Dicle Üniversitesi Fen Bilimleri Enstitüsü Dergisi

Dicle University Journal of the Institute of Natural and Applied Science

https://dergipark.org.tr/tr/pub/dufed

Araştırma Makalesi / Research Article

2D Konvolüsyonun İşleminin Düşük Maliyetli IP Çekirdek Olarak FPGA Tabanlı Gerçeklenmesi

FPGA Based Implementation of 2D Convolution Processing as a Low-Cost IP Core

Mehmet Ali ÇAVUŞLU

Kocaeli Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Gömülü Sistemler Laboratuvarı, 41380, Kocaeli, Türkiye

MAKALE BILGISI

Makale Tarihi

Alınış, 09 Nisan 2021
Revize, 18 Ekim 2021
Kabul, 14 Kasım 2021
Online Yayınlama, 06 Aralık 2021

Anahtar Kelimeler

FPGA, Görüntü İşleme, 2D Konvolüsyon

ÖZ

Bu çalışma kapsamında görüntü işleme uygulamalarında sıklıkla tercih edilen iki boyutlu konvolüsyon işleminin düşük maliyetli IP çekirdek olarak FPGA tabanlı gerçeklenmesi anlatılmıştır. Çalışma kapsamında geliştirilen IP çekirdek ile görüntü üzerinde yatay/dikey Sobel, yatay/dikey Prewitt, kaydır çıkart, alçak geçiren filtre, yüksek geçiren filtre ve Gauss filtre işlemleri kullanıcı tarafından ayarlanan parametre ile kolaylık gerçeklenebilmektedir. IP çekirdek platform bağımsız olarak tasarlanmıştır ve tüm FPGA üreticileri tarafından geliştirilen yazılımlarda sentezlenebilmektedir. IP çekirdeğine ait sentez sonuçları Xilinx firmasının Artix 7 100T FPGA'sı referans alınarak verilmiştir. Sentez sonuçları çalışma kapsamında geliştirilen iki boyutlu konvolüsyon IP çekirdeğinin düşük donanım maliyeti ile FPGA tabanlı gerçeklendiğini göstermiştir.

ARTICLE INFO

Article History

Received, 09 April 2021
Revised, 18 October 2021
Accepted, 14 November 2021
Available Online, 06 December 2021

Keywords

FPGA, Image processing, 2D Convolution

ABSTRACT

In this study, FPGA-based implementation of two-dimensional convolution process, which is frequently preferred in image processing applications, as a low-cost IP core is explained. With the IP core developed within the scope of the study, horizontal/vertical Sobel, horizontal/vertical Prewitt, slide-out, low pass filter, high pass filter, and Gaussian filter operations on the image can be realized easily with the parameter set by the user. The IP core platform is designed independently and can be synthesized in software developed by all FPGA manufacturers. Synthesis results of the IP core are given by taking Artix 7 100T FPGA of Xilinx as a reference. The results of the synthesis showed that the two-dimensional convolution IP core developed within the scope of the study was realized FPGA-based with low hardware cost.

E-posta Adresi: alicavuslu@gmail.com (Mehmet Ali ÇAVUŞLU)

1. GİRİŞ

Yarı iletken teknolojilerindeki gelişmelerin son zamanlarda hız kazanması ile yüksek işlem yükü gerektiren görüntü işleme ve video işleme gibi uygulamaların tek bir donanım üzerinde gerçeklenebilmesi mümkün hale gelmiştir. Birçok alanda uygulanan tek bir donanım üzerinde gerçekleştirilen uygulamalara örnek olarak yüz tanıma, plaka tanıma, araç içerisinde sürücü yorgunluğu verilebilir [1-6].

Günümüzde kamera, fotoğraf makinası gibi görüntü algılayan sistemlerde kullanılan lens teknolojilerindeki gelişmeler ile yüksek çözünürlüklü görüntüler alınabilmektedir. Sistem tarafından alınan görüntünün çözünürlüğü ile analizleri için gerekli işlem yükü doğru orantılı olarak değişmektedir.

Literatürde geleneksel işlemci üzerinde gerçek zamanlı görüntü işleme uygulamalarının gerçeklenmesine yönelik birçok çalışma sunulmuştur. Geleneksel işlemciler üzerinde uygulamalar yazılımsal olarak seri mimari ile gerçeklenebilmektedir. Bu nedenle hızlı bir şekilde çıktı üretmesi beklenen uygulamaların geleneksel işlemciler ile gerçeklenmesi zor problemlerden biri olmaktadır.

