T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GENEL AMAÇLI BİR BELLEK ARAYÜZÜ DONANIMININ ALANDA PROGRAMLANABİLİR KAPI DİZİLERİ İLE GERÇEKLENMESİ

DOĞANCAN DAVUTOĞLU

YÜKSEK LİSANS TEZİ
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ ANABİLİM DALI
ELEKTRONİK PROGRAMI

DANIŞMAN
DR. ÖĞR. ÜYESİ UMUT ENGİN AYTEN

İSTANBUL, 2018

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

GENEL AMAÇLI BİR BELLEK ARAYÜZÜ DONANIMININ ALANDA PROGRAMLANABİLİR KAPI DİZİLERİ İLE GERÇEKLENMESİ

Doğancan DAVUTOĞLU tarafından hazırlanan tez çalışması 30.04.2018 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Dr. Öğr. Üyesi Umut Engin AYTEN

Yıldız Teknik Üniversitesi

Eş Danışman

Dr. Nerhun YILDIZ

Arm Limited

Jüri Üyeleri

Dr. Öğr. Üyesi Umut Engin AYTEN

Yıldız Teknik Üniversitesi _____

Prof. Dr. Vedat TAVŞANOĞLU

Işık Üniversitesi _____

Prof. Dr. Herman SEDEF

Yıldız Teknik Üniversitesi _____

ÖNSÖZ

Öncelikle, yüksek lisans tezimi tamamlama aşamasına geldiğim için mutlu olduğumu belirtmeliyim. Bu aşamaya kadar geçen süreyi iyi değerlendirip çalıştığım konu üzerinde önemli ölçüde birikim kazandığımı düşünüyorum. İleride bu birikimi, teorik bilgilerin pratik uygulamalarla desteklenmesi konusunda yoğun olarak kullanacağıma inanıyorum. Sonraki aşamada da akademik yolculuğumun keyifli olacağını umuyorum.

Şanslıyım ki, yüksek lisans öğrenimim boyunca bana katkıda bulunan çok sayıda değerli insan oldu. Bu kişilerden ilk olarak, akademik hayatımı çok büyük ölçüde şekillendiren ve öğrencisi olmaktan gurur duyduğum Dr. Nerhun Yıldız'a en içten teşekkürlerimi sunarım. Yüksek lisans öğrenimimde yolumun tekrar kesiştiği Prof. Dr. Vedat Tavşanoğlu'na katkıları ve bana kazandırdığı vizyon için çok teşekkür ederim. Tez danışmanım olduğu günden beri yardımlarını benden esirgemeyen Dr. Umut Engin Ayten'e ayrıca teşekkür ederim. Tez döneminde, araştırma görevlisi olduğum sürede ve neredeyse her konuda yardımı dokunan Basri Erdoğan ile lisans öğrenimimde danışmanım olan ve halen danışmaya devam ettiğim Dr. Yusuf Acar'a da desteklerinden dolayı teşekkür ederim. Ayrıca, çok değerli hocalarım Prof. Dr. Oruç Bilgiç'e, Prof. Dr. Ertuğrul Eriş'e, Dr. Ertuğrul Saatçı'ya, Dr. Esra Saatçı'ya, Dr. Sadık Yiğit'e, üzerimde emeği bulunan tüm hocalarıma ve çalışma arkadaşlarıma çok teşekkür ederim.

Beni yetiştiren, bugüne kadar her türlü fedakârlığı yapan ve sahip olduğum için onur duyduğum aileme sonsuz sevgilerimi, son olarak her zaman yanımda olduğu için en kıymetli parçam olan eşim Saniye'ye sonsuz teşekkürlerimi sunarım.

Nisan, 2018

Doğancan DAVUTOĞLU

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

N	Renk derinliğinin bit cinsinden ifadesi
$\mathbf{N}$	Doğal sayılar kümesi
$\mathbf{Z}$	Tam sayılar kümesi
$\mathbf{Z}^+$	Pozitif tam sayılar kümesi
t	Zaman değişkeni
Δt	İşleme giren video kareleri arasındaki zaman farkı
n	Video karelerinin indisi
k	İşleme giren video karelerinin indisleri arasındaki mutlak değer farkı
T_s	İki ardışık video karesi arasındaki süre

KISALTMA LİSTESİ

CCU	Central Control Unit (Merkezi Kontrol Birimi)
DE	Data Enable (Veri Aktif)
DDR	Double Data Rate (Çift Veri Hızı)
DVI	Digital Visual Interface (Sayısal Görüntü Arayüzü)
EMI	External Memory Interface (Harici Bellek Arayüzü)
FIFO	First In First Out (İlk Giren İlk Çıkar)
FPGA	Field Programmable Gate Array (Alan Programlanabilir Kapı Dizisi)
HBP	Horizontal Back Porch (Yatay Arka Boşluk)
HDL	Hardware Description Language (Donanım Tanımlama Dili)
HDMI	High Definition Multimedia Interface (Yüksek Çözünürlüklü Çokluortam
HFP	Horizontal Front Porch (Yatay Ön Boşluk)
HSYNC	Horizontal Synchronization (Yatay Senkronizasyon)
PLL	Phase-Locked Loop (Faz Kilitlemeli Döngü/Çevrim)
RAM	Random Access Memory (Rasgele Erişimli Bellek)
RGB	Red Green Blue (Kırmızı Yeşil Mavi)
Rx	Receiver (Alıcı)
SDRAM	Synchronous Dynamic RAM (Senkron Dinamik RAM)
Tx	Transmitter (Verici)
VBP	Vertical Back Porch (Dikey Arka Boşluk)
VFP	Vertical Front Porch (Dikey Ön Boşluk)
VGA	Video Graphics Array (Video Grafik Dizisi) Arayüzü)
VSYNC	Vertical Synchronization (Dikey Senkronizasyon)

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

GENEL AMAÇLI BİR BELLEK ARAYÜZÜ DONANIMININ ALANDA PROGRAMLANABİLİR KAPI DİZİLERİ İLE GERÇEKLENMESİ

Doğancan DAVUTOĞLU

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Dr. Öğr. Üyesi Umut Engin AYTEN

Eş Danışman: Dr. Nerhun YILDIZ

Günümüzde birçok sayısal sistemde kısa süreli veya uzun süreli veri saklamaya ihtiyaç duyulmaktadır. Yüksek performanslı ve paralel hesaplama işlemleri gerektiren yapay, hücresel veya konvolüsyonel sinir ağları tasarımları, büyük veri yönetimi, görüntü işleme uygulamarı gibi alanlarda verilerin saklanıp geri çağrılabilmesi için bellek elemanlarının sıklıkla kullanılması gerekmektedir. Eğer bu saklama ve geri çağırma işlemlerinin gerçek zamanlı olarak yapılmasına ihtiyaç varsa, bellek üzerinde yapılan yazma ve okuma işlemlerine ait bant genişliğinin yeterli olması çok önemlidir. Bu noktada, bellek elemanının çalışma frekansının yüksek olması gerekliliğinin yanı sıra, bellek ile sistem arasında köprü oluşturan bellek arayüzünün de hem bellek elemanı hem de sistem tarafındaki işlemleri yerine getirebilmesi için yüksek hızda çalışması bir zorunluluktur.

Bu tez çalışmasında ortaya konulan genel amaçlı bellek arayüzü, yüksek hız ve bant genişliği gerektiren özellikle gerçek zamanlı uygulamalarda kullanılabilir olması açısından fayda sağlamaktadır. Daha önce literatürde bulunan gerçek zamanlı hücresel sinir ağları uygulamalarında, zaman türevi gerektiren işlemlerin gerçeklenebilmesi için bellek arayüzü ihtiyacı olduğu bildirilmiştir. Bu çalışmada sunulan genel amaçlı bellek arayüzünün bu eksiği gidermede kullanılması planlanmaktadır.

Tasarlanan arayüz kullanılarak, arayüzün olası kullanım alanları arasında yer alan gerçek zamanlı görüntü işleme uygulamalarında yaygın olarak kullanılan video standartlarından Full-HD 1080p çözünürlüğünde ve saniyede 60 kare işlemeye olanak

sağlayacak bant genişliğine ulaşılmıştır. Arayüzün test edilme aşamasında farklı bellek türleri de kullanılmıştır. Bu belleklerden 16-bitlik DDR SDRAM bellek kullanılarak 133 MHz saat frekansında maksimum 1,98 Gbit/s bant genişliğine, 64-bitlik DDR3 SDRAM bellek kullanılarak 667 MHz saat frekansında maksimum 39,76 Gbit/s bant genişliğine ulaşılmıştır. Tekrar Full-HD 1080p çözünürlüğüne sahip gerçek zamanlı video akışı göz önüne alınırsa, 256 Mbit bellek üzerinde toplamda 15 kareye kadar saklama ve 1 Gbit bellek üzerinde toplamda 60 kareye kadar saklama kapasitesine ulaşılmıştır.

Anahtar Kelimeler: Bellek arayüzü, alan programlanabilir kapı dizisi, sayısal donanım tasarımı, gerçek zamanlı video işleme

ABSTRACT

HARDWARE REALIZATION OF A GENERAL PURPOSE MEMORY INTERFACE BASED ON FPGA

Doğancan DAVUTOĞLU

Department of Electronics and Communications Engineering

MSc. Thesis

Adviser: Assist. Prof. Dr. Umut Engin AYTEN

Co-Adviser: Dr. Nerhun YILDIZ

Today, in many digital systems, it is necessary to store data for short or long duration. Memory elements are often needed to storing and retrieving data in areas such as artificial, cellular or convolutional neural network designs, big data management, image processing applications that require high performance and parallel computation operations. If it is necessary to perform these storage and recall operations in real time, it is very important that the bandwidth of the write and read operations on the memory is sufficient. At this point, the operating frequency of the memory element is required to be high enough, as well as the memory interface that bridges the gap between memory and the system is required to operate at high speed so that it can perform both memory and system operations.

The general purpose memory interface presented in this study has many advantages in many areas that require high speed and bandwidth, especially in real time applications. Previously, it has been reported in the literature that real-time cellular neural network applications require a memory interface to perform time-derivation operations. The general purpose memory interface presented in this work is planned to be used in time-derivative CNN realizations to fulfill the need of memory interface.

Using the designed interface, the video standards that are common in real-time image processing applications, which are among the possible uses of the interface, have reached the bandwidth that allows processing at 60 fps at Full-HD 1080p resolution and at the very least. Different types of memory have been used during the testing of the interface. Using these 16-bit DDR SDRAM memories, a maximum bandwidth of 1.98

Gbit/s at 133 MHz clock frequency and a maximum bandwidth of 39.76 Gbit/s at 667 MHz clock frequency using 64-bit DDR3 SDRAM memory were achieved. If a real-time video stream with Full-HD 1080p resolution is considered, storage capacity of up to 15 frames on 256 Mbit memory and a total of 60 frames on 1 Gbit memory have been achieved.

Keywords: Memory interface, field programmable gate array, digital hardware realization, real-time video processing

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Sayısal ortamda veri saklama ihtiyacının olduğu her alanda ve her uygulamada, bellek elemanlarına da mutlak bir ihtiyaç vardır. Sayısal verinin türünün değişmesine paralel olarak, bellek elemanlarının da türleri ve teknolojileri gün geçtikçe değişmekte ve gelişmektedir. Buna karşın, depolanan veri büyüklüğünün artması ve yüksek hızlarda veri yakalama ve geri çağırma işlemlerinin çok daha kısa zamanlarda gerçekleştirilmesine olan ihtiyacın artması, bellek elemanlarına erişim için kullanılan arayüz tasarımlarının önemini artırmaktadır. Literatürde, verimli bir harici bellek arayüzüne ihtiyaç duyulduğu bildirilen çalışmalar yer almaktadır. Bu eksiğin giderilmesi veya var olan bellek arayüzünün iyileştirilmesi amacıyla bu tez çalışmasında önerilen genel amaçlı bellek arayüzü tasarımından faydalanılabileceği ön görülmektedir.

Ağırlıklı olarak görüntü ve video işleme uygulamaları çerçevesinde ihtiyaç duyulan harici bellek arayüzünün kullanım alanları içerisinde uzay-zamansal filtreleme uygulamaları, hücresel sinir ağları uygulamaları, hareket tespit sistemleri, veri akışı için ara bellekleme işlemleri bulunmaktadır. Daha önce Nerhun Yıldız ve arkadaşlarının çalışmalarında [1-3] geliştirilen hücresel sinir ağı tasarımının uygulanabilirliğinin artırılması konusunda bellek arayüzü kullanım ihtiyacı işaret edilmiştir. Öyle ki, bu hücresel sinir ağı tasarımının, gerçek zamanlı görüntü işleme uygulaması sırasında daha eski video karelerini geri çağırması mümkün değildir. Eski video karelerine dair bilgiye ihtiyaç duyulan diğer bir uygulama olan zaman-türevli hücresel sinir ağları tasarımının simülasyonuna ilişkin çalışmalar da literatürde yer almaktadır [4-6]. Benzer şekilde, literatürde yer alan hücresel sinir ağı işlemcisi ile uzay-zamansal filtreleme

uygulamasının gerçeklenmesi için bir bellek arayüzü ihtiyacı vardır [7]. Bu uygulamanın yüksek çözünürlüklü ve gerçek zamanlı olarak gerçeklenebilmesi için ise bellek arayüzünün yüksek hızda ve paralel işlem kapasitesine sahip olması gerekmektedir. Bahsedilen bu uygulamalar için bellek arayüzü tasarımı ihtiyacı, bu tez çalışmasında ortaya konulan tasarım ile karşılanabilecektir.

Literatürde yer alan gerçek zamanlı hareketli nesne tespit sistemleri çalışmalarında çeşitli video çözünürlükleri ve saniyedeki kare sayıları ile gerçeklemeler yapılmıştır. Bu çalışmalarda kullanılan bellek arayüzü teknolojisinin ve bellek frekansının düşük olmasından dolayı, işlenen video çözünürlüklerinin yükseltilmesi mümkün değildir. Rao ve arkadaşlarının çalışmasında [8] sistemin çözünürlüğü 720×526 piksel ve saniyede 25 karedir. Benzer şekilde Wang ve Gao'nun çalışmasında [9] da çözünürlük ve kare hızı aynıdır. Başka bir çalışmada Xie ve Wang, gerçek zamanlı bir videonun iyileştirilmesi için kullandıkları bellek arayüzü ile 1920×1080 piksel çözünürlüğüne ve saniyede 30 kare hızına ulaşmışlardır [10]. Fakat bu çalışmada kullanılan arayüzün bellek frekansı 162 MHz'e kadar ulaşabilmektedir. Bu tez çalışmasında gerçeklenen harici bellek arayüzünün bellek frekansı 667 MHz'e ulaşmıştır ve 1920×1080 piksel çözünürlüğünde saniyede 60 video karesi işlenmiştir. Wang ve arkadaşlarının yüksek bant genişlikli bellek arayüzü çalışmasında [11] 25.6 Gb/s olan bant genişliği ile kıyaslandığında bu tez çalışmasında yapılan gerçeklemelerde bant genişliğinin 39.76 Gb/s ile daha yüksek olduğu görülmektedir.

1.2 Tezin Amacı

Bu tez çalışması çerçevesinde tasarlanan bellek arayüzünün öncelikli hedefi, yüksek paralel işlem kapasitesi gerektiren, gerçek zamanlı veri işleme uygulamaları için bir arabellek altyapısı oluşturmaktır. Özellikle, gerçek zamanlı video işleme uygulamaları bu arabellek tasarımının kullanımı için uygun bir alandır. İşlenecek olan video sinyalinin çözünürlüğü, saniyedeki kare sayısı, saklanacak olan kare sayısı gibi parametreler değişken olduğundan, bellek arayüzünün de buna paralel olarak kullanıcı tarafından büyük ölçüde konfigüre edilebilir olarak tasarlanması amaçlanmıştır.

Çalışmada ortaya konulan tasarım ile, gerçek zamanlı video sinyalleri için arabellekleme, fark alıcı ve arkaplan çıkarıcı uygulamaları gerçeklenmiştir. Tasarım amaçları doğrultusunda bu gerçeklemelerde 1920×1080 piksel çözünürlüklü ve

saniyede 60 kareye sahip video sinyallerini gerçek zamanlı olarak işleyebilecek bir yapı elde edilmiştir. Kullanıcıya açık olarak sunulan burst sayısı, arabellek boyutu, veri yolu genişliği ve bellek banka sayısı parametreleri ile yapısal olarak esneklik ve ölçeklenebilirlik sağlanmıştır.

1.3 Orijinal Katkı

Bu tez çalışmasında ortaya konulan genel amaçlı bellek arayüzü tasarımı, Nerhun Yıldız, Vedat Tavşanoğlu, Evren Cesur, Kamer Kayaer ve Murathan Alpay'ın literatüre kattığı ikinci ve üçüncü nesil hücresel sinir ağı benzetimi çalışmalarında [1, 3] değinilen bellek arayüzü ihtiyacı doğrultusunda ortaya çıkmıştır. Bahsedilen ikinci nesil hücresel sinir ağı çalışmasında [1], işlenen video piksellerinin sonuçları FPGA içerisinde oluşturulan blok belleklerde kısa süreli olarak saklanmaktadır ve birden fazla video karesi saklama özelliği yoktur. Üçüncü nesil hücresel sinir ağı işlemcisinin yol haritasında [3], çerçeve arabelleği ihtiyacı ve bu arabelleğin yeni nesil işlemciye uzay-zamansal filtreleme işlemi yapma özelliği katacağı belirtilmiştir. Bu doğrultuda, tez çalışmasında açıklanan bellek arayüzü öncelikli olarak hücresel sinir ağları gerçeklemeleri ile birlikte kullanılabilecek şekilde tasarlanmıştır. Fakat arayüz tasarımı, veri akışının düzenlenmesi, kısa veya uzun süreli veri saklanması ve bellek erişimi gerektiren uygulamalarda da kullanılabilecek şekilde genelleştirilmiştir.

BÖLÜM 2

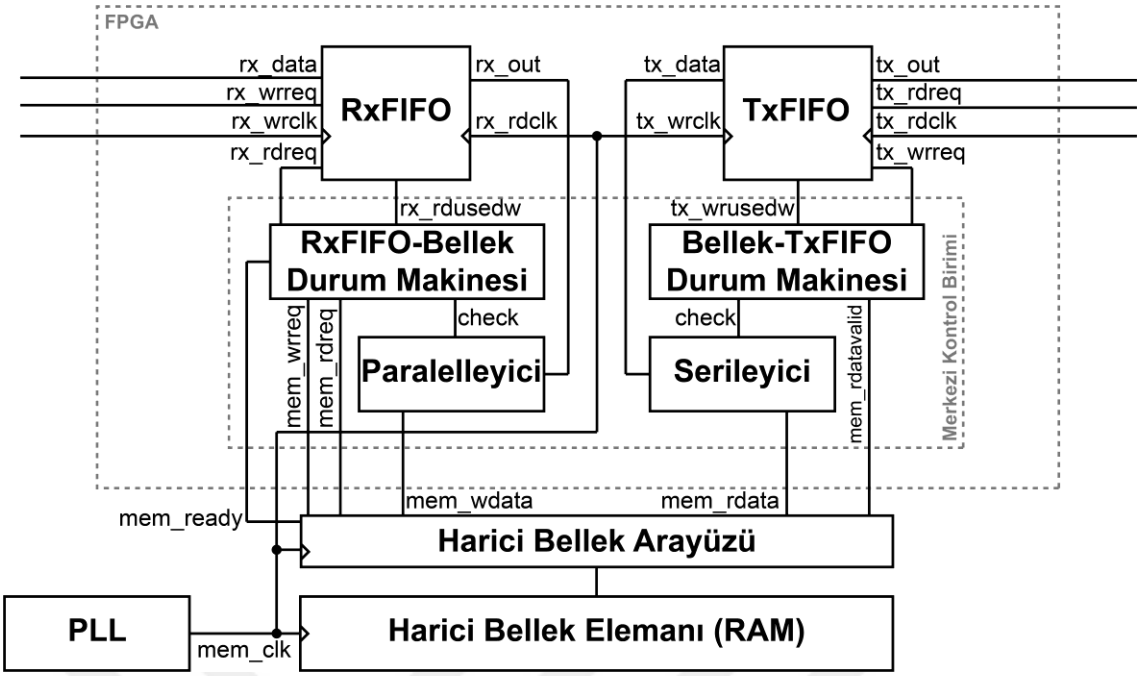
TASARLANAN DONANIMIN YAPISI

Tasarımın temeldeki hedefi, herhangi bir kaynaktan alınan veriyi yakalayan ve bu verileri daha sonraki işlemler içerisinde geri çağırabilen bir mimari oluşturmaktır. Bu doğrultuda tasarlanan donanımın altyapısı, yığın işlem veya gerçek zamanlı, süreksiz veya sürekli veri akışına sahip sinyallerin kaynak olarak kullanılabilmesine olanak sağlayacak şekilde oluşturulmuştur. Bu kaynaklar; sayısal durağan veya video görüntü sinyali, sayısal ses sinyali, metin ve grafik gibi farklı çeşitlerdeki çoklu ortam verisi olabilir. Bu noktada göz önünde bulundurulması gereken tasarım parametrelerinden en önemlileri bant genişliği ve kesintisiz veri akışıdır. Dolayısıyla, tasarımın merkezinde yer alacak olan harici bellek elemanının yüksek bant genişliğine sahip olması ve tasarımın giriş ve çıkış basamaklarında veri akışını düzenlemeye yarayan blokların kullanılması bir gerekliliktir.

