ÇOK ÇEKİRDEKLİ MİMARİ ÜZERİNDE GERÇEK ZAMANLI GÖREV ZAMANLAMA VE İLETİŞİM YÖNETİM ALGORİTMALARININ BİRLİKTE TASARIMI

DESIGN OF REAL-TIME SCHEDULING AND COMMUNICATION MANAGEMENT ALGORITHMS ON MULTICORE ARCHITECTURE

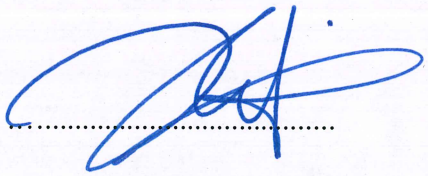
HÜSEYİN TEMUÇİN

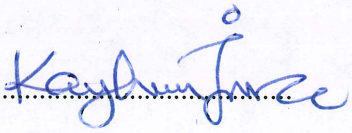
DOÇ. DR. KAYHAN M. İMRE
Tez Danışmanı

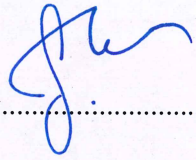
Hacettepe Üniversitesi
Lisansüstü Eğitim – Öğretim ve Sınav Yönetmeliğinin
Bilgisayar Mühendisliği Anabilim Dalı İçin Öngördüğü
DOKTORA TEZİ
olarak hazırlanmıştır.

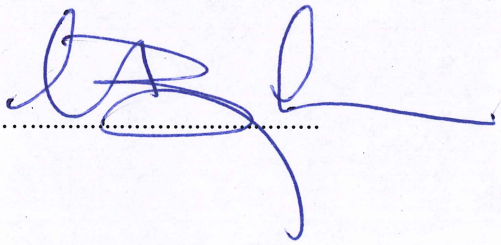
2018

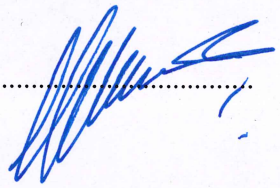
HÜSEYİN TEMUÇİN'in hazırladığı "**Çok çekirdekli mimari üzerinde gerçek zamanlı görev zamanlama ve iletişim algoritmalarının birlikte tasarımı**" adlı bu çalışma jürimiz tarafından **BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**'nda **DOKTORA TEZİ** olarak kabul edilmiştir.

Prof. Dr. Halit Oğuztüzün
Başkan ..... 

Doç. Dr. Kayhan M. İmre
Danışman ..... 

Doç. Dr. Süleyman Tosun
Üye ..... 

Doç. Dr. Cüneyt Bazlamaçcı
Üye ..... 

Yrd. Doç. Dr. Mustafa Ege
Üye ..... 

Bu tez Hacettepe Üniversitesi Fen Bilimleri Enstitüsü tarafından **DOKTORA TEZİ** olarak onaylanmıştır.

Prof. Dr. Menemşe GÜMÜŞDERELİOĞLU
Enstitü Müdürü

YAYINLAMA VE FİKRİ MÜLKİYET HAKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezimin/raporumun tamamını veya herhangi bir kısmını, basılı (kağıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma iznini Hacettepe üniversitesine verdiğimi bildiririm. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları bana ait olacaktır.

Tezin kendi orijinal çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

☐ **Tezimin/Raporumun tamamı dünya çapında erişime açılabilir ve bir kısmı veya tamamının fotokopisi alınabilir.**
(Bu seçenekle teziniz arama motorlarında indekslenebilecek, daha sonra tezinizin erişim statüsünün değiştirilmesini talep etseniz ve kütüphane bu talebinizi yerine getirse bile, tezinin arama motorlarının önbelleklerinde kalmaya devam edebilecektir.)

☒ **Tezimin/Raporumun** 15/03/19 **tarihine kadar erişime açılmasını ve fotokopi alınmasını (İç Kapak, Özet, İçindekiler ve Kaynakça hariç) istemiyorum.**
(Bu sürenin sonunda uzatma için başvuruda bulunmadığım taktirde, tezimin/raporumun tamamı her yerden erişime açılabilir, kaynak gösterilmek şartıyla bir kısmı ve ya tamamının fotokopisi alınabilir)

☐ **Tezimin/Raporumun ..... tarihine kadar erişime açılmasını istemiyorum, ancak kaynak gösterilmek şartıyla bir kısmı veya tamamının fotokopisinin alınmasını onaylıyorum.**

☐ **Serbest Seçenek/Yazarın Seçimi**

15 / 03 / 2018

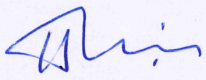
Hüseyin Temuçin

ETİK

Hacettepe Üniversitesi Fenbilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversitede veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.



25.02.2018

HÜSEYİN TEMUÇİN

ÖZET

Çok çekirdekli mimari üzerinde gerçek zamanlı görev zamanlama ve iletişim yönetim algoritmalarının birlikte tasarımı

Hüseyin Temuçin

Doktora Bilgisayar Mühendisliği

Tez Danışmanı: Doç. Dr. Kayhan M. İmre

Şubat 2018

İşlemci mimarilerinin fiziksel sınırlara yaklaşması, ticari ve akademik süreçlerde kullanılan tüm bilgisayar sistemlerinde koşut sistemleri zorunlu hale getirmiştir. Koşut sistemler, bir amaca hizmet etmek üzere çok sayıda işlemcinin bir topoloji kapsamında bir araya getirildiği bilgisayar sistemleridir. Söz konusu amaç belirlenen bir problemin çözülmesi veya bir sistemin hizmet sağladığı görevlerin dağıtılması olabilir. Bu sistemler üzerindeki işlemciler dağıtılmış veya paylaşımlı kaynakları kullanabilirler ve dağıtılmış bellekli sistemlerde işlemciler konumlandırılmış olan bir sistem alan ağı üzerinden aralarında veri paylaşımı sağlarlar.

Genel yaklaşımda, bilgisayar sistemlerinin en temel başarım ölçütleri sırasıyla doğruluk ve etkinlik olarak tanımlanabilir. Genelleştirilmiş bir sistem görevinde doğruluk sadece çıktıların doğruluğu ile ölçülmektedir ve sistemlerin anlık yoğunluğuna göre görevlerin tamamlanmasının gecikebilmesi kabul edilebilir bir durumdur. Gerçek zamanlı sistemler, zamanlılık esasıyla çalışan özelleşmiş bilgisayar sistemleridir ve bu sistemler üzerindeki tüm görevlerin, belirlenmiş zaman sınırı öncesi tamamlanması beklenmektedir. Bu sistemlerde gerçekleştirilen gerçek zamanlı bir görevin zamanında gerçekleştirilmemesi sistemin yanlış çalışmasına sebep olur ve sistemin çalışma alanına bağlı olarak felakete varan sonuçlar doğurabilir. Gerçek zamanlı sistemler başta savunma ve sağlık olmak üzere birçok kritik süreçlerde kullanılan bilgisayar sistemleridir ve günümüzde değişen eğilimler söz konusu sistemlere olan ihtiyaçları arttırmaktadır. Bununla birlikte gerçek zamanlı

sistemlerin artan ve değişen ihtiyaçlar doğrultusunda karmaşıklaşması, sistemlerde gereksinim duyulan işlem gücünü de artırmaktadır.

Tez kapsamında yapılan çalışmada, gerçek zamanlı sistemlerle uyumlu ve çok işlemcili ve dağıtılmış bellekli bir yonga mimarisi ve yonga üzeri ağ yapısı önerilmiştir ve önerilen ağ yapısı üzerinde işletilecek belirlenebilir ve tahmin edilebilir bir iletişim örüntü kümesi tanımlanmıştır. Önerilen iletişim örüntülerinin tahmin edilebilir tasarlanmasından faydalanılarak iletişim süreçleri de gerçek zamanlı görevler olarak ele alınmıştır ve hem iletişim hem de işlem görevlerini birlikte yöneten zaman odaklı bir görev zamanlama algoritması ortaya koyulmuştur. Sistemin iletişim katmanında yonga üzeri fotonik ağlar gibi yeni ve gelecek vaat eden teknolojilerden faydalanılmıştır. Sistemin başarımını ölçmek amacıyla, sistem üzerinde gerçek zamanlı sistemlerden seçilmiş dünya problemleri üzerinden teorik ve benzetim çalışmaları yapılmıştır. Yapılan çalışma sonuçları, önerilen sistem mimarisi ve üzerinde tanımlanan iletişim ve yönetim algoritmalarının gerçek zamanlı sistemler için uygun ve yüksek başarımlı bir yaklaşımı ortaya koyduğunu göstermiştir.

Anahtar Kelimeler: Gerçek zamanlı sistemler, Koşut sistemler, Görev yönetimi, Koşut sistemlerde iletişim, Yonga üzeri fotonik ağlar

ABSTRACT

Design of real-time scheduling and communication management algorithms on multicore architecture

Hüseyin Temuçin

Doctor of Philosophy Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Kayhan M. İmre

February 2018

The reach of processor architectures to physical boundaries has made parallel systems mandatory in all computer systems used in commercial and academic processes. Parallel systems are the computer systems in which a large number of processors are brought together in a topology to serve one purpose. That purpose may be resolving a problem identified or distributing tasks that a system provides. The processors on these systems can use distributed or shared resources and in distributed memory systems the processors shares data between them over a system area network.

In the general approach, the most basic performance measures of computer systems can be defined as accuracy and effectiveness. In a generalized system task, accuracy is only measured by the accuracy of outputs, and it is acceptable to delay the completion of tasks according to the instantaneous load of the systems. Real-time systems are specialized computer systems that operate on a timely basis and all tasks on these systems are expected to be completed before their *deadlines*. Failure to perform a real-time task on time in these systems causes the system to operate improperly and may results apocalyptic results, depending on the system's domain. Real-time systems are computer systems that are used in many critical processes, primarily defense and health, and today, changing trends are increasing the needs of such systems. However, the complexity of real-time systems with increasing and changing needs also increases the processing power required for systems.

In the thesis study, a chip architecture and nework-on-chip structure with multi-processor and distributed memory compatible with real-time systems is proposed and a deterministic and predictable set of communication patterns to be operated on the proposed network structure is defined. In the study communication processes are also considered as real-time tasks and a time-based task scheduling algorithm that manages both communication and transaction tasks has been put forward by taking advantage of the predictable

communication patterns. In the communication layer of the proposed system, today's new and promising technologies such as photonic networks on chip have been utilized. In order to measure the performance of the system, theoretical and simulation studies on the proposed system have been carried out problems selected real time systems. The results of the study show that the proposed system architecture and the communication and management algorithms described above provide a suitable and high performance approach for real-time systems.

Keywords: Real-time Systems, Parallel systems, Task management, Communication in parallel systems, Photonic networks on chip

TEŞEKKÜR

Tez konusunun belirlenmesini sağlayan, tez metninin yazılmasına ve tez çalışmasının hazırlanmasına yardımcı olan ve bu süreçte desteğini eksik etmeyen Sayın Doç. Dr. Kayhan M. İmre'ye,

Tez izleme aşamasında fikirleriyle tezin gelişmesine ve tez metnini inceleyerek biçim ve içerik bakımından son halini almasına yardımcı olan Sayın Yrd. Doç. Dr. Mustafa Ege ve Sayın Doç. Dr. Özcan Öztürk'e,

Tez çalışmamı inceleyerek çok değerli katkılarını sunan Sayın Prof.Dr. Halit Oğuztüzün'e , Sayın Doç.Dr. Süleyman Tosun'a ve Sayın Doç.Dr. Cüneyt Bazlamaçcı'ya

Tez sürecinde maddi ve manevi desteklerini esirgemeyen değerli arkadaşlarım Aydın Kaya'ya, Ali Seydi Keçeli'ye ve Yalın Yalıç'a,

Çalışma hayatım boyunca verdiği desteklerden dolayı Hacettepe Üniversitesi Bilgisayar Mühendisliği Bölümü çalışanlarına,

Hayatımın her aşamasında hep yanımda olan, sevgi ve desteklerini hiç eksik etmeyen çok sevgili aileme,

Sadece doktora çalışmalarımda değil hayatımda aldığım tüm kararlarda hem danışman hem de destekçim olan, sevgili eşim Özge Temuçin'e ve tez çalışmalarımı sabote eden biricik kızım Miray'a

Ve burada ismini zikredemediğim tüm bilge ve tecrübeli destekçilerime

Canı gönülden teşekkür ederim.

