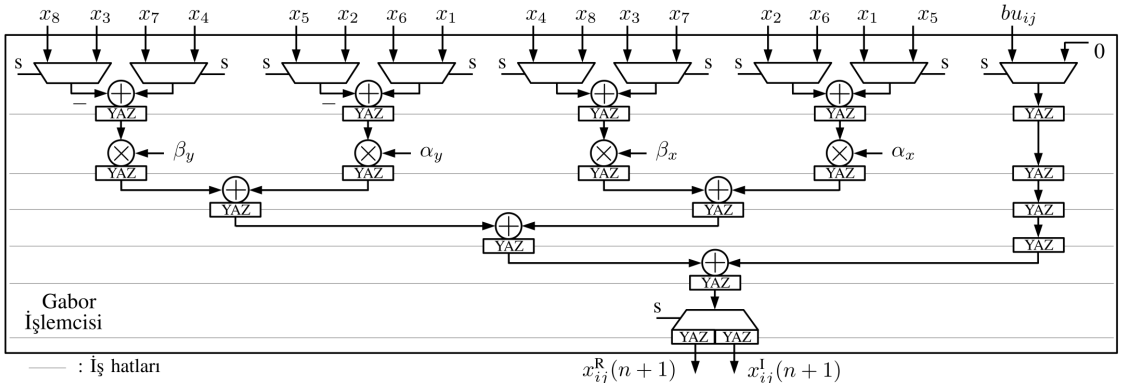
Gabor işlemcisi, aynı HSA işlemcisi gibi bir biri ardına dizilerek Gabor benzeri HSA filtresini oluşturur. Gabor işlemcisinin uç bağlantıları ve iç yapısı Şekil 4.7'de verilen HSA işlemcisi ile aynıdır. HSA işlemcisinin sınır koşulu üretici, veri belleği, sabit bel-

leği, katsayı belleği, yerel kontrol blokları Gabor işlemcisiyle aynıdır. Sadece aritmetik işlem bloğu Gabor benzeri HSA filtresini gerçekleyecek şekilde tasarlanıp, gerçeklenmiştir. Aritmetik işlem bloğu Gabor benzeri HSA filtresine ait Şekil 3.3'de verilen aritmetik işlemleri yapar.

5.2.1 Aritmetik İşlem Bloğu

Gabor benzeri HSA filtresinin işlemci bloğunun bir diğer parçası olan aritmetik işlem bloğunun görevi isminden de anlaşılacağı gibi toplama, çıkarma, çarpma işlemlerini yapmaktır. Her FPGA tümdevresinde üretimde tanımlanmış belirli miktarda çarpıcı (çarpma bloğu) bulunmasından dolayı çarpıcışar FPGA tümdevreleri için değerlidirler. Bu nedenle Gabor benzeri HSA filtresinin aritmetik bloğunun gerçeklemek için yarı sayıda çarpıcı kullanan Şekil 3.3'de verilen işaret akış diyagramı kullanılmıştır. Ayrıca durumların gerçel ve sanal kısımlarını hesaplayan işaret akış diyagramları birbirlerine benzedikleri için sadece birinin gerçeklenmiştir ve çarpıcı sayısı dörde düşürülmüştür. Bloğun çalışma hızı piksel saat hızının iki katıdır.

İş hattı mantığıyla tasarlanan aritmetik blokğun iç yapısı ve uç diyagramı Şekil 5.2'de verilmiştir. İlk saat darbesiyle çoğullayıcılar durumların gerçel kısımlarının hesaplanması için gerekli girişleri toplayıcılara yönlendirir ve toplama sonucu yazmaçlara kaydedilir. İkinci saat darbesiyle yazmaçlardaki sonuçlar katsayılarla çarpılıp, çarpıcı çıkışındaki yazmaçlara kaydedilir. Çarpma bloğu işlemini yaparken çoğullayıcı sanal kısmın hesaplanması için gerekli girişleri toplayıcılara yönlendirir. Üçüncü ve dördüncü saat darbeleriyle çarpma bloğunun çıkışlarındaki değerler iki girişli toplayıcılar ile toplanırlar. Gerçel kısmın hesaplanabilmesi için girişten gelen sabitin de toplam sonucuna eklenmesi gerekmektedir. Sabit girişinin ve 0 değerinin bağlı olduğu çoğullayıcının çıkışı gerçel ve sanal kısımların hesaplanmasına uygun olarak gerekli değerleri yönlendirir. Gerçel kısmın hesaplanmasından bir saat darbesi sonra aynı duruma ait sanal kısmın hesaplanması da biter. İşlemci giriş saat frekansının iki katı hızında çalıştığından dolayı her girişe karşılık bir çıkış üretilir. Bu nedenle her bir saat darbesinde işlemci bloğunun bütün toplayıcı ve çarpıcı elemanları çalışmaktadır.



Şekil 5.2 Gabor benzeri HSA filtresinin aritmetik işlem bloğu

5.2.2 Gabor benzeri HSA Filtresi İşlemcisinin Kaynak Kullanımı

Bu bölümde, çarpıcı ve RAM blokları sınırlı olan FPGA'lerin içine ne kadar Gabor işlemcisi sığabilir, sorusunun cevabı yanı sıra filtrenin toplam gecikmesi ve işlem hacmi konularındaki bilgiler verilmiştir.

$K \times L$ boyutlarındaki filtreyi gerçekleyecek bir işlemci durumlar için iki satır $(2 \times L)$, sabit terim için ise bir satır (L) RAM'e ihtiyaç duyar. FPGA'lerde sabit olarak bulunan blok RAM'lerin boyutu $1Kbayt = 1024Bit$ tir. Bu nedenle kullanılacak bellek elemanının miktarı sadece satır uzunluğuna değil, aynı zamanda da durum ve sabitlerin bit genişliklerine bağlıdır. Bir Gabor işlemcisinde kullanılacak bellek miktarı (5.2.2) denklemiyle hesaplanır.

$$\text{Bellek Miktarı} = 2\lceil\frac{L}{1024}\rceil\lceil\frac{BitG_{durumlar}}{8}\rceil + \lceil\frac{L}{1024}\rceil\lceil\frac{BitG_{sabit}}{8}\rceil \tag{5.1}$$

denklemindeki $BitG_{durumlar}$, $BitG_{sabit}$ ve L sırasıyla durumların toplam bit genişliğini, sabitin bit genişlğini ve ağın sütun sayısını gösterir, $\lceil\ \rceil$ ise büyük tamsayıya yuvarlama işlemidir. Bu hesaplamayla ilgili, üç farklı durum için işlemcinin ihtiyaç duyduğu bellek miktarına ilişkin üç örnek verilmiştir.

Örnek 1: Durumların bit genişlikleri gerçel ve sanal kısım için 12 bit, sabitin bit genişliği 16 bit olan 1080×1920 boyutlarındaki ağın

$$\text{Bellek Miktarı} = 2\lceil\frac{1920}{1024}\rceil\lceil\frac{24}{8}\rceil + \lceil\frac{1920}{1024}\rceil\lceil\frac{16}{8}\rceil = 16$$

tane $1Kbayt$lık blok RAM'e ihtiyacı vardır.

Örnek 2: Durumların bit genişlikleri gerçel ve sanal kısım için 16 bit, sabitin bit genişliği 24 bit olan 720×1280 boyutlarındaki ağın

$$\text{Bellek Miktarı} = 2\lceil\frac{1280}{1024}\rceil\lceil\frac{32}{8}\rceil + \lceil\frac{1280}{1024}\rceil\lceil\frac{24}{8}\rceil = 22$$

tane $1Kbayt$lık blok RAM'e ihtiyacı vardır.

Örnek 3: Durumların bit genişlikleri gerçel ve sanal kısım için 10 bit, sabitin bit genişliği 20 bit olan 480×640 boyutlarındaki ağın

$$\text{Bellek Miktarı} = 2\lceil\frac{640}{1024}\rceil\lceil\frac{20}{8}\rceil + \lceil\frac{640}{1024}\rceil\lceil\frac{20}{8}\rceil = 9$$

tane $1Kbayt$lık blok RAM'e ihtiyacı vardır.

Sütun sayısı 512'den küçük ağlar için $1Kbayt$lık blok RAM2lerin 2×512 byte olarak kullanarak gerçeklenebilmektedir. Bu durumlarda 1024 yerine 512 alınarak gerekli toplam bellek miktarı hesaplanır.

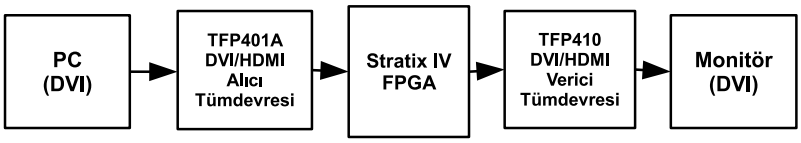
Örnek 4: Durumların bit genişlikleri gerçel ve sanal kısım için 12 bit, sabitin bit genişliği 24 bit olan 512×512 boyutlarındaki ağın

$$\text{Bellek Miktarı} = 2\lceil\frac{512}{512}\rceil\lceil\frac{24}{8}\rceil + \lceil\frac{512}{512}\rceil\lceil\frac{24}{8}\rceil = 9$$

tane $512bayt$lık yada 5 tane $1Kbayt$lık blok RAM'e ihtiyacı vardır.

FPGA'ler içinde hazır olarak bulunan diğer bir blok olak çarpıcı bloklarından, 1) iyleştirilmiş Gabor filtresi için 4/8 tane, 2) kelebek yapısında ise 16 tane çarpıcı bloğu gerekir.

Gabor işlemcisinde bir önceki iterasyona ait durumların değerleri iki satır olarak saklandığından dolayı herbir işlemcinin giriş çıkış gecikmesi $2 \times L + 10$ piksel saat periyo kadardır. İşlemci tamamen iş hattı şeklinde tasarlandığından dolayı her iki iş arasına yazmaç konulması gerekmektedir. Toplam gecikmetedeki 10 değeri bundan kaynaklanmaktadır.

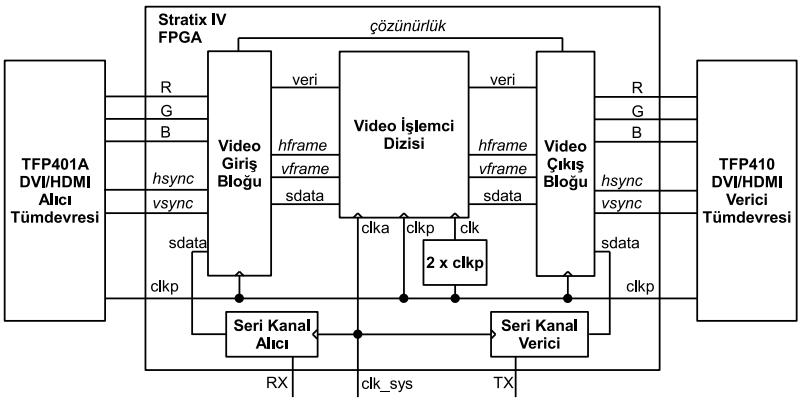


Şekil 5.3 Genel sistemin basitleştirilmiş blok diyagramı

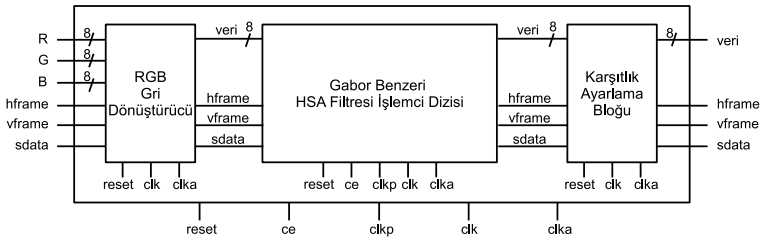
5.3 Prototip Sistemler

Bu alt bölümde prototip sistemin detayları, tasarım aşamaları, kaynak kullanımı ve sonuçları anlatılmıştır. İlk olarak gerçek zamanlı Gabor benzeri HSA filtresinin Altera firmasına ait Stratix IV FPGA'i üzerindeki gerçeklemesinin detayları ve sonuçları, daha sonra ise giriş görüntüsünü 3 sütuna bölerek, her bir sütunu farklı bir filtre ile işleyen prototipin detayları ve sonuçları verilmiştir. Son olarak ise prototip ile literatürdeki diğer gerçeklemelerle olan karşılaştırmalar verilmiştir.