Sonuç olarak, donanımın ana tasarım blokları; giriş-çıkış basamaklarında akış düzenleyici olarak kullanılan 'İlk Giren İlk Çıkar (FIFO)', 'Merkezi Kontrol Birimi (CCU)' ve 'Harici Bellek Arayüzü (EMI)' bloklarından oluşmaktadır. Şekil 2.1'de donanımın genel yapısı verilmiştir.

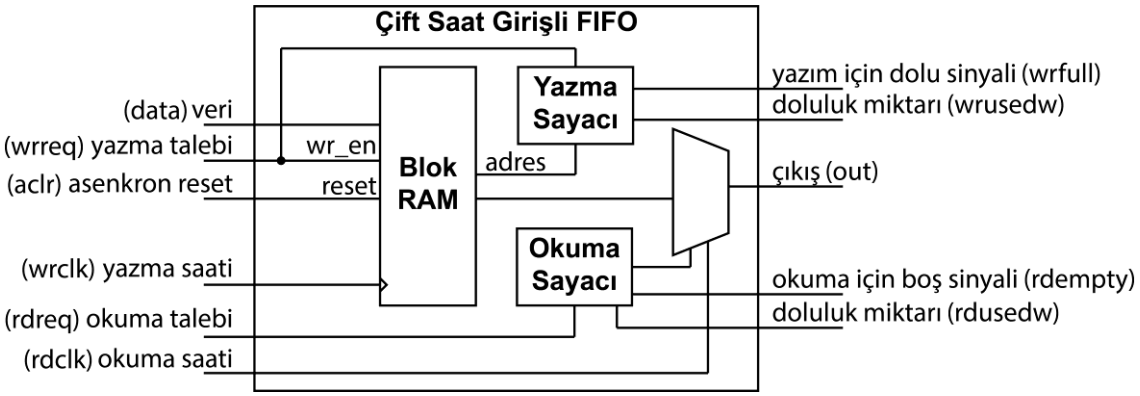
2.1 FIFO Bloğu

FIFO yapısı kısaca; verileri sırayla kısa süreli depolayıp, ihtiyaç olduğunda verilerin ilk saklanan veriden son saklanan veriye kadar sıra ile geri çağrılmasına imkân veren yapıdır. Tasarımda bu yapının kullanılmasının amacı, giriş ve çıkış bloklarında verileri sıralı hale getirmek ve sürekli veri akışına ihtiyaç olduğu durumda kesintisiz bir akış sağlamaktır. Şekil 2.2'de tasarımda kullanılan FIFO'ların genel yapısı gösterilmiştir.



Şekil 2.1 Donanımın genel yapısı

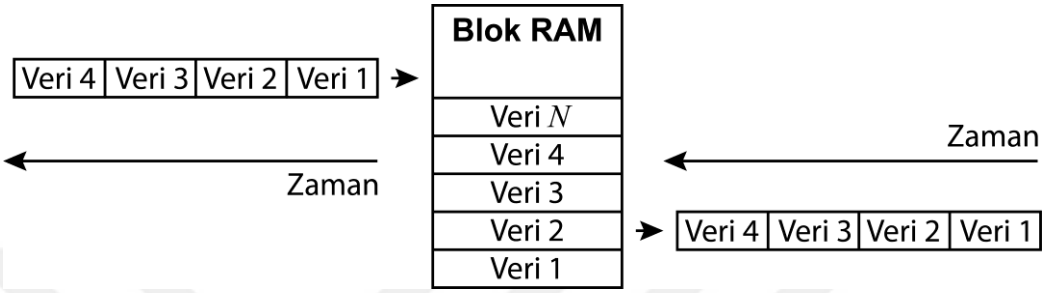
Bellek arayüzünün saati ile giriş – çıkış bloklarındaki verilerin saatleri farklı frekanslarda olacağından, FIFO'ların okuma ve yazma işlemleri için farklı saatlerle çalışmaları gerekmektedir. Bu nedenle, tasarımdaki FIFO'lar çift saat girişli (Dual Clock FIFO) olarak kullanılmıştır [12]. FIFO'ların içerisinde saklananabilen verilerin bit genişliği ve derinliği kullanıcı tarafından ayarlanabilmektedir.



Şekil 2.2 Tasarımda kullanılan FIFO'ların genel yapısı

FIFO'ların yazma ve okuma işlemleri, sırasıyla *wrreq (yazma talebi)* ve *rdreq (okuma talebi)* girişleri ile kontrol edilir. Yazma ve okuma talepleri ilgili işleme ait saat girişleri ile senkron olarak işlenir. FIFO içerisindeki veri miktarı *wrusedw* ve *rdusedw* çıkışları üzerinden takip edilebilir. Veri miktarına ait iki farklı çıkış olmasının sebebi, *wrusedw*

çıkışının yazma işlemine ait saat ile güncellenmesi ve *rdusedw* çıkışının ise okuma işlemine ait saat ile güncellenmesinden dolayıdır. FIFO kapasitesinin dolması durumunda *wrfull (yazma için dolu)* sinyali aktif olur. Benzer şekilde FIFO içerisinde veri kalmadığı durumda *rdempty (okuma için boş)* sinyali aktif olur. FIFO'nun içerisindeki veriler, asenkron reset girişi ile tamamen silinebilir. FIFO elemanının içerisinde bulunan blok RAM yapısı Şekil 2.3'te gösterilmiştir.

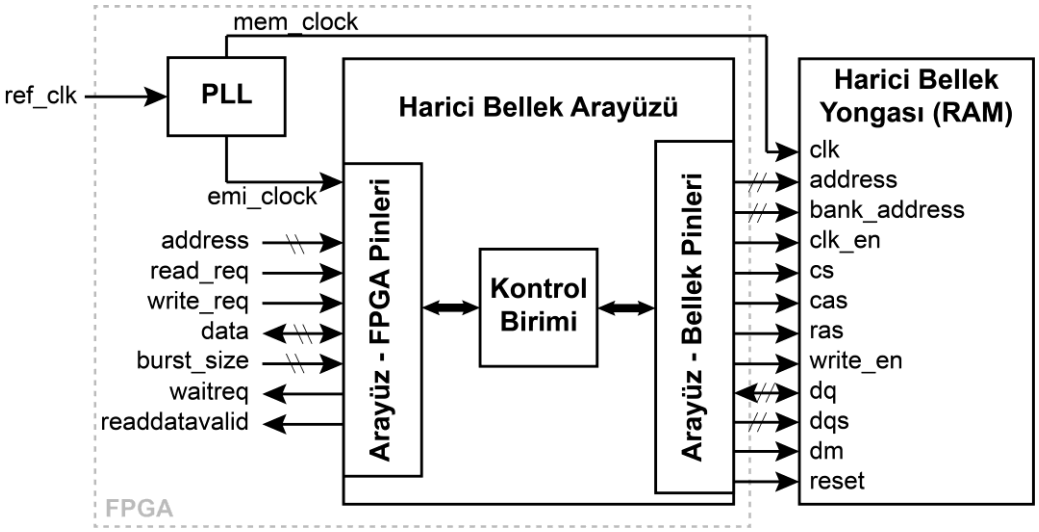


Şekil 2.3 Verilerin FIFO içerisindeki blok RAM'e sırayla yazılması ve okunması

Tasarımda iki adet FIFO kullanılmıştır. Giriş basamağında alıcı FIFO (*RxFIFO*) ve çıkış basamağında verici FIFO (*TxFIFO*) vardır. *RxFIFO*'nun okuma işlemine ait saati ve *TxFIFO*'nun yazma işlemine ait saati, harici bellek arayüzüne ait saat ile senkrondur. Bu saat sinyali, faz kilitlemeli döngü elemanı (*Phase-Locked Loop; PLL*) ile üretilmektedir.

2.2 Harici Bellek Arayüzü (EMI)

Tasarımda harici bellek elemanı ile FPGA arasında köprü oluşturan yapı harici bellek arayüzüdür. Tez çalışması çerçevesinde, FPGA geliştirme kartı üreticilerinin, ürettiği kartlara özgü olarak tasarladığı bellek-FPGA arayüzleri kullanımı tercih edilmiştir [13]. Bu sayede harici bellek elemanına ait kontrol sinyallerinin yönetilmesi, belleğe ait bank, satır ve sütun adreslerinin sıralanması, okuma ve yazma taleplerinin işlenmesine ilişkin işlemler, *dq* ve *dqs* sinyallerinin oluşturulması, ön dolum (pre-charge), yenileme (refresh), kalibrasyon ve ilk kullanıma hazırlama (initialization) işlemleri bu arayüz ile gerçekleştirilir [14]. Sonuç olarak; tasarım içerisinde belleğe ilişkin kontrol edilmesi gereken sinyal sayısı büyük ölçüde indirgenmiştir. İndirgeme sonrasında tasarım tarafından yönetilecek olan sinyaller; okuma ve yazma talep sinyalleri (*read_req, write_req*), bellek adresi, belleğe yazılacak veri ve burst (*çoğuşma*) modu için boyut (*burst_size*) sinyalleridir.

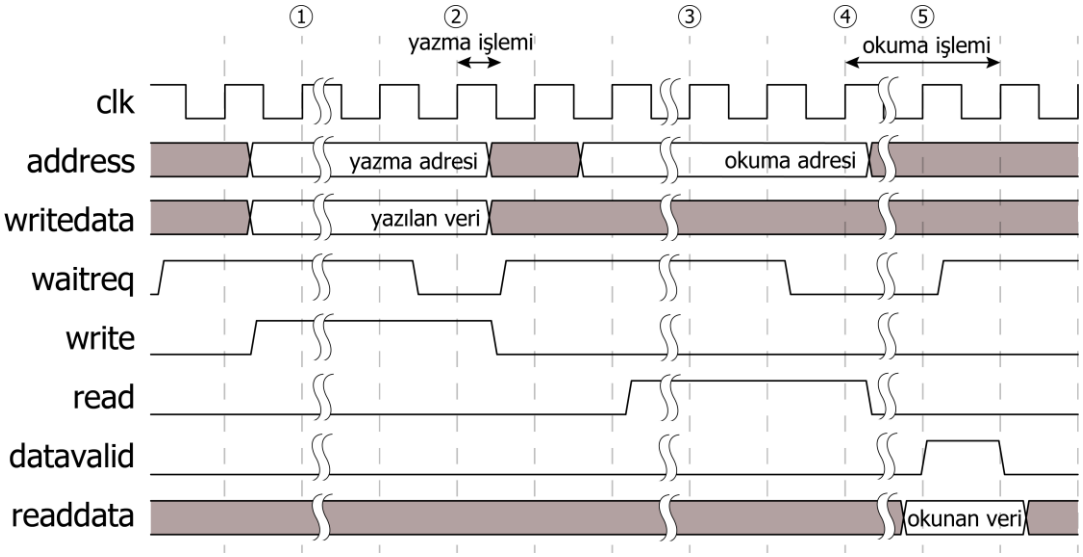


Şekil 2.4 Harici bellek arayüzünün genel yapısı [15]

Harici bellek arayüzünün genel yapısı Şekil 2.4 üzerinde gösterilmiştir. Arayüz içerisindeki kontrol birimi, FPGA ile harici bellek arasındaki işlemleri yönetir.

2.2.1 Burst Olmadan Yazma – Okuma İşlemleri

Bellek arayüzü kullanılarak harici bellek üzerine burst olmadan veri yazma ve harici bellekten veri okuma işlemlerine ilişkin zamanlamalar Şekil 2.5'te verilmiştir. Buna göre, '1' numaralı yükselen saat kenarı ile birlikte yazma (*write*) komutu ilk defa görülür. Fakat arayüzün meşgul olması nedeniyle (*waitreq = '1'*) yazma işlemi başlamaz. Yazma veya okuma işlemleri ancak ve ancak bekleme sinyalinin (*waitreq*) inaktif olduğu ve ilgili işlem sinyalinin aktif olduğu durumda başlatılır. '2' numaralı yükselen saat kenarı ile gerçekleşen işlem bu duruma örnektir. Bekleme sinyali inaktif olduğu için yazma işlemi başlatılır. Sonrasında bellek arayüzü bir süre bekleme modunda kalır. Bekleme modunda geçen saat döngü sayısı, arayüzün arkaplanda kontrol ettiği yenileme ve ön dolum işlemleri dolayısıyla değişkenlik gösterir. Dolayısıyla yazma ve okuma işlemleri için gereken bekleme döngüsünün öngörülen bir sayısı yoktur. '3' numaralı yükselen saat kenarı, okuma sinyalinin aktif olduğu ilk zamandır. Ama bekleme sinyali aktif olduğu için okuma işlemi başlamaz. '4' numaralı yükselen saat kenarı ile birlikte okuma işlemi başlatılır. '5' numaralı yükselen saat kenarı ile birlikte okuma işlemi tamamlanır. Sonraki aşamada, okunan verinin çıkışa gönderildiğine dair *readdatavalid* sinyali aktif olur. Bu sinyalin aktif olduğu sürede veri *readdata* sinyali üzerinden okunabilir.



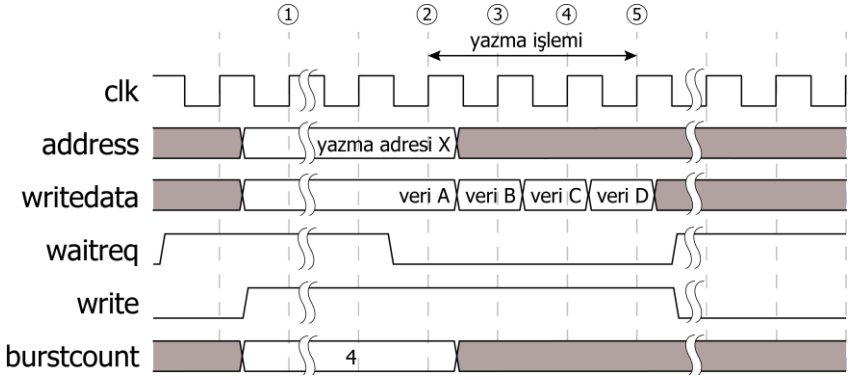
Şekil 2.5 Bellek arayüzü üzerinde burst olmadan yazma ve okuma işlemleri için dalga şekli

2.2.2 Burst Modunda Yazma – Okuma İşlemleri

Bellek üzerine veri yazma ya da bellekten veri okuma işlemleri başlamadan önce bellek üzerinde birtakım hazırlıkların yapılması gerekir. Her bir yazma ve okuma işlemi için adres verisinin alınması, bellek üzerinde ilgili adrese karşı gelen satır, sütun ve bank aktivasyonları, işlem sonunda ilgili adrese dair birimlerin deaktivasyonu gerçekleştirilmektedir. Eğer her seferinde tek bir okuma ve tek bir yazma işlemi yapılırsa, her bir işlem için bu ön hazırlıklar tekrarlanır. Verimliliğin artırılması amacıyla, tek bir yazma veya okuma komutu ile birlikte art arda seri yazma ve okuma işlemlerinin gerçekleştirilebilmesini mümkün kılan burst modu geliştirilmiştir. Bu sayede yazma ve okuma komutlarının öncesinde yapılması gereken ön hazırlıklar bir kereye mahsus olarak yapılır, sonrasında adres otomatik olarak burst süresince burst sayısı kadar artırılır ve burst bittiğinde işlem tamamlanır. Burst modunda gerçekleşen yazma ve okuma işlemlerine dair dalga şekilleri sırasıyla Şekil 2.6 ve Şekil 2.7 üzerinde gösterilmiştir. Kullanılan harici bellek arayüzü burst modunda 1024 seri yazma ve okuma işlemine kadar desteğe sahiptir.

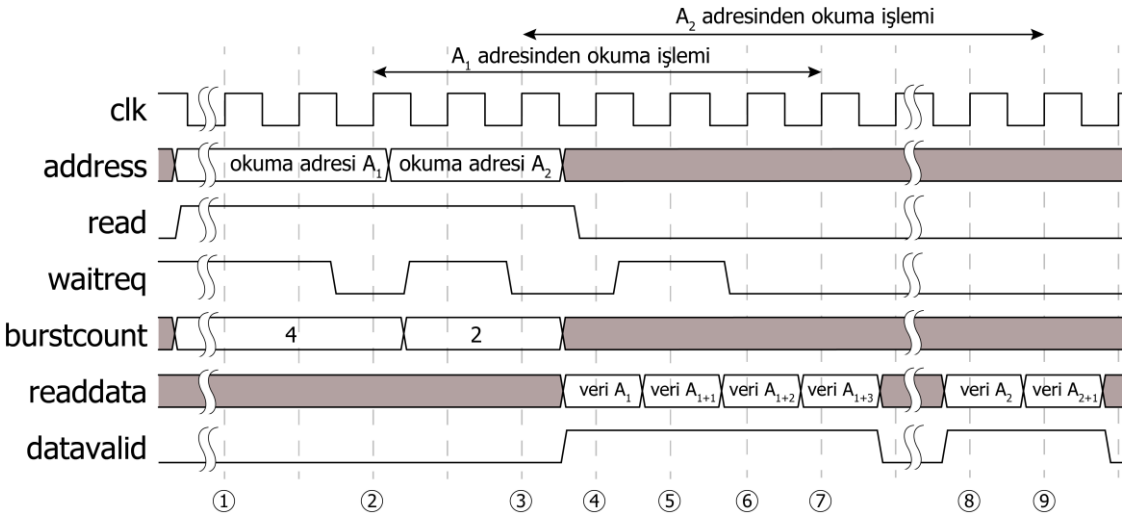
Yazma işlemine ait olan Şekil 2.6 üzerinde ‘1’ numaralı zamanda bekleme sinyali aktif olduğu için yazma talebi işleme konulmaz. ‘2’ numaralı zamanda bekleme sinyali inaktiftir ve yazma talebi ilgili adres üzerinde başlatılır. Aynı anda burst sayısı 4 olarak belirlendiği için, girilen adresten başlayarak her bir burst adımı için adres birer birer otomatik olarak artırılır. ‘2’ numaralı zamanda ‘A’ verisinin ‘X’ adresine

yazılma talebi işlenir. Sonrasında ‘3’ numaralı zamanda ‘B’ verisinin ‘X+1’ adresine yazılma talebi ve devamında ‘4’ ile ‘5’ numaralı zamanlarda sırasıyla ‘C’ ve ‘D’ verilerinin yine sırasıyla ‘X+2’ ve ‘X+3’ adreslerine yazılma istekleri işleme alınır.



Şekil 2.6 Bellek arayüzü üzerinde 4’lü burst modunda yazma işlemi için dalga şekli

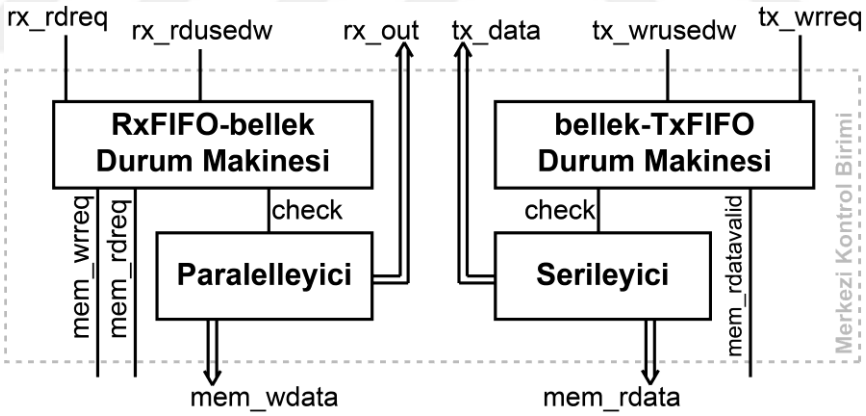
Burst modunda okuma işlemine ait olan Şekil 2.7’de art arda iki farklı burst sayısıyla iki okuma işlemi tekrarlanmıştır. ‘1’ numaralı zamanda bekleme sinyali aktif olduğundan, okuma talebi işleme alınmaz. ‘2’ numaralı zamanda ‘A_1’ adresinden başlayarak 4 adet burst okuma talebi işleme alınır. ‘3’ numaralı zamanda ise ‘A_2’ adresinden başlayarak 2 adet burst okuma talebi işleme alınır. Bu adımların sonrasında ilk olarak ‘4’ numaralı zamanda ‘A_1’ adresinden okunan veri gözlemlenir. Ardından sırasıyla ‘A_1+1’, ‘A_1+2’ ve ‘A_1+3’ adreslerinden okunan veriler gözlemlenir. ‘7’ numaralı zamanda ilk okuma işlemi sona ermiş olur. İkinci okuma işleminin sonucu ilk olarak ‘8’ numaralı zamanda gözlemlenir. ‘8’ numaralı zamanda okunan veri ‘A_2’ adresine aittir. Ardından ‘9’ numaralı zamanda ‘A_2+1’ adresine ait veri okunmuş olur.



Şekil 2.7 Bellek arayüzü üzerinde 4’lü burst modunda okuma işlemi için dalga şekli

2.3 Merkezi Kontrol Birimi (CCU)

Tasarım içerisinde yer alan FIFO blokları, harici bellek arayüzü ve harici bellek birimi arasında bir köprü oluşturması amacıyla ‘Merkezi Kontrol Birimi (CCU)’ ne ihtiyaç duyulmaktadır. Bu birimin görevi, veri akışının özellikle gerçek zamanlı ve kesintisiz olduğu durumda belleğe yazılacak olan verinin sürekli olarak yakalanabilmesini, bellekten okunacak olan verinin de sürekli olarak hedefteki çıkışa gönderilebilmesini sağlamaktır. Veri tipinin gerçek zamanlı bir video sinyali olduğu durum göz önüne alınırsa, bir kaynaktan gelen video kareleri kesintisiz bir şekilde belleğe yazılabilmeli, sonrasında geri çağırılıp bir monitöre veya görüntü işleme aygıtına gönderilebilmelidir. Bu noktada, dinamik tip bellek elemanlarının yenileme sinyaline ihtiyaç duyması ve dolayısıyla bellek üzerindeki yazma ve okuma ilemleri kısa süre için duraklaması nedeniyle harici bellek arayüzünün tek başına kesintisiz bir veri akışı sağlaması mümkün değildir. Kesintisiz veri akışını sağlamak amacıyla daha önceki kısımlarda bahsedilen FIFO bloklarının bellek arayüzü ile birlikte kullanılması gerekmektedir. FIFO blokları ve harici bellek arayüzünün birlikte kullanımı, senkronizasyonu ve kontrolü için ise ‘Merkezi Kontrol Birimi’ görev yapmaktadır.