Son zamanlarda yoğun işlem yükü gerektiren uygulamalarda paralel veri işlem yapabilme ve veri akışı (pipelining) özelliklerini donanım mimarisine aktarabilen FPGA'lar, yoğun işlem yüküne sahip ve kısa süre içerisinde çıktı üretmesi beklenen uygulamaların gerçeklenebildiği platform haline gelmiştir [7,8].

Yoğun işlem yükü gerektiren uygulamaların başında gelen görüntü işleme uygulamalarında da literatürde FPGA sıkılıkla tercih edilen platformlarından biridir [7,8]. Çalışma [1]'de, plaka tanıma sistemi FPGA üzerinde SoC uygulaması ile gerçek zamanlı olarak gerçeklenmiştir. Çalışma kapsamında görüntü işlemede kullanılan yatay/dikey maskeleme ve ikili imge dönüşümü algoritmaları kullanılmıştır. Çalışma [3]'te, sürücülerde gözlerin kapalı olma durumu algılanarak yorgunluk takibi işlemleri FPGA tabanlı gerçeklenmiştir. Gerçeklemelerde gri tonlama projeksiyonu, kenar bulma (Prewitt operatörü kullanarak) gibi görüntü işleme algoritmalarını kullanmışlardır. Çalışma [5]'te, gerçek zamanlı kenar bulma algoritmalarını FPGA tabanlı gerçeklenmiştir. Çalışma [8]'de imge üzerinde kenar tespiti işlemleri FPGA tabanlı olarak gerçekleştirilmiştir. Yerel ve küresel eşik (LGT, Local and Global Threshold) ve mutlak mesafe maskesi algoritmaları kenar tespiti için kullanılmıştır. Çalışma [9]'da FPGA üzerinde plaka tespiti uygulaması gerçekleştirmişlerdir. Çalışmada kullanılan gri imgeye dönüştürme, morfolojik işlemler ve arka plan çıkarma yöntemleri donanıma aktarılmıştır. Çalışma [10]'da plaka yeri bulma işlemlerinin FPGA'da donanımsal gerçeklenmesi aşamasında Sobel parametreleri kullanarak kenara tespiti, ikili imgeye dönüştürme işlemleri gerçekleştirilmiştir. Çalışma [11]'de yine plaka yeri bulma işlemi FPGA tabanlı gerçekleştirilmiştir. Görüntü işleme algoritmaları

olarak filtreleme, yatay/dikey kenar bulma ve ikili imgeye dönüştürme işlemleri gerçekleştirilmiştir. Çalışma [12]'de ortanca filtre, genişletilmiş sıralama modülü, yumuşatma filtresi, Sobel parametreleri ile kenar bulma, hareket bulanıklığı yöntemleri FPGA tabanlı gerçekleştirilmiştir. Çalışma [13]'te FPGA tabanlı gerçekleştirilen yüz tanıma işlemlerinde FFT ve renk uzay dönüşümü işlemleri gerçekleştirilmiştir. Çalışma [14]'te ortanca filtre, hızlı ortanca filtre yöntemleri görüntü işleme uygulamasında FPGA tabanlı olarak gerçekleştirilmiştir. Çalışma [15]'te görüntü işleme yöntemlerinden skaler toplama, Prewitt ve Canny yöntemleri kullanarak kenar bulma, dalgacık dönüşümünü FPGA tabanlı gerçeklemiştir. Çalışma [16]'da 3 farklı yöntem kullanılarak iki boyutlu konvolüsyon işlemini FPGA üzerinde donanımsal gerçeklemişlerdir. Çalışma [17]'de gauss filtre ve kenar bulma işlemlerini farklı çözünürlükteki görüntü üzerinde, [18]'de ortanca filtre ve hızlı ortanca filtre FPGA tabanlı gerçekleştirmişlerdir.

Bu çalışmada yukarıda sunulan literatür çalışmalarının veya benzer çalışmaların FPGA tabanlı gerçekleştirilebilmesini kolaylaştırmak amacıyla iki boyutlu konvolüsyon işlemi, kullanıcı tarafından seçilebilen yönteme göre gerçekleştirilmektedir. Çalışma kapsamında aşağıda listelenen yöntemleri tek bir çatı altında toplanmış, genelleştirilmiş mimari anlayışı ile IP çekirdeği olarak tasarlanmış, platform bağımsız, düşük maliyetli olarak FPGA tabanlı gerçeklenmiştir.