İÇİNDEKİLER

Sayfa

ŞEKİLLER

Sayfa

ÇİZELGELER

Sayfa

SİMGELER, KISALTMALAR VE TERİMLER

$\propto$ = Ağ kurulum maliyeti / τ = Bant genişliği / τ_o= Fotonik bant genişliği

L = İletim boyu

t = Görev

ACK: Acknowledgement

B: Byte

CMOS: Complementary Metal Oxide Semiconductor (Bütünleyici Metal Oksit Yarı İletken)

CRC: Cyclic redundacy check

DMA: Direct Memory Access (Doğrudan Bellek erişimi)

FLOPS: Floating Point Per Second

FPS: Frame per second

GZS: Gerçek zamanlı sistemler

ITRS: International Roadmap of Semiconductors

MIMD: Multiple instructor multiple data

MTS: Minimal Time slicing

NACK: Negative-Acknowledgement

SIMD: Single instructor multiple data

UML: Unified modeling language

NoC: Network-on-chip

PSE: Photonic Switching Element

RGB: Red-Green-Blue

RWA: Routing and wavelength assignment

SPmS: Sample Per micro Second

WDM: Wavelength Division Multiplexing

Actuator: Uyarıcı

All-to-all: Hepsinden hepsine

Bandwidth: Bant genişliği

Barrier Synchronization: Bariyer eş zaman uyumlama

Best fit: En uygun olan

Bit-rate transperancy: İkil-oran şeffaflığı

Broadcast: Yayın

Bus: Veri yolu

Circuit switching: Devre anahtarlama

Client: İstemci

Cluster: İşlemci öbeği

Column exchange: Sütun veri değişimi

Column gather: Sütun toplama

Column multicast: Sütun çoklu gönderme

Component framework: Bileşen çatısı

Component loader: Bileşen yükleyici

Component manager: Bileşen yöneticisi

Component repository: Bileşen ambarı

Container: Kapsayıcı

Control plane: Denetim düzlemi

Cyclic executive: Döngüsel yürütücü

Cyclic code execution: Döngüsel kod yürütme

Data plane: Veri düzlemi

Deadline: Zaman sınırı

Deterministic: Belirlenebilir

Dimension ordered: Boyut sıralı

Direct acyclic graph: Döngüsüz yönlü çizge

Direct networks: Dolaysız ağlar

Direct Memory Access: Doğrudan Bellek erişimi

Dispatching: Görev yönlendirme

Dominating node: Baskın düğüm

Event: Olay

Execution time: İşletim zamanı

Earliest deadline first algorithm: En erken zaman sınırı öncelik algoritması

Feasible scheduling algorithm: Uygulanabilir görev zamanlama algoritması

First fit: İlk uygun olan

Flow Control: Akış denetimi

Hard real-time systems: Zor gerçek zamanlı sistemler

Hard real-time task: Zor gerçek zamanlı görev

Indirect Networks: Dolaylı ağlar

Instance: Görev örneği

Interconnection Network: Ara bağlantılı ağ

Interrupt: Kesilme

Interrupt-only system: Sadece kesilmeli sistem

Inter-task communication: Görevler arası iletişim

Inter-task synchronization: Görevler arası eş zaman uyumlama

Job: İş

Latency: Gecikme

Least laxity First algorithm: En az esnek öncelik algortiması

Local scheduler: Yerel zamanlayıcı

Logging task: Günlük görevleri

Maximum urgency: En yüksek aciliyet

Mesh: Hasır doku

Micro kernel: Mikro çekirdek

Minimal time slicing: Minimal zaman bölütleme

Mirroring: Yansıtma

Mode division multiplexing: Mod bölmeli çoklama

Module: modül

Multithreading: Çok iş parçacıklı

Network-on-chip: Yonga üzeri ağ

Non-blocking: Çakışmaz

Non-preemptive: Ele geçirmesiz

Non-preemptive system: Ele geçirmesiz sistem

Photodetector: Foton algılayıcı

Photonic switching element: Fotonik anahtarlama bileşeni

Pipelining: Ardıl düzen işleme

Poll systems: Seçim tabanlı sistemler

Power aware systems: Güç sezimli sistem

Preemptive: Ele geçirmeli

Preemptive priority system: Öncelik ele geçirmeli sistem

Preemptive system: Ele geçirmeli sistem

Priority: Öncelik

Ready Time: Hazır durum zamanı

Real-time: Gerçek zamanlı

Reduction: İndirgeme

Row exchange: Satır değişimi

Row gather: Satır toplama

Row multicast: Satır çoklu gönderme

Safe-critical hard real-time systems: Güven-kritik gerçek zamanlı sistem

Scan: Tarama

Scatter: Dağıtım

Scheduling: Görev zamanlama

Selective wavelength: Dalga boyu seçmeli

Sensor: Duyarga

Share driven systems: Kaynak odaklı sistemler

Slack time: Zaman payları

Soft real-time systems: Yumuşak gerçek zamanlı sistemler

Soft real-time task: Yumuşak gerçek zamanlı görev

Software defined networks: Yazılım tanımlı ağlar

Spatial switching: Uzamsal anahtarlama

Sporadic: Düzensiz

Stream: Veri akışı

Switch: Anahtar

Task: Görev

Task partitioning: Görev ayırma

Thread: İş parçası

Time division multiplexing: Zaman bölmeli çoklama

Time driven systems: Zaman odaklı sistemler

Time frame patterns: Zaman çerçevesi örüntüsü

Time slots: Zaman aralıkları

Topology: Topoloji

Uninterruptable task: Kesilemez görev

Urgency: Aciliyet

Virtual paralleling: Sanal koşutlaştırma

Waveguide: Dalga kılavuzu

Wavelength Division Multiplexing: Dalga boyu bölmeli çoklama

1. GİRİŞ

İşlemci teknolojilerinde saat sıklığının fiziksel sınırlara yaklaşması, küçük ve büyük ölçekli bilgisayar mimarilerinin bütününde koşut sistemleri zorunlu kılmış ve koşut sistemlerinin kullanımlarını yaygınlaştırmıştır. Çok işlemcili koşut sistemlerin yaygınlaşmasıyla işlemci üretici firmalar düşük maliyetli ve yüksek başarımlı koşut sistemlere ve yonga düzeyinde çok işlemcili ağ sistemlerinin tasarımlarına yönelmişlerdir. Düşük maliyetli ve yüksek başarımlı koşut sistemlere olan ihtiyaç, bu konularda yapılacak akademik ve endüstriyel çalışmalara olan ihtiyacı artırmıştır.

Moore Yasası, donanım karmaşıklığının belli bir kuralla arttırılabileceğini, sistem başarımının her 18 ayda iki katına çıkacağını belirtmektedir [1]. Günümüz işlemci teknolojileri, bu yasayla paralellik göstermiştir ve 1980'lerden beri 2 ila 3 yıl arasında, tek bir çekirdek yongası içine yerleştirilebilen transistor miktarı yaklaşık iki katına çıkmıştır. Moore Yasasının önerdiği teknolojik ilerlemenin silikon transistor üretiminde hala geçerli olduğu söylenebilir. *The International Roadmap of Semiconductors* (ITRS) metal oksit yarı iletken boyutlarının 7nm düzeyine indiğini belirtmektedir [2]. Bununla birlikte, işlemci çekirdeklerin artan tasarımsal ve gerçekleştirim karmaşıklıkları, çekirdek geliştirmelerinin maliyet etkinliğini kaybetmesine sebep olmuştur ve fiziksel düzeyde olmasa da endüstriyel düzeyde sınırlarına ulaşmasına sebep olmuştur [3]. Bu durum artan işlem gücü ihtiyacını karşılayacak ticari işlemcilerin üretilmesini engellemiştir.

Çekirdek karmaşıklığında ulaşılan sınırlar, işlemci üreticilerinin çok işlemcili yonga mimarilere yönelmesine neden olmuşlardır. Bu alternatif mimari yaklaşımında, sistem içindeki birden fazla işlemci, sistemin amacına hizmet etmek için birlikte çalışarak kendilerine dağıtılmış görevleri eş zamanlı olarak gerçekleştirirler. Bu hizmet, sistemin adandığı tek bir problem olabilir veya işlemciler üzerinde birbirinden bağımsız işlemler yürütülebilir. Bununla birlikte, bellek erişim sığasını artırmak için, sistem üzerindeki işlemcilere dağıtılarak veya ön bellekler kullanımları ile bellek erişim sığası sistemdeki işlemci sayısıyla ölçekli biçimde artırılması gerekmektedir. Bu sistemlerde, ihtiyaç doğrultusunda, bellek ve işlemcilere benzer biçimde, çevresel aygıtlarda da birden fazla bileşen eşgüdümlü biçimde kullanılarak koşut çalışması sağlanmaktadır [4]. Bu yaklaşımla, yonga işlemci kapasiteleri, artırılan çekirdek sayısı ile etkin ve ölçeklendirilebilir bir biçimde gelişmeye devam etmektedir [5].

Çok işlemcili yaklaşımlar, günümüzde standart işlemci mimarisi haline gelmiştir ve mobil aygıtlardan yüksek kapasiteli sunuculara kadar bütün yeni nesil bilgisayarlar birden fazla çekirdeği olan işlemci yongaları barındırmaktadırlar. Bu mimari ticari olarak ölçeklendirilebilir işlemcilerin üretilmesini kolaylaştırdığı için sistemlerin ihtiyaçlara göre büyüyebilmesini sağlayabilmektedir. Bununla birlikte, ticari ve teknoloji bağımlılığı azalan tek çekirdek mimarilerdeki gelişmeler, çok işlemci mimarilere doğrudan katkı yapmaktadır. Bu bağlamda Moore yasasının, tek bir yonga üzerine sığdırabilecek işlemci çekirdeği sayısının her geçen yıl yükseleceği şeklinde evrimleştiği yorumlanabilir. ITRS hedefleri doğrultusunda, yakın zamanda standart genel amaçlı bir işlemci yongası içine yüzlerce veya binlerce çekirdeğin sığabileceğini göstermektedir [6]. Birden fazla işlemcinin belli amaç doğrultusunda yazılımsal ve donanımsal düzeyde eş güdümlü çalıştığı sistemler koşut sistemler olarak tanımlanırlar. Koşut sistemlerin etkinliği, sistemdeki tüm işlemcilerden yüksek başarım elde edilmesi ile orantılıdır. Bu durum işlemciler arası iletişiminin önemini artırmaktadır ve düşük iletişim kapasitesine sahip işlemcilerin yüksek hızda çalışması, sistemin istenilen başarıma ulaşması için yetersiz olacaktır. Bu bağlamda, işlemci çekirdek teknolojilerinde yüksek başarım elde edilmesi, bu işlemciler arasında yüksek hızlı iletişim sağlayacak yonga üzeri ağ mimarilerine bağlıdır [7]. Bu mimarilerin yüksek bant genişliği yanında, artan işlemci sayısıyla orantılı ölçeklenebilir sistemler olması gerekmektedir.

Silikon yarı iletken malzeme boyutlarındaki geliştirmelerle yonga içindeki işlemci sayılarının artmasını sağlamasına rağmen, bu işlemcileri birbirine bağlayan metal iletişim bileşenlerinin fiziksel sınırları, gelecek nesil işlemciler için yetersiz kalmaktadır [8]. Bu durum yonga üzeri ağ sistemlerinde elektrik tabanlı iletişime alternatif yeni iletişim mimarilerine ihtiyacı ortaya çıkarmıştır. Fotonik sistemlerin yonga üzeri ağ mimarilerinde kullanılması, geleneksel elektrik iletişimi sistemlerine bir alternatif oluşturmuştur. Bu sistemlerde iletişim, verinin ışık dalgaları üzerinde kodlanarak, ışık hızına yakın bir seviyede iletilmesi yaklaşımına dayanır. Bu bağlamda, optik sistemler, elektrik tabanlı alternatiflerine göre çok daha yüksek hız sağlamaktadır. Optik tabanlı iletişim sistemlerinin bir diğer avantajı; enerji tüketiminin elektrik tabanlı sistemlerde olduğu gibi mesajın uzunluğu ve iletim mesafesine bağlı olmamasıdır. Bu özellik ikil oran şeffaflığı olarak tanımlanır [7].