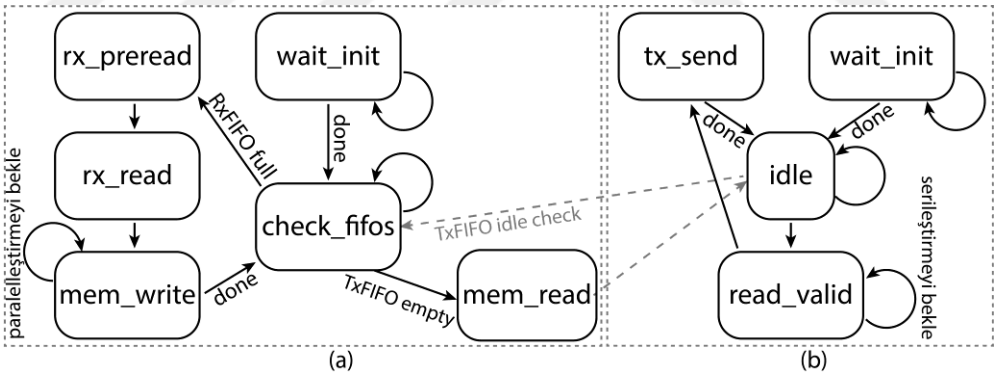


Şekil 2.8 Merkezi Kontrol Birimi’nin genel şeması

2.3.1 Durum Makineleri

Merkezi Kontrol Birimi’nin genel şeması Şekil 2.8’de verilmiştir. Yapının içerisinde *RxFIFO-bellek* arasındaki ve *bellek-TxFIFO* arasındaki işlemleri yürütmek üzere iki adet durum makinesi bulunmaktadır. Giriş bloğundaki verinin belleğe gönderilmesi esnasında diğer bir yandan bellekten okunan verinin çıkış bloğuna aktarılma işleminin yürütülmesi gerektiği için durum makineleri birbirlerine paralel olarak çalışmak

zorundadır. Bahsedilen durum makinelerinin durum diyagramları Şekil 2.9'da gösterilmiştir. *RxFIFO-bellek* ve *bellek-TxFIFO* durum makinelerinin başlangıç durumları, bellek yongasının başlatılmaya hazır olup olmadığının kontrol edildiği *wait_init* durumudur. Bellek arayüzü çalışmaya hazır hale gelene kadar durum makineleri sonraki duruma geçmez. Bellek hazır hale geldikten sonra *RxFIFO-bellek* durum makinesi, *RxFIFO* ve *TxFIFO* içerisindeki doluluk oranlarını kontrol eder. Belleğe okuma talebi gönderilmeden önce yeterli miktarda verinin belleğe yazılmış olması gerekmektedir. Bu nedenle *RxFIFO* elemanından okuma yapmaya yetecek kadar doluluğa ulaşıldığında ilk olarak ön bellekleme işlemi başlar. Başka bir deyişle, harici bellek üzerine depolama yapılır. Kesintisiz bir veri akışı sağlanabilmesi için idealde *RxFIFO* elemanının tamamen boşaltılmış olması, *TxFIFO* elemanının ise tamamen doldurulmuş olması gerekmektedir. Bu koşulun sağlanması amacıyla, eğer *RxFIFO* içerisindeki veri miktarı fazla ise, *RxFIFO-bellek* durum makinesi *RxFIFO*'dan veri okuyup harici belleğe yazma işlemini başlatır. Eğer *TxFIFO* içerisindeki veri miktarı az ise, *RxFIFO-bellek* durum makinesi harici bellekten veri okuyup *TxFIFO*'ya yazmaya karar verir. Kararı *RxFIFO-bellek* durum makinesi vermesine rağmen, harici bellekten veri okuyup *TxFIFO*'ya yazma işlemini *bellek-TxFIFO* durum makinesi denetler.



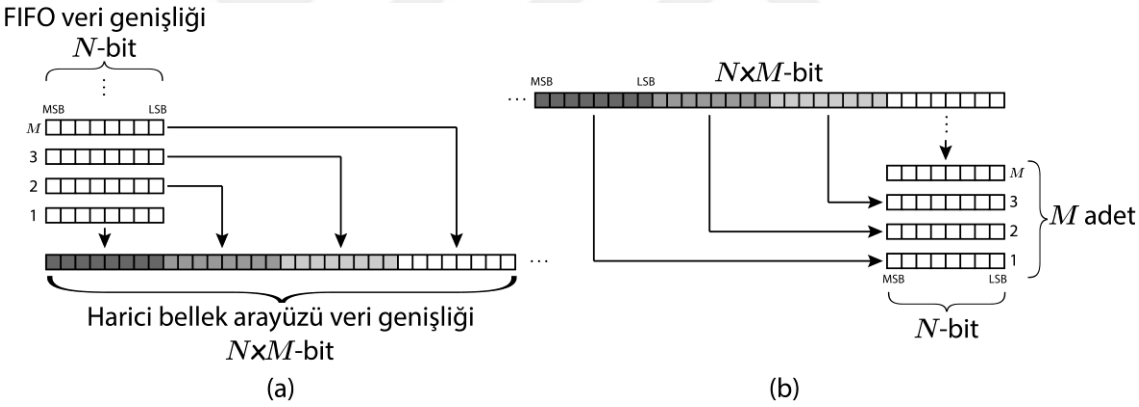
Şekil 2.9 Merkezi Kontrol Birimi'nin içerisindeki durum makinelerine ait durum diyagramları; (a) *RxFIFO-bellek durum makinesi* (b) *bellek-TxFIFO durum makinesi*

RxFIFO'nun tamamen dolması ya da *TxFIFO*'nun boş kalması durumlarını engellemek için, bellek arayüzünün bant genişliği, giriş ve çıkış bloklarındaki veri akışı için gerekli bant genişliğinin en az iki katı olmalıdır. Bunun nedeni, gerçek zamanlı bir veri akışı olduğu durumda giriş ve çıkışın eş zamanlı olarak işlenmesine karşın tasarımda kullanılan harici bellek yongasının tek portlu olması ve bu sebeble tek seferde sadece okuma veya yazma işlemlerinden sadece birini yerine getirebilmesidir.

2.3.2 Paralelleyici ve Serileyici

Merkezi Kontrol Birimi içerisinde FIFO'lar ve bellek arayüzü arasında paralelleyici ve serileyici modüller bulunmaktadır. Bu modüller FIFO'lar ve bellek arayüzünün veri genişlikleri arasında farklılık olması durumunda devreye girer. Paralelleyici modül, veri paketleme işlemi yapar. Serileyici modül ise paketlenmiş olan veriyi parçalara ayırır ve zamanda sıralı hale getirir. Örnek olarak bellek arayüzünün giriş ve çıkış FIFO'larından daha yüksek veri genişliğine sahip olduğu durum incelenirse:

Giriş bloğundaki *RxFIFO*'dan okunan veri harici belleğe gönderilmeden önce paralelleyici modül içerisinde yan yana paketlenir, paket genişliği bellek arayüzünün veri genişliğine ulaştığında harici belleğe yazım işlemi başlar. Harici bellekten okunan verinin, çıkış bloğundaki *TxFIFO*'ya yazılabilmesi için ise serileyici modül, harici bellekten okunan veriyi *TxFIFO*'nun veri genişliği büyüklüğünde parçalara böler ve her bir parçayı ayrı saat döngüsünde *TxFIFO*'ya gönderir.

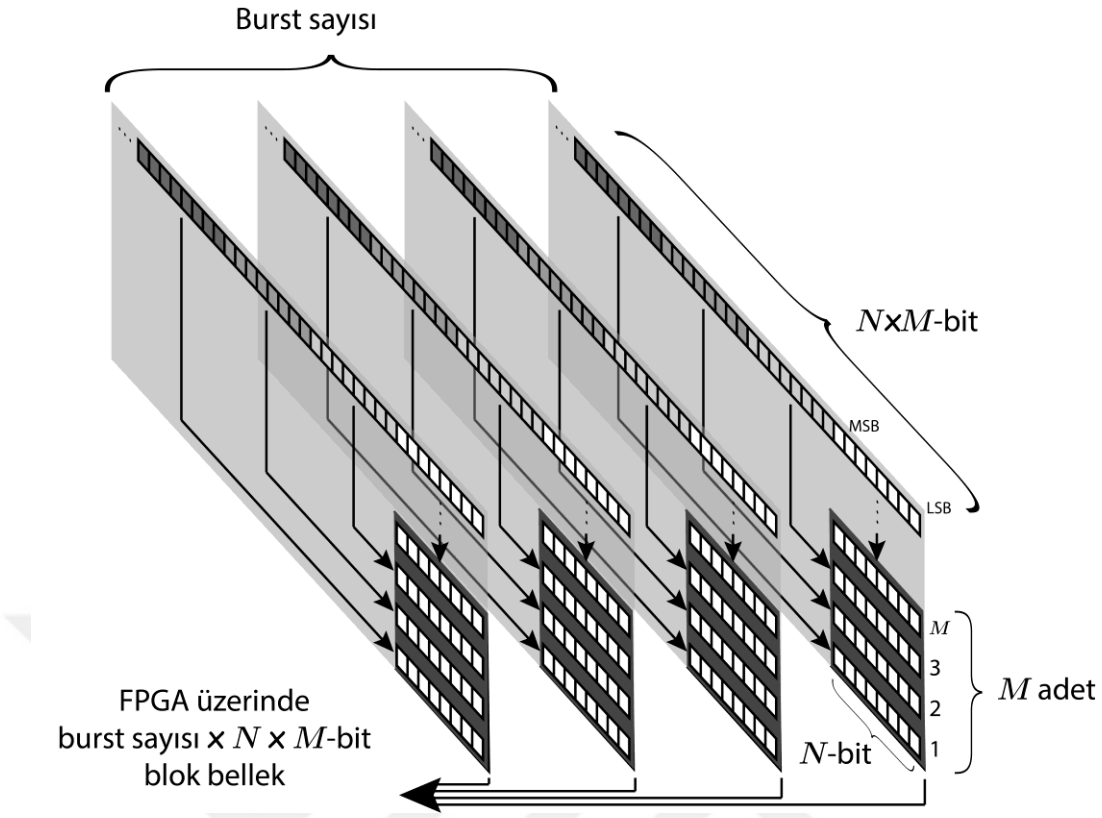


Şekil 2.10 Paralelleme ve serileme işlemleri (a) *Paralelleyici* (b) *Serileyici*

RxFIFO'dan okunan verinin paralelleyici modül kullanılarak paketlenmesi işlemi ve serileyici modül kullanılarak parçalanması işlemi Şekil 2.10'da gösterilmiştir.

2.3.3 Biriktirici Modülü

Paralelleme işleminde veri direkt olarak bellek arayüzünün girişine dizilir. Buna karşın, serileme işleminde eğer burst okuma var ise harici bellekten okunan veriler parçalanmadan önce veri kaybının engellenmesi için sıraya konulup biriktirilmelidir. Bu nedenle serileyici modülün içerisinde burst okuma işleminden elde edilen verilerin saklandığı küçük bir hafıza birimi vardır.



Şekil 2.11 Burst okuma modu için biriktirici modeli

Harici bellek arayüzünün genişliğinin *TxFIFO* verilerinden daha geniş olduğu durumda ve burst okuma modunda iken harici bellekten okunan veriler Şekil 2.11'de gösterildiği biçimde FPGA üzerindeki yazmaçlar kullanılarak oluşturulan blok bellek içerisinde sıralanır ve *TxFIFO*'ya gönderilene kadar kısa süreli depolanır. Sonrasında *TxFIFO*'nun her bir saat döngüsünde veriler çıkışa gönderilir.

BÖLÜM 3

TASARLANAN ARAYÜZÜN KULLANIMI İÇİN ÖRNEK UYGULAMALAR

Bu bölümde, tasarlanan genel amaçlı bellek arayüzünün kullanım alanlarına yönelik bazı çalışmalar yapılmıştır. Temelde kısa süreli ve yüksek hızlı veri aktarımına ihtiyaç duyulan uygulamalarda kullanılması amaçlanan arayüz için gerçek zamanlı görüntü işleme gerçeklemeleri açıklanmıştır.

Uygulamalar için günümüzde yaygın olarak kullanılan Full-HD 1920×1080 piksel çözünürlük ve saniyede 60 kare formatı hedeflenmiştir. Daha düşük çözünürlük ve saniyedeki kare sayıları için de uygulamaların desteklediği formatların içerisindedir. Video giriş ve çıkışı için farklı port türlerinin desteklenmesi de uygulamaların kapsamındadır.

3.1 Uygulamalarda İşlenen Video Sinyalinin Özellikleri

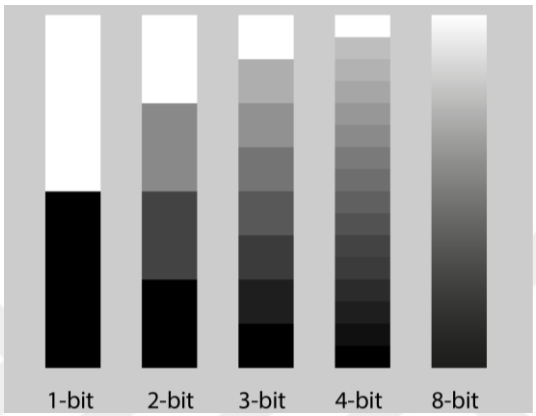
Öncelikle video kavramının, durağan görüntülerin belirli zaman aralıklarıyla art arda akışıyla birlikte oluştuğu göz önünde bulundurulmalıdır. Bir saniye içerisinde art arda kaç durağan görüntü olduğu videonun kare hızını, videoyu meydana getiren durağan görüntüler üzerindeki yatay ve dikey doğrultudaki piksel sayısı da çözünürlüğü ifade eder.

3.1.1 Video Renk Formatı

Tasarım için önerilen uygulamalarda piksellere ait renk bilgisi, yaygın olarak kullanılan RGB[1] formatında temsil edilmiştir. Ek olarak, video sayısal ortamda işlendiği için,

[1] RGB renk formatında R: kırmızı, G: yeşil, B: mavi rengi ifade eder.

piksellerin renk bilgisi bir kuantalama değerine ya da bit derinliğine sahiptir. Buna göre bir video pikselinin bit derinliğine göre elde edilebilecek toplam renk sayısı $2^{(N)}-1$'dir. Bu ifadede N toplam bit derinliğini ifade eder. Örnek olarak, yaygın kullanılan 24-bit renk derinliği için her bir renk kanalı 8-bit'lik bir kuantalamaya sahiptir. RGB formatı için üç renk kanalının toplam bit derinliği $8\times3=24$-bit olmaktadır. Sonuç olarak, 24-bit renk derinliğine sahip bir sistemde görüntülenebilecek toplam renk sayısı hesaplanırsa, $2^{(24)}-1\cong16.8$ milyon renk elde edilir.



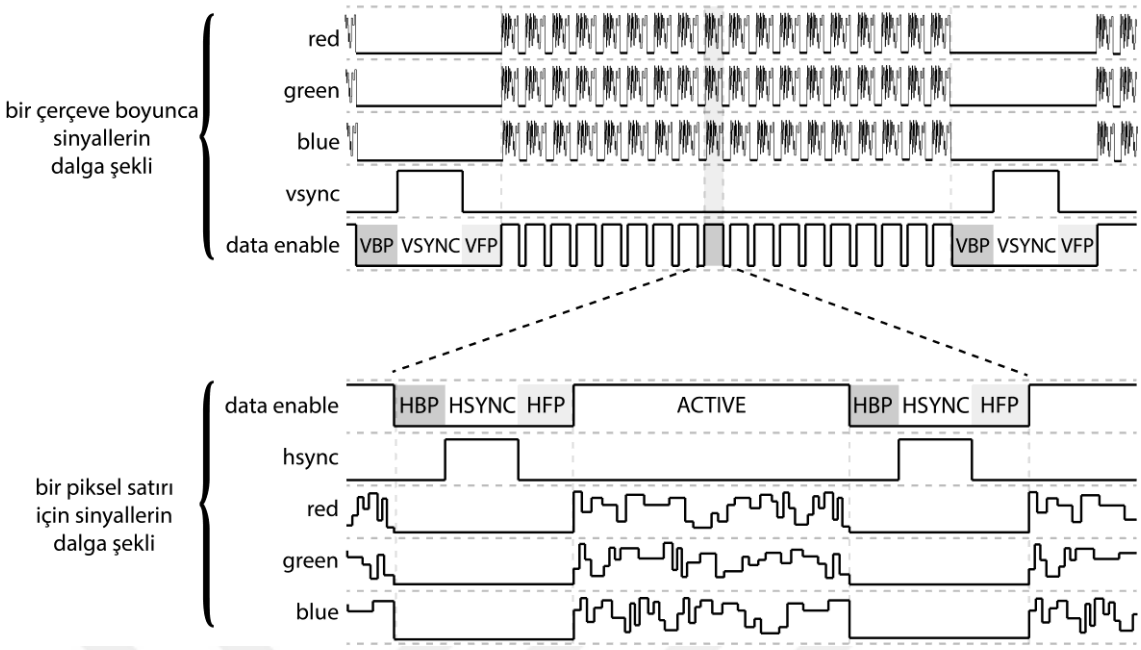
Şekil 3.1 Gri tonlamalı renklerin farklı bit derinliğine göre renk çeşitliliği

Piksel bit derinliği renk çeşitliliğini artırmasına karşın (Şekil 3.1), tasarım üzerindeki veri akışının bant genişliği ihtiyacını artıracaktır. Fakat, prototip geliştirme aşamasında bant genişliğini düşürmek amacıyla işlemler genelde gri-tonlamalı video sinyalleri üzerinde yapılır. Bu sayede görüntü üzerindeki bir piksele ait renk bilgisi, RGB görüntüye ait piksel bilgisinin 1/3'ü kadar bit ile ifade edilmiş olur. 8-bit gri-tonlamalı görüntü akışı göz önünde bulundurulursa, bir pikselin rengi 8-bit ile ifade edilecektir.

3.1.2 Video Sinyal Zamanlamaları

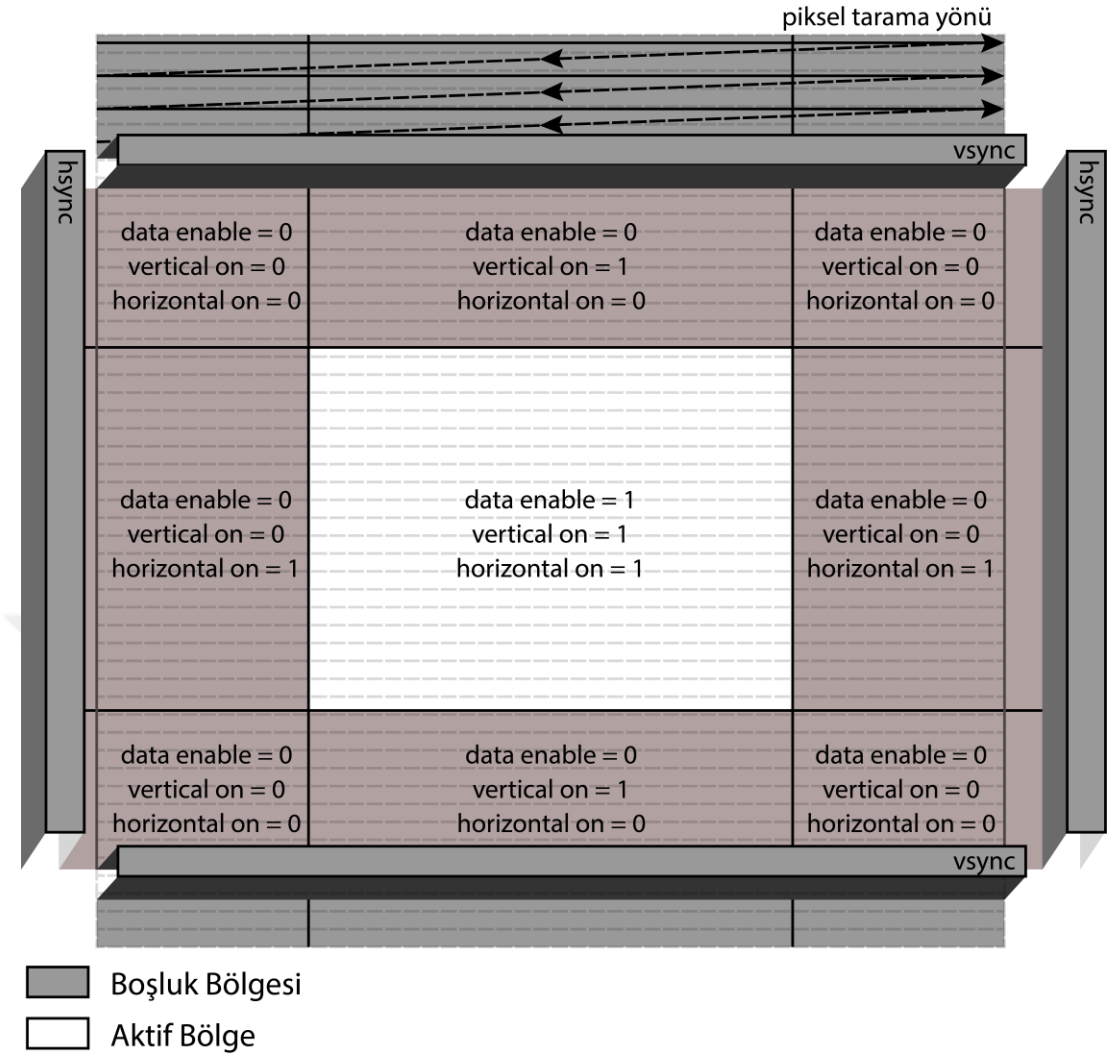
Görüntüleme aygıtlarında yaygın olarak kullanılan video formatlarına ilişkin bazı standartlar oluşturulmuştur. VGA, HDMI, DVI, DisplayPort gibi port türlerinden görüntüleme aygıtlarına aktarılan video sinyallerini temelde üç gruba ayırabiliriz:

- Senkronizasyon sinyalleri: yatay senkronizasyon (*hsync*), dikey senkronizasyon (*vsync*), veri aktif sinyali (*de*).
- Veri sinyalleri (piksel verileri): kırmızı, yeşil, mavi (*red, green, blue*).
- Saat sinyali: piksel saati (*pxclk*).