- Yatay/dikey Sobel,
- Yatay/dikey Prewit,
- Kaydır çıkart,
- Alçak geçiren filtre,
- Yüksek geçiren filtre,
- Gauss filtre

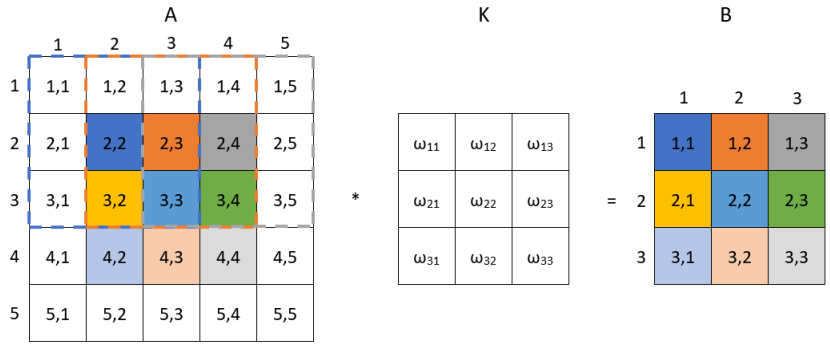
2. İKİ BOYUTLU KONVOLÜSYON İŞLEMİ

Sinyal işleme uygulamalarında olduğu gibi görüntü işleme uygulamalarında da sıkılıkla kullanılan konvolüsyon işlemi giriş imgesinin tabi tutulacağı yönteme ait parametrelerle gerçekleştirilir. Görüntü üzerinde konvolüsyon, işlem yapılacak pikselin komşuluğunda bulunan piksel değerleri ile gerçekleştirilmektedir. Denklem (1)'de iki boyutlu konvolüsyon işlemi gösterilmiştir. Denklem (1)'de A giriş imgesini, K konvolüsyon kernelini ve B konvolüsyon işlemi sonucundaki imgeyi göstermektedir. x ve y giriş imgesinde işlem uygulanacak pikselin satır ve sütun indislerini, i ve j işlem yapılan kernelin satır ve sütun indislerini göstermektedir. M ve N ise kernel uzunluğunun satır ve sütun değerlerini göstermektedir.

$$B(x, y)=\sum_{i=1}^{M}\sum_{j=1}^{N} K(i,j)A(x-i+M-1, y-j+N-1) \quad (1)$$

Konvolüsyon işlemleri imgeye iki şekilde uygulanmaktadır. Birinci uygulamada, Denklem 1'den de görüleceği üzere komşuluk ilişkilerinden dolayı ilk ve son satırlarda/sütunlarda bulunan piksel değerleri için bu işlemleri gerçekleştirememekteyiz.

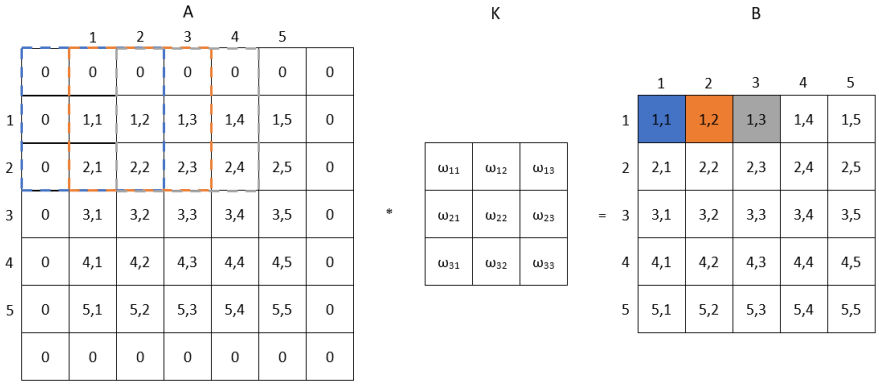
Örneğin X satırlı, Y sütunlu imge üzerinde 3x3'lük bir kernel ile gerçekleştirilecek konvolüsyon ile, (1,1)'den (1, Y)'e, (X,1)'den – (X, Y)' e, (1,1)' den - (X,1)'e ve (1,Y)'den - (X,Y)'e kadar olan pikseller için işlemler gerçekleştirilemeyecektir. Konvolüsyon işlemleri sonucunda elde edilecek imgenin boyutlarında azalma meydana gelecektir. Şekil 1'den de görüleceği üzere 3x3'lük bir kernel 5x5'lik imge üzerine uygulandığında çıkışta 3x3'lük imge elde edilmektedir. İmge satır ve sütün boyutunda azalma kernel boyutuna bağlı olarak değişkenlik göstermektedir.