Gerçek zamanlı sistemler, çalışma etkinliği ve doğruluğunun zamanlılığa yoğun biçimde bağlı olduğu özelleşmiş bilgisayar sistemleridir. Bu sistemlerde herhangi bir görevin doğru

çalışması, elde edilen çıktının mantıksal değeri kadar çıktının üretildiği zamana da bağlıdır [9]. Gerçek zamanlı sistemlerde, sistem bütünlüğü ve tutarlığının sağlanması gerçek zamanlı görevlerin zamanında tamamlanması ile ilişkilidir. Bu bağlamda, gerçek zamanlı bir görevin her bir örneğinin genel olarak zaman sınırı olarak tanımlanan bir zaman kısıtlamasında tamamlanması garanti edilmelidir. Gerçek zamanlı sistemler, zamanlılık etkisine bağlı olarak zor ve yumuşak sistemler olarak gruplanırlar. Zor gerçek zamanlı sistemler, görevlerin zamanında tamamlanmaması durumunda, felaketle sonuçlanan olayların oluşabildiği sistemlerdir ve bu sistemlerde zaman sınırına her zaman uyulması zorunluluktur. Yumuşak gerçek zamanlı sistemlerde kaçırılan zaman sınırı sonuç sistemin doğruluğunu etkilese de zor gerçek zamanlı sistemlere göre daha kabul edilebilir sonuçlar üretirler. Gerçek zamanlı sistemler, başta sağlık, uzay ve savunma sanayinde gömülü sistem veya bilgisayar sistemi olarak yoğun biçimde kullanılmaktadırlar ve zor gerçek zamanlı sistemler çok kritik süreçlerin izlenmesi ve yönetilmesinde etkin rol almaktadırlar.

Tez kapsamında yapılan çalışmada, gerçek zamanlı sistemler için yazılım ve donanımın birlikte tasarlanması hedeflenmiştir. Çalışmada öncelikle çok işlemcili yonga mimarisi işlemci ve iletişim düzeyinde tasarlanmıştır. Tasarlanan bilgisayar mimarisinin iletişim düzeyinde, elektronik tabanlı sistemlere alternatif olan gelişmiş fotonik yonga üzeri ağ mimarilerinden faydalanılması sağlanmıştır. Önerilen donanım mimarisi üzerinde görev ve iletişim süreçlerinin yönetimi için belirlenebilir, ölçeklenebilir ve örüntüsel algoritmaların tasarlanması sağlanmıştır. Geliştirilen görev yönetimi yaklaşımı, sistem kaynaklarının zaman odaklı biçimde iletişim ve hesaplama görevleri arasında tahmin edilebilir ve etkin biçimde paylaşılmasını sağlamıştır. Çalışmanın son adımında gerçek zamanlı sistem senaryolarından seçilecek başarım çalışmaları önerilen sistem üzerinde anahtarlanarak ve benzetim ortamında test edilmiştir.

Tez metni şu şekilde tasarlanmıştır: 2.bölümde koşut ve dağıtılmış sistemler için literatürde geçen bilgiler verilmiş, 3. Bölümde gerçek zamanlı sistemler farklı açılardan ele alınmıştır. 4. bölümde fotonik yonga üzeri ağ teknolojileri ve yapılan çalışmalar ele alınmıştır. 5. Bölümde tez kapsamında önerilen sistem ve özellikleri ele alınmış ve 6. Bölümde önerilen sistem için yapılan teorik ve benzetim çalışmaları açıklanmış ve analiz edilmiştir. 7. bölümde önerilen sistem ve yapılan çalışmalar üzerinde tartışılarak sonuçlar ele alınmıştır.

2. KOŞUT VE DAĞITILMIŞ SİSTEMLER

Koşut sistem mimarileri günümüzde artık genel amaçlı bilgisayarlar için standart haline gelmeye başlamıştır. Bununla birlikte koşut sistemlerin daha etkin kullanımı yeni ihtiyaç ve problemlerin de ortaya çıkmasını sağlamaktadır. Koşut sistemler belli bir amaç doğrultusunda çok sayıda işlemcinin belli bir donanım alt yapısı doğrultusunda bir arada çalıştığı sistemleri tanımlar. Sistem üzerindeki işlemcilerin birlikte çalıştığı amaç; yüksek işlem gücü gereken bir problemi eş zamanlı olarak çözümlemek veya sistemin hizmet ettiği süreç üzerindeki görevlerin işlemci düzeyinde paylaşılması olabilir. Bu sistemlerin sistem ihtiyaçları temel olarak; aynı anda bir veya birden fazla problemin çözülmesi için gerekli işlemlerin, birbirinden bağımsız biçimde, eş kaynaklar üzerinde koşut şekilde çalışması ile benzer problem veya problemlerin çözülmesi için çalışan işlemlerin eş kaynak ve veri üzerinde koşut şekilde çalışması olmak üzere iki ayrı grupta toplanmaktadırlar. Birinci grup işlemler bilgisayar kaynakları üzerinde paylaşımın kritik olduğu durumlardır. Bu işlemler genellikle birbirleri ile iletişim ihtiyaçları düşük seviyede olan koşut işlemler grubunu içerirler. İkinci grup işlemler benzer problem üstünde çalışan ve bilgi paylaşımı ihtiyaçları yüksek olan işlem gruplarıdır ve sadece kaynakların değil verinin paylaşımı noktasında ihtiyaçları vardır. Birinci grup işlemlerin veri üzerinde yaptığı işlemler benzerlik göstermezken ikinci grup işlemlerin veri üzerinde yaptığı işlemler ciddi benzerlikler içerir. Birinci grup işlemler işlem yoğunluklu hesaplamaları kapsar ve genel amaçlı kullanıma daha uygundur. İkinci grup işlemler ise veri yoğunluklu hesaplamaları kapsar ve daha özelleşmiş sorunların çözümünü kapsar.

2.1. Mimari

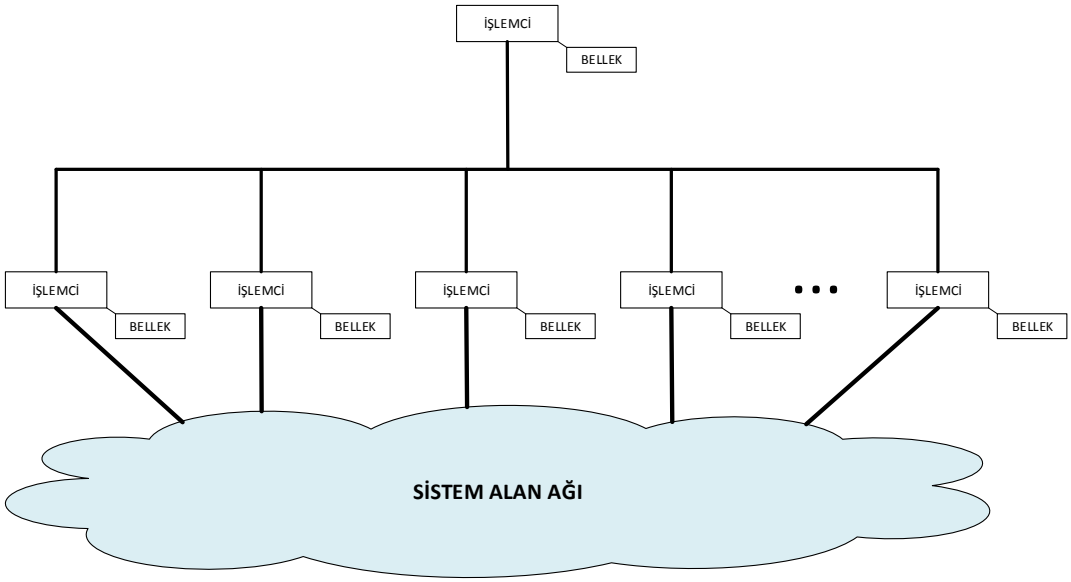
Koşut sistemlerdeki farklı ihtiyaçlar farklı mimarilere olan ihtiyaçların ve bu problemlere göre birbirinden farklı koşut işlemci mimarilerinin ortaya çıkmasını sağlamıştır. Bilgisayar mimarileri için yaygın kullanılan sınıflandırma M.Flynn tarafından yapılmıştır. Flynn, temel olarak işlemci mimarilerini dört temel grupta toplamıştır [10].

SISD (Single Instruction, Single Data): Bilinen geleneksel mimari tek çekirdekli işlemcileri kapsar. Tek bir işlemci tek bir bellek alanındaki verileri tekil komut akışlarıyla işler [11].

SIMD (Single Instruction, Multiple Data): Aynı işlemin, benzer özellikli farklı grup veriler üzerinde koşut olarak gerçekleştirildiği mimarilerdir. Bu mimarilerde, denetleyici işlemci olarak da tanımlanan bir komut işlemcisi ve komut istemcisinin gönderdiği komutu

işleyen çok sayıda işlemci bulunur [11]. Görüntü işleme amaçlı işlemcilerin büyük bir çoğunluğunun bu mimaride olduğu söylenebilir. Dağıtılmış bellekli bir SIMD modeli Şekil 2-1 ile gösterilmiştir.

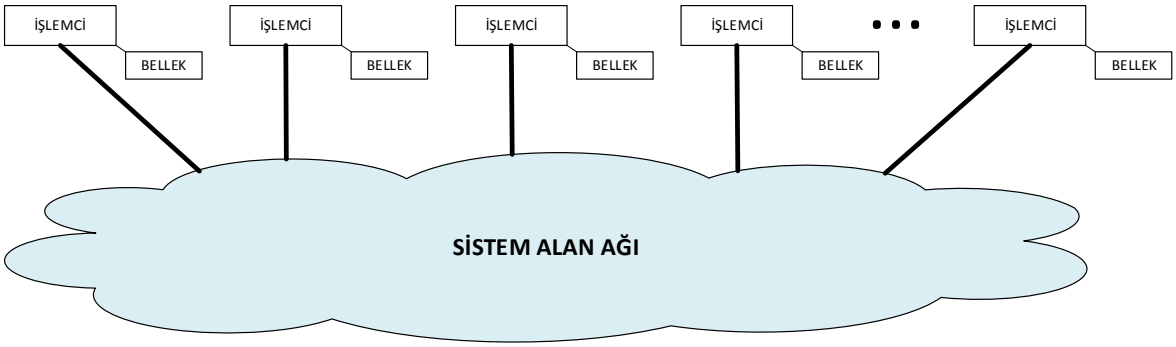
MISD (Multiple Instruction Single Data): Birden fazla birimin aynı veri üzerinde çalıştığı mimari sınıfıdır ve bu sınıf ardıl düzen işleme kullanan bilgisayarları içerir [11].



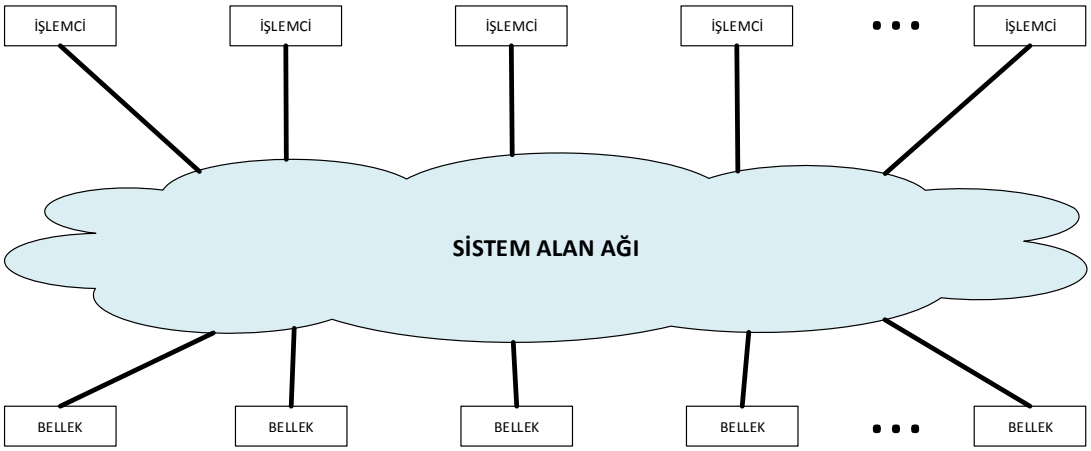
Şekil 2-1: Dağıtılmış bellekli SIMD koşut system modeli

MIMD (Multiple Instruction Multiple Data): MIMD mimarisindeki işlemciler, SIMD mimarisinden farklı olarak, komut düzeyinde zaman uyumsuz çalışırlar ve sistemde denetleyici bir işlemci bulunmaz. Bununla birlikte daha üst katmanlarda zaman uyumlu veya uyumsuz çalışabilirler [11]. MIMD mimarisi içindeki işlemcilerin birbirlerinden bağımsız çalışan özel birer komut yönetim birimleri bulunur ve işlemciler kendilerine atanmış veriler üzerinde komut çalıştırırlar. Genel amaçlı çok çekirdekli işlemcilerin büyük bir çoğunluğunun bu mimariye sahip olduğu söylenebilir.