Gerçek zamanlı Gabor benzeri HSA filtresinin blok diyagramı Şekil 4.5'de verilmişti. Gabor benzeri HSA filtresi bloğunun yerine video işlemci dizisi konularak genel amaçlı bir gerçek zamanlı video işleme sistemi elde edilir. Prototip sistemler Altera firması tarafından üzetilen Stratix IV GX230 FPGA üzerinde bulunduran "Audio Video Development Kit, Stratix IV GX Edition" geliştirme bordu ve Bitec firması tarafından üretilen "DVI Input/Output Card" video giriş/çıkış kartı kullanılarak gerçeklenmiştir. Prototip sistemleri 1080×1920 çözünürlüğü, ve 60 Hz yenileme hızındaki (1080p@60Hz, full-HD) videoyu işlemektedir. Giriş videosu karanlık bölgeler dahil edildiğinde çözünürlüğü 1125×2200 olur, 60 Hz yenileme hızıyla piksel saat frekansı $60 \times 1125 \times 2200 = 148.5$ MHzbulunur. Sadece görünür bölge için hesaplanırsa $60 \times 1080 \times 1920 = 124.4MHz$ olduğu görünür. Diğer bir değişle saniyedeki işlem hızı 148.5 MHz ile 124.4 milyon piksel işlenmektedir. Bu video işareti PC tarafıdan üretilmekte, TFP401A DVI/HDMI alıcı tümdevresi ile alınmakta, Stratix IV FPGA ile işlenmekte ve TFP410 DVI/HDMI verici tümdevresi ile işlenen görüntü monitöre aktarılır. Sistemin basitleştirilmiş blok diyagramı Şekil 5.3'de verilmiştir..



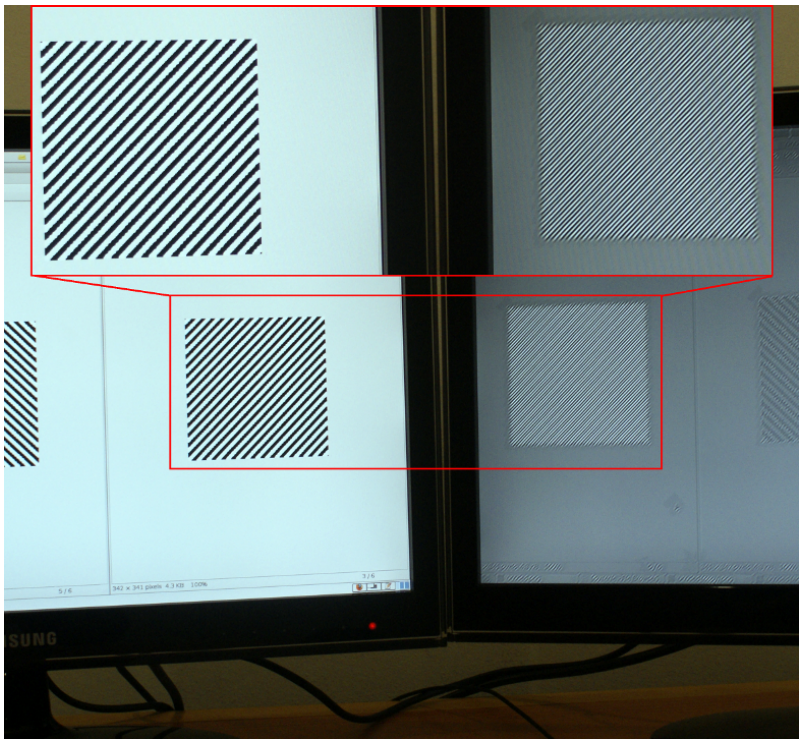
Şekil 5.4 Gerçek zamanlı video işleme sistemi



Şekil 5.5 Gerçek zamanlı video işleme sistemi

5.3.1 1. Prototip Sistem

İlk prototip sistemde Şekil 5.4'deki video işlemci dizisi yerine, 24 bit renkli video görüntüsünden RGB (Red, Green, Blue) 8 bit renk derinliğine sahip gri video görüntüsü elde eden, Gabor benzeri HSA işlemci dizisi ve girişindeki görüntünün karşıtlığını parametrelerine göre ayarlanan bloklar bir dizi oluşturacak şekilde konulmuştur (Şekil 5.5). Prototip sisteminde her biri Euler iterasyonuna karşılık gelen 50 işlemci gerçeklenmiştir. Filtre katsayıları 16 bit (1.15), durumların gerçel ve sanal kısımları 8 bit (1.7) ve sabitlerin bit genişliği 24 bit (2.22) olacak şekilde ayarlanmıştır. Bu bit genişlişliklerine göre her bir işlemci 14 Kbayt RAM kullanmaktadır. Filtrenin parametreleri $\omega_{x0} = \pi/4$, $\omega_{y0} = \pi/4$ ve $\lambda = 0.1$ seçilmiştir. Bu değerler için Gabor işlemcilerinin katsayıları $\alpha_x = 0.1763$, $\alpha_y = 0.1763$, $\beta_x = 0.1763$, $\beta_y = 0.1763$ ve $b = 0.0025$ değerleriyle programlanmıştır. Bu katsayılara göre programlanan sistemin giriş ve çıkışına ait görüntü Şekil 5.6'de verilmiştir. Soldaki görüntü sistemin girişi, sağdaki görüntü ise sistemin çıkışıdır. Filtre giriş görüntüsündeki kare dalganın üçüncü harmoniğini seçecek şekilde ayarlanmıştır. Bu sayede çıkışta, girişteki kare dalganın üçüncü harmoniği görünmektedir. Tüm bloklar ve



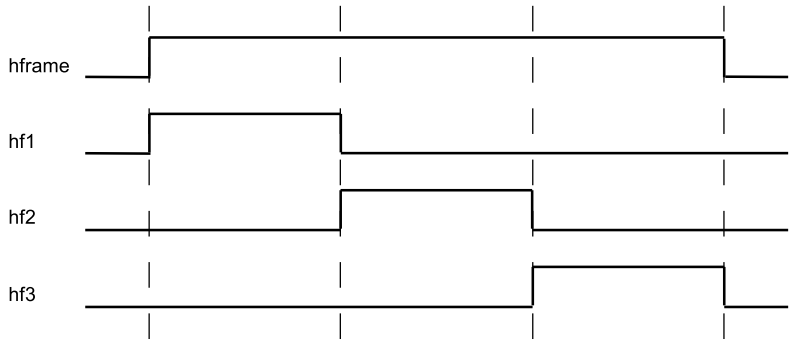
Şekil 5.6 Prototip sistemin giriş görüntüsü ve filtre tarafından işlenen görüntü

işlemcilere seri haberleşme kanalı bağlandığından dolayı tüm blokların parametreleri çalışma anında değiştirilebilmetedir.

Her bir işlemcinin toplam gecikmesi yaklaşık olarak 2 satır olduğu düşünüldüğünde, prototip sistemin toplam giriş çıkış gecikmesi yaklaşık olarak $2 \times 50 \times 2200 = 1.48$ ms olur. Toplam giriş çıkış gecikmesinin hesaplamasında iş hattı ve senkronizasyon yazmaçlarının sayısı az olduğu için ihmal edilmiştir. Altera Stratix IV GX 230 FPGA tümdevresi için prototip sistemin kaynak kullanımı Çizelge 5.1'de verilmiştir.

Çizelge 5.1 Prototip sistemin kaynak kullanımı.

Bloklar	**Kombinasyonal ALUT (Combinational Alut's)**	**Lojik Yazmaçlar (Logic Registers)**	**Bellek (Kb) (Block Memory)**	**18 Bit DSP**
Gabor İşlemcisi	13472 (%7.4)	20321 (%11.2)	871 (%67.6)	202 (%15.5)
Bağlantı Lojiği	613 (%0.4)	381 (%0.2)	0 (%0)	0 (%0)
Toplam	14085 (&7.8)	20702 (%11.4)	871 (%67.6)	202 (%15.5)



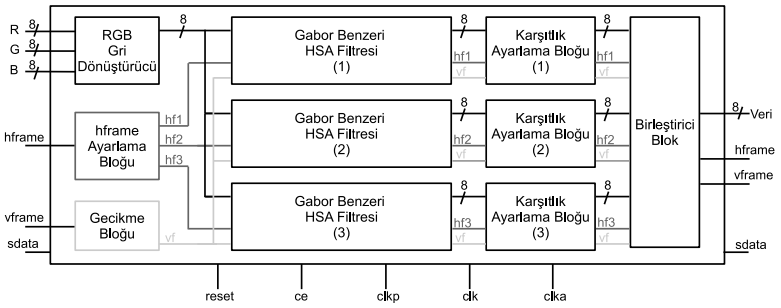
Şekil 5.7 2. prototip sistemin kontrol sinyalleri

hx = 0 vframe = 0	hf1 = 1 vframe=1	hf2 = 1 vframe=1	hf3 = 1 vframe=1	hx = 0 vframe = 0
hx = 0 vframe = 1	hf1 = 1 vframe=1	hf2 = 1 vframe=1	hf3 = 1 vframe=1	hx = 0 vframe = 1
hx = 0 vframe = 0	hf1 = 1 vframe=1	hf2 = 1 vframe=1	hf3 = 1 vframe=1	hx = 0 vframe = 0

Şekil 5.8 2. prototip sistemin kontrol sinyallerinin belirlediği bölgeler

5.3.2 2. Prototip Sistem

İkinci prototip sistemde ekran 640×1080 çözünürlükteki üç parçaya bölünerek işlenmektedir [33]. Her bölge farklı bir Gabor filtresi tarafından filtrelenmektedir. Giriş görüntüsünün üç sütuna bölünebilmesi için *hframe* kontrol sinyali kullanılmıştır. Şekil 5.7'de video giriş bloğu tarafından üretilen *hframe* kontrol sinyali, üç sütun oluşturacak şekilde üç farklı kontrol sinyaline dönüştürülmüştür. Bu kontrol sinyalleri sayesinde giriş görüntüsü Şekil 5.8'de olduğu gibi üç parçaya bölünür. Şekil 5.4'deki video işlemci dizisi yerine Şekil 5.9'deki işlemci dizisi konulmuştur. 24 bit renkli giriş görüntüsünü, 8 bit gri görüntüye dönüştüren RGB'den griye dönüştüren blok çıkışı üç farkı Gabor işlemci dizisine girmektedir. Gabor işlemci dizilerinin çalışmalarını kontrol eden kontrol sinyallerinden *vframe* sinyali sadece iş hattının bozulmaması (senkronizasyonun) için kendi sevisindeki bloklar kadar geciktirilerek Gabor işlemci dizilerine bağlanmıştır. *hframe* kontrol sinyalinden ise Şekil 5.7'deki gibi üç tane 640 piksel genişliğinde sinyaller elde edilmiş ve Gabor işlemci dizilerine bağlanmıştır. RGB'den griye dönüştüren blok, *hframe* ayarlama

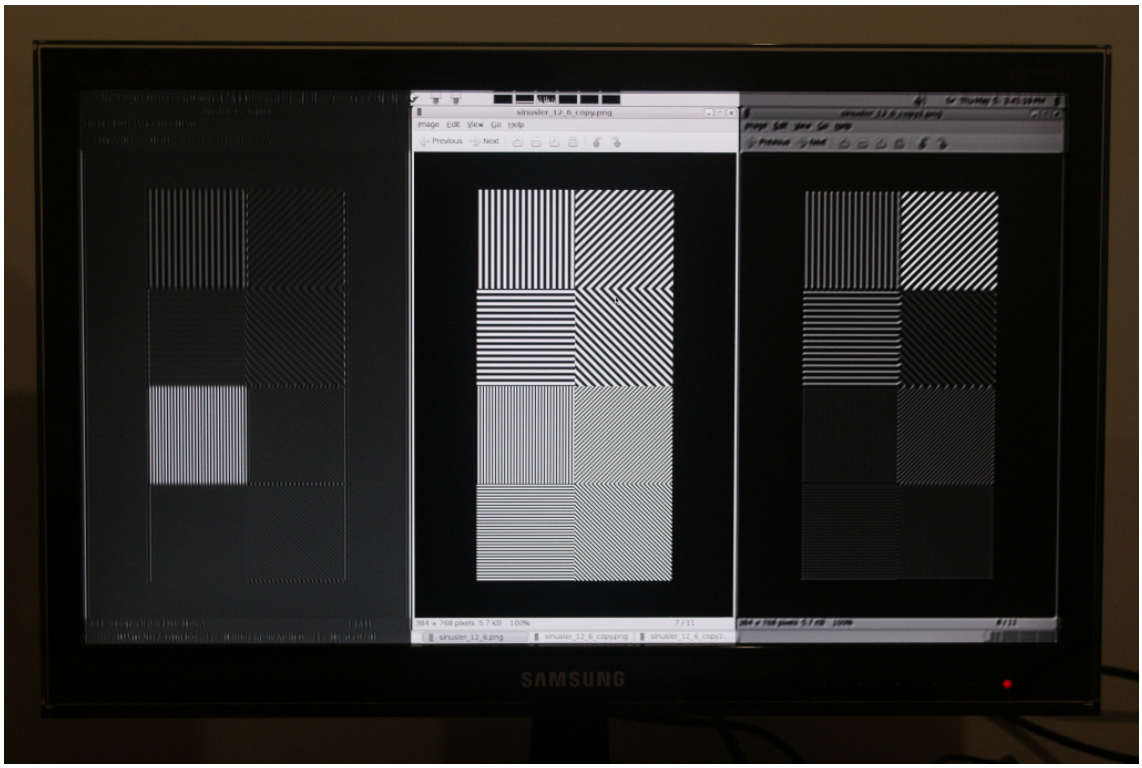


Şekil 5.9 2. prototip sistemin uç bağlantıları ve basitleştirilmiş blok diyagramı