Şekil 3.2 Video sinyallerine ilişkin zaman diyagramı

Bir video karesi, *vsync* sinyalinin aktif olduğu anda başlamış olur. Videonun senkronizasyon sinyaline ait polariteye göre *vsync* ve *hsync* sinyalinin aktif olduğu mantıksal seviye '1' veya '0' olabilir. Şekil 3.2'te *vsync* ve *hsync* sinyallerinin pozitif (+) polaritede olduğu bir video formatı için dalga şekilleri verilmiştir. Dolayısıyla, video karesi *vsync* sinyalinin '1' olduğu anda başlamaktadır. Görüntüleme aygıtının ekranında görünen alanın dışında kalan inaktif alanda, video senkronizasyonunun sağlanması için gerekli sürelerin doldurulması gerekir. Bu inaktif alanda, aktif sürenin öncesinde dikey doğrultuda; dikey senkronizasyon *vsync*, ön boşluk *vertical front porch (VFP)* ve arka boşluk *vertical back porch (VBP)*, yatay doğrultuda; yatay senkronizasyon *hsync*, ön boşluk *horizontal front porch (HFP)* ve arka boşluk *horizontal back porch (HBP)* sürelerinin sağlanması gerekmektedir. Aktif alanda ise videoya ait piksel bilgisi iletilir. İletilen pikseller görüntüleme aygıtı ekranının sol üst köşesinden sağa doğru ekranı doldurmaya başlar. Yatay doğrultuda bir satırın aktif ve inaktif alanının tamamlanmasının ardından sonraki satırın sol tarafından tarama devam eder. Videonun bir çerçevesine ait bütün piksellerin aktarımı tamamlandıktan sonra, *vsync* sinyalinin tekrar aktif olmasıyla birlikte videoya ait yeni çerçevenin aktarımı başlamış olur. Video sinyaline ait bir karenin aktif ve inaktif bölgelerinin 2-boyutlu gösterimi Şekil 3.3'de verilmiştir.



Şekil 3.3 Video sinyaline ait bir karenin aktif ve inaktif bölgelerinin 2-boyutlu gösterimi

Görüntüleme aygıtlarının video sinyallerini algılayıp, doğru bir şekilde görüntüleyebilmeleri için bazı standartlar oluşturulmuştur. Bu video parametrelerine dair standartlar Çizelge 3.1'de gösterilmiştir.

Çizelge 3.1 Yaygın kullanılan bazı video formatlarına ilişkin standartlar [16]

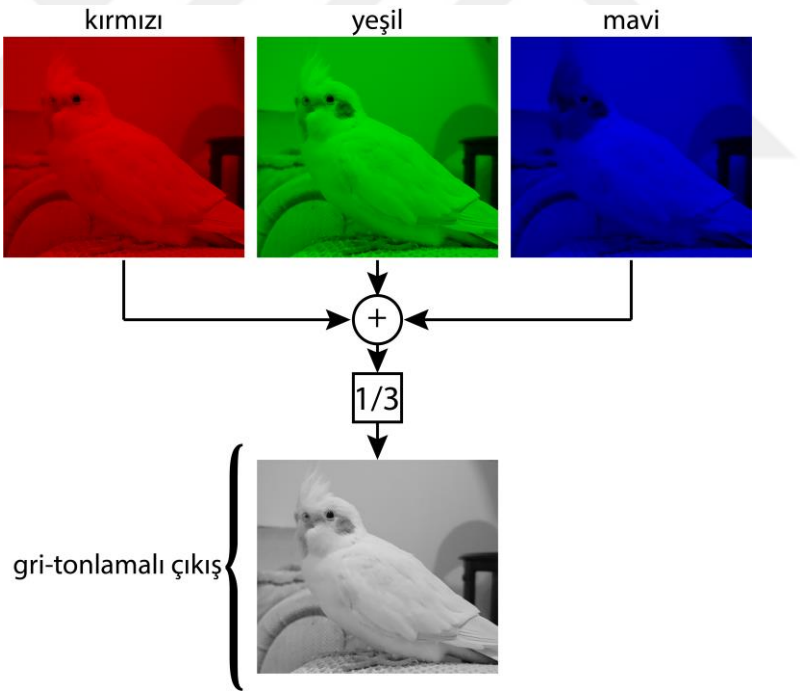
Çözünürlük	**Yatay çöz.**	**Dikey çöz.**	**Piksel saat frekansı**	**VSYNC (satır)**	**VFP (satır)**	**VBP (satır)**	**HSYNC (piksel)**	**HFP (piksel)**	**HBP (piksel)**	**Satırda topl. piksel**	**Topl. satır**
1024x768@60	1024	768	65MHz	6	3	29	136	24	160	1344	806
1280x720@60	1280	720	74.25MHz	5	5	20	40	110	220	1650	750
1366x768@60	1366	768	85.5MHz	3	3	24	143	70	213	1792	798
1920x1080@60	1920	1080	148.5MHz	5	4	36	44	88	148	2200	1125

3.2 Uygulamalar için Tasarlanan Ortak Video Modülleri

Uygulamalarda ortak olarak kullanılan video modülleri bu başlık altında açıklanmıştır. Ortak olarak kullanılan modüllerin yanı sıra, uygulamaya özgü olarak tasarlanan modüller de ilgili uygulamaya ait başlıklar çerçevesinde ele alınmıştır.

3.2.1 RGB'den Gri-Tonlamalı Videoya Dönüştürme Modülü

Öncelikle, arayüzün kullanılabilirliğini kanıtlamak amacıyla uygulamalar gri-tonlamalı video üzerinde yapılmıştır. Renkli video girişini gri-tonlamalı videoya dönüştürmek için ortalama yöntemi kullanılmıştır (Şekil 3.4). Bu yöntem, videonun renklerine ait olan bilgi üzerinde basit bir şekilde ortalama işlemi gerçekleştirmektedir. Şöyle ki, videoya ait olan üç adet renk bilgisi sinyalinin değerleri toplanır ve renk sayısına[1] bölünür. Bu işlem sonucunda videoya ait veri genişliği üçte bir oranında azalır ve harici bellek arayüzünde kullanılacak olan bant genişliği azaltılmış olur.



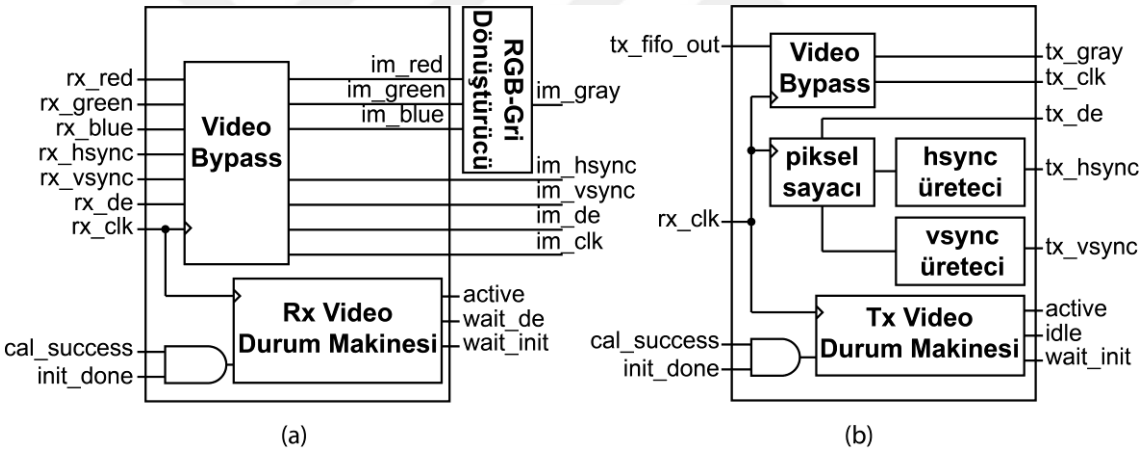
Şekil 3.4 Ortalama alma yöntemiyle gri-tonlama dönüşümü uygulaması

[1] Yaygın olarak kullanılan RGB modelinde bir pikselin içerdiği renk bilgisi sayısı 3'tür. Nadir olarak kullanılan RGBY gibi bazı modellemelerde bir pikseli temsil eden renk sayısı 4'e çıkabilir.

3.2.2 Video Giriş Modülü (Video Rx)

Video kaynağından veri alabilmek için öncelikle giriş bloğunun video sinyallerini işleyebilmesi gerekir. Bu amaçla, video giriş modülü içerisinde temel olarak bir bypass bloğu ve bir durum makinesi oluşturulmuştur (Şekil 3.5 (a)). Bypass bloğu, senkronizasyon sinyalleri ve saat sinyalini ara bloklara doğrudan iletir. Renk verilerine ait sinyaller ise RGB-Gri tonlama dönüştürücü bloğuna iletilir. Gri tonlamaya dönüştürülen renk sinyalleri, senkronizasyon ve saat sinyalleri ile birlikte ara bloklarda kullanılmak üzere yönlendirilir.

Video kaynağından veri alma işleminin başlatılması ile ilgili kararı durum makinesi verir. Bellek kalibrasyonu ve başlatılma işlemleri tamamlandıktan sonra *wait_init* sinyali inaktif olur ve bu aşamadan sonra video sinyalinin ilk aktif olduğu an beklenir. Video için veri aktif *de* sinyali aktif olduktan sonra *rx video durum makinesi* aktif duruma geçer ve video verisinin yakalanma işlemi de başlatılmış olur.



Şekil 3.5 Tasarlanan video giriş - çıkış modüllerinin yapısı. (a) Giriş modülü (b) Çıkış modülü

3.2.3 Video Çıkış Modülü (Video Tx)

Video çıkış modülüne gelen görüntüye ilişkin senkronizasyon, bellek arayüzünden okunan veri ile alakalı olduğu için giriş bloğuna gelen görüntünün senkronizasyon sinyalleri çıkış modülü için kullanılamayacaktır. Bu nedenle çıkış modülünde senkronizasyon sinyallerinin bellekten okunan veriye ait zamanlama ile uyumlu olacak şekilde yeniden üretilmelidir. Video çıkış modülü içerisinde senkronizasyon sinyallerinin üretildiği *hsync üreteci* ve *vsync üreteci* blokları bulunmaktadır (Şekil 3.5 (b)). Bu bloklar, çıkış modülü içerisinde bulunan *tx video durum makinesi*

kontroluündedir. Video giriş modülünde olduğu gibi, bu durum makinesinin de başlangıç koşulunda bellek kalibrasyonu ve belleğin başlatılma işleminin tamamlanmasını beklemektedir. Kalibrasyon ve başlatılma işlemi tamamlandıktan sonra durum makinesi bellek arayüzünden gelecek olan video verisini beklemeye başlar. Bu durum *idle* olarak adlandırılmıştır. Bellek üzerinde okunmaya yetecek miktarda veri depolandıktan sonra senkronizasyon sinyalleri oluşturulmaya başlar ve bellekten video verisi çağırılır. Çağırılan video verisi bir işlem bloğundan geçirilecek ise ilgili bloğa yönlendirilir ve son olarak çıkışa aktarılır. Giriş video formatının gerektirdiği zamanlama standartları çıkış bloğunda da aynı şekilde sağlanır.

3.3 Gerçek Zamanlı Video için Arabellek Uygulaması

Bu uygulamanın hedefi, video kaynağındaki çerçeveleri yakaladıktan sonra bir süre bellek üzerinde saklayıp, daha sonra gerçek zamanlı olarak bir video görüntüleme aygıtına göndermektir. Bu durumda video kaynağından gelen verinin kesintisiz olduğu ve çıkış bloğunda da aynı şekilde kesintisiz bir veri akışı sağlanması gerektiği göz önünde bulundurulmalıdır.

Bu uygulama kapsamında, ikinci bölümde yapısı açıklanan bellek arayüzünün giriş ve çıkışında *Video Rx* ve *Video Tx* modülleri kullanılmıştır. *Video Rx* modülü aracılığıyla yakalanan video sinyali, *RGB-Gri dönüştürücü* modüle aktarılıp gri tonlamalı video sinyaline dönüştürüldükten sonra *RxFIFO* modülüne gönderilir. *RxFIFO-bellek durum makinesi, RxFIFO* içerisindeki video verisinin belleğe yazılması ile ilgili işlemleri kontrol eder. Bellekten okunacak video verileriyle ilgili işlemleri de *bellek-TxFIFO durum makinesi* kontrol eder. Bellekten okunan video verileri *TxFIFO* modülüne gönderildikten sonra çıkış bloğuna bağlı olan görüntüleme aygıtına gönderilmeye hazır olur. Bu işlem, (3.1) denkleminde ifade edilmiştir. Bu denklemlde *t* zaman değişkenini, Δt işleme giren video kareleri arasındaki zaman farkını, pikseller görüntüyü oluşturan noktaları, x ve y ise piksellerin sırasıyla dikey ve yatay koordinatlarını temsil eder.

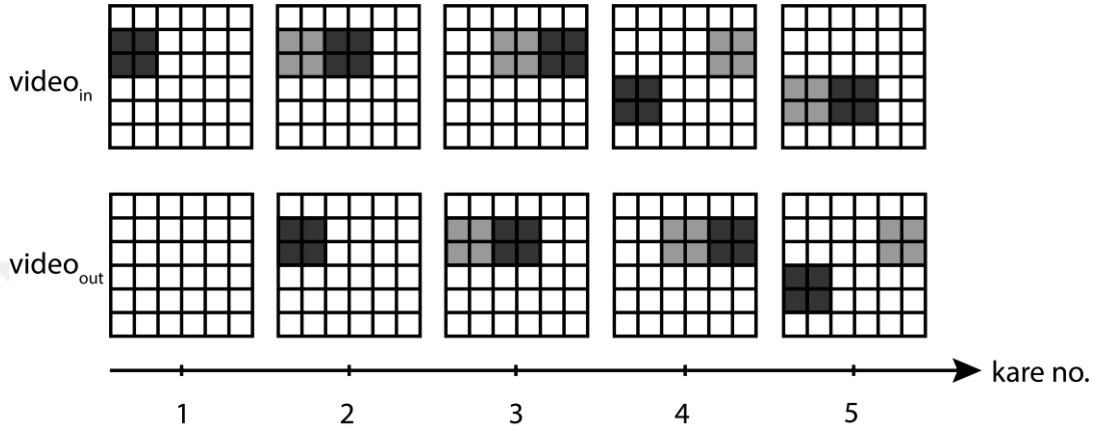
$$\text{piksel}_{(x,y)_{out}}(t) = \text{piksel}_{(x,y)_{in}}(t - \Delta t) \tag{3.1}$$

Günümüzde kullanılan görüntü aygıtları yaygın olarak sayısal görüntü formatlarıyla çalışmaktadırlar. Bu nedenle bellekleme işleminin sayısal işlemlere uygun olarak ifade edilmesi gerekir. Eğer ardışık video kareleri arasındaki zaman farkı T_s olarak ifade edilirse, denklem (3.2)'deki gibi yazılabilir. Sonrasında, denklem (3.3)'te ayrık

zamanda ifade edilen bellekleme işlemi gösterilmiştir. Bu denklemde n video karesinin indisini, k işleme giren video karelerinin indisleri arasındaki mutlak değer farkını temsil eder.

$$\text{piksel}_{(x,y)_{out}}(nT_s) = \text{piksel}_{(x,y)_{in}}(nT_s - kT_s), \quad n \in \mathbf{Z}, \quad k \in \mathbf{Z}^+ \tag{3.2}$$

$$\text{piksel}_{(x,y)_{out}}[n] = \text{piksel}_{(x,y)_{in}}[n-k], \tag{3.3}$$

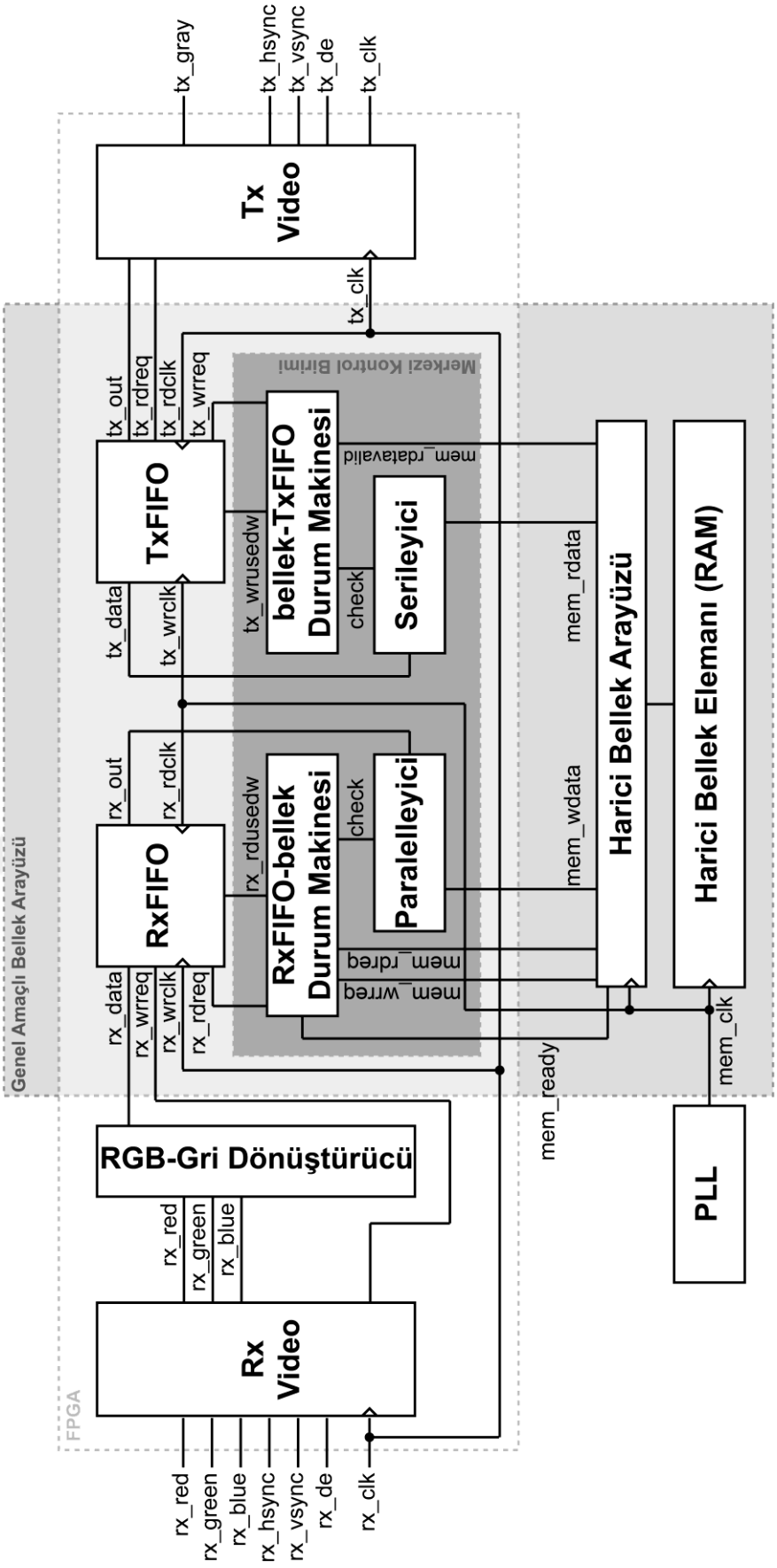


Şekil 3.6 Örnek giriş video karelerine karşılık çıkış video kareleri

Bellek üzerinde depolanacak olan video verisinin artırılıp, bellekten okunacak olan video verisinin okunma işleminin geciktirilmesi, başka bir deyişle k değerinin artırılması, çıkış bloğuna bağlanan görüntüleme aygıtı üzerinde görüntülenecek olan video karesinin gecikmesini belirleyecektir. Bu sayede giriş ve çıkış görüntüsü arasında gecikme ayarının yapılabilmesi mümkün olacaktır. Ayrıca bellekten okunan önceki video kareleri üzerinde sinyal işleme uygulamalarının gerçekleştirilmesi de mümkündür. Bu gerçek zamanlı uygulamaya ait blok diyagram Şekil 3.7'de gösterilmiştir.