MIMD mimarileri bellek yaklaşımlarına göre temel olarak paylaşımlı bellek ve dağıtılmış bellek olmak üzere iki temel gruba ayrılmaktadırlar. Paylaşımlı bellekli sistemlerde tüm işlemciler aynı bellek alanını paylaşırlar ve veri iletişimi bellek üzerinden olur. Bu işlemcilerde bellek tutarlılığı yönetilmesi gerekli bir sorundur. Dağıtılmış bellekli mimarilerde ise her işlemcinin kendisine adanmış bir bellek alanı vardır ve veri iletişimi işlemciler arası kurulmuş sistem ağı üzerinden gerçekleştirilmektedir. Yaygın iki MIMD mimarisi Şekil 2-2 ile gösterilmiştir.



DAĞITILMIŞ BELLEK MIMD



PAYLAŞIMLI BELLEK MIMD

Şekil 2-2: MIMD mimari yapısı

2.2. İletişim

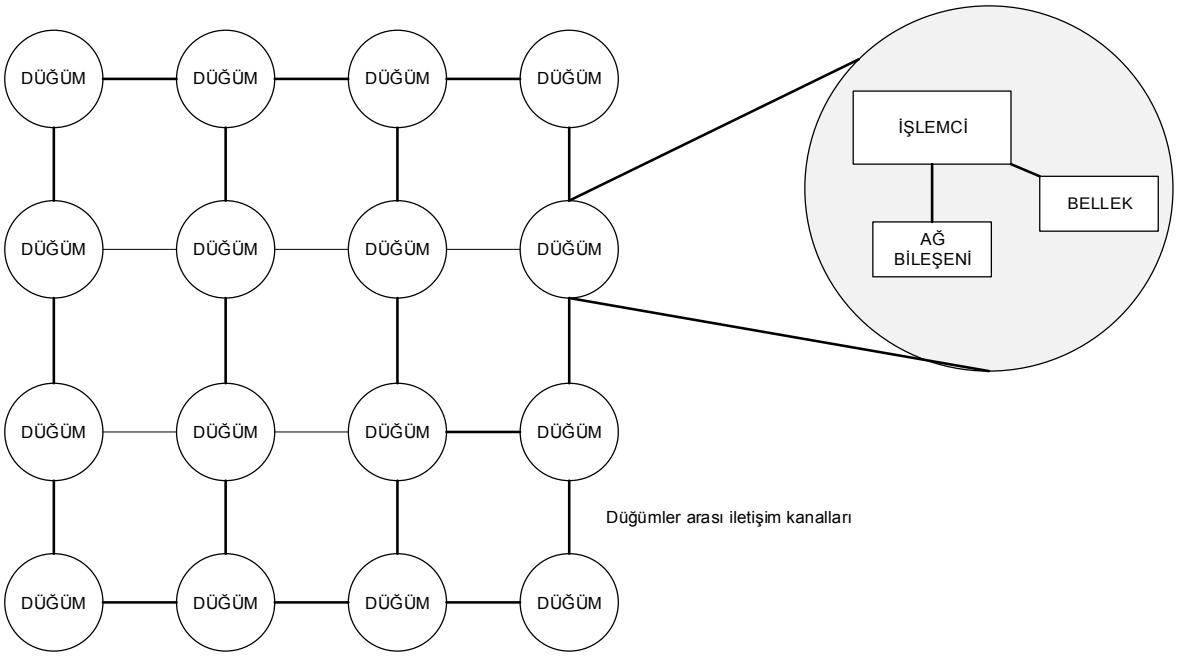
Sayısal bir sistem üç temel bileşenden oluşur: *mantık, bellek* ve *iletişim* [12]. Mantık verinin işlenmesi, Bellek işlenen ve işlenecek verinin saklanması ve zamanı geldiğinde tekrar çağrılması ve İletişim ise verinin birimler arasında taşınmasını belirtir. Bu yaklaşımla, bilgisayar sistemleri bileşenlerinin işlevleri kapsamında düşünüldüğünde, mantık işlemci, iletişim ise sistem ağıyla eşleşmektedir.

Koşut sistemler belli bir amaç doğrultusunda çok sayıda işlemcinin belli bir donanım alt yapısı doğrultusunda bir arada çalıştığı sistemleri tanımlar. Belli bir amaç doğrultusunda birlikte çalışan işlemciler, sıklıkla veri paylaşımına ve eş zaman uyumlamaya ihtiyaç duyarlar. Bu bağlamda koşut sistemler işlemcileri arasında veri ve bilgi paylaşımını sağlamak üzere bir takım ağ katmanı donanımlarına ihtiyaç duyarlar. Söz konusu ağ

bileşenleri, işlemcilerin konumlandırıldığı aynı yonga üzerindeki veri yolları, devre elemanları veya farklı bilgisayar sistemleri arasında yapılandırılmış ağ sistemleri olabilirler. Koşut sistem üzerindeki işlem birimlerin ağ üzerinden gelen mesajları çözümlemek ve ağ içinde kendi mesajlarını iletmek için ağ bileşenlerine ihtiyaç duyarlar ve bu ağ bileşenleri, sistemin ağ mimarisine bağlı biçimde, anahtar ve yönlendirici olabilirler.

Belirtildiği üzere çok işlemcili koşut sistemlerinin işlemci mimarileri genel olarak MIMD ve SIMD mimarilerinden birine sahiptir. Bu sistemlerin bellek bileşenleri yapı olarak diğer bilgisayar sistemlerindeki bellek bileşenlerinden farklı bir şekilde çalışmazlar. Fakat bellek bileşenlerinin sistem içindeki işlevine göre dağıtılmış bellek veya paylaşımlı bellek olmak üzere iki temel yaklaşıma sahiptirler. Dağıtılmış bellek mimarilerde bellek bileşenleri işlemcilere dağıtılırlar ve veri iletişimi ağ üzerinden iletiler ile sağlanır. Paylaşımlı bellek mimarilerinde, ortak bellek bileşenleri tüm işlemcilerin bir protokol üzerinden eriştiği bir iletişim bileşeni üzerinden işlemcilere paylaştırılırlar.

Koşut sistemlerde ağ protokolü temel olarak dolaylı ve dolaysız ağlar olmak üzere iki gruba ayrılırlar [4], [11]. Dolaysız ağlarda işlemcilerin / bilgisayarların (düğümler) birbirlerine noktadan noktaya (doğrudan) bir veya birden fazla bağlantısı bulunur ve ileti gönderirken düğümler arası bulunan kanalları kullanırlar. Bu ağ mimarisinde düğümler birbirlerine dolaysız olarak ileti gönderebilirler. Dolaylı ağlarda düğümlerin birbirleriyle doğrudan bağlantıları yoktur ve gönderilen iletilerin bir veya birden fazla ağ bileşeni üzerinden geçmesi gerekmektedir. Genel olarak dolaylı ağlarda ağ bileşeni olarak anahtar kullanılırken, dolaysız ağlarda ise her düğüme atanmış bir yönlendirici bulunur [12]. Dolaylı ağ yapısı Şekil 2-3 ile gösterilmiştir.



Şekil 2-3: Dolaysız ağ bağlantılı hasır doku çok işlemci mimarisi örneği

Koşut sistemlerde ağ bileşeni temel olarak dört temel katmandan oluşur.

- Topoloji
- Anahtarlama
- Yönlendirme
- Akış denetimi

2.2.1. Topoloji

Koşut sistemler belli bir amaç doğrultusunda çok sayıda işlemcinin belli bir donanım alt yapısı ile bir arada çalıştığı sistemleri tanımlar. Bu sistemler işlemci birimleri, bellekler ve veri iletişimi birimlerinin bir araya gelmesinden oluşurlar. İşlemciler arasındaki veri paylaşımı ve eş zaman uyumlama, ağ bileşenleri arasında bulunan iletişim kanalları üzerinden gönderilen veri paketleri ile sağlanır. Dolaylı ağlar üzerinde her bir işlemci, bellek ve ağ bileşeni bloğu bir düğümü tanımlar [4].

Topoloji sistemdeki düğümlerin ve kanalların hangi örüntüde konumlanacağını ve birbirlerine nasıl bağlanacağını matematiksel olarak tanımlar [4]. Bu bağlamda, topoloji, N elemanlı bir düğüm kümesinin, C elemanlı bir kanal kümesi üzerinden birbirlerine bağlanmalarının matematiksel olarak ortaya konulması olarak tanımlanabilir. Ortaya koyulan matematiksel model, herhangi iki düğüm arasında veri iletişimi için kurulacak yol ve rotanın en kısa veya en maliyetli olacağını da belirlemektedir.

Tasarlanan bir sistemin ilk ve en önemli adımı topolojinin belirlenmesi olacaktır. Belirlenen topolojiye göre sistemin veri iletişimi, akış denetimi ve yönlendirme örüntüleri doğrudan topolojiye bağlıdır. Sistem tasarımında topoloji iki kıstas kapsamında seçilmektedir: maliyet ve ağ performansı [4]. Ağ performansı istenilen verinin en hızlı biçimde aktarılmasına bağlıdır. Bu durum iki kıstas ile ölçülür: bant genişliği ve gecikme. Ağ performansını, iletişim bileşenlerinin bütünü belirlediği bilinmektedir fakat topoloji özellikle bant genişliğine doğrudan etki etmektedir. Ağ üzerinde gecikmelerin azalması, temel olarak ağ üzerindeki eş zamanlı iletimlerde kullanılan rotalarda oluşacak çakışmaların en az indirgenmesi ile orantılıdır. Çakışmaz topoloji, bir düğüm üzerinden aynı boyuttaki hedef düğüme gitmek isteyen iki ileti olmadıkça çakışma olmayacağını garanti eden topolojilerdir[13]. Bu durum yönlendirme algoritmaları, yönlendirici mimarisi veya kanal çoklama üzerinden sağlanabilir veya geliştirilebilir.

Topolojinin etkilediği diğer bir olgu sistem maliyetidir. Topoloji, sistem üzerindeki düğümlerin birbirlerine kaç kanalla ve nasıl şekilde bağlanacağını tanımlamaktadır. Dolayısıyla iletişimde kullanılacak kanal sayısının sayısı topoloji ile belirlenmektedir ve topolojinin sistem maliyetlerini de doğrudan etkilemesine neden olmaktadır.

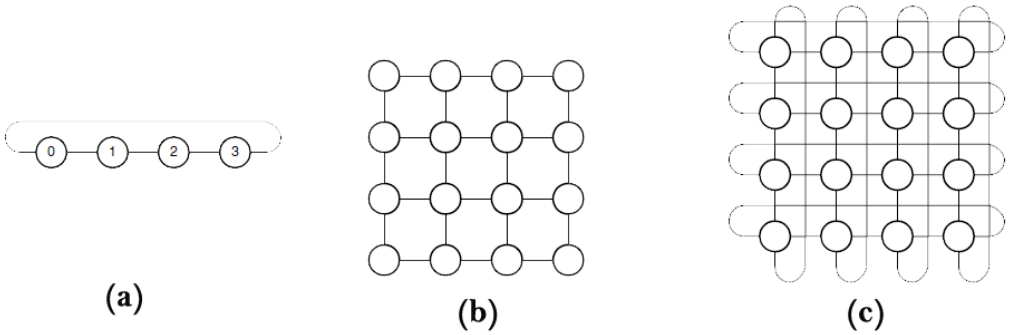
Koşut sistemlerde ideal topoloji; N adet düğüm bulunan bir sistemde tüm düğümlerin birbirlerine doğrudan bağlantısı olduğu ve her düğümün herhangi bir diğer düğüme doğrudan ileti gönderebildiği yapıdır. Fakat bu yapı büyük sistemler için çok maliyetlidir ve kullanılmaz [11].

k-ary n-tip topolojiler, farklı boyutlardaki koşut sistemlerde tercih edilmekte olan bir topoloji grubudur. Bu topolojiler, genel olarak her boyutta k tane düğümün olduğu n boyutlu grid topolojiler olarak tanımlanabilirler. Bu bağlamda, her boyutta k tane düğüm olan n boyutlu bir küpte $N = kxn$ adet düğüm bulunmaktadır [4]. *k-ary n-tip* topolojiler, özellikle dolaylı ağ mimarisine sahip koşut sistemlerde sıklıkla tercih edilen topolojilerdir ve günümüz çok işlemcili yonga üzeri ağlarda da kullanılmaktadırlar [7].