Şekil 1. *Konvolüsyon Uygulama Yöntemi - 1*

İkinci uygulamada ise X satırlı, Y sütunlu imge üzerinde 3x3'lük bir kernel ile gerçekleştirilecek konvolüsyon için giriş imgesinin boyutlar X+2 ve Y+2 olacak şekilde yatay ve dikey kenarlara 0 eklenmektedir. Böylece giriş imgesi üzerinde (1,1)'den (1, Y)'e, (X,1)'den – (X, Y)' e, (1,1)'den - (X,1)'e ve (1,Y)'den - (X,Y)'e kadar olan pikseller içinde işlemleri gerçekleştirlmekte ve çıkış imgesinin boyutu değişmemektedir. Şekil 2'den de görüleceği üzere 5x5'lik giriş imgesi 0 eklemeleri yapılarak boyutu 7x7'e çıkmıştır. Yeni imge üzerine 3x3'lük bir kernel uygulandığında çıkışta 5x5'lik imge edilmektedir. Giriş imgesine eklenecek satır ve sütün değerleri kernel boyutuna bağlı olarak değişkenlik göstermektedir.

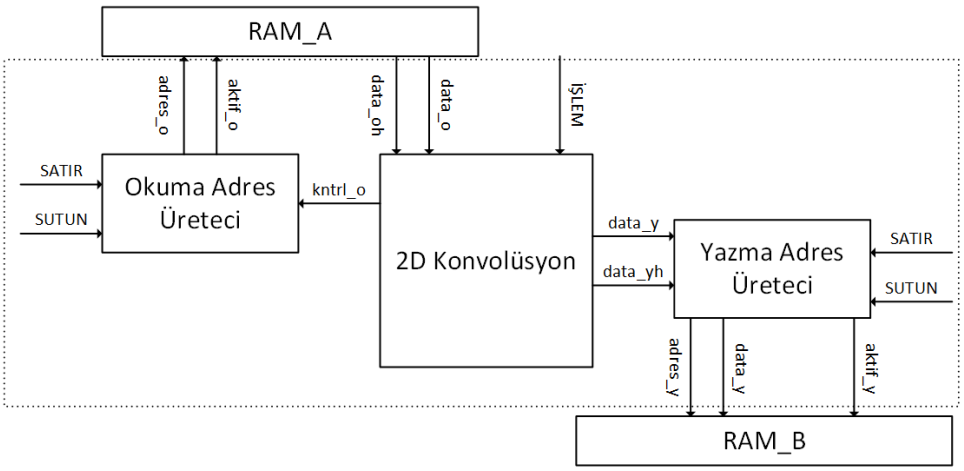
Bu çalışma kapsamında geliştirilen IP çekirdeği 1. yönteme uygun olarak tasarlanmıştır.



Şekil 2. *Konvolüsyon Uygulama Yöntemi - 2*

3. IP ÇEKİRDEĞİN FPGA TABANLI GERÇEKLEMESİ

Şekil 3'te çalışma kapsamında gerçekleştirilen 2D konvolüsyon IP çekirdeğinin çalışma akışı gösterilmiştir. IP tasarımı 3 bloktan meydana gelmektedir. **Okuma Adres Üreteci** ve **Yazma Adres Üreteci** bloğu kullanıcı tarafından jenerik parametre olan imgenin satır ve sütun sayısının tanımlandığı **SATIR** ve **SUTUN** parametrelerini referans alarak parametre üretmektedir. Bu bloklar **2D Konvolüsyon** tarafından aktif edildikten sonra hafıza biriminden okunacak/yazılacak veri ile ilgili adres değerini üretir.