bloğu ve gecikme bloğunun toplam gecikmeleri senkronizasyonun bozulmaması için aynı olacak şekilde tasarlanmıştır. Üç Gabor filtresi kendilerine bağlanan kontrol sinyallerinin lojik '1' olduğu durumlarda çalışmakta, diğer durumlarda ise çalışmalarını durdurmaktadır. Her üç Gabor filtresi de birbirinden bağımsız olacak şekilde farklı parametrelere göre filtreleme yapabilmektedir. İşlenen görüntü histogramının ayarlanabilmesi için karşıtlık ayarlama bloklarına girmektedir. En son blok ise üç *hframe* sinyalinden tekrar tek *hframe* sinyalini elde eder. Ayrıca elde edilen sinyale *vframe* ve verilerin senkronizasyonunuda bu blok sağlar ve çıkış tümderesine iletilmek üzere video çıkış bloğuna gönderir. Tüm bloklar ve işlemcilere seri haberleşme kanalı bağlandığından dolayı tüm blokların parametreleri çalışma anında değiştirilebilmetedir.

Prototip sistemde gerçeklenen her bir Gabor işlemci dizisinin uzunluğu 20 olacak şekilde toplamda 60 işlemci kullanılmıştır. Filtre katsayılarının 16 bit (1.15), durumların gerçel ve sanal kısımları 12 bit (1.11) ve sabitlerin bit genişliği 24 bit (2.22) olacak şekilde ayarlanmıştır. Bu bit genişlişliklerine göre her bir işlemci 9 Kbayt RAM kullanmaktadır.

Sistemin giriş ve çıkışına ait görüntü Şekil 5.10'de verilmiştir. Soldaki görüntü sistemin girişi, sağdaki görüntü ise sistemin çıkışıdır. Filtre giriş görüntüsündeki üstteki kare dalganın üçüncü harmoniği ve alttaki kare dalgenın temel harmoniğini seçecek şekilde ayarlanmıştır. Bu sayede çıkışta, girişteki kare dalganın üçüncü harmoniği görünmektedir. Her bir işlemcinin toplam gecikmesi yaklaşık olarak 2 satır olduğu düşünüldüğünde, prototip sistemin toplam giriş çıkış gecikmesi yaklaşık olarak $2 \times 20 \times 2200 = 592.6\mu s$ olur. Toplam giriş çıkış gecikmesinin hesaplamasında iş hattı ve senkronizasyon yazmaçları sayısı az olduğu için ihmal edilmiştir. Altera Stratix IV GX 230 FPGA tümdevresi için prototip



Şekil 5.10 2. prototip sistemin çıkış görüntüsü

sistemin kaynak kullanımı Çizelge 5.2'de verilmiştir.

5.4 Gerçekleme Sonuçları ve Karşılaştırmalar

Prototip sistem 480×640 çözünürlükten 1080×1920'e, 50 Hz çerçeve yenileme hızından 85 Hz'e ve 1 iterasyondan 100 iterasyona kadar çeşitli durumlar için test edilmiştir. Giriş/çıkış tümdevrelerinin sınırlarından dolayı test edilebilen en yüksek iş yükü $1080 \times 1920@60Hz$ için 124.4 MP/s (mega piksel/saniye) olmuştur. Buna rağmen, kaynak iyileştirmesi devre dışı bırakıldığında işlemci dizisi 373.2 MP/s işlem kapasitesine sahiptir.

Çizelge 5.2 Prototip sistemin kaynak kullanımı.

Bloklar	**Kombinasyonal ALUT (Combinational Alut's)**	**Lojik Yazmaçlar (Logic Registers)**	**Bellek (Kb) (Block Memory)**	**18 Bit DSP**
Gabor İşlemcisi	13472 (%7.4)	20321 (%11.2)	871 (%67.6)	202 (%15.5)
Bağlantı Lojiği	613 (%0.4)	381 (%0.2)	0 (%0)	0 (%0)
Toplam	14085 (&7.8)	20702 (%11.4)	871 (%67.6)	202 (%15.5)

Çizelge 5.3 FIR Gabor filtresi ve Gabor benzeri HSA filtresinin kaynak kullanımı

	FIR Gabor Filtresi		Gabor Benzeri HSA Filtresi	
Bant Genişliği	Şablon Boyutu	Çarpıcı/Blo RAM Sayısı	İşlemci Sayısı	Çarpıcı/Blo RAM Sayısı
$\lambda \geq 0.2$	11×11	$242/20$	30	$240/480$
$\lambda \geq 0.1$	21×21	$882/40$	50	$400/800$
$\lambda \geq 0.02$	51×51	$5202/100$	115	$920/1840$

Table 5.3'de FIR Gabor filtresi ve Gabor benzeri HSa filtresi FPGA gerçeklemeleri için kullanışlı bazı istatistikler verilmiştir. Bu sonuçlar, $\lambda \geq 0.2$ bant geniliği için her iki tasarımda da neredeyse aynı sayıda çarpıcı gerektiğini göstermesine rağmen, FIR filtrenin çarpıcı kullanımı λ'nın düşmesiyle sert bir şekilde artmaktadır. Diğer taraftan, Gabor benzeri HSA filtresi gerçeklemelerinde FIR gerçeklemelerine göre daha fazla RAM kullanılmaktadır. Table 5.3'de verilen 9 Kbit Blok RAM kullanım istatistikleri, Full-HD giriş ve sırasyıla u_{ij}, $x_{ij}^{\mathrm{R}}(n+1)$, $x_{ij}^{\mathrm{I}}(n+1)$ and bu_{ij}'ların bit genişlikleri 8,12,12 ve 16 için verilmiştir.

Gerçeklenen sistem ile literatürde yapılmış çalışmaların karşılatırması Çizelge 5.4'de verilmiştir. Farklı frekanslarda çalışan teknolojiler ile gerçeklenen birçok mimarinin birbirleriyle karşılatırılması oldukça zordur. Bu nedenle ölçekleme için saniyedeki iş yükü (iş yükü/çalışma frekansı, throughtput divided by frequency) kullanılmıştır. Kaynak iyileştirmesi yapılmadığı durumda karşılaştırma sonuçlarıgercek zamanlı Gabor benzeri HSA filtresi, [14]'da sunulmuş olan GPU gerçeklemesinden (giriş/çıkış gecikmeleri dahil) 2 kat hızlıdır. Gercek zamanlı Gabor benzeri HSA filtresi [18]'de verilen Gabor benzeri HSA filtresinden 4.2 kat, [15], [16] ve [17]'de verilen iki boyutlu FIR gerçeklemelerinden ise sırasıyla 2.27, 4.94 ve 42 kat daha hızlıdır. Kaynak iyileştirmesi yapıldığı durumda hızı yarı yarıya düşmesine rağmen, [14]'deki GPU gerçeklemesi ile aynı hızda, diğer gerçeklemelerden ise daha hızlıdır. Bu karşılaştırma sonuçlarına göre gerçek zamanlı Gabor benzeri HSA filtresi tezin yazıldığı ana kadar gerçeklenen en hızlı Gabor filtresidir.

5.5 Sonuç

Bu bölümde önceki bölümde detayları verilen gerçek zamanlı HSA gerçeklemesindeki işlemci dizisi yerine Gabor işlemci dizisi konularak elde edilen gerçek zamanlı Gabor benzeri HSA filtresi ve prototiplerin detaylarıyşa literatürdeki diğer gerçeklemelerle olan karşılaştırmalar verilmiştir.

Aritmetik işlem bloğu kullanılan çarpıcı sayısının azaltmak için Şekil 3.3'de verilen tasarım kullanılarak gerçeklenmiştir. Piksel saat hızının iki katı hızda çalışan bu blok, 3 saat darbesinde sonuç vermektedir ve her giriş için bir çıkış üretmektedir.

Gabor benzeri HSA filtresinin katsayıların, ilk koşulların değerlerinin, sınır koşullarının ve diğer blokların çalışma parametrelerinin çalışma anında değiştirilebilmesi için seri bir programlama arayüzü tasarımı yapılmıştır. Bu arayüz sayesinde tekrar sentezleme gibi zaman kaybına yol açacak işin yapılmasına gerek kalmamıştır. Aynı zamanda seri programlama arayüzü ile çalışma anında sistemin durumu izlenerek herhangi bir sorun olup olmadığına bakılabilmektedir. Bu sayede sistem sadece PC ortamında değil donanım üzerinde de test edibilmektedir.

Gabor benzeri HSA filtresinin çıkışının karşıtlığı belli bir bölgede yoğunlaştığı için insan gözü detayları algılayamamaktadır. Bu nedenle karşıtlığın iyileştirilmesi için karşıtlık germesi (Contrast Streching) yöntemi kullanılarak iyileştirme yapılmıştır.

1. prototip 50 Euler iterasyonunu gerçekleyen 50 işlemci kullanılarak gerçeklenmiştir. Giriş çıkış toplam gecikmesi 100 satır, yani 1.48ms dir. İşlemci çalışma saat frekansı 297 MHz dir. Prototip sistem tamamem iş hattı mantığı ile tasarlandığı için her saat darbesin de tüm çarpıcı ve toplayıcılar çalışmaktadır, bu durum da saniyedeki toplam çarpma ve toplama işlem sayısı yaklaşık 119 Milyar dır ki süper bilgisayar hızı olan 10^12'nın yaklaşık 8'de biridir. Saniyede işlenen piksel sayısı karanlık bölgeler hariç 124.4 milyon, dahil 148.5 milyondur.

2. prototip toplam da 60 euler iterasyonunu gerçekleyen 3 paralel hatta 20 işlemci kullanılarak gerçeklenmiştir. Giriş çıkış toplam gecikmesi 40 satır, yani 592.6μs dir. İşlemci çalışma saat frekansı 297 MHz dir. Prototip sistem tamamem iş hattı mantığı ile tasar-

landığı için her saat darbesin de tüm çarpıcı ve toplayıcılar çalışmaktadır, bu durum da saniyedeki toplam çarpma ve toplama işlem sayısı yaklaşık 143.5 Milyar dır ki süper bilgisayar hızı olan 10^12'nın yaklaşık 6'da biridir. Saniyede işlenen piksel sayısı karanlık bölgeler hariç 124.4 milyon, dahil 148.5 milyondur.

Kaynak iyileştirmesi yapıldığında işlemci dizisi 297 MP/s işlem kapasitesine sahip iken, kaynak iyileştirmesi devre dışı bırakıldığında işlemci dizisi 373.2 MP/s işlem kapasitesine sahip olmaktadır. Litesatürdeki Gabor filtresi gerçeklemeleriyle karşılaştırıldığında, kaynak iyileştirmesi devre dışı iken en hızlı gerçekleme olan [14]'dan 2 kat hızlıdır. Kaynak iyileştirme devrede iken yine [14]'deki gerçeklemeyle aynı hızdadır. Sonuç olarak şu ana kadar gerçeklenen en hızlı Gabor filtresidir.