Koşut mimarilerde en çok tercih edilen özelleşmiş *k-ary n-tip* topolojiler *k-ary n-küp* torus ve *k-ary n-mesh* hasır doku ağlardır. Hasır doku ağlar, her bir düğümün, en yakınındaki komşu ile doğrudan iletişim kanalı bağlantısına sahip olduğu topolojilerdir. Bu topolojide, düğümler bağlantı örüntüsü olarak üç grupta toplanabilirler:

- *Köşe düğüm:* k-ary n-mesh bir ağda, n boyutun hepsinde en küçük indis sahibi düğümlerdir. n boyutlu bir topolojide, her bir köşe düğümün n adet komşusu bulunur.
- *Dış çeper düğüm:* k-ary n-mesh bir ağda, n boyutun herhangi birinde en küçük indis sahibi düğümlerdir. n boyutlu bir topolojide, her bir köşe düğümün $(2n-1)$ adet komşusu bulunur.
- *İç düğümler:* k-ary n-mesh bir ağda, n boyutun hiç birinde en küçük indis sahip olmayan düğümlerdir. n boyutlu bir topolojide, her bir iç düğümün $2n$ adet komşusu bulunmaktadır.

k-ary n-küp torus ağlar topolojik olarak hasır ağlara benzerler. Fakat bu topolojide, köşe ve dış çeper indislerinde bulunan düğümler, her boyutta kendilerine en uzak olan düğüme bir kanal bağlantıları bulunur. Bu bağlamda bu topolojideki tüm düğümler simetrik ve homojen bir yapıya sahiptir ve n boyutlu bir küpte her düğümün $2n$ adet komşusu bulunur. Örnek torus ve hasır doku topolojileri Şekil 2-4 ile gösterilmiştir.



Şekil 2-4: k-ary n-type topoloji örnekleri. (a) 4-ary 1-küp (b) 4-ary 2-mesh (c) 4-ary 2-küp [4]

Torus ve hasır doku ağ topolojilerinin koşut sistemlerde kullanılması birçok avantajları bulunmaktadır. Öncelikle bu mimariler ölçeklenebilir topolojilerdir. Bu topolojilerde düğüm sayısı arttıkça, kanal sayısı ve bant genişliği düzeyinde benzer ölçüde genişlemektedir. Bu mimarilerdeki düğümlerin belirgin kanal örüntülerine sahip olmaları, bu topolojiler üzerinde yönlendirme ve iletişim algoritmalarının geliştirilmesi ve genelleştirilmesini kolaylaştırır. Bu durum, özellikle torus topoloji üzerinde rota çeşitliğinin ve ağ üzerinde etkin yük dağılımını sağlar. Bununla birlikte k-ary n-type topolojiler dolaysız ağ topolojileridir ve özellikle hasır doku ağlarda uzak düğümler arası rotalamada birçok düğümden geçilmesi gerekebilmektedir.

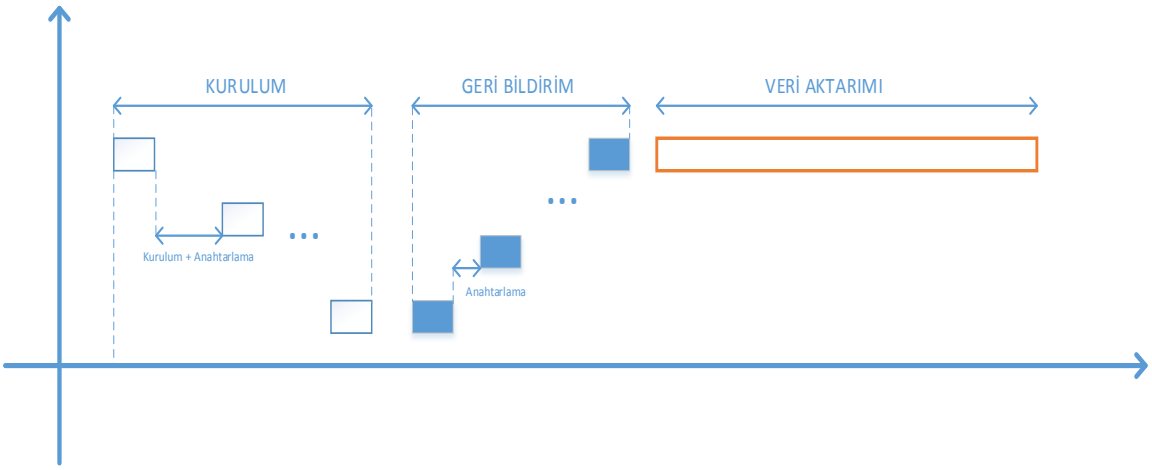
Çok işlemcili mimarilerin yonga üzerinde tercih edilmeleri, yonga üzeri veri iletişimini artırmıştır. Yonga üzerinde artan ağ ihtiyaçları karşısında işlemci yongalarında kullanılan veri yolu mimarileri yetersiz kalmaya başlamıştır. Veri yollarının fazla işlemcilerde darboğaz oluşturması durumu, üretici ve araştırmacıları yonga üzeri ağ mimarilerine yöneltmiştir. Yakın zamanda üretilen çok çekirdekli işlemci ve yardımcı işlemcilerde geleneksel veri yolu mimarilerinin yerini farklı topolojilerdeki yonga üzeri ağlar almaktadırlar [14]. Torus ve hasır doku ağları geliştirme maliyetlerinin uygunluğu ve vaat ettiği yüksek bant genişliği bağlamında, yonga üzerinde gerçekleştirime uygun topolojilerdir ve günümüz yonga üzeri ağlarda bu topolojilerin 2 boyutlu modelleri kullanılmaktadır [7].

2.2.2. Anahtarlama

Anahtarlama yaklaşımı, rotası belirlenmiş bir iletinin, akış denetimi yaklaşımı ile belirlenen aktarım yolu üzerindeki kanal ve anahtar kaynaklarının hangi şekilde atanacağı ve kullanılacağını belirler ve bu süreçte akış denetimi yaklaşımı ile birlikte çalışır [4]. Koşut yazılım katmanında düğümler arası gönderilen iletiler, ağ katmanında paket veya alt paketlere bölünür ve bu paketler ağ ortamı üzerinden ilgili düğüme iletilerek orada tekrar birleştirilir.

Koşut sistemlerde en yaygın kullanılan anahtarlama yaklaşımları devre anahtarlama, paket anahtarlama ve *wormhole* anahtarlama ve yaklaşımlarıdır [4], [11]. Paket anahtarlamada ileti tek bir paket üzerinden hedefe gönderilir ve ileti her kanaldan tek parça halinde iletilir. Paket anahtarlama genel amaçlı bilgisayar iletişimlerinde yazılım düzeyinde sıklıkla tercih edilen bir iletişim örüntüsüdür. Internet üzerinde çalışan standart iletişim protokolleri paket anahtarlama yaklaşımı üzerine tanımlanmışlardır.

Devre anahtarlamada, kaynak ve hedef düğümler arasındaki tüm kanallar arasında tek bir kanal gibi bir devre kurulduktan sonra bu devre üzerinden bir seferde iletilir. Devre anahtarlama ile yapılan bir iletimin zaman ekseninde davranışı Şekil 2-5 ile gösterilmiştir.



Şekil 2-5: Devre anahtarlama ile veri iletişiminin zaman ekseninde gösterimi [4]

Wormhole anahtarlamada ileti yine alt paketlere bölünür fakat devre kurulumu gerçekleşmez. Bu yapıda ilk alt paket yönlendirme bilgisi içerir ve son paket gelene kadar ilk ve ara düğümler gelen alt paketleri belirlenen kanal yönünde iletirler. Paket anahtarlama kısa ve sık paketlerin kullanıldığı yapılarda, devre anahtarlama ise uzun ve seyrek mesajlaşma içeren durumlarda daha etkin sonuç verir [4], [15]. Wormhole anahtarlama diğer iki yaklaşımdan da özellikler içerir ve yönlendirici tabanlı koşut sistemlerde etkin sonuçlar vermektedir.

2.2.3. Yönlendirme

Dağıtılmış bellekli bir mimaride, iki düğüm arasındaki iletişim bu düğümler arasında ileti geçilerek sağlanır. Bu ileti herhangi iki düğüm arasındaki bir iletişim iletisi olabileceği gibi, birden fazla düğüme yayınlanan bir toplu ileti de olabilir. Ara bağlantılı çok işlemcili bir mimaride, hedef ve kaynak düğümler arasında iletişimin sağlanması için, bu iki düğüm arasında bir iletişim yolunun kurulması gerekir ve dolaylı ağ topolojilerinde kurulacak iletişim yolu birden fazla ara düğüm üzerindeki kanalların tahsisini gerektirebilir. Çok işlemcili bir mimaride, sıklıkla işlemciler arasında bilgi alışverişine ihtiyaç duyulur ve söz konusu iletişim iletiler üzerinden sağlanacağı için, bu sistem üzerinde düğümlerdeki kanalların ve diğer ağ kaynaklarının etkin yönetilerek iletiler arasındaki çakışmaların asgari düzeyde tutulması sistem başarımını doğrudan etkilemektedir.

Yönlendirme, iki düğüm arasında bir iletinin geçeceği yolun belirlenmesidir. Yönlendirme algoritması sistem topolojisi ile doğrudan ilişkilidir ve yönlendirmenin nasıl yapılacağı topolojiye bağlı olarak karar verilir. Yönlendirmenin nasıl sağlanacağı, seçilen yönlendirme algoritması ile belirlenir.

Dolaylı ağlarda en bilinen yönlendirme algoritması, boyut sıralı yönlendirme algoritmasıdır. Boyut sıralı yönlendirmede, ileti hedef düğüme ulaşması için gerekli yolu önce X (yatay), sonra Y (dikey) ve varsa Z boyutunda ilerleyerek hedef düğüm indisine ulaşır.

Yönlendirme algoritmaları yönlendirmenin yapıldığı düğüm (merkezi-dağıtılmış), yönlendirme davranışı (belirlenebilir – uyarlamalı) veya yol uzunluğu (minimal-dolaylı) gibi özellikleri doğrultusunda birçok sınıfa ayrılır. Yönlendirme algoritmalarının detaylı incelenmesi bu çalışma kapsamının dışındadır.

2.2.4. Akış Denetimi

Akış denetimi anahtar yastıkları ve iletişim kanalları gibi sistem kaynaklarının düğümlere tahsis edilmesi ve bu kaynakların etkin ve adil yönetilmesi için geliştirilmiş yapılar bütünüdür. İyi bir akış denetimi yönetimi sistem kaynaklarının etkin kullanılmasını sağlayarak, düşük ve tahmin edilebilir gecikmelere izin verirken kötü bir akış denetimi çakışmalara sebep olarak diğer sistem kaynaklarının boşta kalmasına sebep olur ve bu durum düşük bant genişliği ile iletişim yapılmasına sebep olur [12].

Akış denetimi yapısı sistem için iki temel düzeyde işlev sağlar: kaynakların atanması ve çakışmaların çözülmesi. Kaynakların atanması iletilerin ihtiyacı olan kanal ve yastıkların atanması ilgili iletilere atanması ve iletim tamamlandığında serbest bırakılarak diğer iletişimlere atanması süreçlerinin yönetilmesi sürecidir.

Akış denetimi temel düzeyde yastıklı veya yastıksız akış denetimi olmak üzere iki gruba ayrılır. Yastıksız bir akış denetiminde ara düğümde ihtiyacı olan kanalın başka bir iletime atandığı için bekletilmesi gereken bir ileti bileşeni (ileti, paket veya flit), bekletileceği bir yastık olmadığı için sistem tarafından düşürülür. Böyle bir sistemde akış denetimi sadece ilgili kanalların atanması ve başarısız olan ileti veya ileti parçalarının (paket veya flit) tekrar iletilmesi süreçlerini yönetir. Yastıklı akış denetiminde düğümlerin ağ bileşenlerini sınırlı sayıda ileti bileşenin bekletilebileceği yastıklar bulunur ve çakışmadan dolayı ilgili kanal kaynağını bekleyen ileti parçaları bu yastıklarda bekletilirler. Bu bağlamda, bu denetim yapısında kanal ve yastık yönetimi bir arada yapılır ve yastıklı akış denetiminde bir iletişim anında, iletinin yapıldığı hedef veya ara düğüme yastık durumu hakkında bilgi iletimi de sağlanmalıdır.