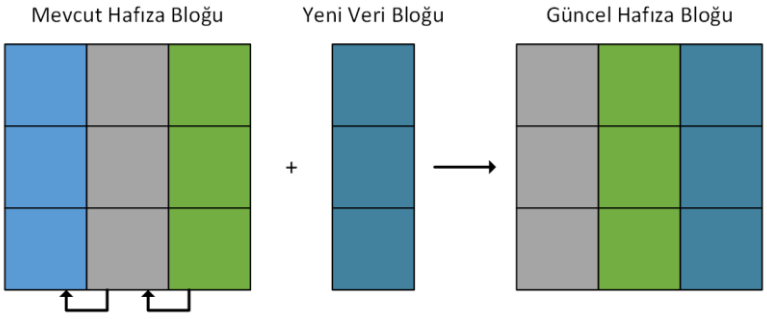
Şekil 3. *Konvolüsyon IP Tasarımı Blok Şeması*

Konvolüsyon işlemlerinde her bir piksel değerinin hesaplanması için kernel boyut kadar piksel değerinin hafıza biriminden okunması gerekmektedir. Örneğin 3x3'lük bir kernel için toplam 9 piksel değerinin hafızadan okunması gerekmektedir. Yan yana bulunan iki piksel değeri için okunması gereken 6 adet piksel değeri aynı olmaktadır. Bu doğrultuda tekrar okuma yapmamak amacı ile çalışmada Okuma Adres Üreteci birimi sadece aynı sütunda 3 piksel değeri okuyacak şekilde tasarlanmıştır. **2D**

Konvolüsyon işlem bloğunda konvolüsyon işleminde kullanılacak verilerin saklandığı hafıza bloğu kernel boyutu ile eşlenik olarak kaydırmalı saklayıcı şeklinde tasarlanmıştır (Şekil 4).

Şekil 4'ten de görüleceği üzere okunan güncel verilerin hafıza bloğuna yazılma işlemlerinde blokta bulunan veriler sütunda kaydırılmakta ve en sağdaki sütun verileri bloktan çıkarılmaktadır. Güncel veriler ise hafıza bloğunun en solundaki sütuna yazılarak hafıza bloğu güncellenmektedir. Hafıza bloklarından okunacak verilerin uzunlukları piksel değerlerinin kaç bit ifade edileceğine bağlı olarak kullanıcı tarafından tanımlanacak **VERI_UZUNLUGU** parametresi ile belirlenmektedir.

Adres üreteci bloklar tarafından gerçekleştirilecek işlemlerde imge piksel değerlerinin kaç bit ile ifade edeceğine göre adresler oluşturulmaktadır. İmgelerin saklanacağı hafıza birimlerinde her bir adres değerinde 8 bitik (1 Byte) veri saklanabilmektedir. Çalışma kapsamında geliştirilen IP ile piksel değerlerinin saklanacağı Byte değerleri Denklem (2)'deki gibi hesaplanmaktadır. Denklem (2)'de **Q** piksel değeri için hafıza biriminde kaç Byte'lık alan ayrılması gerektiğini göstermektedir. Örneğin 3,6 ve 8 bit uzunluğu için **Q** 1 Byte, 9, 13 ve 16 bit uzunluğu için Q 2 Byte olmaktadır.



Şekil 4. *Hafıza Bloğuna Verilerin Yerleştirilme İşlemleri*

$$Q=\text{ceil}\left(\log_8 \text{VERI\_UZUNLUGU}\right) \quad (2)$$

Okuma ve yazma işlemlerinde adres üretim işlemleri Denklem (3) ve Denklem (4)'deki gibi gerçekleştirilmektedir. Denklem (3)'de **adres_o** ve **adres_y** sırası ile okunacak ve yazılacak adres değerlerini göstermektedir. **ni** satırda indisini, **nj** sütun indisini göstermektedir. **SUTUN** parametresi ise kullanıcı tarafından imge sütun sayısını göstermektedir.

$$\text{adres\_o}=Q(ni*SUTUN+nj) \quad (3)$$

$$\text{adres\_y}=Q(ni*(SUTUN-2)+nj) \quad (4)$$

2D Konvolüsyon işlem bloğu ile kullanıcı tarafından ISLEM parametresi ile belirlenebilecek kerneller Tablo 1'de listelenmiştir. Tablo 1'den de görüleceği üzere 8 farklı yönteme uygun olarak

işlemler gerçekleştirilebilecektir. Çıkış değerlerinin yazma birime aktarılması sırasında sadece Gauss kerneli sonucu 16'ya bölünmektedir. Tablo 1'de gösterilen kerneller geliştirilebilmektedir.