Çizelge 5.4 Gabor filtre gerçeklemelerinin karşılaştırılması

Gerçekleme	Donanım	İşlemci Frekansı	Method	Şablon Boyutları	Çözünürlük (Çerçeve Hızı)	Throughput (MP/s)	Throughput/ Hertz
Sunulan [a]	FPGA (Altera Stratix IV 230)	297.0 MHz	2-B HSA	3×3 (yineleme)	1080×1920 (60)	124.4	0.42
		148.5 MHz [b]			480×640 (310) [c]		0.84 [b]
					32×32 (121500) [c]		
maksimum hız [d]	FPGA (Altera Stratix IV 230)	445.5 MHz [d]	2-B HSA	3×3 (yineleme)	1080×1920 (180) [d]	373.2 [d]	0.84
					480×640 (930) [d]		
					32×32 (364500) [d]		
Cheung et al. (2006) [18]	FPGA (Virtex II 1000-4)	120 MHz	2-B HSA	3×3 (yineleme)	32×32 (23000)	23.5 [e]	0.20
Wang and Shi (2010) [14]	GPU (NVIDIA GeForce 9800 GTX+)	738 MHz (çekirdek)	2-B FIR (ayrış.)	65×65	512×512 (—)	666.6 (çekirdek) [e]	0.90 (çekirdek)
						306.7 (toplam) [e]	0.42 (toplam)
Norouznezhad et al. (2010) [15]	FPGA (Xilinx Virtex 5 XC5VSX50)	296 MHz	2-B FIR	7×7	480×640 (30×8 banks)	110.5 [e]	0.37
Cho et al. (2011) [16]	FPGA (Xilinx Virtex 6 SX475T)	200 MHz	2-B FIR	11×11	—	33.33 ($\times$6 kanal)	0.17 ($\times$6 kanal)
Liu et al. (2010) [17]	ASIC (SMIC $0.13\mu m$)	250 MHz	2-B FIR	11×11	256×256 (32 [e])	2.1	$<$0.01
					256×256 (47 [e])	3.1	0.01
					256×256 (77 [e])	5.1	0.02

[a] Önerilen yapı tamamen iş hattı mantığıyla çalışır bir prototip olarak gerçeklenmiştir.

[b] Bu değerler kaynak iyileştirmesi yapılmamış, direk gerçekleme için verilmiştir.

[c] Bu çözünürlük ve/veya çerçeve hızları throughputlardan tahmini olarak hesaplanmıştır, ve fonksiyonel VHDL benzetimleriyle doğrulanmıştır.

[d] Zamanlama direk gerçekleme için verilmiştir. RTCNNP–v2'nin [31] en kötü durumdaki zamanlamaları kullanılmıştır, buna rağmen gerçek sonuçlar beklenenden daha iyi olacaktır [e] Bu değerler referanslarına ilişkili olarak hesaplanmıştır.

BÖLÜM 6

HIZ AYARLI FİLTRELER VE TASARLANAN BİR SAYISAL DEVRE GERÇEKLEMESİ

Bu bölüm video görüntüsü sabit hızlı hareketin algılanmasında kullanılan hız ayarlı uzay-zamansal filtrelerin [19] matematiksel incelemesi, gerçeklemesi ve önerilen sayısal devre gerçeklmesini içermektedir. İlk olarak sabit hızlı hareketin uzay-zaman ve frekans düzlemindeki matematiksel modeli, Hız Ayarlı Filtrenin (HAF) matematiksel detayları, daha sonra HAF'ın analog devre gerçekmesi [19] ve sayısal benzetimleri [34] ve son olarak da HAF için önerilen sayısal devre verilecektir.

6.1 Matematiksel İnceleme

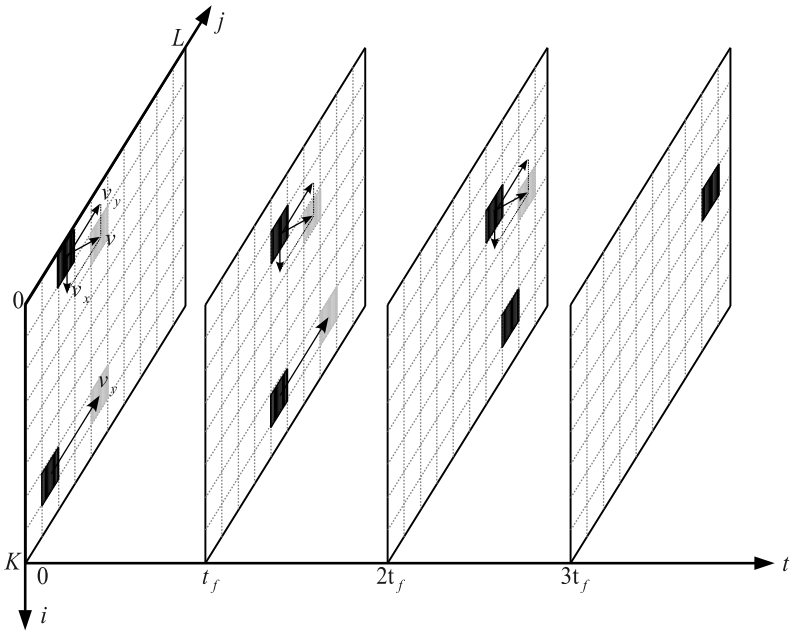
Video görüntüsü üzerindeki çerçeveler arasındaki parlaklığın değişimi

$$\begin{aligned} s(i,j,t) &= s(i-v_x t, j-v_y t, 0) \\ &= s_0(i-v_x t, j-v_y t) \end{aligned} \tag{6.1}$$

şeklinde tanımlanır. i ve j uzamsal koordinatat, v_x ve v_y hız bileşenleri, t zaman ve $s_0(i-v_x t, j-v_y t)$ ise referans çerçevedir. Referans çerçeveden Δt kadar sonra gelen çerçevedeki parlaklık değişimi (6.2)'deki gibi hesaplanır.

$$s(i,j,t+\Delta t) = s(i-v_x \Delta t, j-v_y \Delta t, \Delta t) \tag{6.2}$$

Şekil 6.1'de sabit hızla giden bir cismin hareketi, başlangıç konumu ve hız bileşenleri cinsinden verilmiştir.



Şekil 6.1 Hareketli görüntüdeki hız vektörleri.

6.2 Hız Ayarlı Filtreler

Şekil 6.1'deki sabit hızla hareket eden bir cismin algılanabilmesi için HAF adındaki bir uzay-zamansal hareket seçici filtreler Torralba [19] tarafından sunulmuştur. Şekil 6.2'de HAF'nin analog devresi verilmiştir, bu devredeki $x_{ij}(t)$ düğümü için Kirchoff akım denklemi yazıldığında

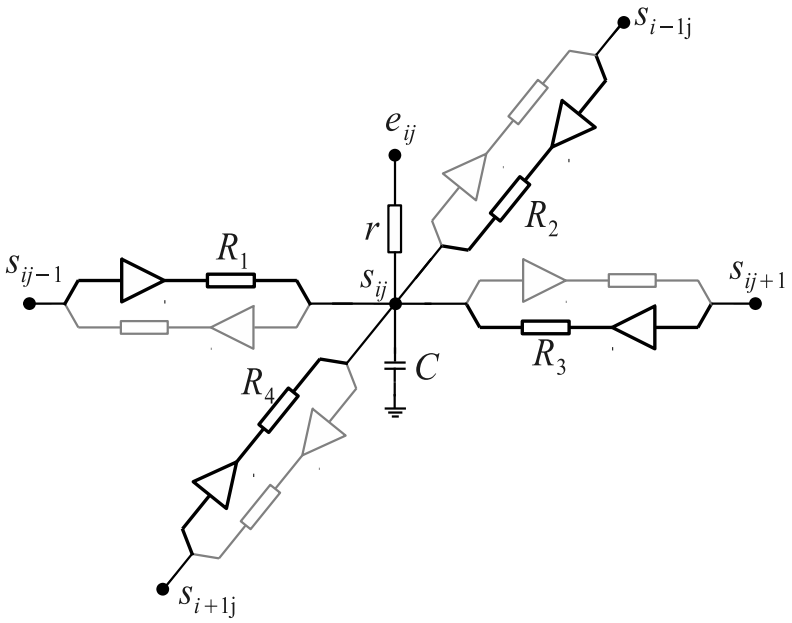
$$C\frac{dx_{ij}(t)}{dt} = \frac{u_{ij}(t)-x_{ij}(t)}{r} + \frac{x_{i-1j}(t)-x_{ij}(t)}{R_1} + \frac{x_{ij-1}(t)-x_{ij}(t)}{R_2} + \frac{x_{i+1j}(t)-x_{ij}(t)}{R_3} + \frac{x_{ij+1}(t)-x_{ij}(t)}{R_4} \tag{6.3}$$

elde edilir. Terimler yeniden düzenlenirse

$$u_{ij}(t) = x_{ij}(t)(1+4\gamma^2) - a_x x_{i-1j}(t) - a_y x_{ij-1}(t) - b_x x_{i+1j}(t) - b_y x_{ij+1}(t) + \tau\frac{dx_{ij}(t)}{dt} \tag{6.4}$$

elde edilir. Buradaki $4\gamma^2 = a_x + a_y + b_x + b_y$, $a_x = r/R_1$, $a_y = r/R_2$, $b_x = r/R_3$, $b_y = r/R_4$ ve $\tau = rC$ dir. (6.4), (2.16)'deki şablon nokta çarpımı şeklinde düzenlenirse

$$\frac{dx_{ij}(t)}{dt} = \frac{1}{\tau}\begin{bmatrix} 0 & a_y & 0 \\ a_x & -(1+4\gamma^2) & b_x \\ 0 & b_y & 0 \end{bmatrix} \circledast \begin{bmatrix} x_{i-1,j-1}(t) & x_{i-1,j}(t) & x_{i-1,j+1}(t) \\ x_{i,j-1}(t) & x_{i,j}(t) & x_{i,j+1}(t) \\ x_{i+1,j-1}(t) & x_{i+1,j}(t) & x_{i+1,j+1}(t) \end{bmatrix} + \frac{1}{\tau}u_{ij}(t) \tag{6.5}$$



Şekil 6.2 Hız ayarlı filtre analog devre gerçeklemesi

elde edilir. (6.5)'de görüldüğü gibi HAF, Doğrusal HSA durum denklemi ile aynı yapıdadır. a_x, b_x, a_y ve b_y değerleri yerine sırasıyla $\gamma^2+(a_x-b_x)/2$, $\gamma^2-(a_x-b_x)/2$, $\gamma^2+(a_y-b_y)/2$ ve $\gamma^2-(a_y-b_y)/2$ konulduğunda (6.5)

$$\frac{dx_{ij}(t)}{dt}=\frac{1}{\tau}\begin{bmatrix} 0 & \gamma^2+\frac{a_y-b_y}{2} & 0 \\ \gamma^2+\frac{a_x-b_x}{2} & -(1+4\gamma^2) & \gamma^2-\frac{a_x-b_x}{2} \\ 0 & \gamma^2-\frac{a_y-b_y}{2} & 0 \end{bmatrix} \circledast \begin{bmatrix} 0 & x_{i-1,j}(t) & 0 \\ x_{i,j-1}(t) & x_{i,j}(t) & x_{i,j+1}(t) \\ 0 & x_{i+1,j}(t) & 0 \end{bmatrix} + \frac{1}{\tau}u_{ij}(t) \tag{6.6}$$

halini alır. HAF'ın transfer fonksiyonunu bulmak için (6.6)'nin Fourier dönüşümü alınıp, gerekli düzenlemeler yapıldığında (6.7) denklemi elde edilir.

$$H(e^{j\omega_x},e^{j\omega_y},j\Omega_t)=\frac{1}{1+2\gamma^2(1-\cos\omega_x)+2\gamma^2(1-\cos\omega_y)+\tau\left(j\Omega_t+\frac{a_x-b_x}{\tau}\sin\omega_x+\frac{a_y-b_y}{\tau}\sin\omega_y\right)} \tag{6.7}$$

Frekans yanıtının daha basit hale getirmek için düşük uzaysal açısal frekanslardaki davranışını incelendiğinde, $1-\cos\Omega_x\simeq\Omega_x^2/2$ ve $\sin\Omega_x\simeq\Omega_x$ yaklaşıklıkları yapılabilir. Bu yaklaşıklar sonucunda frekans yanıtı

$$H(e^{j\omega_x},e^{j\omega_y},j\Omega_t)=\frac{1}{1+\gamma^2\omega_x^2+\gamma^2\omega_y^2+j[(a_x-b_x)\omega_x+(a_y-b_y)\omega_y+\tau\Omega_t]} \tag{6.8}$$

şeklinde elde edilir. (6.8)'deki frekans yanıtı a_x, b_x, b_x, b_y ve τ değerlerine bağlı olarak sınıflanbabilir. Bu sınıflar şunlardır:

- Dairesel simetrik: $a_x = b_x = a_y = b_y$ ve $C = 0$ koşulları sağlandığında frekans yanıtı dairesel simetrik,
- Yönlü simetrik: $a_x = b_x <> a_y = b_y$ ve $C = 0$ koşulları sağlandığında frekans yanıtı yönlü simetik,
- Yönlü asimetrik: $a_x + b_x = a_y + b_y > 0$ ve $C = 0$ koşulları sağlandığında frekans yanıtı yönlü asimetik,
- Uzay-zamansal yönlü: $a_x + b_x = a_y + b_y > 0$ ve $C > 0$ koşulları sağlandığında frekans yanıtı uzay-zamansal yönlü filtre.