Yastıksız akış denetimleri arasında sıklıkla tercih edilen bir yöntem ACK-NACK akış denetimidir. ACK-NACK akış denetiminde, hedef düğüm başarıyla aldığı her ileti bileşeni

için karşı tarafa başarılı iletim anlamında bir ACK paketi yollar ve iletimi başarısız olan her iletim içinde ACK iletisinin karşıtı olan NACK iletisi yollar. Kaynak düğüm, her ACK mesajı aldığında ilgili iletime ait bir sonraki ileti bileşenini yollar ve aldığı NACK iletiler veya zaman aşımına uğrayan durumlarda son yolladığı ileti bileşenini tekrar yollar.

Fotonik iletişimde, optik yastık yapılarının ticari olarak yetersiz olması sebebiyle optik verilerin ara düğümlerde bekletilmesi olanaksızdır. Bu durum optik iletişimde sadece yastıksız akış denetiminin kullanılabilmesine olanak verir.

2.3. İletişim Protokolleri

Koşut sistemlerin amacı genellikle tekil işlemcilerle çözümü maliyetli olan problemlerin eş zamanlı çalışan işlemciler arasında paylaştırılarak daha kısa zamanda çözülmesini sağlamaktır. Bu yapı eş zamanlı çalışan işlemciler arasında yoğun biçimde veri paylaşımı ve iletişimini zorunlu kılar. Koşut sistemlerde veri iletişimi genellikle düğümler arası ileti gönderilerek sağlanır. Gönderilen iletiler koşut sistemin işlemci bileşenlerinin ileti geçebildiği bir iletişim evreni üzerinden sağlanır. Söz konusu iletişim evreni koşut sistem mimarisine bağlı biçimde ağ ve bellek bileşenlerinin farklı düzeylerde çalışması üzerinden sağlanır.

Koşut sistem bileşenleri olan düğümler üzerinde çalışılan verinin dağıtılması ve tekrar ilgili düğümlerde toplanmasında artık standart haline gelmiş bir grup iletişim protokollerini kullanırlar. Bu iletişim protokolleri toplu iletişim protokolleri olarak isimlendirilirler. Toplu iletişim protokolleri farklı amaçlar doğrultusunda, ilgili verinin ilgili işlemci / düğümlerce gönderilmesi ve hedef düğüme iletilmesi süreçlerini kapsar.

Toplu iletişim protokolleri genel olarak üç grupta toplanırlar [16], [17]: veri iletimi, genel hesaplama ve süreç denetimi. Veri iletimi protokolleri hesaplama süreçlerinde ilgili verinin işlemci grubunun bütünü veya bir grubuna iletilmesi amacıyla çağrılırlar. Bu gruptaki protokoller en maliyetli olan ve koşut hesaplamada en çok kullanılan toplu iletişim grubunu kapsar [18]. Veri iletim protokolleri çok farklı amaçla kullanılabilirler ve çalışma süreçleri kendi içinde ciddi farklılıklar içerebilir. Örneğin yayın bir düğümdeki verinin / iletinin diğer bütün işlemcilere iletilmesini amaçlar. Bununla birlikte, dağıtım bir düğümdeki bir verinin bölünerek ağdaki her bir düğüme farklı parçasını iletmesini sağlar.

Genel hesaplama protokollerinde, tüm ağdaki işlemcilere uygulanan genel işlemleri kapsar. Genel hesaplama protokolleri indirgeme ve tarama işlemlerini içerir. İndirgeme işleminde genel bir hesaplama işlemi tüm düğümlerdeki verilere uygulanır ve bu hesaplama

sonucunda her bir düğüm indirgenmiş verinin bir kopyasını elde eder [17]. Tarama işlemi indirgeme işleminin tüm düğümlere değil belli bir işlem grubuna uygulanmasıdır.

Süreç denetimi, ağ içindeki düğümler içinde eş zaman uyumlamanın sağlanması için kullanılır. Koşut programların hesaplamanın bazı noktalarında, eş zaman uyumlama sağlanarak, diğer bütün düğümlerin tüm düğümler o noktaya erişince kadar beklemesi gerekmektedir. Süreç denetimi bariyer işlemi sayesinde sağlanır. Toplu iletişim protokolleri Çizelge 2-1 ile özetlenmiştir.

Kategori	**İşlem**	**Tanım**
Veri İletimi	Yayın *(broadcast)*	Bir düğüm aynı iletiyi tüm düğümlere iletir.
	Dağıtım *(scatter)*	Bir düğüm bir iletinin farklı parçalarını tüm düğümlere iletir.
	Topla *(gather)*	Her bir düğüm bir düğüme farklı bir ileti gönderir ve ileti hedef düğümde birleştirilir.
	Hepsinden hepsine yayın *(All – to – all broadcast)*	Her bir düğüm bir yayın işlemi gerçekleştirir.
	Hepsinden hepsine dağıtım/toplama *(All – to – all scatter-gather)*	Her bir düğüm bir dağıtım işlemi gerçekleştirir.
Süreç denetimi	Bariyer *(barrier synchronization)*	Tüm düğümler bu noktaya erişmeden koşut sistem yazılımı devam edemez.
Genel hesaplama	İndirgeme *(reduction)*	Dağıtılmış veride genel bir hesaplama işlemi gerçekleştirir.
	Tarama *(scan)*	İndirgeme işlemini belli bir grup düğümde gerçekleştirir.

Çizelge 2-1: Koşut sistemlerde İletişim Protokolleri [12]

3. GERÇEK ZAMANLI SİSTEMLER

Bir sistem en temel düzeyde belli girdi grubunu bir çıktı grubuna anahtarlayan bileşenler grubu olarak betimlenebilir [3]. Bir bileşen grubunun bir sistem olarak tanımlanabilmesi için aşağıdaki özelliklere sahip olması gerekmektedir [19]

1. Bir sistem bir grup bileşenin örgütsel bir bağlamda bir araya gelmesinden oluşur.
2. Sistemin herhangi bir bileşeninin değişmesi, sistemin yapısal olarak değişmesine sebep olur.
3. Bir sistemin bir çalışma amacı vardır ve bu sistemin devamlılığı derecelendirilebilir.
4. Bir sistem belli bir amaca göre tanımlanır.

Bilgisayar sistemleri, üzerinde tekrarlı biçimde çalışan yazılımlar sayesinde, bir veya birden fazla amaca hizmet eden elektronik donanım bileşenleri olarak tanımlanabilirler. Sistem bünyesinde çalışan ve her birinin özelleşmiş amaçları olan yazılım bileşenleri genel olarak görev olarak tanımlanırlar (t_i). Görevlerin doğru çalışması, görev sonunda doğru mantıksal çıktıların elde edilmesi ile ölçülür. Sistem görevleri, sistemin yaşam döngüsü boyunca tekrarlı olarak yaratılarak hizmet verirler ve sistem görevlerinin her bir tekrarlı örneği iş olarak tanımlanır (j). Burada i. bir görevin j. kez oluşan örneği (j_{ij}) olarak gösterilebilir. Bir bilgisayar sisteminde, her bir görevin, gerekli kaynakları elde ettiği ve hizmet vermeye hazır olduğu zamanı betimleyen bir hazır durum (r_i) ve işini tamamlaması için gerekli olan bir çalışma zamanı (e_i) bulunur. Bu yaklaşımla görev ve iş bileşenleri eşitlik (1) ile gösterilebilir.

$$t_i = \{r_i, e_i\}$$
$$j_{ij} = \{r_{ij}, e_i\} \quad (1)$$

Gerçek zamanlı görevler, görevlerin doğruluğunun mantıksal çıktı tutarlılığı ile birlikte zamanında çalışması ve tamamlanmasına da bağlı olduğu özelleşmiş sistem görevleridir. Bu bağlamda her bir gerçek zamanlı görev, görevin hazır olduğu andan itibaren maksimum tamamlanma süresine sahiptir ve bu süre zaman sınırı (D_i) olarak tanımlanmıştır. Bu yaklaşımla, sistem başlama anına göreli olarak (r_i) anında hazır olan bir göreve ait iş oluşumunun, sistem ve görevin tutarlılık ve doğruluğunu sağlaması için $r_i + D_i$ zamanına kadar tamamlanması gerekir [20]. Bir görev anahtarlama algoritması ile yönetilen gerçek zamanlı bir görev grubunda, her bir görevin zaman sınırı (D_i) gelmeden önce

tamamlanması gerekir ve tamamlanması durumunda görevin önemine bağlı olarak sistemde felaket durumları oluşabilir [21].

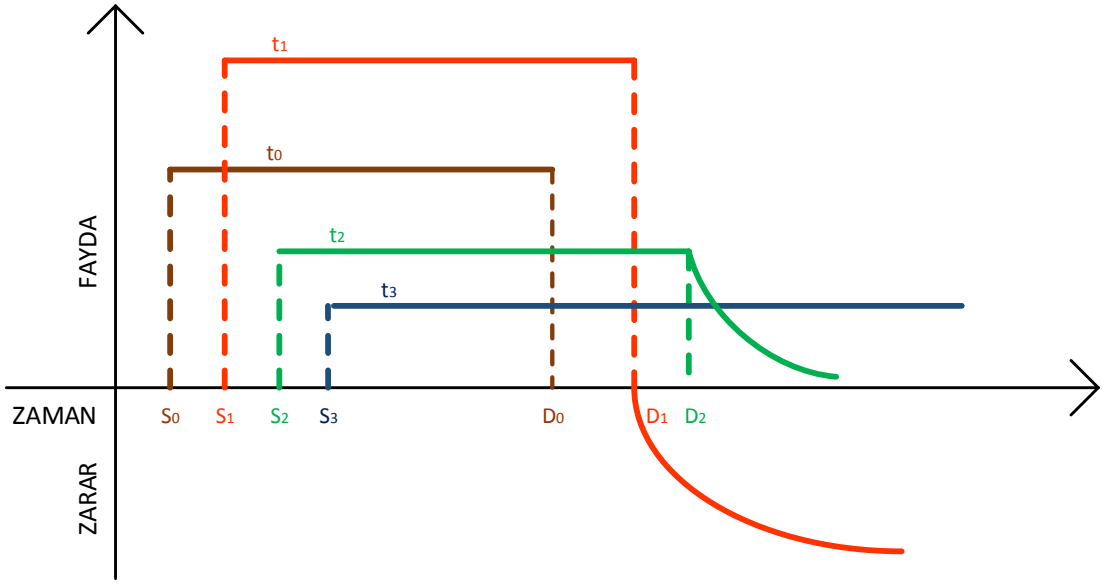
Gerçek zamanlı görevler, tekrarlama davranışlarına göre periyodik, aperiyodik veya düzensiz olarak üç gruba ayrılırlar. Periyodik görevlerin iş oluşumları belli bir frekans doğrultusunda tekrarlanırlar dolayısıyla bu görevlerin belli bir periyotları (p_i) bulunur. Bu doğrultuda, ilk oluşumu R_i anında oluşan bir periyodik görevin, k. oluşumu $R_i + kp_i$ anında oluşur. Aperiyodik gerçek zamanlı görevlerin belli bir periyotları bulunmaz ve bu görevler rasgele zamanlarda oluşurlar. Düzensiz görevler genel olarak aperiyodik görevlere benzer bir davranış gösterirler fakat bu görevlerin iki oluşumu arasında bir maksimum oluşma periyodu bulunur. Bu bakış açısıyla eşitlik (1) deki görev tanımı, gerçek zamanlı görevler için eşitlik (2) gibi genişletilebilir.

$$t_i = \begin{cases} \{r_i, e_i, D_i, p_i\}, & t\ periyodik/sporadik \\ \{r_i, e_i, D_i\}, & t\ aperiyodik \end{cases} \qquad (2)$$

Herhangi bir gerçek zamanlı görev içeren sistemler *gerçek zamanlı sistem* (GZS) olarak tanımlanırlar ve bu sistemlerin bütünlüğü ve tutarlılığı, sistem içindeki gerçek zamanlı görevlerin zamanında çalışmasına bağlıdır. GZS işlevsel bağlamda bir bilgisayar sistemi olmakla birlikte birçok açıdan genel amaçlı bilgisayar sistemlerinden ayrışırlar. GZS, geleneksel sistemlerden çok daha kısıtlı bir ortamda çalışırlar ve bu sistemlerin güvenlik önceliği diğer bilgisayarlarla karşılaştırılamayacak düzeydedir [3]. GZS oluşacak hatalar veya yanlış çıktılar çok daha yüksek maliyete veya can kaybına sebep olabilirler. Bu bağlamda gerçek zamanlı sistemler bir isteği beklenen zamanda mutlaka karşılamak zorundadır ve sistemin ele alınmamış herhangi bir aykırı durumu olamaz. Bu bağlamda gerçek zamanlı sistemler, bir isteği, daha önce belirlenen zamanda cevaplamayı garanti eden bilgisayar sistemleri olarak tanımlanabilirler [22].