Kernel tanımlamaları için oluşturulan tip tanımlamaları ve kernel değerlerinin sabit olarak tanımlamaları Şekil 5'te gösterilmiştir.

```
type K_SUTUN is array (0 to 2) of integer;
type K_MATRIS is array (0 to 2) of K_SUTUN;

type KERNEL array (0 to 7) of K_MATRIS;
constant c_KERNEL : KERNEL;
```

Şekil 5. *Kernel Tip Tanımlamaları*

Kullanıcı tarafından ayarlanabilecek parametreler oluşturulan IP blok tasarımında varlık tanımlamalarına eklenmiştir ve Şekil 6'da ***generic*** tanımlama içerisinde gösterilmiştir. ISLEM parametresi geliştirilen uygulamada 8 adet kernel olduğu için maksimum 8 değer alacak şekilde tasarlanmıştır. Kernel sayısının arttırılması durumunda bu aralığın güncellenmesi yeterli olacaktır.

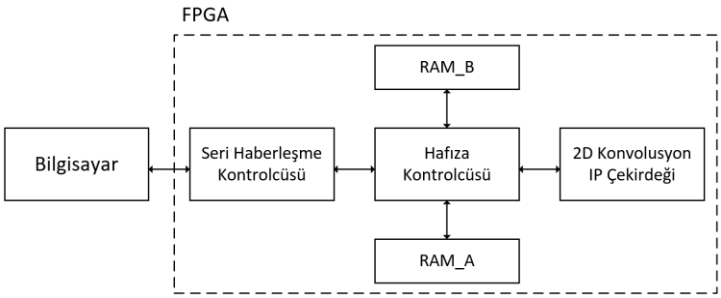
Tablo 1. ISLEM parametresi ile gerçekleştirilen kerneller

İşlem	Kernel			İşlem	Kernel		
Yatay Sobel	1	2	1	**Kaydır Çıkart**	0	0	0
	0	0	0		0	1	0
	-1	-2	-1		0	0	-1
Dikey Sobel	1	0	-1	**Alçak Geçiren Filtre**	0	1	0
	2	0	-2		1	0	1
	1	0	-1		0	1	0
Yatay Prewitt	1	1	1	**Yüksek Geçiren Filtre**	-1	-1	-1
	0	0	0		-1	8	-1
	-1	-1	-1		-1	-1	-1
Dikey Prewitt	1	0	-1	**Gauss**	1	2	1
	1	0	-1		2	4	2
	1	0	-1		1	2	1

```
generic(
    SATIR : integer;
    SUTUN : integer;
    ISLEM : std_logic_vector(2 downto 0);
    VERI_UZUNLUGU : integer );
```

Şekil 6. *IP Tasarımına ait generic yapısı*

Şekil 7'de çalışma kapsamında geliştirilen IP çekirdeğin FPGA tabanlı gerçekleştirilen testlerini tanımlayan blok şema gösterilmiştir. Testlerin ilk adımında kullanılacak imgeye ait piksel değerleri 1 Byte veri olarak MATLAB programı ile okunduktan sonra yine aynı program vasıtası ile seri haberleşme üzerinden FPGA'ya aktarılmaktadır.



Şekil 7. *IP bloğunun FPGA tabanlı gerçekleştirilen testlerine ait blok şema*

İmgeye ait tüm piksel değerleri FPGA üzerinde **Seri Haberleşme Kontrolcüsü** birimi ile alındıktan sonra **Hafıza Kontrolcüsü** birimine aktarılmaktadır. **Hafıza Kontrolcüsü** birimi ile alınan tüm piksel değerleri hafıza bloğuna (Blok RAM) yazılmaktadır.

Hafıza bloğuna yazma işlemleri tamamlandıktan sonra konvolüsyon işlemi başlatılmak üzere IP çekirdeğe başla komutu iletilir. IP çekirdek imge üzerinde işlem yapacağı piksellere ait adres değerlerini üreterek **Hafıza Kontrolcüsü**'ne iletir. **Hafıza Kontrolcüsü** adrese ait piksel değerini IP bloğuna iletir ve IP çekirdek tarafından konvolüsyon işlemi sonucunda elde edilen piksel değeri üretilir. Bu değer **Hafıza Kontrolcüsü** vasıtası ile IP çekirdek tarafından üretilen hafıza adresine yazılmaktadır. Tüm piksel değerleri için bu işlemler tekrarlandıktan sonra **Seri Haberleşme Kontrolcüsü** ile konvolüsyon işlemi sonucunda elde edilen imge bilgisayara aktarılarak dosyaya yazılmaktadır. Dosyada bulunan imge bilgileri MATLAB programı ile görselleştirilmektedir.