4. madde için uzay-zamansal yönlü filtre, HAF olarak kullanılır. HAF'nin parametreleri analog şu şekilde tanımlanır:

- Bant genişliği: γ. Filtrenin uzaysal bant genişliğini belirler.

$$\gamma^2 = \frac{1}{2}(a_x + b_x) = \frac{1}{2}(a_y + b_y) \tag{6.9}$$

- Hız: $\mathbf{v}_0^T = (v_{x0}, v_{y0})$. Giriş $\mathbf{v}_0$ ile hareket ettiğinde çıkışın enerjisi en büyük olur.

$$\mathbf{v}_0^T = \frac{1}{rC}(a_x - b_x, a_y - b_y) \tag{6.10}$$

- Zaman sabiti: τ. Giriş hızı optimum hızdan farklı olduğu zaman filtrenin hassasiyetini belirler.

$$\tau = rC \tag{6.11}$$

6.3 Benzetim Sonuçları

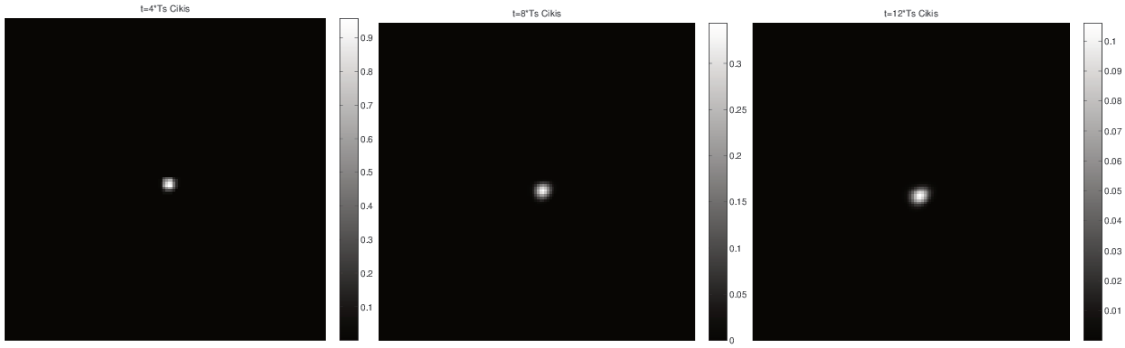
HAF'nin sayısal benzetiminin yapılabilmesi için öncelikle durum denklemlerinin ayrıklaştırılarak fark denklemleri haline getirilmesi gerekmektedir. HAF'nin durum denklemleri (2.17)'deki gibi paketlendikten sonra türev operatörü yerine Euler ileri yaklaşıklığı uygulanır ve (2.25) denklemindekilere benzer denklemler elde edilir. Çözümün buluna-

bilmesi için her fark denklemine ilk koşul verilerek çıkış hesaplanır ve sabit giriş görüntüsü için bu çıkış sabit kalıncaya kadar devam ettirilir. HAF için giriş görüntüsü zamanla değişmektedir, bu nedenle HAF devresinin zaman sabiti τ ile görüntü çerçevelerinin geliş hızı t_f arasında bir ilişki tanımlanması gerekmektedir. Çünkü devrenin zaman sabiti çerçeve geliş süresinden daha küçük ise $(\tau < t_f)$ ise devreye sabit giriş uygulanmış gibi olur ve devre hız algılamada kullanılamaz. Bundan dolayı $\tau > t_f$ seçilmesi gerekir, hız algılamanın daha iyi yapılabilmesi için zaman sabitinin mümkün olduğunca büyük seçilmesi gerekir.

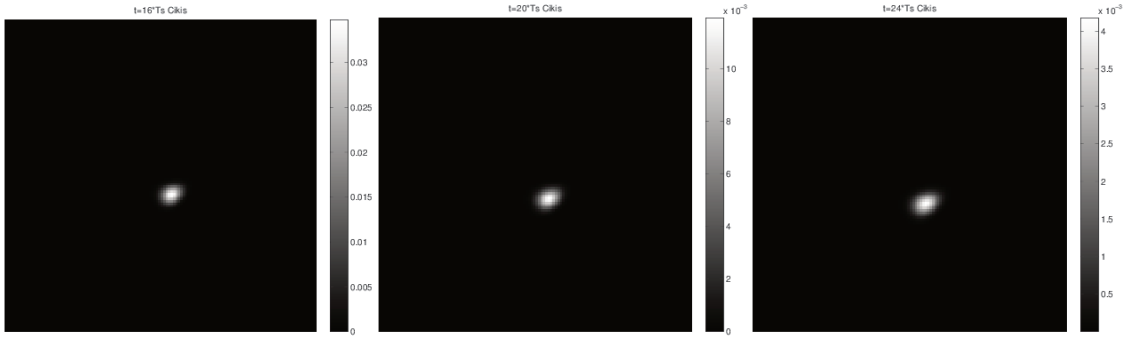
HAF'nin sayısal benzetimi için [34]'deki gibi yapılmıştır. [34]'de giriş her 3 iterasyonda değiştirilmekte, elde edilen sonuç bir sonraki giriş görüntüsü için fark denklemlerinin ilk koşulu olarak kullanılmaktadır. Benzetim adımları

$$
\begin{aligned}
\mathbf{x}(1) &= \mathbf{x}(0) + T_s\{\mathbf{A}\mathbf{x}(0) + \mathbf{B}\mathbf{u}(\mathbf{0})\} \\
\mathbf{x}(2) &= \mathbf{x}(1) + T_s\{\mathbf{A}\mathbf{x}(1) + \mathbf{B}\mathbf{u}(\mathbf{0})\} \\
\mathbf{x}(3) &= \mathbf{x}(2) + T_s\{\mathbf{A}\mathbf{x}(2) + \mathbf{B}\mathbf{u}(\mathbf{0})\} \\
\mathbf{x}(4) &= \mathbf{x}(3) + T_s\{\mathbf{A}\mathbf{x}(3) + \mathbf{B}\mathbf{u}(\mathbf{1})\} \\
\mathbf{x}(5) &= \mathbf{x}(4) + T_s\{\mathbf{A}\mathbf{x}(4) + \mathbf{B}\mathbf{u}(\mathbf{1})\} \\
\mathbf{x}(6) &= \mathbf{x}(5) + T_s\{\mathbf{A}\mathbf{x}(5) + \mathbf{B}\mathbf{u}(\mathbf{1})\} \\
&\vdots \\
\mathbf{x}(N) &= \mathbf{x}(N-1) + T_s\{\mathbf{A}\mathbf{x}(N-1) + \mathbf{B}\mathbf{u}(\lceil (N-1)/3 \rceil\}
\end{aligned}
\tag{6.12}
$$

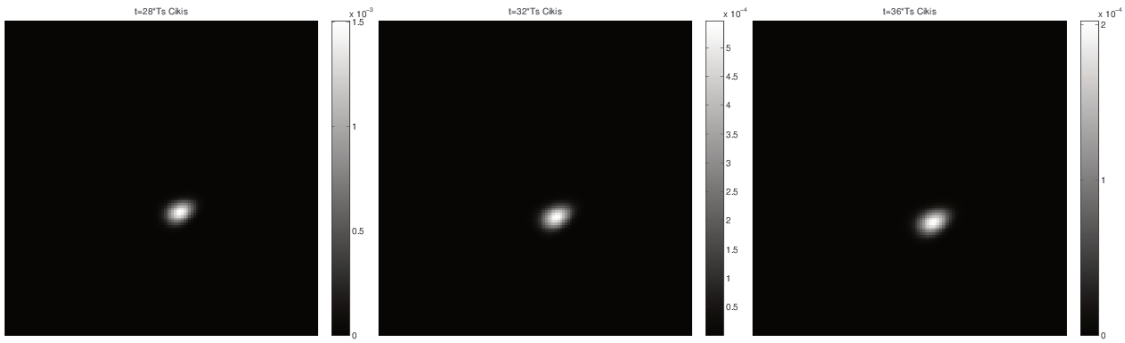
şeklinde yapılır. 3 iterasyon benzetim sonuçlarında elde edilmiştir. T_s adım aralığı, [27]'deki ideal adım aralığı $T_s = \frac{1}{|a_{00}|}$ alınmıştır. Bu adım aralığı kullanıldığında düşük hızlar için HAF'ın ayrık $\mathbf{A}$ matrisinin mutlak değeri en büyük özdeğerinin 0.4 ten küçük olduğu görülmüştür. Bundan dolayı 3. iterasyondan sonra $0.4^3 = 0.064$ değerine düşer, bu değerde yeterince küçük olduğu için uzaysal filtrelemenin yapıldığı düşünülür ve bir sonraki giriş görüntüsü filtreye uygulanır. Şekil 6.3'de HAF'ın uzay-zamansal impuls yanıtı verilmiştir. Şekil 6.4'de farklı yön ve hızlarda hareket eden 5 karenin bulunduğu hareketli giriş görüntüsü, $v_x = 1$ $v_y = 2$ piksel/s hızlarına ayarlanmış HAF'nin çıkışı verilmiştir.



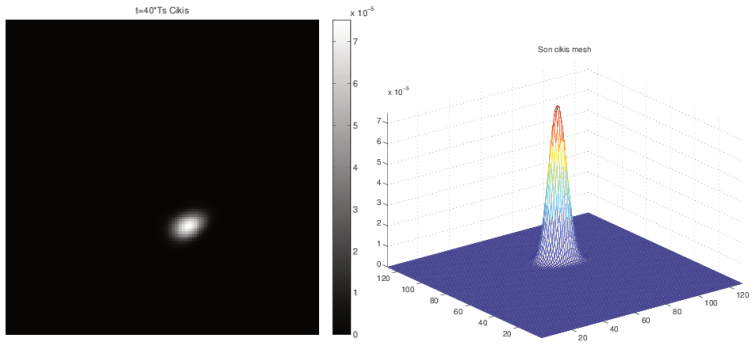
(a) (b) (c)



(d) (e) (f)

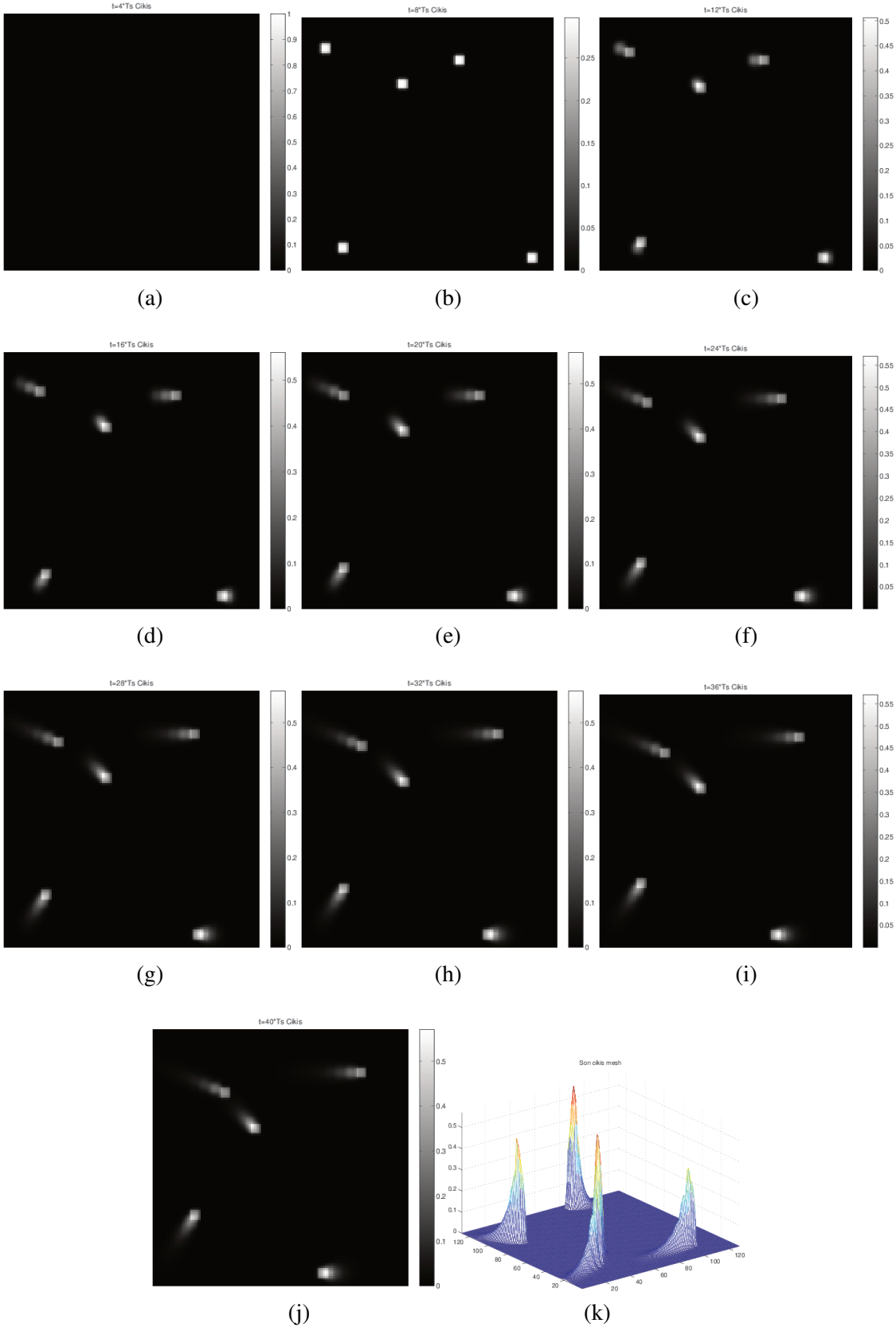


(g) (h) (i)



(j) (k)

Şekil 6.3 $v_x = 1$, $v_y = 2$ piksel/s hızına ayarlı HAF'nin uzay-zamansal impuls yanıtı.