Bir gerçek zamanlı sistemde, herhangi bir görevin sisteme katkısını fayda olarak tanımlayabiliriz [23]. Burada fayda, görevin tamamlanması sonucunda sistemde oluşacak pozitif veya negatif etkiyi tanımlayan bir işlev olarak düşünülmektedir. Gerçek zamanlı bir sistem görevinin zamanlılığı doğruluğu kadar önemli olduğu için, fayda işlevi görevin tamamlanma süresi ile de doğrudan ilişkilidir. Bununla birlikte, normal bir sistem görevinin gecikmeli olarak tamamlanması sadece sistem faydasının gecikmesine neden olurken, gerçek zamanlı bir görevin geç tamamlanması sisteme doğrudan zarar verebilir.

Gerçek zamanlı bir görevin geç tamamlanmasının sisteme vereceği zararın işlevinin boyutu, görevin zamanlılık sınıfını belirlemektedir. Zor gerçek zamanlı bir görev için, fayda işlevi hazır zamanında (r_i) en yüksek değere sahiptir ve zaman sınırı anında bu işlev değeri sıfır olur. Zor gerçek zamanlı bir görevin zaman sınırı sonrası tamamlanmasının sisteme pozitif bir faydası olmamakla birlikte, bu görevlerin gecikmesi sisteme veya sistem çevresine zarar verebilir. Güven-kritik zor görevlerde, fayda işlevinin negatif bir etkisi vardır ve negatif değerin zamansal değişimi görev özelliklerine göre değişiklik gösterir. Zor olanlar dışındaki gerçek zamanlı görevlerin fayda değeri zaman sınırı bittiği anda sıfıra düşmez ama zamanla orantılı biçimde pozitif etkisi azalır. Bu sınıftaki görevler yumuşak gerçek zamanlı görevler olarak tanımlanırlar. Görevlerin zamanlılık düzeyindeki sertliği veya yumuşaklığı, bu görevleri içeren gerçek zamanlı sistemlerin davranışını da belirler. Video gecikmesinin istenmeyen bir durum olduğu fakat gecikmesinin sisteme zarar vermediği bir video konferansı sistemi yumuşak gerçek zamanlı sistem olarak tanımlanırken, nükleer çekirdeğin sıcaklığının izlenmesini sağlayan bir nükleer santral yönetimi sistemi zor gerçek zamanlı sistem olarak tanımlanacaktır. Gerçek zamanlı görevlerin fayda davranışları Şekil 3-1 ile gösterilmiştir.



Şekil 3-1: Görevlerin fayda davranışları (t_0: zor gerçek zamanlı görev, t_1: güven-kritik gerçek zamanlı görev, t_2: yumuşak gerçek zamanlı görev, t_3: gerçek zamanlı olmayan görev, S: başlangıç zamanı, D: zaman sınırı)

Gerçek zamanlı sistemlerin giriş / çıkış aygıtları da diğer bilgisayar sistemlerinden farklılaşırlar. Gerçek zamanlı sistemlerin girdi grubu farklı imge türlerinde çeşitlenmiş

duyarga veya kamera gibi aygıtlardan oluşurken, çıktı grupları belli girdilere göre farklı bileşenleri tetikleyen işleyicilerden oluşur. Bu sistemler birinci düzeyde insan kullanıcılara da sahipse genel amaçlı sistemlerdeki giriş/çıkış aygıtlarını da ek olarak içerebilirler [22].

Gerçek zamanlı sistemler, sistem tasarımı, davranışı ve sistem içindeki görevleri ele alma yaklaşımları doğrultusunda birçok sınıfa ayrılmaktadırlar. İleriki kısımlarda, sistemlerin özellikleri ele alınmış ve bu özellikler doğrultusunda sınıflandırılmaları yapılmıştır.

3.1. Sistem tasarımlarına göre

Bilgisayar sistemleri en temel düzeyde Yazılım, Donanım ve Ağ bileşenleri olmak üzere üç temel bileşen / katmandan oluşurlar. Donanım katmanı sistemin işlevselliğine bağlı olarak bir araya getirilmiş elektronik veya elektromekanik bileşenleri kapsar. Donanım katmanı, üzerinde çalışan sıralı komutları işleterek sistemin amacına yönelik hizmet etmesini sağlar. Donanım üzerinde çalışan sıralı komutlar yazılım olarak tanımlanırlar. Donanım bileşenlerinin belli bir amaca göre özelleşmesi ve insanların istedikleri hizmetleri sağlamaları yazılımlarla sağlanır. Ağ bileşenleri bilgisayar sisteminin kendi içindeki bileşenleri veya dışardaki diğer bilgisayar sistemleri ile haberleşmesini sağlayan iletişim bileşenlerini kapsar.

Bilgisayar sistemleri, sistemin kullanıcı grubu ile sistemi tasarlayıp geliştirecek geliştirici grubunun birlikte yürüteceği analiz, tasarım ve test süreçleri sonucunda mantıksal düzeyde tasarlanırlar. Ortaya çıkarılan mantıksal tasarımlara göre geliştirilen sistem kullanıcı testine ve/veya hizmetine açılır. Sistem üzerinde gerçekleştirilen analiz, tasarım, gerçekleştirim ve test süreçleri; sistemin devreye alınması sonrası süreçte de döngüsel olarak devam eder.

Bilgisayar sistemlerinin tasarımı, geliştirimi ve geliştirme sonrası hedeflenen sistem özelliklerinin belirlenmesinde birçok kıstas etkindir. Bu kıstaslar içinde en belirleyici olanlar istenilen sistemin kullanıcı gereksinimleri ve sistemi tasarlayan geliştiricilerin sistem tercihleri olacaktır. Bununla birlikte sistemin çalışacağı ortamın fiziksel özellikleri veya sistemin enerji gereksinimleri gibi durumlar da sistemin tasarımını veya durumunu etkileyecektir.

Bilgisayar sistemleri genel amaçlı veya belli bir hizmete özelleşmiş olabilirler. Genel geçer hizmet amacıyla tasarlanan bilgisayar sistemleri genel amaçlı bilgisayarlar olarak tanımlanırlar. Bir amaca adanmış ve özelleşmiş sistemler gömülü sistemler olarak tanımlanırlar. Genel amaçlı sistemler tek başlarına çalışan sistemler iken, gömülü sistemler genellikle farklı bir sistemin içinde belli bir süreci yönetmek üzere

konumlandırılırlar. Genel amaçlı bilgisayarların tasarım ve geliştirme süreçleri gömülü sistemlere göre büyük farklılıklar içerirler. Genel amaçlı bilgisayarlar genel geçer donanım özelliklerine ve mimarilere sahiptirler. Bu sistemlere yönelik yapılan geliştirmelerde belli olan donanımsal niteliklere yönelik yazılım sistemi tasarımı ve geliştirmeleri yapılır. Bununla birlikte, genel amaçlı sistemlerin çoğunluğunda bir işletim sistemi yazılımı bulunur ve geliştiriciler tasarımlarında bu sistemin hizmetlerini doğrudan kullanırlar. Genel amaçlı sistemlerin bütününde işletim sistemi üzerinden sağlanan bir ağ katmanı hizmeti bulunur ve geliştiriciler yazılım haberleşmesinde bu hizmetlerden faydalanırlar. Dolayısıyla ağ bileşenleri için de bir geliştirme özel durumlar dışında yapılmaz. Bazı güvenlik ve/veya etkinliğin yüksek önem arz ettiği bazı özelleşmiş yazılımlarda, geliştiriciler sistemin sağladığı ağ bileşen hizmetlerini kullanmayarak bu katmanı kendileri geliştirebilmektedirler[1].

Gömülü sistemlerde donanım bileşenleri genel amaçlı bilgisayarlar gibi genelleştirilmiş özelliklere sahip olmazlar ve her adanmış sistem amacına yönelik farklı donanımsal mimarilere ve özelliklere sahiptirler. Bu bağlamda, gömülü sistemler kullanıcı ve sistem gereksinimleri doğrultusunda donanımsal bileşenleri ve mimarisi; daha sonra tasarlanan donanım mimarisi üzerinde çalışacak yazılım sistemi tasarlanarak geliştirilir. Bu sistemlerin çoğunluğunda işletim sistemi görevini sağlayan bir yazılım katmanı bulunmasına rağmen, bu sistem geliştiriciler tarafından sağlanabilir veya üçüncü parti ticari veya açık kaynak çözümlerden faydalanılabilir [22].

Adanmış sistemlerin diğer sistemlerle iletişimi genel amaçlı bilgisayarlardan farklılık gösterebilir veya sistemin dış iletişime ihtiyacı olmayabilir. Dolayısıyla ağ bileşenleri ve özellikleri de sistemin gereksinimleri doğrultusunda diğer sistem bileşenleri ile birlikte tasarlanır. Bununla birlikte orta ve büyük ölçekli sistemlerde yazılım standartlarında kullanılan ağ protokolleri tercih edilebilir.

Gerçek zamanlı sistemler (GZS) bilindiği üzere birer bilgisayar sistemleridir. Bu sistemlerin çalışma prensipleri ve özellikleri GZS adanmış sistemler grubuna dâhil etmektedir. Dolayısıyla GZS tasarım ve gerçekleştirme süreçlerinde yazılım donanım ve ağ bileşenleri birlikte tasarlanır ve gerçekleştirilirler. GZS için önerilen ve seçilecek mimarilerin özelliklerin belirlenmesi veya farklılaşmasında, diğer bilgisayar sistemleriyle benzer biçimde, kullanıcı ve sistem gereksinimleri temel belirleyici etmenleri

[1]Bilgisayar sistemlerinin analizi ve tasarımı özelleşmiş bir akademik alan olup yapılan çalışma bu alanı kapsamamaktadır.

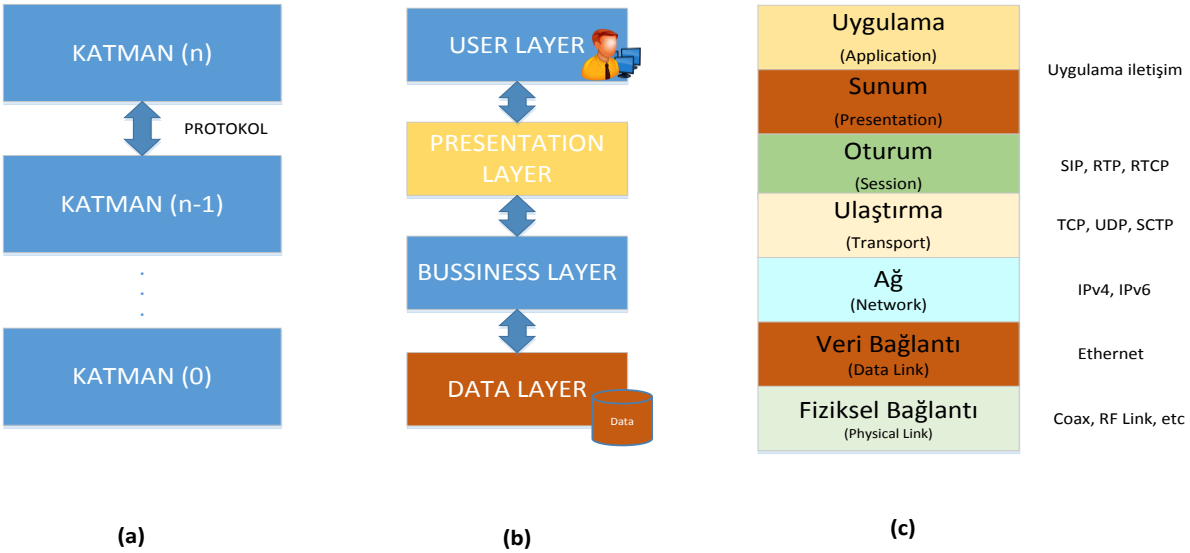
oluşturmaktadır. Bu sistemlerin izlediği ve yönettiği çevrenin özellikleri sistemin mimari özelliklerini belirleyen en belirgin kavramlardır.