Çalışma kapsamında geliştirilen IP bloğun Şekil 7'de verilen akışa göre FPGA tabanlı gerçeklenmesi ile elde edilen sonuçlar Şekil 8'de gösterilmiştir. Şekil 8a'da orijinal imge, Şekil 8b'de yatay Sobel kerneli uygulanarak elde edilen yeni imge, Şekil 8c'de dikey Sobel kerneli uygulanarak elde edilen yeni imge, Şekil 8d'de yatay Prewitt kerneli uygulanarak elde edilen yeni imge, Şekil 8e'de dikey Prewit kerneli uygulanarak elde edilen yeni imge, Şekil 8f'de kaydır çıkart kerneli uygulanarak elde edilen yeni imge, Şekil 8g'de alçak geçiren filtre kerneli uygulanarak elde edilen yeni imge, Şekil 8h'de yüksek geçiren filtre kerneli uygulanarak elde edilen yeni imge ve Şekil 8i'de Gauss kerneli uygulanarak elde edilen yeni imge gösterilmiştir.

IP bloğu kullanarak FPGA üzerinde ve MATLAB 2020b programı ile 512x512 boyutunda imge üzerinde Sobel filtresinin uygulanması için gerekli işlem sürelerinin karşılaştırılması Tablo 2'de verilmiştir. IP bloğunun işlem süresi 100 MHz'lik saat darbesi frekansı ile hesaplanmıştır. MATLAB kodları ise Windows 10 işletim sistemine sahip, i7 işlemcili, 16 GB RAM'li ve 250 GB SSD harddiske sahip bilgisayar üzerinde koşturulmuştur. MATLAB işlem süresi 100 koşumun ortalaması olarak verilmiştir. Tablo 2'den de görüleceği üzere FPGA tabanlı gerçekleme MATLAB'a göre daha düşük sürede işlemleri gerçekleştirmiştir.

Şekil 8. *2D Konvolüsyon IP bloğunun imge üzerine uygulanması (a) Orijinal imge, (b)Yatay Sobel, (c)Dikey Sobel, (d) Yatay Prewit, (e) Dikey Prewit, (f)Kaydır ve Çıkart, (g) Alçak Geçiren, (h) Yüksek Geçiren, (i) Gauss*

Tablo 2. Gerçekleştirilen IP Bloğunun İşlem Süresinin Karşılaştırılması

Platform	***Süre (ms)***
FPGA	13.005
MATLAB	48.145

Geliştirilen 2D konvolüsyon IP bloğunun Xilinx firmasına ait Artix 7 100T FPGA'sı referans alınarak elde edilen sentez sonuçları Tablo 3'de gösterilmiştir.

Tablo 3. FPGA Kaynak Kullanımı

Kaynak	***Kullanılan***	***Mevcut***	***Oran (%)***
LUT	427	63400	0.67
FF	90	126800	0.2
BUFG	1	32	3.13

4. SONUÇLAR

FPGA tasarımlarının gerçekleştirilebilmesi yazılım uygulamalarına göre daha fazla zaman ve kullanıcı tarafından donanım tanımlama dili bilmesi zorunluluğu gerektirmektedir. Bu çalışma kapsamında kullanıcı tarafından değişikliklerin hızlı yapılabilmesi ve donanım tanımlama dili hakimiyeti olmadan işlemlerini gerçekleştirilebileceği 2D Konvolüsyon IP çekirdeği tasarımı gerçekleştirilmiştir. Jenerik mimari olarak çalışma kapsamında geliştirilen 2D Konvolüsyon IP çekirdeği, kullanıcı tarafından belirlenen parametreler ile hızlıca güncellenerek tasarlanacak sisteme uyarlanabilir hale gelebilmektedir. Böylelikle sistemde değişiklikler hızlıca yapılabilmektedir ve farklı sistemlere kolaylıkla uyumlandırılabilmektedir. Tablo 2'de verilen işlem süreleri referans alındığında geliştirlen IP bloğununun MATLAB'a göre daha kısa sürede işlemleri tamamladığı görülmektedir. IP bloğunun işlem süresi daha yüksek saat darbesi frekansında ve boru hattı (pipeline) mimari tasarımı eklenmesi ile daha da azaltılabilecektir. Tablo 3'de verilen sentez sonuçlarından da görüleceği üzere, gerçekleştirlen IP düşük donanım tüketimine sahiptir. Böylelikle IP kısıtlı donanım kaynaklarında da rahatlıkla kullanılabilmektedir.