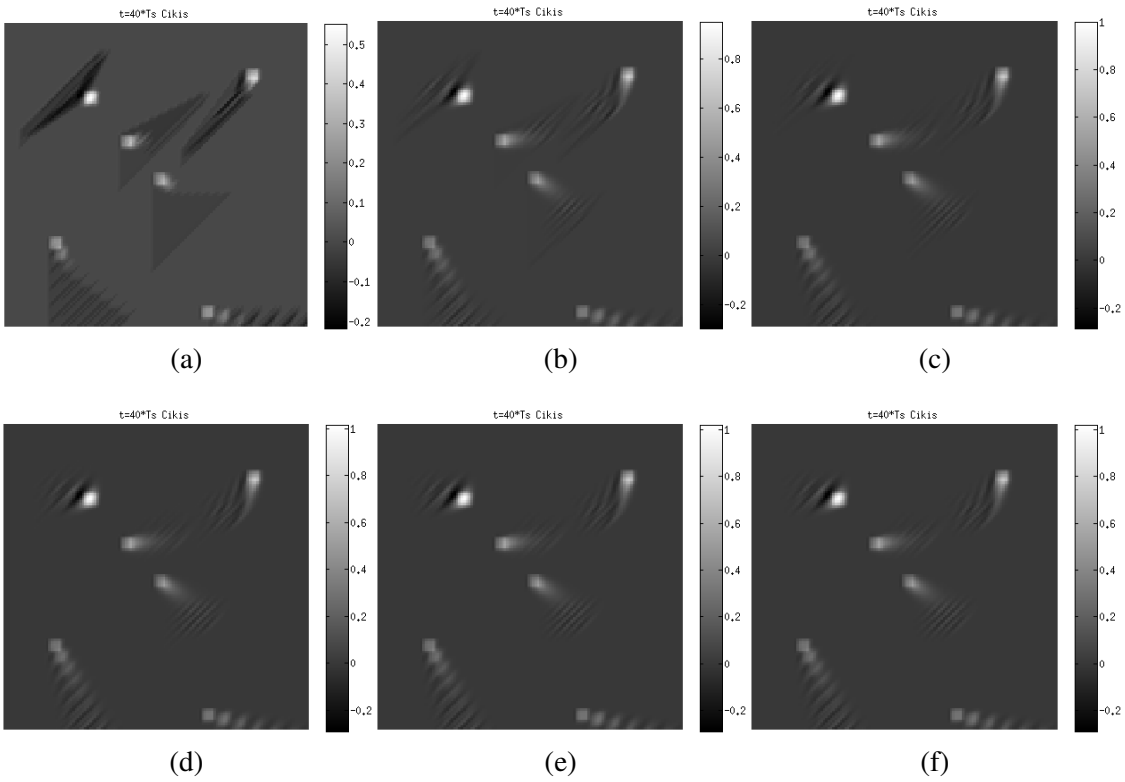
Şekil 6.4 Hareketli giriş görüntüsünün $v_x = 1$, $v_y = 2$ piksel/s hızına ayarlı HAF'si ile işlenmesi.

6.3.1 Sabit Noktalı Benzetim

FPGA tümdevreleri ile aritmetik işlem yapmak gerektiğinde, işleme giren sayıların belli bir bit genişliğinde olması tercih edilir. Bundan dolayı sayılar sabit noktalı olarak ifade edilirler. Bu alt bölümde HAF'nin FPGA tümdevresi ile gerçeklenmesi için katsayılarının, durumlarının ve sabitinin bit genişliklerinin testpiti gerekir. Gabor benzeri HSA filtresinin gerçeklenmesinden görüldüğü gibi bellek ihtiyacı çok fazladır. Bu nedenle HAF'nın gerçeklemesinde kullanılan işlemcilerin en az sayıda bellek kullanması amaçlanmıştır. Şekil 6.5'de durumların bit genişliklerine göre benzetim yapılmıştır. HAF'nin şablon katsayıları işaretli 8 bit, $\mathbf{Bu}(\mathbf{N})$ değeri işaretli 16 bit olarak alınıp ve durumun bit genişliği 5, 8, 10, 12, 16 alınarak $40T_s$ uzunluğunda benzetim yapılmıştır. Benzetime tüm değerler kayan noktalı sayılar kullanılarak tekrar edilmiştir. Benzetim sonuçlarından görüldüğü gibi durumun bit genişliği 5 bit seçilsebile filgrenin ayarlı olduğu hızı seçmesi kayan noktalı benzetim ile aynı olduğu görülmüştür. Benzetimlerin sonucunda durumların bit genişliği ez az 5 bit olması gerektiğine karar verilmiştir..

6.4 Gerçek Zamanlı Hız Ayarlı Filtrenin Sayısal Devre Gerçeklemesi

Gerçek zamanlı Gabor benzeri HSA filtresinde kullanılan giriş/çıkış blokları ve işlemciler gerçek zamanlı HAF'ın gerçeklemesi içinde kullanılmıştır. Gabor filtresi girişi durağan görüntü olmasına rağmen HAF'ın girişi zamanla değişen görüntüdür. Ayrık zamanlı HAF'nin çıkışı (6.12)'de verilen iterasyonlarla bulunmaktadır. Her 4 iterasyon da giriş görüntüsü bir sonraki görüntü ile değiştirilmekte ve önceki çıkış ilk koşul olarak kullanılmaktadır. Çıkışın tekrar işlemci dizisin de kullanılmasından dolayı eski çıkışların saklanması gerekir. Tasarımdaki asıl problem bellek yetersizliğidir. 640x480 gibi düşük bir çözünürlükte dahi gelişmiş Stratix IV FPGA tümdevresinde iki çerçeve saklanabilirken 1920x1080 çözünürlükteki bir çerçeve FPGA'e sığmamaktadır. Gerçekleme 640×480 çözünürlükte yapılsa dahi 6000$ değerindeki bir FPGA'i RAM olarak kullanmak ne endüstriyel, ne de akademik bir değer taşır. Karşılaşılan gerçekleme problemi sonucunda FPGA'in dışındaki DDR3 SDRAM'leri kullanan çerçeve tampon belleğinin kullanılmasına karar verilmiştir. Bundan sonra, sırasıyla gerçek zamanlı HAF'nin tasarım ile işlemci



Şekil 6.5 Hareketli giriş görüntüsünün $v_x = 1$, $v_y = 2$ piksel/s hızına ayarlı sabit noktalı sayı formatı kullanılarak HAF ile işlenmesi. Şablon katsayıları işaretli 8 bit (1.7), sabit işaretli 16 bit (2.14) alınmıştır, durum değişkenleri ise işaretli (a) 5 bit (1.4), (b) 8 bit (1.7), (c) 10 bit (1.9), (d) 12 bit (1.11), (e) 16 bit (1.15) alınmıştır. (f) kayan noktalı sonucu göstermektedir.

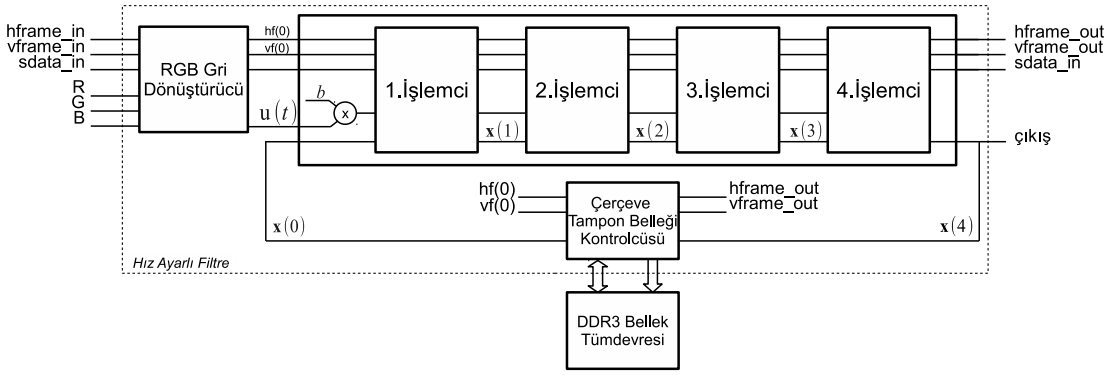
bloğu aritmetik bloğunun detayları verilmiştir.

6.4.1 Gerçek Zamanlı Hız Ayarlı Filtrelerin Mimarisi

Gerçek zamanlı Gabor benzeri HSA filtresinin blok diyagramı Şekil 4.5'de verilmişti, bu blok diyagramında işlemci dizisi yerine Şekil 6.6 konularak gerçek zamanlı HAF elde edilmiştir.

Tasarımda daha önce kullanılmayan DDR3 tümdevreleri ve çerçeve tampon belleği kontrol bloğu vardır. DDR3 tümdevreleri ile olan veri alış verişi Altera firmasına ait UniPHY DDR3 RAM kontrolcüsü ile yapılmıştır. Bu kontrolcü ile haberleşme ise yine Altera firmasına ait avalon veri yoluyla yapılmıştır.

Video giriş bloğundan gelen 24 bit RGB görüntü RGB'den griye dönüştürücü bloğu tarafından 8 bit gri görüntüye dönüştürülür ve kontrol sinyalleriyle senkron olarak işlemci



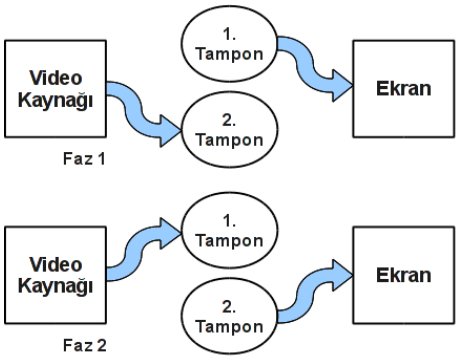
Şekil 6.6 Hız ayarlı filtrenin işlemci dizisinin basitleştirilmiş blok diyagramı.

dizisine aktarılır. İşlemci dizisi girişinde giriş görüntüsü HAF'nin B şablonunun orta elemanıyla b_{00} çarpılır, bu değer 4 iterasyon boyunca sabit kalır. Bu nedenle işlemci dizisi girişinde bir kere hesaplanır ve herbiri bir iterasyonu gerçekleyen işlemciler boyunca birbirlerine aktarılır ve kullanılır. İlk işlemcinin sonucunu hesaplamak için ihtiyaç duyduğu ilk koşullar, bir önceki çıkışın saklandığı çerçeve tampon belleğinden gelmektedir.