3.4 Gerçek Zamanlı Video Çerçeveleri için Fark Alıcı Uygulaması

Bu uygulama kapsamında giriş bloğuna uygulanan gerçek zamanlı bir video kaynağından alınan görüntünün, çıkış bloğuna bağlı bulunan görüntüleme aygıtına gönderilmeden önce kendisinden önceki video kareleri ile fark işlemine tabi tutulması amaçlanmıştır. Video kareleri üzerinde uygulanan fark alma işlemi, zamanda türev anlamı taşımaktadır. Öyle ki; (3.4) denklemi sürekli zamanda türev işleminin ifadesidir.



Şekil 3.7 Gerçek zamanlı videolar için arabellek uygulamasına ait blok diyagram

$$\text{piksel}_{(x,y)_{out}}(t) = \frac{\text{piksel}_{(x,y)_{in}}(t) - \text{piksel}_{(x,y)_{in}}(t - \Delta t)}{\Delta t} \tag{3.4}$$

(3.4) denklemi ayrık zamanda türev işlemi olarak ifade edilmek istenirse, sürekli zaman değişkeni, video karesine ait indis cinsinden ifade edilir. Böylece, T_s ardışık video kareleri arasındaki zaman farkı olmak üzere (3.5) ve (3.6) denklemleri ile ayrık zamanda türev işlemi ifade edilmiş olur. (3.6) denkleminde n video karesinin indisi, k ise işleme giren video karelerinin indisleri arasındaki mutlak değer farkıdır.

$$\text{piksel}_{(x,y)_{out}}(nT_s) = \frac{\text{piksel}_{(x,y)_{in}}(nT_s) - \text{piksel}_{(x,y)_{in}}(nT_s - kT_s)}{kT_s}, \qquad n \in \mathbf{Z}, \quad k \in \mathbf{Z}^+ \tag{3.5}$$

$$\text{piksel}_{(x,y)_{out}}[n] = \frac{\text{piksel}_{(x,y)_{in}}[n] - \text{piksel}_{(x,y)_{in}}[n-k]}{k} \tag{3.6}$$

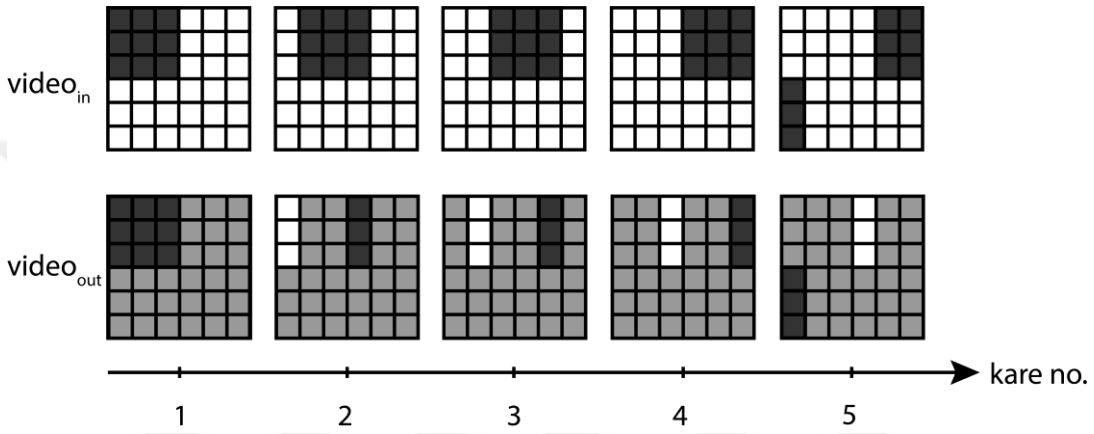
Fakat (3.6) ifadesinin, görüntüleme aygıtlarında bir karşılığı olması için piksel değerlerinin aralığının görüntüleme aygıtlarının çalışabileceği $[0,2^{N-1}]$ değer aralığına uydurulması gerekir. Burada N renk derinliğinin bit cinsinden ifadesidir. O hâlde; (3.7) ifadesi ile piksel değerlerinin bulunduğu aralık, görüntüleme aygıtlarının piksel değeri aralığı olan $[0,2^{N-1}]$ aralığına göre düzenlenmiş olur. Ek olarak, sayısal formatta tam sayılara ihtiyaç olduğu için (3.7) denkleminin sonucuna yuvarlanma işlemi uygulanması gerekmektedir.

$$\text{piksel}_{(x,y)_{out}}[n+1] = \frac{\left(\text{piksel}_{(x,y)_{in}}[n] - \text{piksel}_{(x,y)_{in}}[n-k]\right) + \text{maks}\left(\text{piksel}_{(x,y)_{in}}[n]\right)}{2k} \tag{3.7}$$

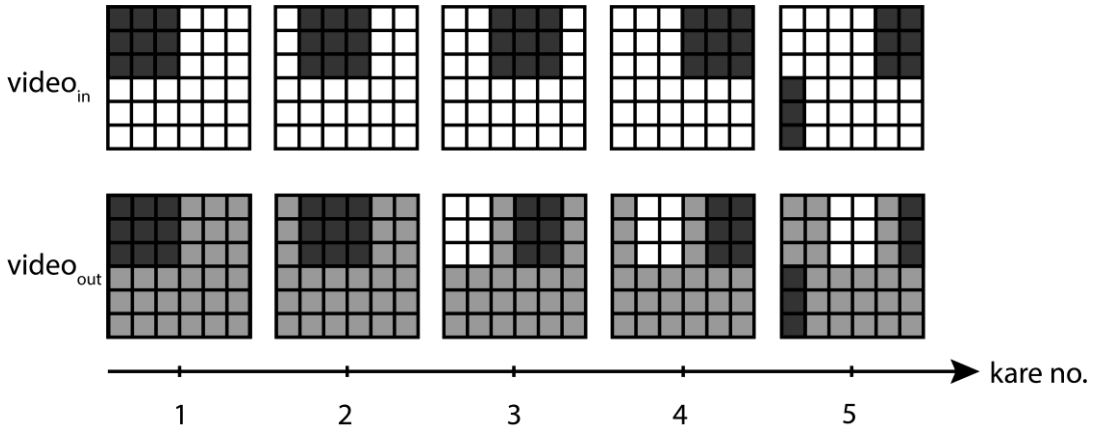
Uygulamaya ait blok diyagram Şekil 3.10'de gösterilmiştir. Uygulama için genel amaçlı bellek arayüzü bloğuna ek olarak, 3.2.1 bölümünde gösterilen RGB'den gri tonlamaya dönüştürme modülü (*RGB-Gri Dönüştürücü*), 3.2.2 bölümünde gösterilen video giriş modülü (*Video Rx*) ve 3.2.3 bölümünde gösterilen video çıkış modülü (*Video Tx*) kullanılmıştır.

Giriş bloğundaki video yakalama işlemi, 3.3 bölümündeki uygulamaya benzer şekilde *Video Rx* modülü aracılığıyla gerçekleştirilir. Yakalanan video sinyali *RGB-Gri dönüştürücü* modüle aktarılıp gri tonlamalı video sinyaline dönüştürüldükten sonra *RxFIFO* modülüne gönderilir. Yine benzer şekilde *RxFIFO-bellek durum makinesi, RxFIFO* içerisindeki video verisinin belleğe yazılması ile ilgili işlemleri kontrol eder.

Bellekten okunması gereken video verileriyle ilgili işlemleri de *bellek-TxFIFO durum makinesi* kontrol eder. Bu aşamadan sonra, giriş bloğunda yakalanmakta olan video verisi ile bellekten okunan daha eski video karelerine ait veriler arasındaki fark alma işlemi *Video Kare Farkı Alıcı* modül üzerinde gerçekleştirilir. Elde edilen sonuç, çıkış tarafında yer alan *Video Tx* modülüne gönderilir. Örnek görüntü üzerinde işlemin uygulanması Şekil 3.8'te ve Şekil 3.9'da gösterilmiştir. Bu örneklerde n=1 değerinden önceki görüntülerin boş olduğu, başka bir deyişle video karelerine ait piksellerin sıfır olduğu varsayılmıştır.

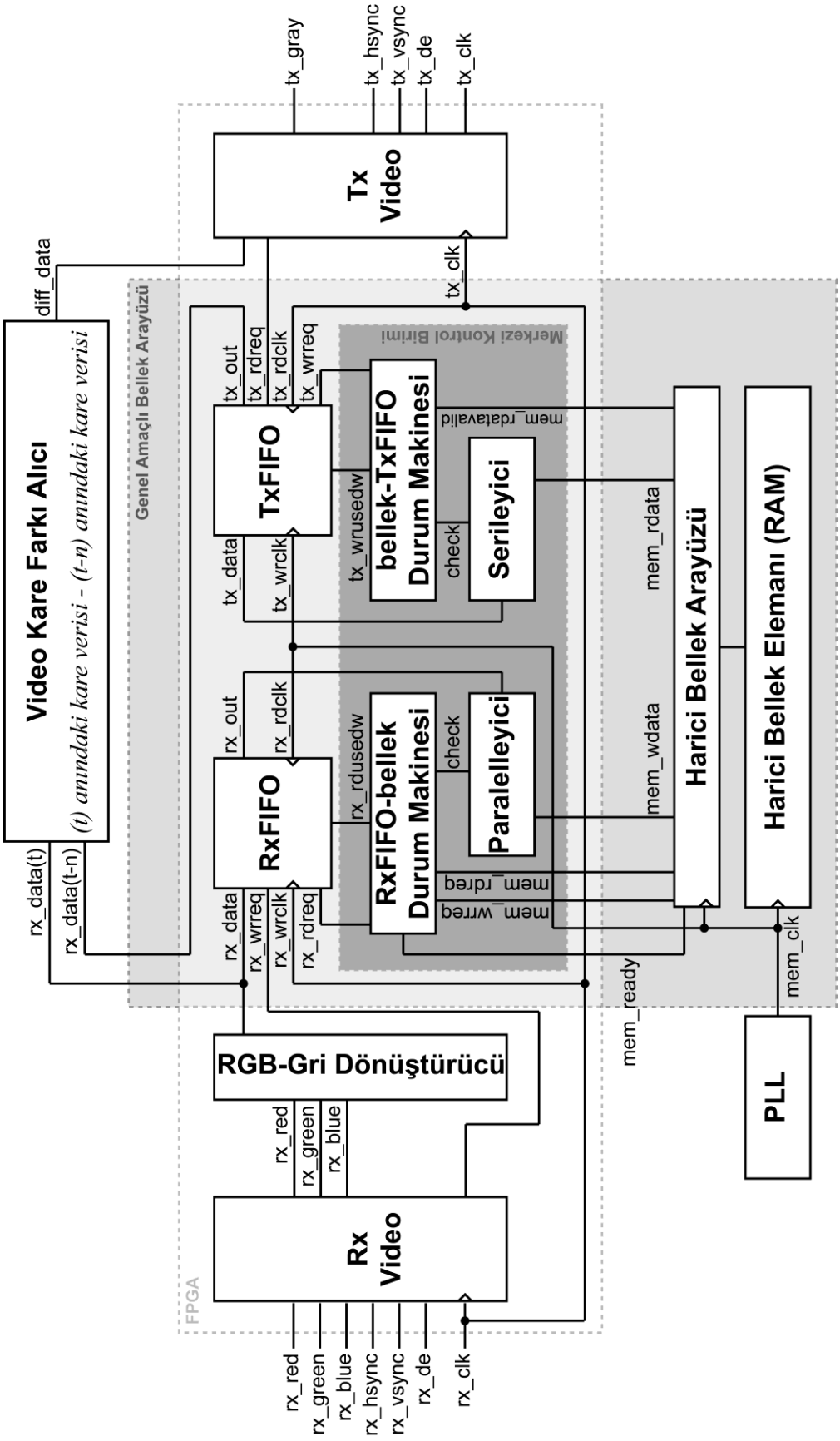


Şekil 3.8 k=1 için video kareleri üzerinde fark alma işlemi



Şekil 3.9 k=2 için video kareleri üzerinde fark alma işlemi

Sonuç olarak; anlık video karesine ait piksel verisi ile daha önceki video karelerine ait piksel verileri arasında bir fark işlemi gerçekleştirilip, görüntüleme aygıtına aktarılmış olur. Bu sayede video karelerinde hareketli olan objelerin görüntü üzerinde tespit edilmesi ve hareketsiz bölgelerin görüntü üzerinde yok edilmesi işlemi mümkün olmaktadır.



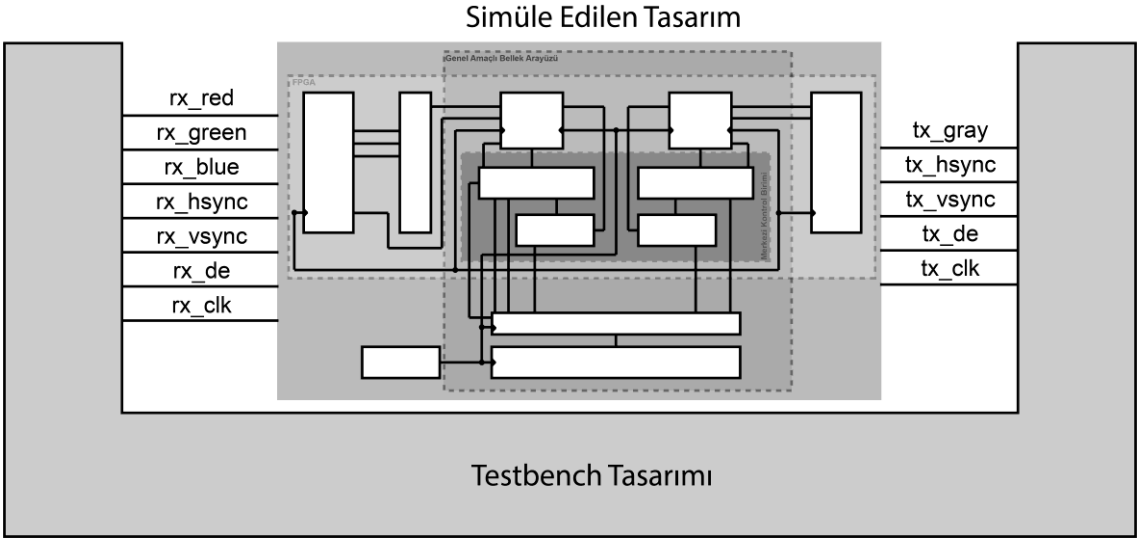
Şekil 3.10 Gerçek zamanlı video çerçeveleri için fark alıcı uygulamasının blok diyagramı

BÖLÜM 4

ÖRNEK UYGULAMALARIN SİMÜLASYONLARI

Donanım tanımlama dili (HDL) kullanılarak tasarlanan yapılar, hedeflenen kriterlere uygunluklarının test edilebilmesi için gerçekleme aşamasından önce simülasyon işlemine tabi tutulmalıdır. Bu sayede tasarımın en dış katmanda yer alan giriş çıkış sinyallerinin yanı sıra, modüllerin içerisinde yer alan ve gerçeklemeden sonra gözlemlenmesi kolay olmayan sinyallerin de test edilmesi mümkün olmaktadır. En önemlisi de, HDL tasarımlarının simülasyonu sırasında tasarıma ait sinyallerin zamanlamaları rahatlıkla gözlemlenebilmektedir.

Simülasyon işlemi, ModelSim uygulaması kullanılarak gerçekleştirilmiştir. Simülasyon için testbench oluşturma aşamasında VHDL tercih edilmiştir. Ek olarak, gerçekleme için kullanılacak olan Altera FPGA kartlarına ait bazı simülasyon modülleri ModelSim dosyası içerisinde kullanılmıştır.



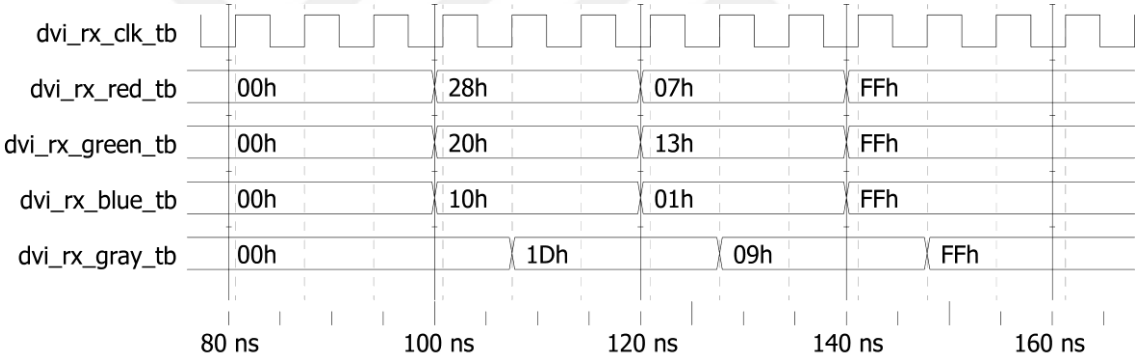
Şekil 4.1 Testbench ve simüle edilen tasarım ilişkisini gösteren diyagram

4.1 Uygulamalarda Kullanılan Ortak Modüllerin Simülasyonları

Tasarıma ait sinyal davranışlarının incelenmesi için, tasarımın portlarının ve iç bağlantılarının simülasyon için oluşturulması gereken *testbench*[1] ortamı içerisinde yer alması gerekir (Şekil 4.1). Tasarımın bütün modüllerinin kendi içerisinde ayrı ayrı simüle edilip test edilmesi gerçekleme adımına geçmeden önce büyük önem taşır. Bu nedenle genel amaçlı bellek arayüzü tasarımına ait modüller kendi içerisinde ayrıca simüle edilmiştir.

4.1.1 RGB'den Gri-Tonlamalı Videoya Dönüştürme Modülü Simülasyonu

Tasarlanan RGB'den gri-tonlamaya dönüşüm modülünün simülasyonu için oluşturulan *testbench* ile 148.5 MHz frekansına sahip bir piksel saati, 8-bit renk derinliğine sahip üç renk giriş kanalı (kırmızı, yeşil, mavi) ve 8-bit renk derinliğine sahip gri-tonlamalı çıkış kanalı simüle edilmiştir. Simülasyona ait dalga şekli Şekil 4.2'de gösterilmiştir.



Şekil 4.2 RGB'den gri-tonlamalı videoya dönüştürme modülü simülasyonuna ait dalga şekli

Çizelge 4.1 RGB'den gri-tonlamalı videoya dönüştürme modülü simülasyonuna ait sinyallerin listesi

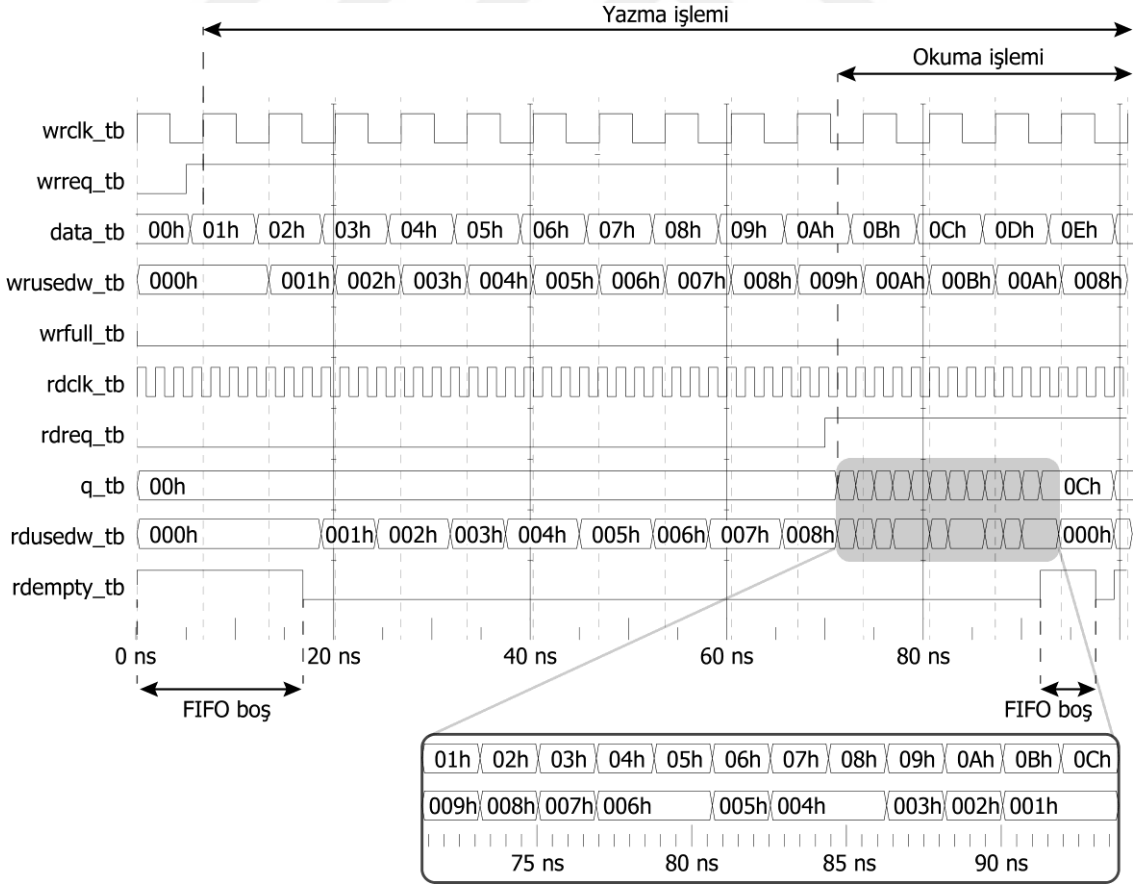
Sinyal İsmi	**Sinyal Türü**	**Sinyal Genişliği**	**Açıklama**
dvi_rx_clk_tb	Giriş	1-bit	148.5 Mhz saat sinyali
dvi_rx_red_tb	Giriş	8-bit	Kırmızı renk kanalına ait sinyal
dvi_rx_green_tb	Giriş	8-bit	Yeşil renk kanalına ait sinyal
dvi_rx_blue_tb	Giriş	8-bit	Mavi renk kanalına ait sinyal
dvi_rx_gray_tb	Çıkış	8-bit	Çıkış sinyali (Gri-ton)

[1] Testbenchler, HDL kullanılarak oluşturulan tasarımların simüle edilebilmesi için gerekli giriş ve çıkış sinyallerinin yapay olarak tanımlandığı, buna bağlı olarak tasarımın özelliklerinin gözlemlenebilmesine imkân veren yardımcı tasarımlardır.