ÇIKAR ÇATIŞMASI

Yazar, çıkar çatışması olmadığını bildirmektedir.

KAYNAKLAR

[1] N. Bellas, S. M. Chai, M. Dwyer and D. Linzmeier, "FPGA implementation of a license plate recognition SoC using automatically generated streaming accelerators", 20th International Parallel and Distributed Processing Symposium, 2006

[2] F. Smach, M. Atri, J. Mitéran and M. Abid, "Design of a Neural Networks Classifier for Face Detection", *Journal of Computer Science*, vol. 2, no. 3, pp. 257-260, 2006.

[3] F. Wang and H. Qin, "A FPGA based driver drowsiness detecting system", IEEE International Conference Vehicular Electronics and Safety, 2005.

[4] K. Appiah and A. Hunter, "A single-chip FPGA implementation of real-time adaptive background model", IEEE International Conference Field-Programmable Technology Proceedings 2005.

[5] P. Y., Hsiao, L.T. Li, C. H. Chen, S. W. Chen and S.J. Chen, "An FPGA architecture design of parameter-adaptive real-time image processing system for edge detection", Emerging Information Technology Conference 2005.

[6] K. Ratnayake and A. Amer, "An FPGA-Based Implementation of Spatio-Temporal Object Segmentation", IEEE International Conference Image Processing, 2006.

[7] G. Wall, F. Iqbal, X. Liu and S. Foo, "A Fast FPGA Implementation of a Unique Multi-level Tree-based Image Classifier", Florida A&M University - Florida State University.

[8] P. Y. Hsiao, L. T. Li, C. H. Chen, S. W. Chen and S. J. Chen, "An FPGA architecture design of parameter-adaptive real-time image processing system for edge detection", Emerging Information Technology Conference.

[9] Y. H. Tan, Y. Xin and X. J. Zhai, "A FPGA-Based Method for License Plate Localization", International Conference on Electrical, Automation and Mechanical Engineering, 2015.

[10] M. A. Çavuşlu, K. Karakaya, and H. Altun, "ÇKA Tipi Yapay Sinir Ağı Kullanılarak Plaka Yeri Tespitinin FPGA'da Donanımsal Gerçeklenmesi", Akilli Sistemlerde Yenilikler ve Uygulamalar Sempozyumu, 2008.

[11] S. Chhabra, H. Jain, and S. Saini, "FPGA based hardware implementation of automatic vehicle license plate detection system". International Conference Computing, Communications and Informatics (ICACCI), 2016

[12] M. I. AlAli, K. M. Mhaidat, and I. A. Aljarrah, "Implementing image processing algorithms in FPGA hardware", IEEE Applied Electrical Engineering and Computing Technologies (AEECT), 2013.

[13] R. K. Mondol, M. I. Khan, A. M. Hye, and A. Hassan, "Hardware architecture design of face recognition system based on FPGA". IEEE International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015.

[14] R. Lu, X. Liu, X. Wang, J. Pan, K. Sun and H. Waynes, "The Design of FPGA-based Digital Image Processing System and Research on Algorithms", *International Journal of Future Generation Communication and Networking*, vol. 10, no. 2, pp. 41-54, 2017

[15] B. A. Draper, J. R. Beveridge, A. W. Bohm, C. Ross, and M. Chawathe, "Accelerated image processing on FPGAs", *IEEE transactions on Image Processing*, vol. 12, no.12, pp. 1543-1551, 2003.

[16] F. Cardells-Tormo and P. L. Molinet, "Area-efficient 2-D shift-variant convolvers for FPGA-based digital image processing", IEEE Signal Processing Systems Design and Implementation, 2005.

[17] J. A. Kalomiros and J. Lygouras, "Design and evaluation of a hardware/software FPGA-based system for fast image processing", *Microprocessors and Microsystems*, vol. 32, no. 2, pp. 95-106, 2008.

[18] S. A. Fahmy, P. Y. Cheung, and W. Luk "Novel FPGA-based implementation of median and weighted median filters for image processing". International Conference on Programmable Logic and Applications, 2005