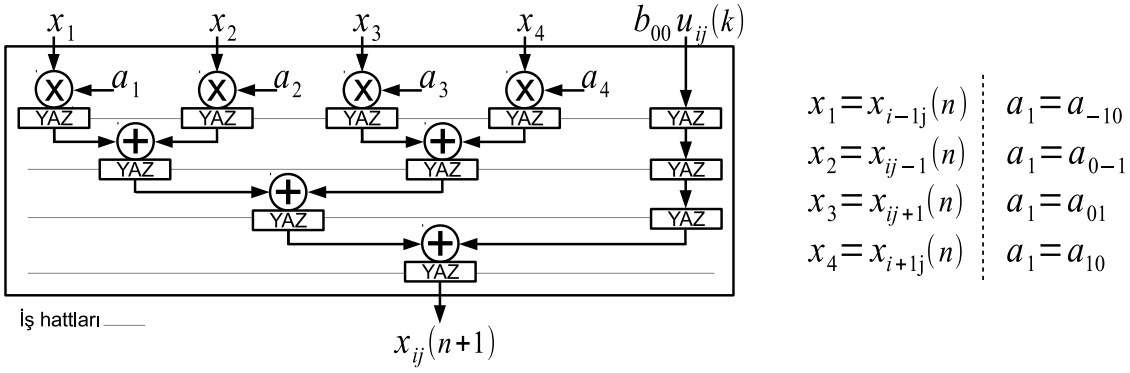
İşlemci sayısı (6.12)'de verildiği üzere 4'tür. Her işlemci kontrol sinyallerine göre çalışmakta ve gerekli işlemeleri yapıp bir sonraki işlemciye sonucu ve daha önce hesapalana sabit değeri iletir. 4. işlemcinin çıkışı video çıkış bloğuna aktarılırken aynı zamanda bir sonraki giriş için ilk işlemcinin ilk koşulu olması için çerçeve tampon belleğinde saklanır.

Şekil 6.7'de ikili tampon belleğin fazları verilmiştir. Birinci fazda video kaynağından gelen çerçeve ikinci tampon belleğe kaydedilirken eski çerçeve kullanılır. Çerçevenin sonunda bellek arayüzü ikinci faza geçer ve yeni çerçeve en eskisinin üzerine yazılırken bir önceki çerçeve kullanılır. Böylece eski çerçeve saklanmış olur ve çerçeveler arasında ilişki kuran algoritmalara uygun bir altyapı hazırlanır.

Şekil 6.7'de çalışma mantığı açıklanan ikili çerçeve tampon belleği, HAF gerçeklenirken birinci çerçeve tampon belleğe sonuç yazılırken, ikinci çerçeve tampon bellekten 1. işlemcinin ilk koşulu için okuma yapılır. Yazma işlemi 4.işlemcinin çıkışındaki kontrol sinyalleri ile kontrol edilirken, çerçeve tampon belleğinden okuma 1. işlemcinin girişindeki kontrol sinyalleriyle kontrol edilir. 4. işlemcinin çıkışındaki *vframe* kontol sinyali lojik '0' değerini aldığında yazma ve okuma yapılan çerçeve tampon bellekeri yer değiştirir.



Şekil 6.7 İkili çerçeve tampon belleğin fazları.



Şekil 6.8 Gerçek zamanlı HAF'ın işlemcisinin aritmetik işlem bloğu

6.4.2 Gerçek Zamanlı HAF İşlemci Bloğu

Hız ayarlı filtrenin gerçeklenmesinde her bir iterasyonu gerçeklemektedir. Gerçek zamanlı Gabor benzeri HSA filtresinde kullanılan işlemci bloğuyla neredeyse tamamen aynıdır. Yerel kontrol bloğu, veri ve sabit belleği, sınır koşul ve seri haberleşme blokları aynı kalamsına rağmen aritmektik işlem bloğu hız ayarlı filtre için değiştirilmiştir. Aritmetik işlem bloğu dışındaki blokların detayları 4. bölümde verilmiştir.

Hız ayarlı filtre işlemci bloğu ile Gabor benzeri HSA filtresi işlemci bloğu arasındaki fark aritmetik işlemlerin yapıldığı aritmetik işlem bloğudur. İş hattı mantığıyla tasarlanan aritmetik işlem bloğunun iç yapısı ve uç diyagramı Şekil 6.8'de verilmiştir. Durumlara ait girişler ile filtre katsayıları çarpılır ve sonuçlar ikili olarak toplanır ve sonuç çıkışa aktarılır. Sabit değeri iş hattının bozulmaması için 3 defa yazmaçtan geçirilmiştir. İşlemci bloğu Gabor filtresinden farklı olarak piksel saat frekansında çalışmakta ve toplam gecikmesi 4 piksel saati kadardır.

6.5 Sonuç

Bu bölümde hareketli video görüntüsüdeki belli bir doğrultuda ve sabit hızla hareket eden bir nesnenin algılanması amaçlanmıştır. Bunun için [19]'de sunulan analog devre incelenmiş ve [34]'deki sayısal yöntem kullanılarak benzetimler yapılmıştır. Benzetim sonucuna göre 4 iterasyon hız algılama için yeterli olduğu görülmüştür. Bu nedenle tasarlanan sayısal devrede her iterasyona karşılık 4 işlemci kullanılmıştır.

Benzetim, hız ayarlı filtrenin katsayılarının kaç bit ile ifade edilebileceği için tekrar edilmiştir. Bu benzetimler sonucunda durumun bit genişliği 5 bit, şablon katsayıları 8 bit sabit, ise 16 bit işaretli sayı olacak şekilde belirlenmiştir.

Her 4 iterasyondan sonra giriş görüntüsü değişmekte ve eski çıkışa ait değerler yeni görüntü için ilk koşul olarak kullanılmaktadır. Eski çıkışların yeni girişte kullanılabilmesi için bellekte saklanması gerekmektedir. 1080×1920 boyutlarındaki giriş için 2 Mbayt belleğe ihtiyaç vardır. Senkronizasyon sorunu için ikili çerçeve tampon belleği kullanıldığından dolayı 4 Mbayt belleğe ihtiyaç vardır. Bu büyüklükteki bellek FPGA tümdevresinin içinde bulunmadığından dolayı dış bellek elemanları kullanılması gerekmektedir. Bunun için FPGA geliştirme kitindeki DDR3-SDRAM ler kullanılmıştır.

Her işlemci bloğu 2 satır sakladığı için HAF'ın toplam giriş/çıkış gecikmesi 8 görüntü satırıdır. Her işlemcide bulunan aritmetik işlem bloğununun gecikmesi ise 4 piksel saat darbesi kadardır.

Hız ayarlı filtre bilgisayar ortamında test edilmiş ve hatasız olarak çalışmıştır. Ancak hareketli görüntü standartları ile ekran tazeleme standartları insan gözünü kandırmaya yönelik olarak tasarlandığından monitöre gönderilen DVI işaretinin arasına girerek bu tür bir filtreleme yapmak elimizdeki donanımla mümkün görülmemiştir.

BÖLÜM 7

SONUÇ VE ÖNERİLER

Bu tez kapsamında ilk olarak 108E023 numaralı TÜBİTAK projesindeki HSA gerçeklemelerinin ortak kısımlarının bir kısmı tasarlanmış ve gerçeklenmiştir. İlgili kısımlar bu tezde ele alınmış, diğer kısımlar ise bu tezin kapsamı dışında bırakılmıştır.

İkinci olarak bilgisayarla görme ve görüntü işleme uygulamalarında yön secici özelliğiyle ön işlem bloğu olarak kullanılan Gabor filtrelerinin hızlı ve verimli bir gerçeklemesi önerilmiştir. Sistemin mimarisi, birçok uygulamada farklı çözünürlüklerdeki giriş görüntülerin farklı yönlere ayarlanmış Gabor filtreleri ile işlenmesi gerektiği göz önüne alınarak tasarlanmıştır. Dolayısıyla yapı tekrar ayarlanabilir, ölçeklenebilir ve çalışma anında programlanabilir olacak şekilde tasarlanmış ve gerçeklenmiştir. Bu yapı aynı zamanda günümüze kadar gerçeklenmiş en hızlı Gabor benzeri filtresidir.

Son olarak hareketli giriş görüntülerindeki sabit hız ve doğrultuda hareket eden cisimleri algılayabilen hız ayarlı filtreler için bir gerçekleme önerilmiştir.

Tezler

1. Tez: Tasarımı ve Gerçeklemesi Yapılan Gerçek Zamanlı HSA Emülatörü Bloklarının Uygulanabilirliği

TÜBİTAK tarafından desteklenen proje kapsamında Kamer Kayaer tarafından tasarımı ve gerçeklemsi yapılan gerçek zamanlı HSA emülatörünün sahip olduğu; 1) Sabit çözünürlük, 2) tek bir model FPGA üzerinde çalışması, 3) çalışma anında programlanamaması, 4) işlemci sayısının sabit olması ve 5) merkezi kontrolün sistemi ağırlaştırması gibi dezavantajların giderilerek yeni nesil bir tasarım önerilmiştir. Bu noktada belirtilmelidir ki

yukarıdaki sorunların çözülmesi ile modüler, tekrar programlanabilir, tekrar kullanılabilir ve üreticiden/modelden bağımsız tüm FPGA'ler üzerinde çalışabilen bir gerçeklemenin yapılması bir ekip gerektirmektedir. İlgili çalışmaların bütünü 108E023 numaralı TÜBİTAK projesinin ekibi tarafından yürütülmüştür.

2. nesil gerçek zamanlı HSA emülatörünün tasarımındaki diferansiyel denkleminin ayrıştırılması aşaması 1. neslininki ile neredeyse aynıdır. 2. nesil HSA emülatörünün en büyük farkı, merkezi kontrol gerektirmeyen bir yerel kontrol mantığının önerilmiş olmasıdır. Bu sayede her işlemci merkezi bir blok tarafında değil, kendinden önceki işlemci tarafından kontrol edilir ve bu sayede emülatörün modülerliği ve tekrar kullanılabilirliği sağlanır. Tasarım tamamen iş hattı mantığıyla yapılmış ve VHDL dilinde gerçeklenmiştir. VHDL dilinin yetenekleri sayesinde de sentezleme öncesinde tasarımın çözünürlük, işlemci sayısı gibi parametreleri değiştirilebilmektedir.

Bu tez kapsamında 2. nesil gerçek zamanlı HSA emülatörünün: 1) Yerel kontrol mantığı ve kontrol sinyallerinin tasarımı, 2) satır tampon bellepğinin tasarımı ve gerçeklenmesi, 3) kontrol sinyallerinin senkronizasyon için geciktirilmesini sağlayan geciktirme bloğu tasarımı ve gerçeklemesi, 4) çalışma anında kullanılabilen seri haberleşme arayüzü ile bu arayüzün mesaj yapısı tasarlanmıştı.

Sonuç olarak tasarlanan emulatörün prototipi $1080 \times 1920@60Hz$ basit taramalı video işaretini gerçek zamanlı olarak işleyebilmektedir.

2. Tez: Tasarımı ve Gerçeklemesi Yapılan Gerçek Zamanlı Gabor Benzeri HSA Filtresi

Bertram E. Shi 1996 ve 1998 yıllarında yayınladığı iki makaleyle HSA ile gerçeklenebilen Gabor benzeri filtreleme yöntemini ortaya koymuştur. [35]'de bant geçiren bir filtrenin tek katmanlı HSA ile oluşturulabilmesi için şablon boyutunun en az 5×5 olması gerektiğini belirtmişlerdir. Shi, 3x3 şablonlu iki katmanlı bir HSA yapısı ile bant geçiren Gabor benzeri filtre gerçekleyebilmiştir. [35]'deki bant geçiren filtre için A ve B şablonları $5x5$ boyutlarındadır ve tüm şablon değerleri sıfırdan farklıdır. Sayısal olarak gerçeklemek istendiğinde toplamda 50 çarpma işlemi yapılması gerekecektir ve bir işlemci içinde 4

satırlık görüntü bilgisinin tutulması gerekecektir. Oysa Shi'nin önerdiği iki katmanlı ve 3×3 şablon boyutlarıan sahip Gabor benzeri filrenin bu tez kapsamında önerilen yapıda gerçeklenmesi ile sadece 4 çarpıcının kullanılması ve 2 satırlık görüntü bilgisinin tampon bellekte saklanması yeterlidir.

Gerçeklenen Gabor benzeri filtrenin yerel kontrolü, durum ve sabit tampon bellekleri, sınır koşulu üreteci ve seri programlama arayüzü olarak 2. nesil gercek zamanlı HSA emülatörünün blokları kullanılmıştır. Toplama ve çarpma işlemlerinin yapıldığı aritmetik işlem bloğu Gabor benzeri filtre için yeniden tasarlanıp gerçeklenmiştir. Ayrıca Gabor benzeri filtrenin çıkışının daha rahat algılanabilmesi için histogram hesaplama ve karşıtlık ayarlama blokları tasarlanmış ve gerçeklenmiştir.