Testbench içerisinde giriş renkleri için ilk olarak 80 ns anında ‘0’ değeri uygulanmıştır ve çıkışta ‘0’ görülmüştür. Daha sonra 100 ns anında kırmızı renk girişine onaltılık sistemde ‘28’ (onluk sistemde ‘40’), yeşil renk girişine ‘20’ (onluk sistemde ‘32’) ve mavi renk girişine ‘10’ (onluk sistemde ‘16’) uygulanmıştır. Ortalama alma işlemi sonucunda (40+32+16)/3=29,33 elde edilir. Bu sonuç modül çıkışına aktarılmadan önce yuvarlama işlemi uygulanır. Sonuç olarak ‘1D’ (onluk sistemde ‘29’) değeri çıkış sinyaline atanır. Modüle uygulanan bu girişlere ilişkin sonuç, iki saat döngüsü sonunda çıkışta görülmektedir. Test edilmek üzere 120 ns ve 140 ns anında uygulanan iki farklı giriş dizisi daha Şekil 4.2 üzerinde gözlemlenebilir.

4.1.2 RxFIFO Modülü Simülasyonu

Örnek uygulama gerçeklemelerinde *RxFIFO* modülü üzerine bir video kaynağından veri yazılacağı için, *RxFIFO* modülünün simülasyonunda, FIFO’ya yazma işlemi için 148.5 MHz saat frekansına sahip *rdclk_tb* sinyali kullanılmıştır. Bu saat frekansı full-HD 1080p çözünürlüklü, saatte 60 kareye sahip bir video sinyalinin piksel frekansıdır.



Şekil 4.3 RxFIFO modülü simülasyonuna ait okuma-yazma işlemi sinyal zamanlamaları

RxFIFO modülünden bellek arayüzü aracılığıyla veri okunup, harici bellek üzerine yazılacağından, FIFO'dan okuma işlemi için 533.33 MHz saat frekansına sahip *wrclk_tb* sinyali kullanılmıştır. Bu saat frekansı, gerçekleme aşamasında kullanılacak olan bellek arayüzü frekanslarından birisidir.

Testbench içerisinde, *RxFIFO* modülüne ilk olarak birer birer artan ardışık veriler sırayla yazılmıştır. Daha sonra *RxFIFO* modülüne okuma isteği gönderilip, veriler sırayla okunmuştur. İlk sırada yazılan verinin, okuma işlemi esnasında yine ilk sırada okunması gözlemlenmiştir (Şekil 4.3). Tasarımda kullanılan FIFO modüllerine ait senkronizasyon ve gecikme karakteristiklerine bağlı olarak, FIFO içerisindeki veri miktarını gösteren sinyallerde ve taşma veya boş kalma durumlarına ait sinyallerde bazı gecikmeler oluşmaktadır [12]. Bu nedenle, FIFO modülünün doluluk miktarını gösteren sinyallerin kullanımında bu gecikmeler göz önüne alınmalıdır. Alternatif olarak, yazılan ve okunan veriler ile ilgili ek bir sayaç bloğu tasarıma eklenebilir.

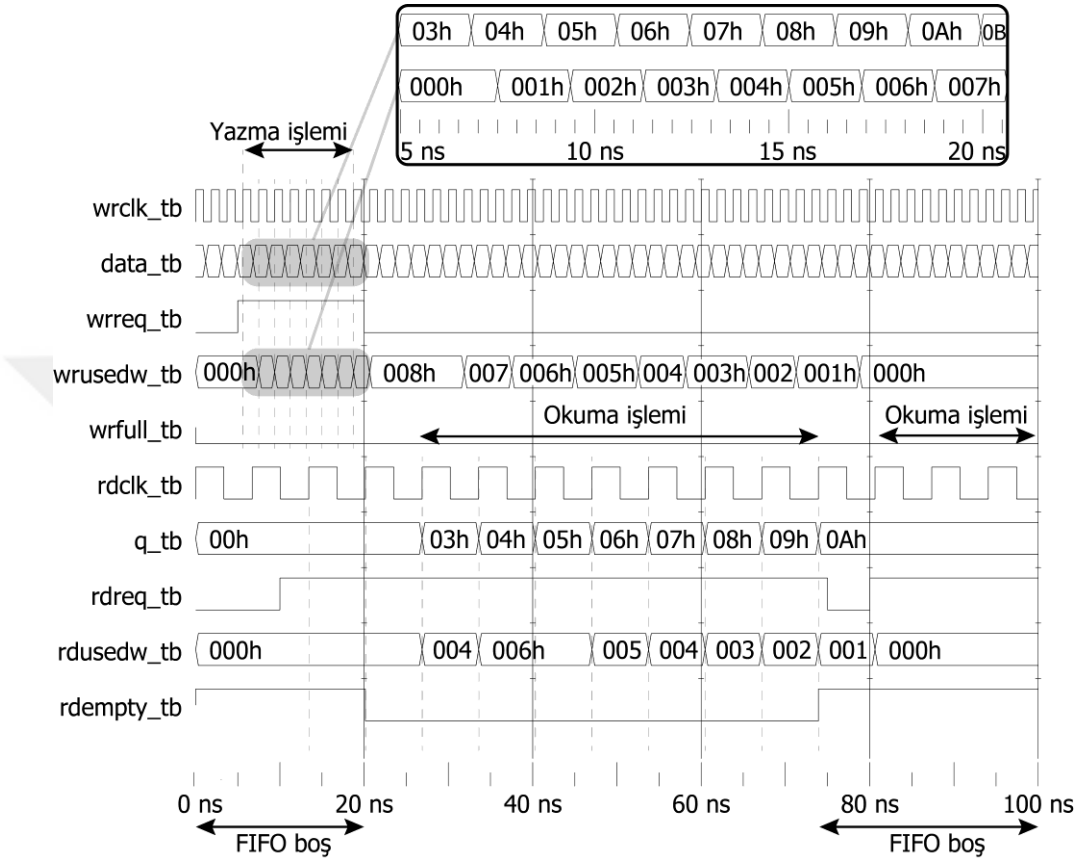
Çizelge 4.2 RxFIFO modülü simülasyonuna ait sinyallerin listesi

Sinyal İsmi	**Sinyal Türü**	**Sinyal Genişliği**	**Açıklama**
wrclk_tb	Giriş	1-bit	533.33 Mhz yazma işlemi saati
wrreq_tb	Giriş	1-bit	Yazma talebi girişi
data_tb	Giriş	8-bit	Yazma işlemi için veri girişi
wrusedw_tb	Çıkış	11-bit	Yazma işlemi sayısı
wrfull_tb	Çıkış	1-bit	FIFO dolu bildirim sinyali
rdclk_tb	Giriş	1-bit	148.5 Mhz okuma işlemi saati
rdreq_tb	Giriş	1-bit	Okuma talebi girişi
q_tb	Çıkış	8-bit	Okuma işlemi için veri çıkışı
rdusedw_tb	Çıkış	11-bit	Okuma işlemi sayısı
rdempty_tb	Çıkış	1-bit	FIFO boş bildirim sinyali

4.1.3 TxFIFO Modülü Simülasyonu

TxFIFO modülünün simülasyonu, *RxFIFO* modülünün simülasyonuna benzer şekilde gerçekleştirilmiştir. Örnek uygulama gerçeklemelerinde *TxFIFO* modülü üzerine harici bellekten okunan veri yazılacağı için, FIFO'ya yazma işlemi için 533.33 MHz saat frekansına sahip *wrclk_tb* sinyali kullanılmıştır. Bu saat frekansı, gerçekleme aşamasında kullanılacak olan bellek arayüzü frekanslarından birisidir.

TxFIFO modülü, tasarımların çıkış bloğuna yakın olacaktır. Çıkış bloğunda da görüntüleme aygıtı bulunacağı için, FIFO'dan veri okuma işlemi için 148.5 MHz saat frekansına sahip *rdclk_tb* sinyali kullanılmıştır. Bu saat frekansı full-HD 1080p çözünürlüklü, saatte 60 kareye sahip bir video sinyalinin piksel frekansıdır.



Şekil 4.4 TxFIFO modülü simülasyonuna ait okuma-yazma işlemi sinyal zamanlamaları

Testbench içerisinde, *TxFIFO* modülüne ilk olarak birer birer artan ardışık veriler sırayla yazılmıştır. Daha sonra *TxFIFO* modülüne okuma isteği gönderilip, veriler sırayla okunmuştur. İlk sırada yazılan verinin, okuma işlemi esnasında yine ilk sırada okunması gözlemlenmiştir (Şekil 4.4). Tasarımda kullanılan FIFO modüllerine ait senkronizasyon ve gecikme karakteristiklerinden dolayı, FIFO'nun boş bildirimi ve doluluk oranının gözlemlenmesinde bir yada birkaç saat döngüsü gecikme olmaktadır [12].

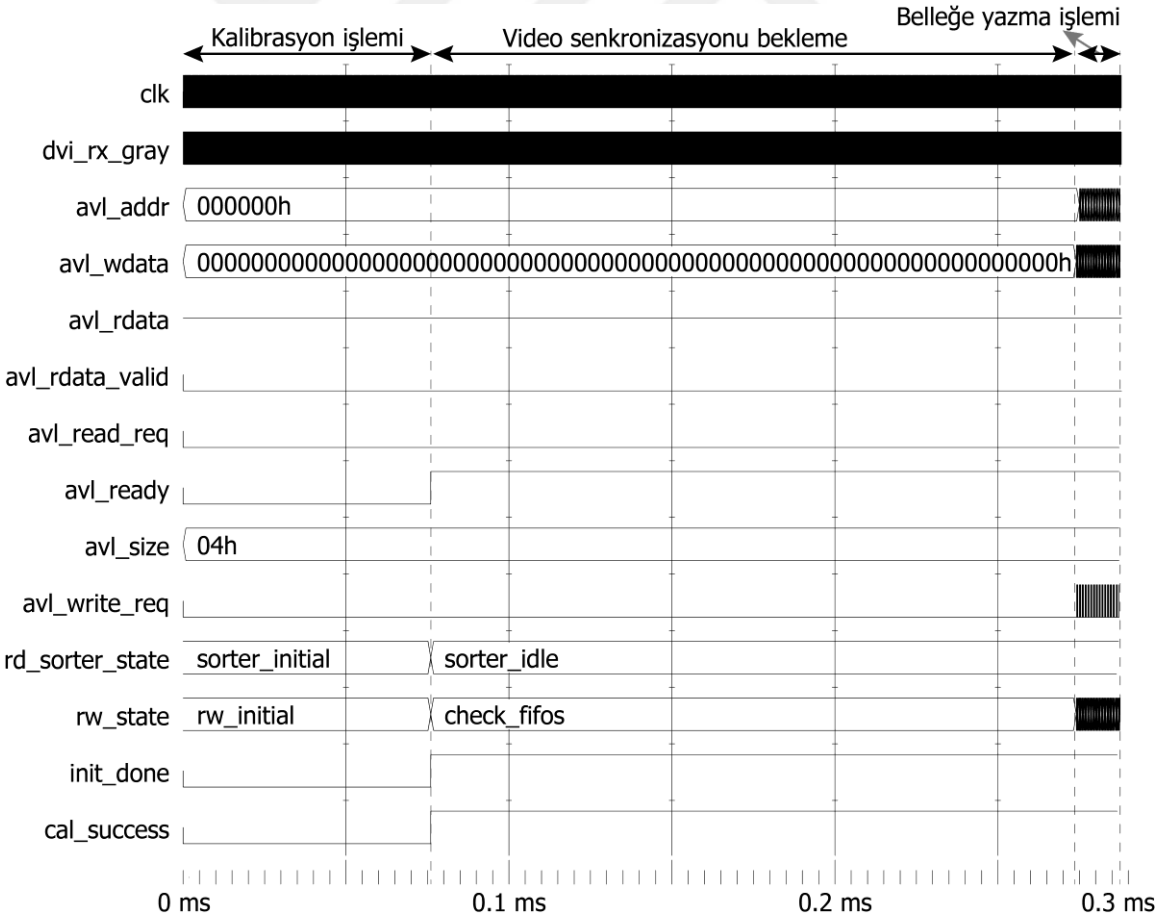
Şekil 4.4'te gözlemlenen 80 ns anındaki ikinci okuma isteği sonrasında, *TxFIFO* modülü boş olduğu için başarılı bir okuma işlemi gerçekleştirilememiştir. Sonuç olarak, *TxFIFO* modülünün çıkışında (*q_tb*) önceki çıkış değeri korunmuştur.

Çizelge 4.3 TxFIFO modülü simülasyonuna ait sinyallerin listesi

Sinyal İsmi	**Sinyal Türü**	**Sinyal Genişliği**	**Açıklama**
wrclk_tb	Giriş	1-bit	148.5 Mhz yazma işlemi saati
wrreq_tb	Giriş	1-bit	Yazma talebi girişi
data_tb	Giriş	8-bit	Yazma işlemi için veri girişi
wrusedw_tb	Çıkış	11-bit	Yazma işlemi sayısı
wrfull_tb	Çıkış	1-bit	FIFO dolu bildirim sinyali
rdclk_tb	Giriş	1-bit	533.33 Mhz okuma işlemi saati
rdreq_tb	Giriş	1-bit	Okuma talebi girişi
q_tb	Çıkış	8-bit	Okuma işlemi için veri çıkışı
rdusedw_tb	Çıkış	11-bit	Okuma işlemi sayısı
rdempty_tb	Çıkış	1-bit	FIFO boş bildirim sinyali

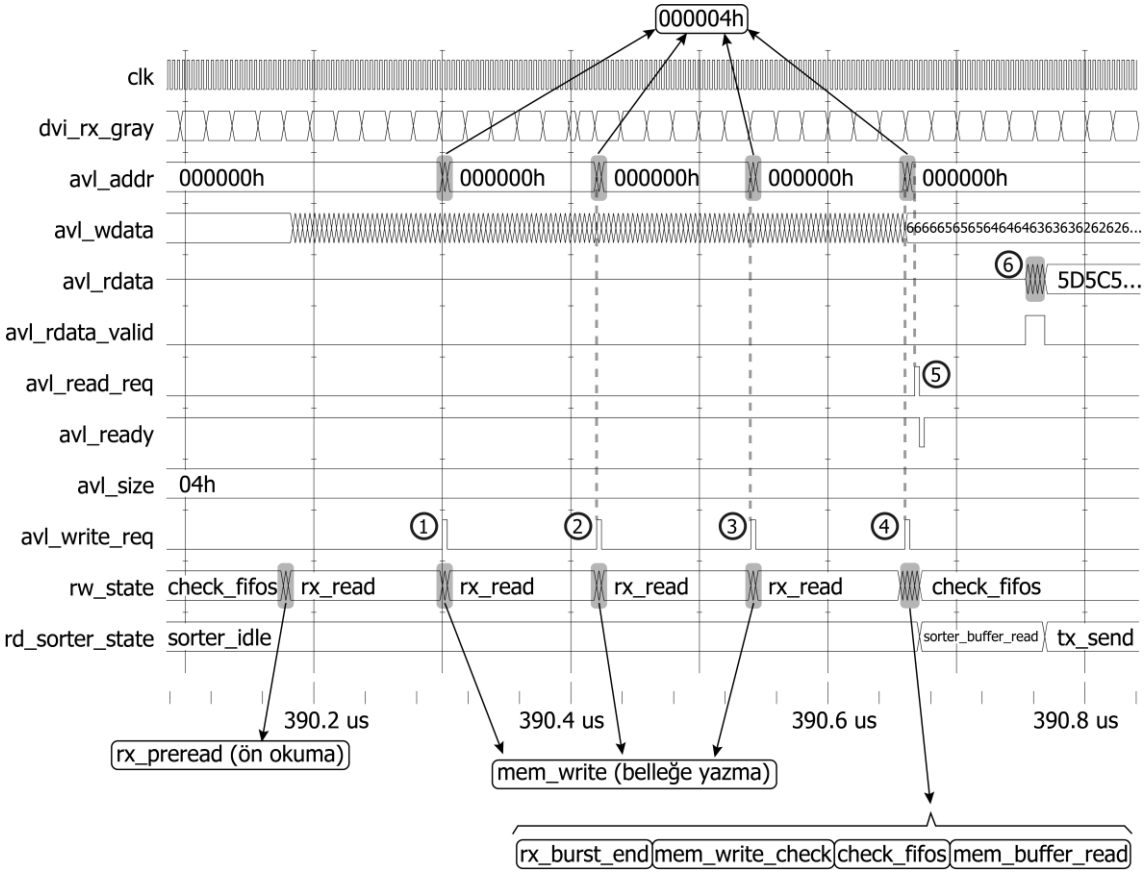
4.1.4 Harici Bellek Arayüzü Simülasyonu

Harici bellek arayüzünün simülasyonu için, 1920×10 çözünürlüğünde video test sinyali kullanılmıştır. Bu sinyalin belleğe yazılma ve bellekten okunma işlemleri gözlenmiştir.



Şekil 4.5 Harici bellek arayüzü simülasyonuna ait kalibrasyon, senkronizasyon ve yazım işlemlerinin zaman diyagramı

Belleğin yazma ve okuma işlemlerine başlayabilmesi için ilk olarak kalibrasyon ve ilk kullanıma hazırlık (initialization) aşamasının tamamlanması gerekir. Simülasyonda DDR3 SDRAM bellek modeli için bu aşama 76µs sürmektedir. DDR SDRAM bellek modeli için ise 17µs sürmektedir. Sonrasında, bellek kalibrasyonunun ardından gelen ilk yeni video karesinin başlangıç anı referans alınarak önbellekleme işlemi başlatılır. Bu aşamaya kadar geçen sürede video senkronizasyonu beklenmiş olur (Şekil 4.5).



Şekil 4.6 Harici bellek arayüzü simülasyonuna ait burst modunda yazma ve okuma işlemlerine ait zaman diyagramı

Harici bellek arayüzüne ait detaylı durum değişimleri, burst modunda yazma ve okuma işlemleri zamanlamaları Şekil 4.6'da verilmiştir. Diyagramda bulunan *rw_state* durum sinyali *RxFIFO*'dan belleğe yazma durumunu ve bellekten veri okuma durumunu göstermektedir. Diğer bir durum sinyali *rd_sorter_state* ise bellekten okunan verinin *TxFIFO*'ya aktarılma aşamasına ait durumu bildirir. Buna göre Şekil 4.6 üzerinde;

- Kalibrasyon ve initialization işlemlerinden sonraki işlemler detaylandırılmıştır.
- Belleğe yazma ve bellekten okuma işlemleri için 4'lü burst sayısı (*avl*_size) seçilmiştir.

- (1), (2), (3), (4) numaralı zamanlarda belleğin ‘000004h’ adresinden başlayıp ‘000007h’ numaralı adresine kadar toplamda 4’lü ardışık adrese burst modunda yazma işlemi gerçekleştirilmiştir.
- (5) numaralı zamanda bellekten ‘000000h’ adresinden başlayarak ardışık olarak ‘000003h’ adresine kadar 4’lü burst modunda okuma isteği gerçekleştirilmiştir.
- (6) numaralı zamanda, okuma isteğinin karşılığında 4’lü paket halinde bellekten veri okunmuştur.