Bu tez sırasında, FPGA içindeki çarpıcı ve bellek elemanlarının sayısının sonlu olması nedeniyle FPGA içine gömülebilecek Gabor işlemcisi sayısı önemli bir konu olmaktadır. Sonuca ulaşmak için gereken minimum iterasyon sayısıyla filtrenin bant genişliği arasında bir ilişki var olduğundan dolayı FPGA içinde gerçeklenebilen işlemci sayısının bilinmesi önemlidir. 4. bölümde iterasyon sayısıyla bant genişliği arasındaki ilişki bir benzetim ile deneysel olarak ortaya konmulştur. Bu sayede gerçeklenen sistemin sınırları belirlenebilmektedir.

Prototip bir sistemde 60 Euler iterasyonunu 3 paralel hatta 20'şer işlemci kullanacak şekilde gerçeklenmiştir. Bu durumda $1080 \times 1920@60Hz$ basit taramalı video işareti için giriş çıkış toplam gecikmesi 40 video görüntüsü satırı, yani yaklaşık $592.6\mu s$, işlemcilerin saat frekansları ise 297 MHz'dir. Prototip sistem tamamen iş hattı mantığı ile tasarlandığı için her saat darbesinde tüm çarpıcı ve toplayıcılar çalışmaktadır, bu durumda saniyedeki toplam çarpma ve toplama işlem sayısı yaklaşık 143.5 milyardır, ki süper bilgisayar hız sınırı olan 10^{12}'nin yaklaşık 6'da biridir. Saniyede işlenen piksel sayısı karanlık bölgeler hariç 124.4 milyon, karanlık bölgeler dahilken ise 148.5 milyondur.

3. Tez: Tasarımı Yapılan Hız ayarlı Filtre

[19]'de sunulan analog devre incelenmiş ve [34]'deki sayısal yöntem kullanılarak benzetimler yapılmıştır. Benzetim sonucuna göre 4 iterasyon hız algılama için yeterlidir, dola-

yısıyla tasarlanan sayısal devrede yalnızca 4 HSA işlemcisi kullanılmıştır. Çıkışların bir sonraki çerçeve ile beraber yeni giriş olarak kullanılabilmesi için bellekte saklanması gerekmektedir. 1080×1920 boyutlarındaki giriş için en az 2 Mbayt belleğe ihtiyaç vardır. Senkronizasyon sorunu için ikili çerçeve tampon belleği kullanıldığından dolayı 4 Mbayt belleğe ihtiyaç duyulur. Bu büyüklükteki bellek FPGA tümdevresinin içinde bulunmadığından dolayı dış bellek elemanları kullanılması gerekmektedir. Bunun için FPGA geliştirme kitindeki DDR3 SDRAM'ler kullanılmıştır.

Hız ayarlı filtre bilgisayar ortamında test edilmiş ve hatasız olarak çalışmıştır. Ancak hareketli görüntü standartları ile ekran tazeleme standartları insan gözünü kandırmaya yönelik olarak tasarlandığından dolayı monitöre gönderilen DVI işaretinin alınmasıyla doğrudan işlemede bazı sorunlar çıkmaktadır. Dolayısıyla bu sistemin çalışan bir prototipi bulunmamaktadır.

Gelecek Çalışmalar

İleride Gabor benzeri filtrelerin el yazısı tanıma gibi bir uygulamasının bir FPGA tümdevresi üzerinde gerçeklenmesi hedeflenmiştir. Ayrıca hız ayarlı filtrelerdeki video sinyal kaynaklarıyla ilgili sorunların giderilmesi için bir video test sinyali üreticisinin tasarlanması ve gerçeklenmesi ile tez kapsamında yaşanan test problemleri ortadan kalkacaktır.

KAYNAKLAR

[1] Gabor, D., (1946). "Theory of communication. Part 1: The analysis of information", Electrical Engineers - Part III: Radio and Communication Engineering, Journal of the Institution of, 93: 429-441.

[2] Daugman, J. G., (1985). "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters", Journal of the Optical Society of America A, 2: 1160-1169.

[3] Clark, M., Bovik, A. ve Geisler, W., (1987). "Texture segmentation using a class of narrowband filters", Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '87., April 1987.

[4] Jain, A. K. ve Bhattacharjee, S., (1992). "Text segmentation using Gabor filters for automatic document processing", Mach. Vision Appl., 5: 169-184.

[5] Porat, M. ve Zeevi, Y., (1989). "Localized texture processing in vision: analysis and synthesis in the Gaborian space", Biomedical Engineering, IEEE Transactions on, 36: 115-129.

[6] Mehrotra, R., Namuduri, K. R. ve Ranganathan, N., (1992). "Gabor filter-based edge detection", Pattern Recognition, 1479-1494.

[7] Daugman, J., (1988). "Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression", Acoustics, Speech and Signal Processing, IEEE Transactions on, 36: 1169-1179.

[8] Heeger, D. J., (1987). "Model for the extraction of image flow", J. Opt. Soc. Am. A, 4: 1455-1471.

[9] Casasent, D. P., Smokelin, J. S. ve Ye, A., (1992). "Wavelet and Gabor transforms for detection", Optical Engineering, 31: 1893-1898.

[10] Super, B. J. ve Bovik, A. C., (1991). "Three-dimensional orientation from texture using Gabor wavelets", 574-586.

[11] Marčelja, S., (1980). "Mathematical description of the responses of simple cortical cells$*$", J. Opt. Soc. Am., 70: 1297-1300.

[12] Shi, B., (1998). "Gabor–type filtering in space and time with cellular neural networks", Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 45: 121-132.

[13] Chua, L. ve Yang, L., (1988). “Cellular neural networks: theory”, Circuits and Systems, IEEE Transactions on, 35: 1257-1272.

[14] Wang, X. ve Shi, B., (2010). “GPU implemention of fast Gabor filters”, Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, June 2010.

[15] Norouznezhad, E., Bigdeli, A., Postula, A. ve Lovell, B., (2010). “Robust object tracking using local oriented energy features and its hardware/software implementation”, Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on, December 2010.

[16] Cho, Y., Bae, S., Jin, Y., Irick, K. ve Narayanan, V., (2011). “Exploring Gabor Filter Implementations for Visual Cortex Modeling on FPGA”, Field Programmable Logic and Applications (FPL), 2011 International Conference on, September 2011.

[17] Liu, baoJ., Wang, S., Li, Y., Han, J. ve Zeng, yangX., (2010). “Configurable Pipelined Gabor Filter implementation for fingerprint image enhancement”, Solid-State and Integrated Circuit Technology (ICSICT), 2010 10th IEEE International Conference on, November 2010.

[18] Cheung, O., Leong, P., Tsang, E. ve Shi, B., (2006). “A Scalable FPGA Implementation of Cellular Neural Networks for Gabor-type Filtering”, Neural Networks, 2006. IJCNN ’06. International Joint Conference on, July 2006.

[19] Torralba, A. ve Herault, J., (1999). “An efficient neuromorphic analog network for motion estimation”, Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 46: 269-280.

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

[21] Yıldız, N., (2013). Design of a Cellular Neural Network Emulator and its Implementation on an FPGA Device, Doktora Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

[22] Alpay, M., (Bitmedi). Çok Katmanlı Bir Hücresel Sinir Ağı Emülatörünün FPGA Mimarisinin Tasarımı ve Gerçeklemesi, Doktora Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

[23] Tavsanoglu, V. ve Saatci, E., (2000). “Feature extraction for character recognition using Gabor-type filters implemented by cellular neural networks”, Cellular Neural Networks and Their Applications, 2000. (CNNA 2000). Proceedings of the 2000 6th IEEE International Workshop on, 2000.

[24] Wang, Y. ve Zhu, S.-C., (2004). “Analysis and synthesis of textured motion: particles and waves”, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 26: 1348-1363.

[25] Saatci, E., Cesur, E., Tavsanoglu, V. ve Kale, I., (2007). "An FPGA implementation Of 2-D CNN gabor-type filter", Circuit Theory and Design, 2007. ECCTD 2007. 18th European Conference on, August 2007.

[26] Cesur, E., Yildiz, N. ve Tavsanoglu, V., (2012). "On an Improved FPGA Implementation of CNN-Based Gabor-Type Filters", Circuits and Systems II: Express Briefs, IEEE Transactions on, 59: 815-819.

[27] Saatci, E. ve Tavsanoglu, V., (2002). "On the optimal choice of integration time-step for raster simulation of a CNN for gray level image processing", Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on, February 2002.

[28] Chua, L. ve Roska, T., (2002). Cellular neural networks and visual computing: foundation and applications, Cambridge University Press.

[29] Drewery, J., Dept, B. R. ve Dept, B. B. C. E. D. R., (1978). The Zone Plate as a Television Test Pattern, Research Department, Engineering Division, BBC.

[30] Yildiz, N., Cesur, E. ve Tavsanoglu, V., (2010). "A new control structure for the pipelined CNN processor arrays", Cellular Nanoscale Networks and Their Applications (CNNA), 2010 12th International Workshop on, February 2010.

[31] Cesur, E., Yildiz, N. ve Tavsanoglu, V., (2010). "Architecture of The Next Generation Real Time CNN Processor: RTCNNP-v2", Nonlinear Theory and its Applications (NOLTA), 2010 International Symposium on, September 2010.

[32] Kayaer, K. ve Tavsanoglu, V., (2008). "A new approach to emulate CNN on FPGAs for real time video processing", Cellular Neural Networks and Their Applications, 2008. CNNA 2008. 11th International Workshop on, July 2008.

[33] Cesur, E., Yildiz, N. ve Tavsanoglu, V., (2012). "Demo: An improved FPGA implementation of CNN Gabor-type Filters", Cellular Nanoscale Networks and Their Applications (CNNA), 2012 13th International Workshop on, August 2012.

[34] Polat, S. N. T., (2010). Uzaysal ve uzay-zamansal hücresel sinir ağı filtreleri, Doktora Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

[35] Crounse, K. ve Chua, L., (1995). "Methods for image processing and pattern formation in Cellular Neural Networks: a tutorial", Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 42: 583 -601.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Evren CESUR

Doğum Tarihi ve Yeri : 12/09/1980 Fatih

Yabancı Dili : İngilizce

E-posta : evrencesur@yahoo.com

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Y.Lisans	Elektronik ve Hab. Müh.	Yıldız Teknik Üni.	2006
Lisans	Elektrik-Elektronik Müh.	Sakarya Üni.	2002
Lise	Fen-Mat	Cibali Lisesi	1997

İŞ TECRÜBESİ

Yıl	Firma/Kurum	Görevi
2009-2012	Yıldız Teknik Üni.	Araştırma Görevlisi
2008-2009	PAVO Tasarım Üretim Elektronik A.Ş	Uzman AR-GE mühendisi
2005-2008	Yıldız Teknik Üni.	Araştırma Görevlisi

YAYINLARI

Makale

1. Cesur, E., Yildiz, N. ve Tavsanoglu, V., (2012). "On an Improved FPGA Implementation of CNN-Based Gabor-Type Filters," Circuits and Systems II: Express Briefs, IEEE Transactions on, Early Access

Bildiri

1. Yildiz, N., Cesur, E. ve Tavsanoglu, V., (2012). "Demonstration of the Second Generation Real-Time Cellular Neural Network Processor: RTCNNP-v2," 12th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA), aug. 2012

2. Cesur, E., Yildiz, N. ve Tavsanoglu, V., (2012). "Demo: An improved FPGA implementation of CNN Gabor-type Filters," 12th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA), aug. 2012

3. Cesur, E., Yildiz, N. ve Tavsanoglu, V., (2011). "An improved FPGA implementation of CNN Gabor-type filters," IEEE International Symposium on Circuits and Systems (ISCAS), may. 2011

4. Cesur, E., Yildiz, N. ve Tavsanoglu, V., (2010). "Architecture of The Next Generation Real Time CNN Processor: RTCNNP-v2," International Symposium on Nonlinear Theory and its Applications (NOLTA), sept. 2010

5. Yildiz, N., Cesur, E. ve Tavsanoglu, V., (2010). "A new control structure for the pipelined CNN processor arrays," 12th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA), feb. 2010

6. Saatci, E., Cesur, E., Tavsanoglu, V. ve Kale, I., (2007). "An FPGA implementation Of 2-D CNN gabor-type filter," 18th European Conference on Circuit Theory and Design (ECCTD), aug. 2007

Proje

1. Bursiyer, "Durağan Ve Video Görüntü İşleyen Hücresel Sinir Ağı Yapısının Yeni Bir Fpga Mimarisi İle Tasarım Ve Gerçeklemesi," TÜBİTAK, 108E023, 2009–2011

2. Bursiyer, "Parmak izi tanıyan hızlı bir sistemin hücresel analog işlemci dizileri kullanarak tasarımı ve uyarlanması," Yıldız Teknik Üniversitesi BAPK, 25-04-03-01, 2005–2008