Çizelge 4.4 Harici Bellek Arayüzü simülasyonuna ait sinyallerin listesi

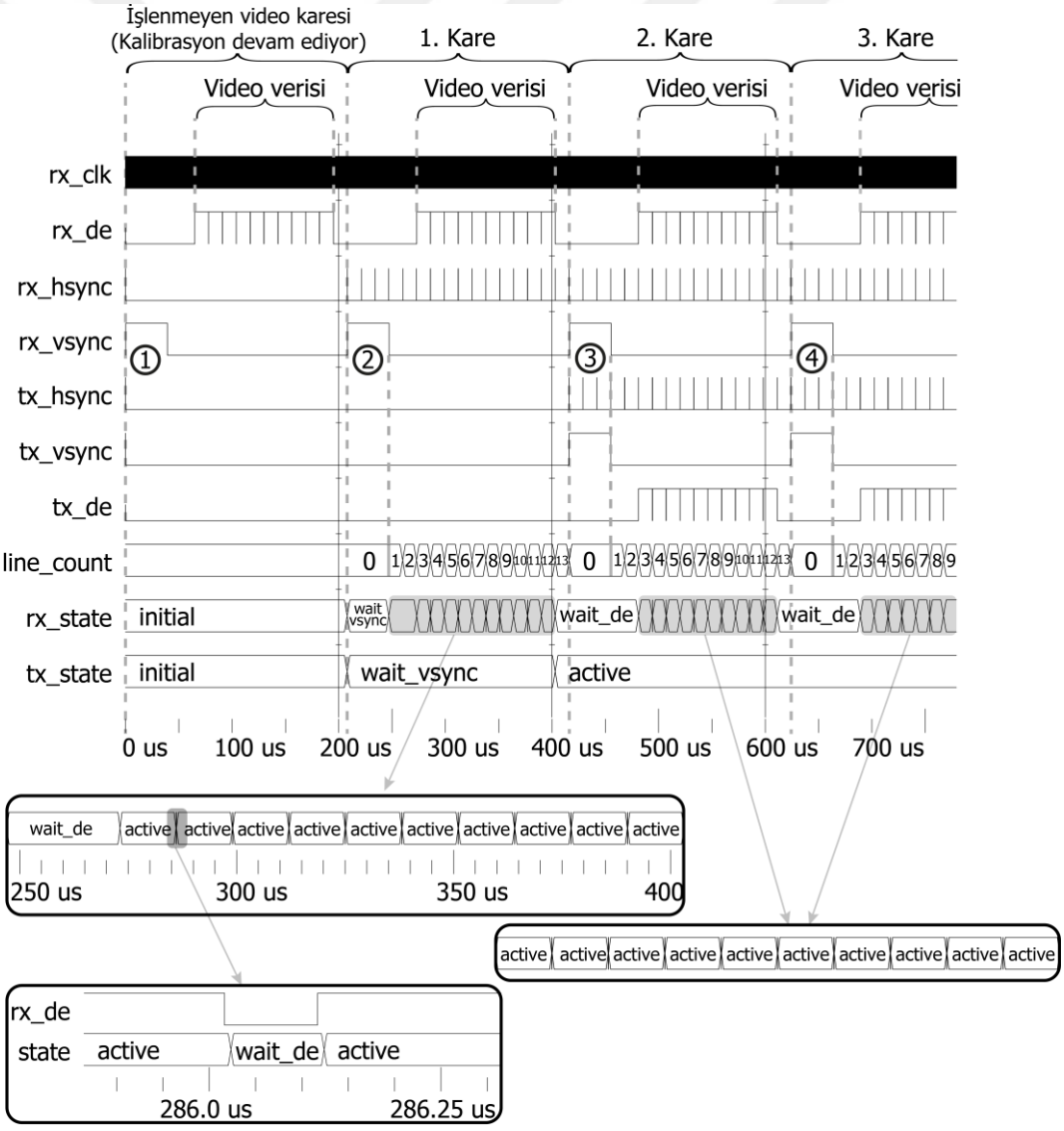
Sinyal İsmi	**Sinyal Türü**	**Sinyal Genişliği**		**Açıklama**
clk	Giriş	1-bit		533.33 MHz bellek saati
dvi_rx_gray	Çıkış	8-bit		Gri-tonlamaya dönüştürülmüş video sinyali
avl_addr	Giriş	24-bit		Bellek adresi
avl_wdata	Giriş	**DDR3**	**DDR**	Belleğe yazım için veri girişi
		256-bit	32-bit	
avl_rdata	Çıkış	**DDR3**	**DDR**	Bellekten okunan veri çıkışı
		256-bit	32-bit	
avl_rdata_valid	Çıkış	1-bit		Bellekten okunan verinin geçerliliğine ait bildirim
avl_read_req	Giriş	1-bit		Bellekten okuma talebi
avl_ready	Çıkış	1-bit		Bellek hazır sinyali
avl_size	Giriş	7-bit		Bellek burst büyüklüğü
avl_write_req	Giriş	1-bit		Belleğe yazma talebi
rw_state	Durum bildirimi	-		*RxFIFO*’dan belleğe yazma ve bellekten okuma işlemini yöneten durum makinesi
rd_sorter_state	Durum bildirimi	-		Bellekten okunan verilerin *TxFIFO*’ya yazılması işlemini yöneten durum makinesi

rw_state durum bilgisi takip edildiğinde, *RxFIFO*’dan okuma işlemi başlatılmadan önce bir ön okuma (*rx_preread*) bildirimi görülmektedir. Sonrasında *rx_read* bildirimi her aktif olduğunda, FIFO’dan okunan 8-bitlik piksel verilerinin 256-bitlik bellek veri genişliğini doldurabilmesi için *RxFIFO*’dan 32 adet okuma işlemi gerçekleştirilir. Belleğe yazma işlemi *mem_write* bildirimi aktifken gerçekleşir. Ayrıca *mem_write_check* bildirimi de son burst sayısına gelindiğini belirtir. Son olarak, *mem_buffer_read* durumunda bellekten okuma isteği işleme alınmış olur.

rd_sorter_state durum bilgisi takip edildiğinde, bellekten okuma isteğinin oluştuğu (5) numaralı andan sonra, okuma işleminin sonucunda elde edilecek olan paket verinin gelmesi beklenmektedir. Bu sırada *sorter_buffer_read* bildirimi aktiftir. (6) numaralı zamanda 4 adet 256-bitlik okuma verisi gözlemlendiği anda veriler 8-bitlik parçalara ayrılmak üzere serileyici modülüne gönderilir. Bu aşamada ise *rd_sorter_state* durum makinesinde *tx_send* bildirimi gözlemlenir.

4.2 Gerçek Zamanlı Video için Arabellek Uygulamasının Simülasyonu

Yüksek çözünürlüklü video sinyalleri için gerçek zamanlı arabellek oluşturma amacıyla gerçekleştirilen tasarım, temel çalışma prensibinin kanıtlanması amacıyla 1920×10 çözünürlüğünde saniyede 60 kareye sahip bir video giriş sinyali ile simüle edilmiştir.



Şekil 4.7 Gerçek zamanlı video için arabellek simülasyonunun video sinyalleri

Video girişinin ilk 4 karesi için simülasyon yapılmıştır. Arabellek boyutu 1 video karesi büyüklüğündedir. Video giriş sinyalinin parametreleri Çizelge 4.5'te verilmiştir.

Çizelge 4.5 Video giriş sinyaline ait parametreler

Çözünürlük	**Yatay çöz.**	**Dikey çöz.**	**Piksel saat frekansı**	**VSYNC (satır)**	**VFP (satır)**	**VBP (satır)**	**HSYNC (piksel)**	**HFP (piksel)**	**HBP (piksel)**	**Satırda topl. piksel**	**Topl. satır**
1920x10	1920	10	148.5MHz	3	1	2	8	3	4	1935	16

Simülasyondaki video sinyallerine ait zaman diyagramı Şekil 4.7'de verilmiştir. Buna göre;

- Her yeni video karesinin başlangıcı *vsync* sinyalinin yükselen kenarı ile senkronizedir.

- Video girişinin ilk karesi (1) numaralı zamanda gelmektedir. Bu sırada bellek kalibrasyon işlemi tamamlanmadığı için bu video karesi işleme alınmamaktadır. Sonraki video karesi gelene kadar geçen sürede, *rx_state* ve *tx_state* durum makineleri başlangıç (*initial*) durumundadır.

- Kalibrasyon işlemi bittikten sonraki ilk video karesinin başlangıcıyla beraber (2) numaralı zamanda aktif olan *rx_vsync* sinyali inaktif olduktan sonra ilk video karesi belleğe alınmaya başlar. Bu işlem *rx_state* durum makinesinin *active* durumuna geçmesi ile gerçekleşir. Bu sırada *Tx Video* modülünde çıkış gözlemlenmez. Ayrıca *tx_state* durum makinesi vsync bekleme (*wait_vsync*) durumundadır.

- İşlenen '2.' video karesinin başlangıcı olan (3) numaralı zamandan itibaren, '2.' video karesi belleğe alınırken *Tx Video* modülünün çıkışında bir önceki video karesi olan '1.' kare gözlemlenmektedir. Bu sırada *tx_state* durum makinesi *active* durumuna geçmiştir.

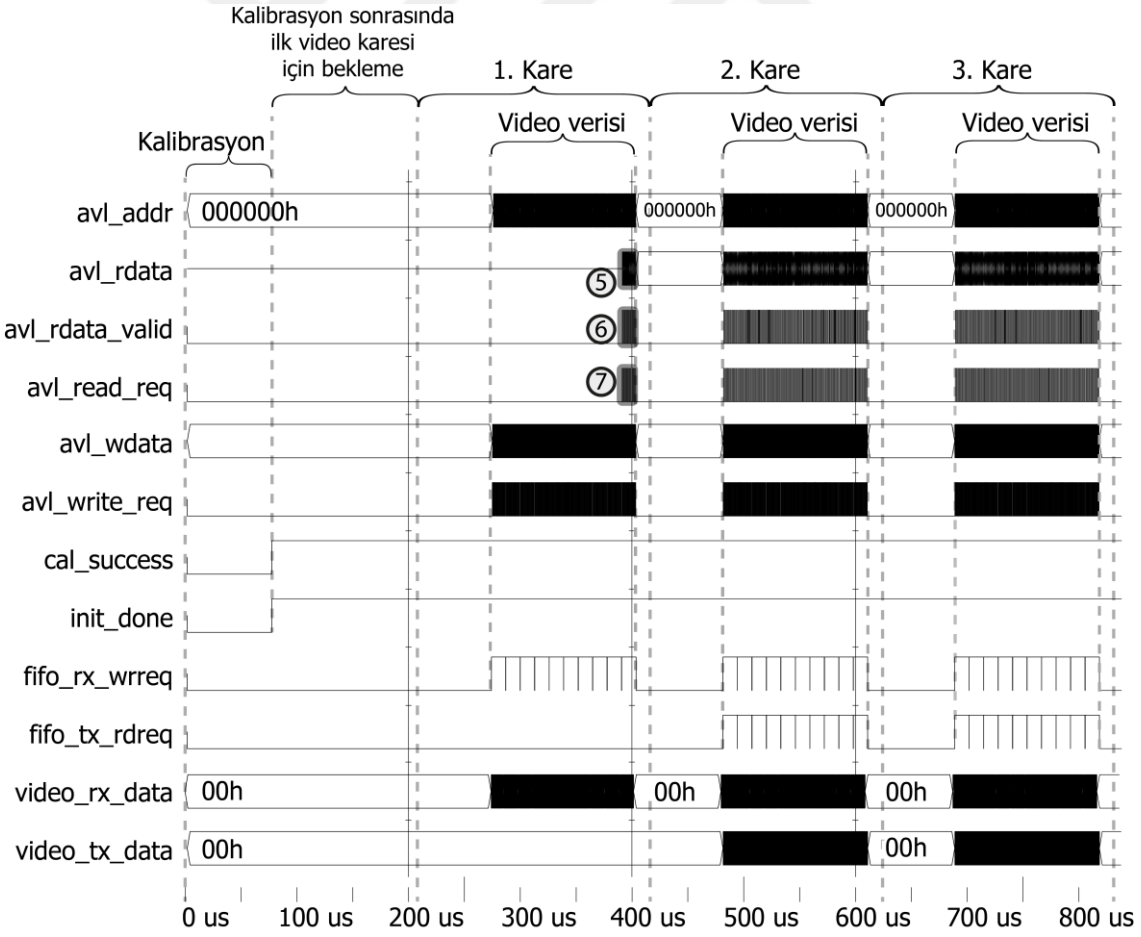
- (4) numaralı zamanda da '3.' video karesi belleğe alınmaya başlarken, '2.' video karesi çıkışa gönderilmektedir. Sonuç olarak, çıkışta görüntülenen video sinyali, video giriş sinyaline göre arabelleğe alınan video karesi sayısı kadar gecikmeli aktarılmış olmaktadır.

Video giriş ve çıkış modüllerini kontrol eden *rx_state* ve *tx_state* durum makinelerinin durum bilgisini gösteren tablo Çizelge 4.6'de verilmiştir. *rx_state* durum makinesi 4 farklı duruma, *tx_state* durum makinesi 3 farklı duruma sahiptir.

Çizelge 4.6 rx_state ve tx_state durum makinelerine ait durumların listesi

Durum Makinesi	**Durum**	**Açıklama**
rx_state	initial	Bellek kalibrasyonu bitene kadar geçerli durum
	wait_vsync	Kalibrasyondan sonraki ilk video karesinin beklenmesi
	wait_de	Video karesinin veri içeren kısımlarının beklenmesi
	active	Görüntü yakalamanın başladığı durum
tx_state	initial	Bellek kalibrasyonu bitene kadar geçerli durum
	wait_vsync	Arabelleğe alınacak video kare sayısı kadar bekleme
	active	Görüntü aktarımının başladığı durum

Arabellek simülasyonu sırasında bellek ve FIFO modüllerinin zamana göre değişimleri Şekil 4.8'te verilmiştir. Buna göre; (5), (6) ve (7) numaralı zamanlarda bellekten ön okuma işlemi gerçekleştirilmiştir. Bu sırada bellekten okunup *TxFIFO*'da biriktirilen önceki video karesine ait veri, çıkışa 1 video karesi gecikme ile aktarılmaktadır. Geciktirilmiş video verisi *video_tx_state* sinyali üzerinde gözlemlenmektedir. Gecikme için seçilen kare sayısı önceden ayarlanabilmektedir.



Şekil 4.8 Gerçek zamanlı video için arabellek simülasyonunun bellek ve FIFO sinyalleri

Simülasyon aşamasında gözlemlenen önemli sinyallerin yer aldığı tablo Çizelge 4.7'de verilmiştir.

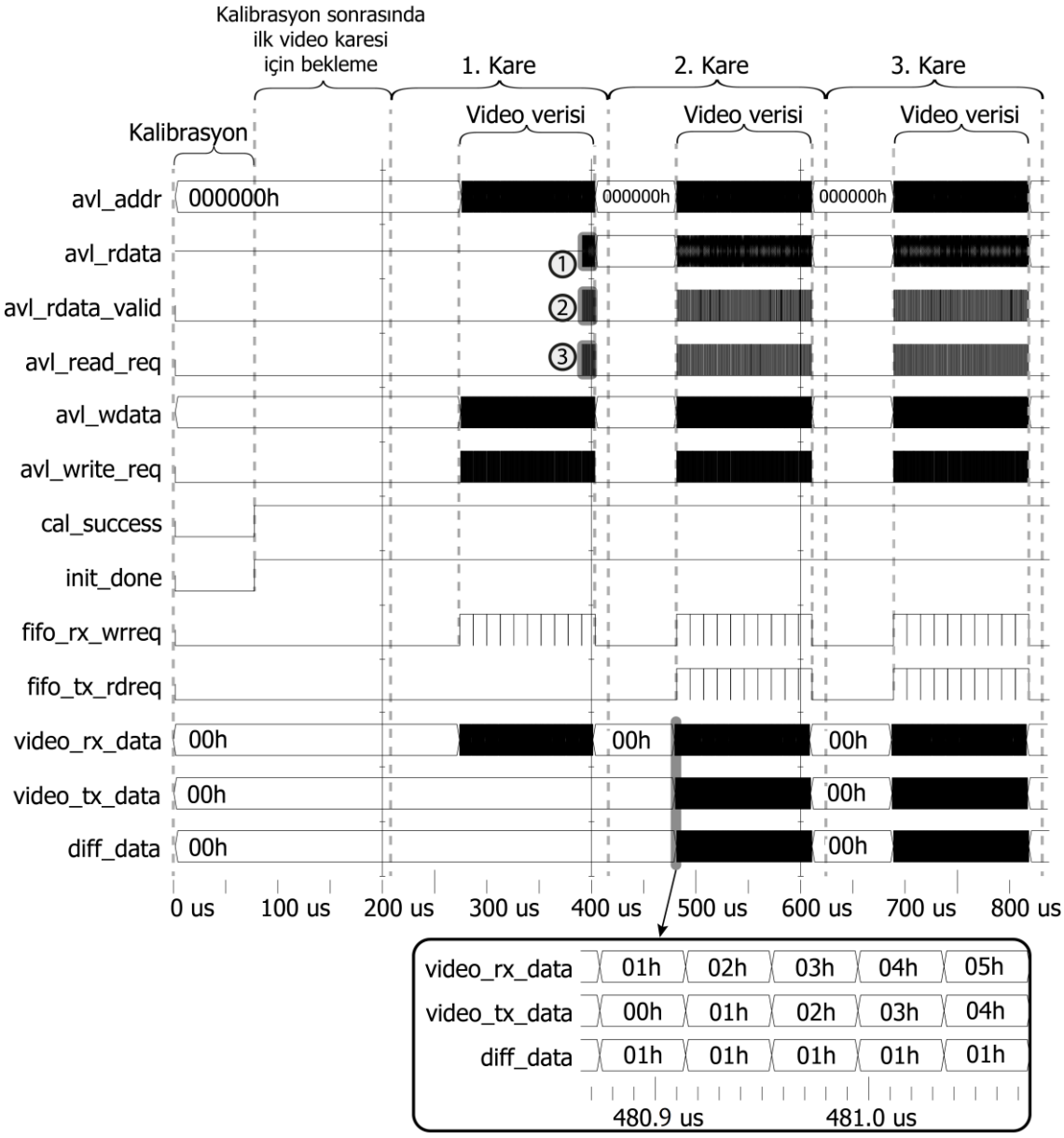
Çizelge 4.7 Gerçek zamanlı video için arabellek simülasyonuna ait sinyallerin listesi

Sinyal İsmi	**Sinyal Türü**	**Sinyal Genişliği**		**Açıklama**
rx_clk	Giriş	1-bit		148.5 MHz piksel saati
rx_hsync	Giriş	1-bit		Rx yatay senkronizasyon sinyali
rx_vsync	Giriş	1-bit		Rx dikey senkronizasyon sinyali
rx_de	Giriş	1-bit		Rx veri aktivitesi sinyali
tx_hsync	Çıkış	1-bit		Tx yatay senkronizasyon sinyali
tx_vsync	Çıkış	1-bit		Tx dikey senkronizasyon sinyali
tx_de	Çıkış	1-bit		Tx veri aktivitesi sinyali
line_count	Çıkış	12-bit		Görüntü satır sayısı sinyali
rx_state	Durum bildirimi	-		Video giriş sinyalinin yakalanması işlemini yöneten durum makinesi
tx_state	Durum bildirimi	-		İşlenen video sinyalinin çıkışa aktarılması işlemini yöneten durum makinesi
avl_addr	Giriş	24-bit		Bellek adresi
avl_rdata	Çıkış	**DDR3**	**DDR**	Bellekten okunan veri çıkışı
		256-bit	32-bit	
avl_rdata_valid	Çıkış	1-bit		Bellekten okunan verinin geçerliliğine ait bildirim
avl_read_req	Giriş	1-bit		Bellekten okuma talebi
avl_wdata	Giriş	**DDR3**	**DDR**	Belleğe yazım için veri girişi
		256-bit	32-bit	
avl_write_req	Giriş	1-bit		Belleğe yazma talebi
cal_success	Çıkış	1-bit		Kalibrasyon başarılı sinyali
init_done	Çıkış	1-bit		Initialization başarılı sinyali
fifo_rx_wrreq	Giriş	1-bit		*RxFIFO*'ya yazma talebi
fifo_tx_rdreq	Giriş	1-bit		*TxFIFO*'dan okuma talebi
video_rx_data	Giriş	8-bit		Video giriş sinyali
video_tx_data	Çıkış	8-bit		Video çıkış sinyali

4.3 Gerçek Zamanlı Video Çerçeveleri için Fark Alıcı Uygulamasının Simülasyonu

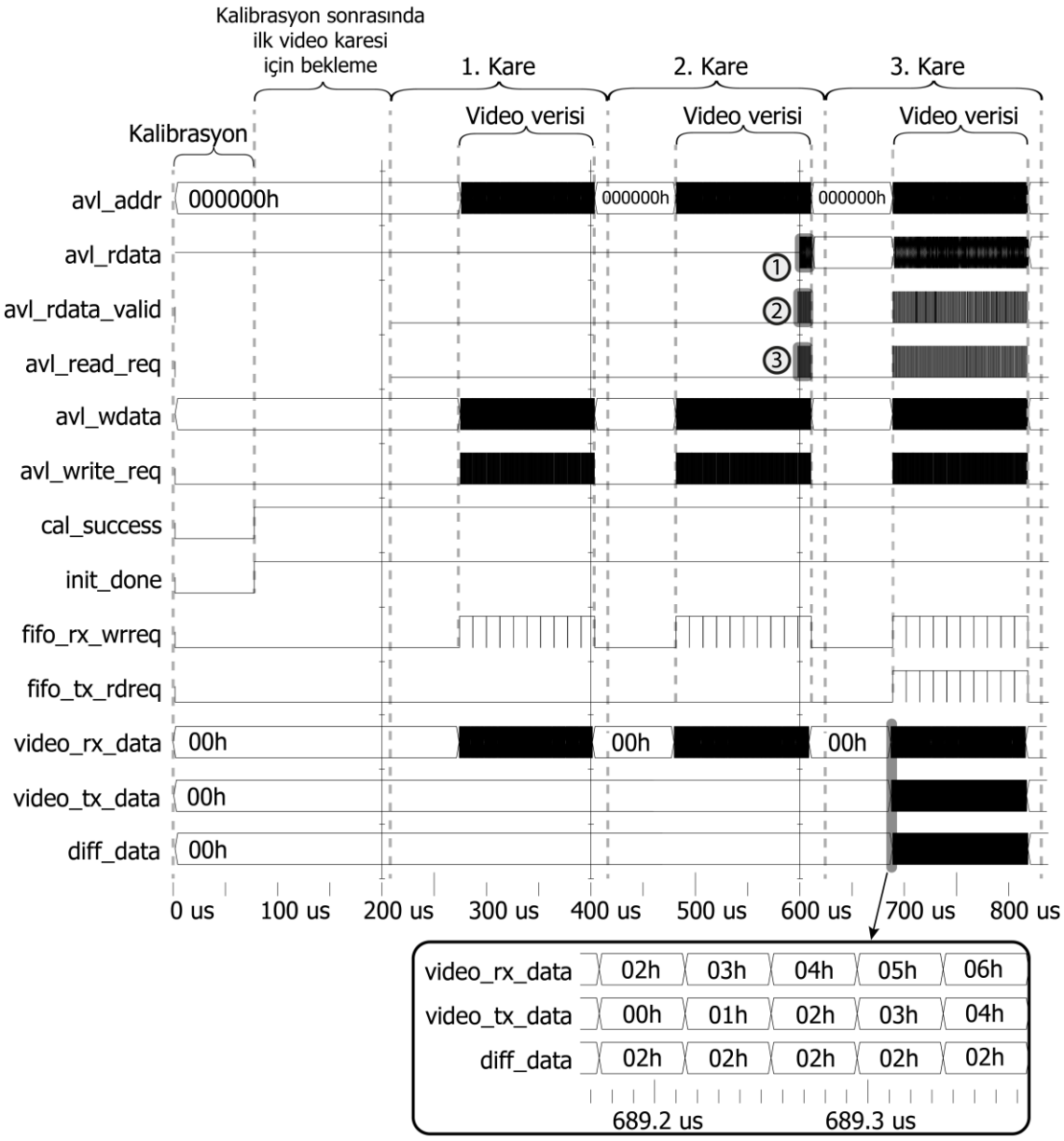
Yüksek çözünürlüklü video sinyalleri için gerçek zamanlı video karesi farkı alma amacıyla gerçekleştirilen tasarım, önceki başlıklarda yer alan simülasyonlarda olduğu gibi, temel çalışma prensibinin kanıtlanması amacıyla 1920×10 çözünürlüğünde saniyede 60 kareye sahip bir video giriş sinyali ile simüle edilmiştir. Simülasyon aşamasında video kareleri için farklı *k* büyüklükleri test edilmiştir.

Farkı alınan video kareleri arasındaki kare sayısı *k*=1 için, simülasyona ait video sinyallerinin zamana göre değişimi Şekil 4.7'deki gibi olmaktadır. Zamanlamalarla ilgili açıklamalar, 4.2 numaralı başlıklı konunun içerisinde verilmiştir.



Şekil 4.9 *k*=1 için, gerçek zamanlı fark alıcı simülasyonunun video sinyalleri

Fark alma işleminin simülasyonuna ait bellek ve FIFO sinyalleri Şekil 4.9'da gösterilmiştir. (1), (2) ve (3) numaralı zamanlarda ön okuma işlemi başladıktan sonra, 1 kare gecikme ile *RxFIFO* modülünün çıkışından *TxFIFO* modülünün çıkışı çıkarılır. Kare gecikmesi *k*=1 olduğu için, *video_tx_data*(n) sinyali *video_rx_data*(n-1) sinyalini temsil eder. Sonuç olarak, *video_rx_data*(n) – *video_rx_data*(n-1) işleminin sonucu *diff_data* sinyali üzerinden gözlenlenmiş olur.



Şekil 4.10 *k*=2 için, gerçek zamanlı fark alıcı simülasyonunun video sinyalleri

Kare gecikmesinin *k*=2 olduğu durum için, simülasyon işlemine ait zaman diyagramı Şekil 4.10'da verilmiştir. Bu durumda gecikme 2 video karesi kadar olduğu için, video giriş sinyali işleme alındıktan 2 kare sonra, (1), (2) ve (3) numaralı zamanlarda ön okuma işleminin başlamasından sonra çıkış sinyali üretilmeye başlar. Buna göre, *video_tx_data*(n) sinyali *video_rx_data*(n-2) sinyalini temsil eder. Sonuç olarak, *video_rx_data*(n) – *video_rx_data*(n-2) işleminin sonucu *diff_data* sinyali üzerinden gözlenlenmiş olur. Kullanılan video giriş sinyaline ait parametreler Çizelge 4.5'te verilmiştir. Simülasyon aşamasında gözlemlenen diğer sinyallerin açıklamaları ise Çizelge 4.7'de verilmiştir.

BÖLÜM 5

ÖRNEK UYGULAMALARIN FPGA ÜZERİNDE GERÇEKLENMESİ

Önceki bölümlerde bahsi geçen "Gerçek Zamanlı Video için Arabellek Uygulaması" ve "Gerçek Zamanlı Video Çerçeveleri için Fark Alıcı Uygulaması" ile ilgili FPGA gerçeklemeleri bu bölümde gösterilmiştir. Gerçekleme sırasında Altera Stratix IV GX Development Kit ve Altera Cyclone III Starter Kit kartları kullanılarak, uygulamaların üst sınıf ve alt sınıf FPGA geliştirme kartları üzerinde gerçeklenebilirliği test edilip kanıtlanmıştır.

5.1 Gerçek Zamanlı Video için Arabellek Uygulamasının Gerçeklenmesi

Gerçek zamanlı arabellek uygulamasının gerçeklemesi Cyclone III Starter ve Stratix IV GX geliştirme kartları üzerinde yapılmıştır. Farklı burst sayıları ve video kare gecikmeleri test edilmiştir.

Şekil 5.1 1024×768 piksel 60 Hz video girişi için Cyclone III Starter kartı üzerinde 15 video karesi büyüklüğünde arabellek gerçeklemesi

Bir film sonu jeneriği üzerinde test edilen 15 video karesi büyüklüğündeki arabellek gerçeklemesinin görüntüsü Şekil 5.1'te verilmiştir. Aşağıdan yukarıya doğru kayan yazıların, görüntünün sağ tarafında sol tarafına göre gecikmeli olarak çıkışa aktarıldığı görülmektedir.

Şekil 5.2 Full-HD 1080p 60 Hz video girişi için Stratix IV GX kart üzerinde 60 kare büyüklüğünde arabellek gerçeklemesi

Şekil 5.2'te ise arabellek boyutu 60 kareye çıkartılmıştır. Bu nedenle görüntünün sağ tarafında yer alan çıkış, Şekil 5.1'deki çıkış sinyaline göre daha fazla gecikme ile gözlemlenmektedir.

Çizelge 5.1 Gerçeklenebilir burst büyüklükleri, arayüz genişliği ve FPGA kartına göre kullanılan kaynaklar

FPGA kartı	**MaksimumVideo Çözünürlüğü**	**Burst**	**Arayüz Genişliği**	**Blok Bellek Elemanı**	**Toplam Pin**	**Toplam Yazmaç**
Cyclone III Starter	1024×768 60 Hz	8	16-bit	43872	114	3855
	1024×768 60 Hz	16	16-bit	44384	114	3698
	1024×768 60 Hz	32	16-bit	44896	114	3697
Stratix IV GX	1920×1080 60 Hz	4	16-bit	360666	124	7544
	1920×1080 60 Hz	8	16-bit	360660	124	8047
	1920×1080 60 Hz	16	16-bit	360654	124	8962
	1920×1080 60 Hz	2	64-bit	297562	190	8381
	1920×1080 60 Hz	4	64-bit	297556	190	8940
	1920×1080 60 Hz	8	64-bit	297550	190	9956
	1920×1080 60 Hz	16	64-bit	297544	190	12014
	1920×1080 60 Hz	32	64-bit	297538	190	16137

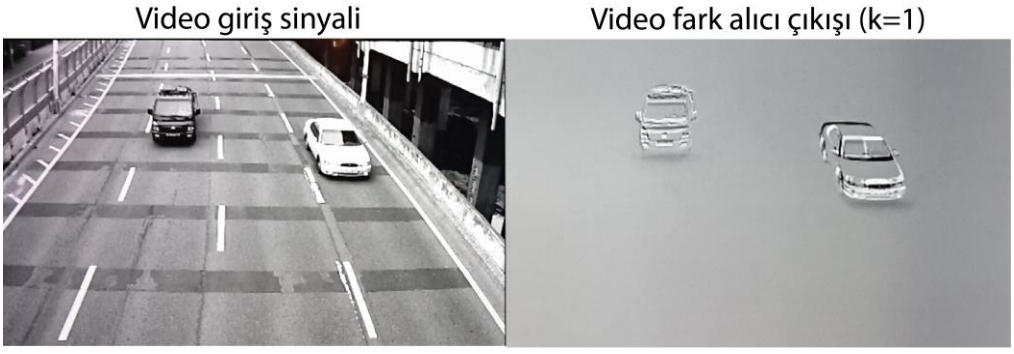
Çizelge 5.2 Bellek türü, bellek frekansı ve burst sayısına göre desteklenen video formatları ve piksel frekansları

Bellek Türü	Veri Genişliği	Minimum Burst	Test Edilen Bellek Frekansı	Maksimum Video Çözünürlüğü	Piksel Frekansı
DDR SDRAM	16-bit	8	133 MHz	1024×768 60 Hz	65 MHz
DDR3 SDRAM	16-bit	8	400 MHz	1920×1080 60 Hz	148.5 MHz
DDR3 SDRAM	16-bit	4	533 MHz	1920×1080 60 Hz	148.5 MHz
DDR3 SDRAM	16-bit	2	667 MHz	1920×1080 60 Hz	148.5 MHz
DDR3 SDRAM	64-bit	4	400 MHz	1920×1080 60 Hz	148.5 MHz
DDR3 SDRAM	64-bit	2	533 MHz	1920×1080 60 Hz	148.5 MHz
DDR3 SDRAM	64-bit	1	667 MHz	1920×1080 60 Hz	148.5 MHz

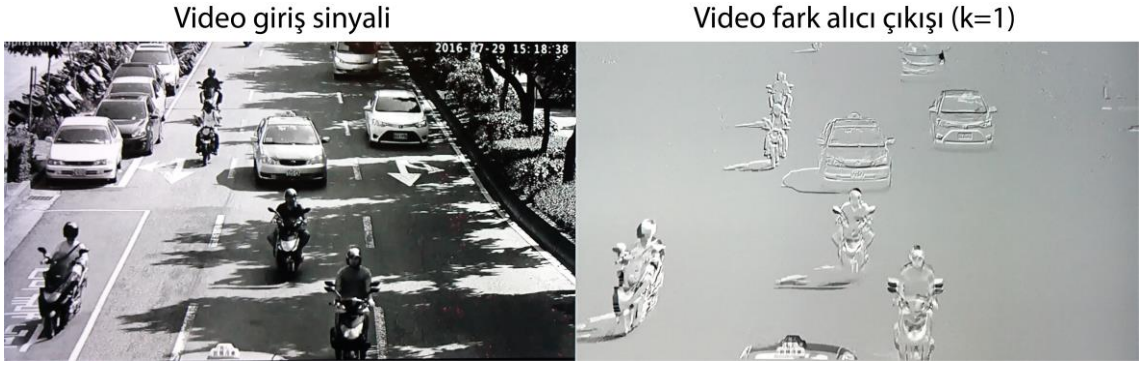
Test edilen FPGA kartlarına ve video formatlarına göre kullanılan kaynaklara ilişkin veriler Çizelge 5.1'de verilmiştir. Bellek türü, bellek veri genişliği ve bellek frekanslarına göre elde edilen veriler ise Çizelge 5.2'de gösterilmiştir.

5.2 Gerçek Zamanlı Video Çerçeveleri için Fark Alıcı Uygulamasının Gerçeklenmesi

Gerçek zamanlı video çerçeveleri için fark alıcı uygulamasının gerçeklemesi Cyclone III Starter ve Stratix IV GX geliştirme kartları üzerinde yapılmıştır. Farklı burst sayıları ve video kare gecikmeleri test edilmiştir. Test videosu olarak 1024×768 ve 1920×1080 piksel çözünürlüğündeki saniyede 60 kareye sahip trafik kameraları görüntüleri kullanılmıştır. Görüntü üzerinde hareket eden objeler bulunduğu zaman, video çerçevelerinin fark alma işleminin sonucu olarak sadece hareket eden objelerin gözlemlenebildiği, kamera konumu sabit olduğu için zamanla değişmeyen arkaplan görüntüsünün süzüldüğü kanıtlanmıştır.



Şekil 5.3 1024×768 piksel 60 Hz video girişi için Cyclone III Starter kartı üzerinde k=1 için video çerçeve fark alıcısı gerçeklemesi



Şekil 5.4 Full-HD 1080p 60 Hz video girişi için Stratix IV GX kartı üzerinde k=1 için video çerçeve fark alıcısı gerçeklemesi

Test için kullanılan 1024×768 piksel çözünürlüğünde ve saniyede 60 kareye sahip video için fark alıcı uygulaması Cyclone III Starter kartı üzerinde gerçeklenmiş ve sonuç görüntüsü Şekil 5.3'te verilmiştir. Sonrasında aynı uygulama 1920×1080 piksel çözünürlüğünde ve saniyede 60 kareye sahip video kullanılarak Stratix IV GX kartı üzerinde gerçeklenmiş ve sonuç görüntüsü Şekil 5.4'te verilmiştir. Video girişinde hareketsiz olan nesneler, sonuç görüntüsünde gri olarak gözlemlenmektedir.

Bu uygulama aşamasında test edilen FPGA kartları ve görüntü formatlarına ilişkin veriler Çizelge 5.3'te verilmiştir. Bellek türü, bellek veri genişliği ve bellek frekanslarına göre elde edilen veriler için

Çizelge 5.3 Gerçekleme sırasında kullanılan FPGA kartları ve video çözünürlüklerine ilişkin bilgiler

FPGA kartı	**MaksimumVideo Çözünürlüğü**	**Burst**	**Arayüz Genişliği**	**Blok Bellek Elemanı**	**Toplam Pin**	**Toplam Yazmaç**
Cyclone III Starter	1024×768 60Hz	8	16-bit	43872	114	3890
	1024×768 60Hz	16	16-bit	44384	114	3733
	1024×768 60Hz	32	16-bit	44896	114	3732
Stratix IV GX	1920×1080 60Hz	4	16-bit	360666	124	7579
	1920×1080 60Hz	8	16-bit	360660	124	8082
	1920×1080 60Hz	16	16-bit	360654	124	8997
	1920×1080 60Hz	2	64-bit	297562	190	8416
	1920×1080 60Hz	4	64-bit	297556	190	8975
	1920×1080 60Hz	8	64-bit	297550	190	9991
	1920×1080 60Hz	16	64-bit	297544	190	12049
	1920×1080 60Hz	32	64-bit	297538	190	16172

BÖLÜM 6

SONUÇ VE ÖNERİLER

Tez çalışması kapsamında, genel amaçlı bir bellek arayüzü tasarımı açıklanmıştır ve bu tasarımın simülasyonuna ek olarak FPGA geliştirme kartları üzerinde çeşitli örnek uygulamalar baz alınarak gerçeklemesi yapılmıştır. Simülasyon ve gerçekleme aşamasında, bellek arayüzünün kullanım alanları arasında yer alan gerçek zamanlı video işleme uygulamalarına öncelik verilmiştir. Bu uygulamalarda, 1920×1080 piksel çözünürlüğüne ve saniyede 60 kare yenileme hızına kadar video giriş sinyali işleme desteği sağlanmıştır. Bellek arayüzünün farklı bellek elemanı türleri üzerinde çalışabilirliği DDR SDRAM ve DDR3 SDRAM bellek tipleri ile test edilmiştir. Bellek arayüzü frekansı incelendiğinde, 1024×768 piksel çözünürlüklü ve saniyede 60 kare video giriş sinyali kullanıldığında DDR SDRAM için 133 MHz'e kadar, 1920×1080 piksel çözünürlüklü ve saniyede 60 kare video giriş sinyali kullanıldığında DDR3 SDRAM için 667 MHz'e kadar ulaşılmıştır. Bu maksimum bellek arayüzü frekanslarında 16-bitlik DDR SDRAM bellek türü için 1.98 Gbit/s bant genişliğine, 64-bitlik DDR3 SDRAM bellek türü için 39.76 Gbit/s bant genişliğine ulaşılmıştır.

Günümüzde Full-HD 1920×1080 piksel çözünürlüğü ve saniyede 60 kare yenileme hızı yaygın kullanılmasına karşın, görüntüleme aygıtları teknolojisinin gelişmesiyle birlikte 4K Ultra-HD 3840×2160 piksel çözünürlüğü ve 8K Ultra-HD 7680×4320 piksel çözünürlüğü de yaygınlaşmaya başlayacaktır. Bu tez çalışması kapsamında incelenen gerçek zamanlı video işleme uygulamalarının, tasarlanan bellek arayüzü ile 4K ve 8K çözünürlük desteği ile yakın gelecekte gerçeklenmesi ön görülmektedir.

Ek olarak, bellek arayüzünün öncelikli kullanım alanları arasına görüntü üzerinde uzam-zamansal filtreleme, konvolüsyonel sinir ağları ve zaman-türevli hücresel sinir ağı uygulamaları dâhil edilerek, bu uygulamalar için gerekli olan arabellek ihtiyacının bu tez kapsamındaki tasarım ile giderilmesi planlanmaktadır.

KAYNAKLAR

[1] Yıldız, N., Cesur, E., Kayaer, K., Tavşanoğlu, V., ve Alpay, M., (2015), "Architecture of a Fully Pipelined Real-Time Cellular Neural Network Emulator", IEEE Transactions on Circuits and Systems I: Regular Papers, 62 (1): 130 - 138.

[2] Yıldız, N., Cesur, E., ve Tavşanoğlu, V., (2014), "Design of a Third Generation Real-Time Cellular Neural Network Emulator", 14th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA), 29-31 Temmuz 2014, Notre Dame.

[3] Yıldız, N., Cesur, E., ve Tavşanoğlu, V., (2016), "On the way to a Third Generation Real-Time Cellular Neural Networks Processor", CNNA 2016; 15th International Workshop on Cellular Nanoscale Networks and their Applications, 23-25 Ağustos 2016, Dresden.

[4] Polat, S. N. T., ve Tavşanoğlu, V., (2009), "A New Simulation Method for Time-Derivative Cellular Neural Networks", 17th European Signal Processing Conference, 24-28 Ağustos 2009, Glasgow.

[5] Polat, S. N. T., Yavuz, O., ve Tavşanoğlu, V., (2012), "Efficient Simulation of Time-Derivative Cellular Neural Networks", IEEE Transactions on Circuits and Systems I: Regular Papers, 59 (11): 2638 - 2645.

[6] Yavuz, O., Polat, S. N. T., ve Tavşanoğlu, V., (2010), "On the Simulation of Time Derivative Cellular Neural Networks", 18th European Signal Processing Conference, 23-27 Ağustos 2010, Aalborg.

[7] Yıldız, N., Cesur, E., ve Tavşanoğlu, V., (2016), "A Discussion on Spatiotemporal Filtering on a Third Generation Real-Time Cellular Neural Network Processor", CNNA 2016; 15th International Workshop on Cellular Nanoscale Networks and their Applications, 23-25 Ağustos 2016, Dresden.

[8] Rao, M. V. G., Kumar, P. R., ve Prasad, A. M., (2016), "Implementation of Real Time Image Processing System with FPGA and DSP", International Conference on Microelectronics, Computing and Communications (MicroCom), 23-25 Ocak 2016, Durgapur.

[9] Wang, Q., ve Gao, Z., (2008), "Study on a Real-Time Image Object Tracking System", International Symposium on Computer Science and Computational Technology, 20-22 Aralık 2008, Şangay.

[10] Xie, D., ve Wang, Y., (2017), “High Definition Wide Dynamic Video Surveillance System Based on FPGA0”, IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 25-26 Mart 2017, Chongqing.

[11] Wang, B., Du, J., Bi, X., ve Tian, X., (2015), “High Bandwidth Memory Interface Design Based on DDR3 SDRAM and FPGA”, International SoC Design Conference (ISOCC), 2-5 Kasım 2015, Gyeongju.

[12] Altera Corporation, SCFIFO and DCFIFO IP Cores User Guide, https://www.altera.com/en_US/pdfs/literature/ug/ug_fifo.pdf, 14 Ocak 2018.

[13] Allan, G., ve Defazio, J., (2007), "IP solves the increasing challenge of implementing an interface to off-chip DDR SDRAM", International Engineering Consortium DesignCon 2007, 29 Ocak - 1 Şubat 2007, Santa Clara, Kaliforniya.

[14] Altera Corporation, External Mermoy Interface Handbook Volume 3: Reference Material, https://www.altera.com/en_US/pdfs/literature/hb/external-memory/emi_ip.pdf, 20 Ocak 2018.

[15] Video Electronics Standards Association (V. E. S. A.), VESA and Industry Standards and Guidelines for Computer Display Monitor Timing (DMT), http://caxapa.ru/thumbs/361638/DMTv1r11.pdf, 2 Şubat 2018.

[16] Video Electronics Standards Association (V. E. S. A.), Proposed Monitor Timing Standard, https://wenku.baidu.com/view/b378c2d026fff705cc170a5e.html, 2 Şubat 2018.

[17] Altera Corporation, Stratix IV GX FPGA Development Board Reference Manual, https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/rm_sivgx_fpga_dev_board.pdf, 20 Ocak 2018.

[18] Altera Corporation, Cyclone III FPGA Starter Board Reference Manual, https://www.altera.com/en_US/pdfs/literature/manual/rm_ciii_starter_board.pdf, 20 Ocak 2018.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Doğancan DAVUTOĞLU

Doğum Tarihi ve Yeri : 07.07.1991

Yabancı Dili : İngilizce

E-posta : dogancandavutoglu@gmail.com

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Y. Lisans	Elektronik ve Haberleşme Mühendisliği, Elektronik Programı	Yıldız Teknik Üniversitesi	2018
Lisans	Elektronik Mühendisliği	İstanbul Kültür Üniversitesi	2015
Lise	Fen-Matematik	Fahreddin Kerim Gökay Anadolu Lisesi	2009

İŞ TECRÜBESİ

Yıl	**Firma/Kurum**	**Görevi**
2015 -	İstanbul Kültür Üniversitesi	Araştırma Görevlisi

YAYINLARI

Uluslararası Bildiri

[1] Davutoğlu, D.,Yıldız, N., Ayten, U. E., ve Tavşanoğlu, V., (2018), "Real-time Frame Buffer Implementation based on External Memory using FPGA", International Congress of Information and Communication Technology (ICICT 2018), 27-28 Ocak 2018, Xiamen.