GZS kapsamında ele alınacak bir sistemin tasarlandığını düşünelim. Belirtildiği üzere, sistemin tasarımı ve mimarisinde temel etmen kullanıcı ve sistem gereksinimleri olacaktır. Bu gereksinimlerin sisteme ve tasarımına etkilerini, iç ve dış çevrede meydana gelen olay ve durum değişiklikleri ve bu doğrultuda oluşacak verilerin yapısal ve akış örüntüleri olarak tanımlayabiliriz. Oluşan olaylar ve sistem düzeyinde olayları betimleyen verilerin özellikleri, sistemden beklenen veri işleme yapılarını ve sistemin olaylara cevap verme üst sınırlarını da belirlemektedir. GZS için en kritik ölçütün, olaylara doğru ve zamanında dönüş sağlaması olduğu düşünüldüğünde, sistemin mimarisi ve özellikleri sistem gereksinimlerinin ortaya koyduğu olay ve olay verileri örüntüsüne göre tasarlanacak veya seçilecektir.

GZS için önerilmiş birçok sistem mimarileri bulunmaktadır. Yapılan çalışma kapsamında, sistem gereksinim ve veri akışı özellikleri bağlamında altı örüntü tanımında toplanabilir.

3.1.1. Katmanlı sistem örüntüsü

Katmanlı mimari yaklaşımı birçok yazılım sorununda kullanılan bir yazılım örüntüsüdür. Bu örüntüde sistem işlev gruplarına göre katmanlara ayrılır. Her katman belli bir iş mantığından sorumludur. Bu örüntüde her katman sadece kendisinden bir üst veya bir alt katmanla iletişim kurar veya tanır ve katmanlar birbirleri arasında tanımlanan protokollerle haberleşirler. Çok katmanlı mimari basit düzeyde Şekil 3-2 ile gösterilmiştir.



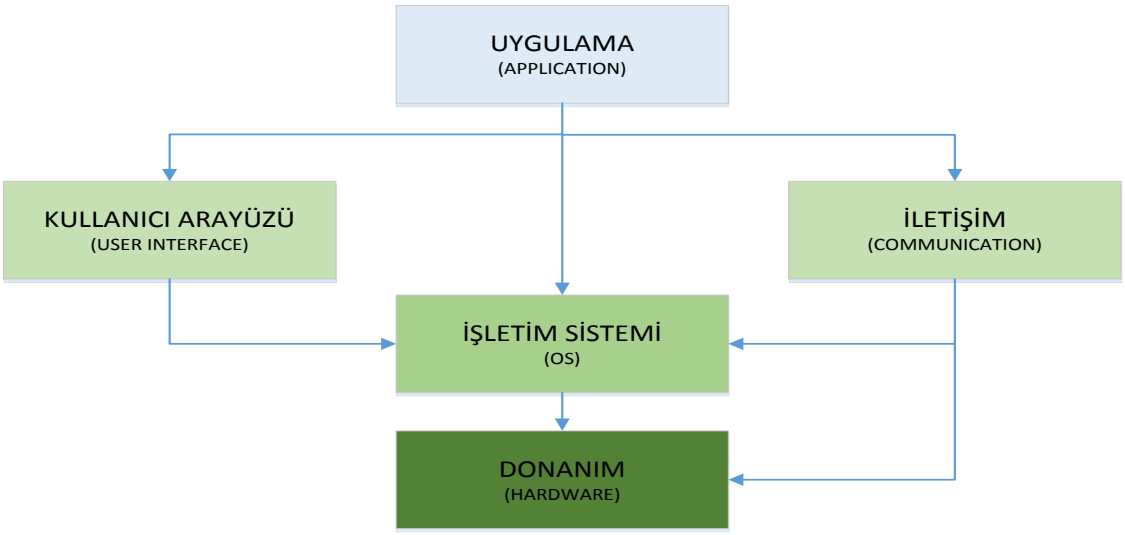
Şekil 3-2: a) Çok katmanlı mimari, b) 3-katmanlı web mimarisi c) OSI İletişim mimarisi

Bu yapıda haberleşme iki yönlü doğrusal olarak gerçekleşir. İletişim üst katmandan alt katmana doğru veya alt katmandan üst katmana doğru gerçekleşir ve genelde süreç en üst katmandan tetiklenir. Çok katmanlı mimari güvenlik ve soyutlama düzeylerinde geliştiricilere başarılı bir protokol sunar. Bu bağlamda birçok süreçte standart haline gelmiş çözüm yaklaşımlarına temel sağlamış bir örüntüdür. Web mimarilerinde kullanılan üç katmanlı mimari ve bilgisayar iletişim standardı olan OSI Katman yaklaşımları Şekil 3-2 ile gösterilmiştir.

Gerçek zamanlı sistemlerde; duyarga ve işletici gibi G/Ç aygıtlarının çalıştığı fiziksel katman, işlemci ve bellek gibi elektronik bileşenlerinin bulunduğu donanım katmanı ve donanım katmanı üzerinde soyut düzeyde çalışan yazılım katmanı olmak üzere üç temel katmandan söz edebiliriz. Bu katmanlar sistemin işlev ve özelliklerine göre başta yazılım katmanı olmak üzere, kendi içinde alt katmanlara ayrılabilir. Bununla birlikte, gerçek zamanlı sistemlerde etkinlik ve güvenlik diğer bilgisayar sistemlerine göre daha yüksek önem arz etmektedir. Çok katmanlı mimaride katmanlar arası sağlanan soyutlama ve güvenlik, mimari örüntü yaklaşımlarının gerçek zamanlı sistemler için yazılım ve sistem düzeyinde uygun ve kullanışlı olmasını ve gerçek zamanlı sistemlerde tercih edilmesini sağlamıştır.

Gerçek zamanlı sistemlerde mimari tasarım yapılırken duyar ve uyarıcı ve donanım gibi katmanların birbirinden ayrılarak katmanların oluşturulması beklenen bir durumdur. Bununla birlikte GZS'de bu mimari örüntüsü yazılım katmanlarının oluşturulmasında da tercih edilmektedir. Bu yaklaşımda birlikte çalışan yazılım modülleri, işlev ve hizmet gruplarına göre katmanlara ayrılırlar ve her katman alt katmanından hizmet alırlar. En alt katman yazılım sistemi ayrıca donanım ve fiziksel aygıtların takibi ve yönetimi hizmetini de sağlar.

Gerçek zamanlı sistemlerde kullanılan Beş katman örüntüsü katmanlı mimari yaklaşımına benzer bir örüntüdür [22]. Bu yaklaşımda sistem donanım, işletim sistemi, iletişim, kullanıcı ara yüzü ve uygulama olmak üzere beş katmana ayrılır. Fakat diğer katmanlı mimarilerde olduğu gibi katmanlar arasında hiyerarşik protokol bulunmaz. Diğer katmanlar da birbirleriyle iletişim kurmaktadırlar. Bu mimari Şekil 3-3 ile gösterilmiştir.



Şekil 3-3: Beş katmanlı mimari

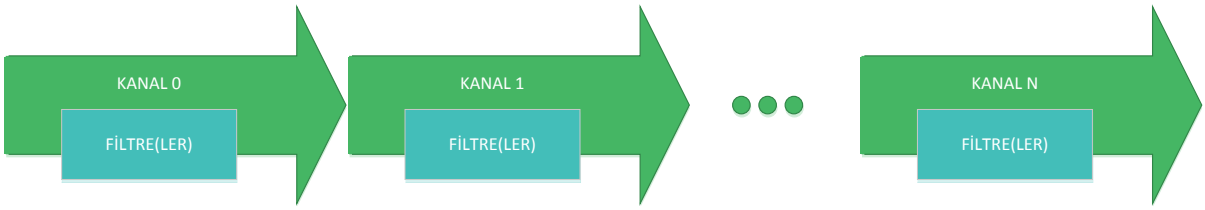
3.1.2. Kanal örüntüsü

Modern bilgisayar sistemlerinin, diğer bilgisayar sistemleri ile iletişime geçmeden tek başına çalışması olası değildir. Günümüz bilgisayar sistemlerinin yönettiği iş akışlarının büyük bir çoğunluğu, birbirleri arasında kurulan farklı iletişim ağları üzerinden veri iletişimi sağlamaları temeline dayanır. Bununla birlikte bilgisayar sistemlerinin kendi içindeki bileşenleri, tanımlanan veri yolu yapıları üzerinden sürekli olarak birbirleriyle veri alışverişi yapmak zorundadırlar. Bu bileşenler sistem üzerindeki farklı işlevlere sahip donanım bileşenleri olabileceği gibi, eş zamanlı çalışan birden fazla yazılım bileşeni de olabilir. Bu bağlamda iç ve dış sistemlerle veri iletişiminin bilgisayar sistemlerinin temel yapılarından birini oluşturduğu söylenebilir.

Kanal terimi, bilgisayar sistemleri bağlamında bir girişten gelen veriyi bir çıkışa taşıyan/ileten bir hat olarak tanımlanabilir [22]. Bilgisayar sistemlerinin iç ve dış sistemlerle olan iletişimlerinin bütünü söz konusu kanallar üzerinden sağlanır. Kanallar yazılım düzeyinde tanımlanmış bir veri yapısı veya fiziksel olarak var olan elektronik bir bileşen olabilir. Bununla birlikte kanallardaki girdi ve çıktılar bir bellek alanı veya diğer bir kanalın girdisi olabilir. Bu yaklaşım birden fazla kanalın birbirlerine bağlanabilmesini olanak sağlar. Birden fazla kanalın birbirlerine bağlanması ardıl düzen olarak da tanımlanır.

Kanallar kullanıldıkları iş akışı özellikleri kapsamında, gelen veriyi doğrudan veya bir takım işlemlerden geçirdikten sonra çıkış noktasına aktarabilirler. Kanallarda uygulanan veri işleme işlemleri filtre olarak tanımlanırlar. Filtreler kanallarla benzer şekilde

birbirlerine ardıl düzen işleme mantığı ile bağlanabilirler. Bu bağlamda her kanalda birden fazla filtre bulunabilir. Birden fazla kanalın birlikte çalıştığı sistemlerde her bir kanalın girdisi, kendinden bir önceki kanalın çıktısı olmaktadır. Genelleştirilmiş bir kanal yapısı Şekil 3-4 ile gösterilmiştir.

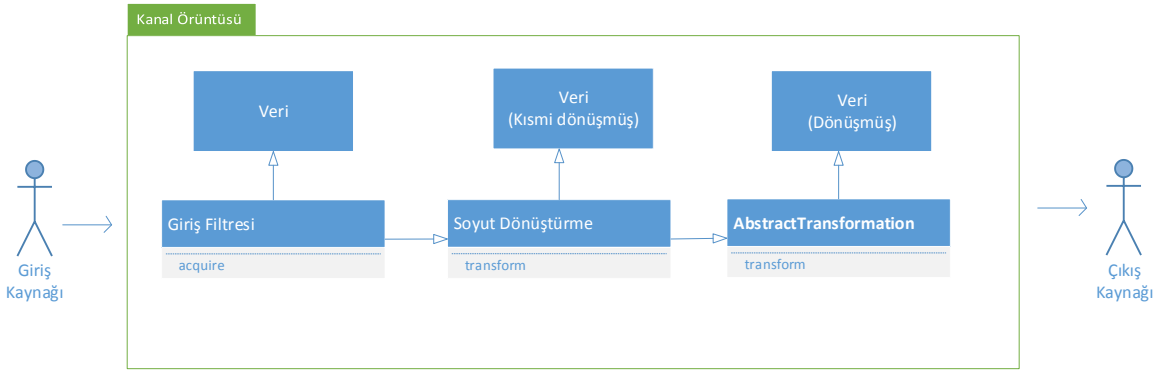


Şekil 3-4: Genelleştirilmiş kanal yapısı

Kanal örüntüsü kanal yapısı üzerine geliştirilmiş bir mimari örüntüdür. Bu örüntüde sistem mimarisi bir grup çıktıyı alıp işledikten sonra bir grup çıktıya yönlendiren bir ardıl düzen işleme gibi çalışır. Gerçek zamanlı bir sistem kapsamında, girdiler izlediği çevreden gelen veriler; çıktılar ise uyarıcılar veya verileri aktaracağı uzak bir kaynak için kullandığı bir iletişim kanalı olacaktır.

Kanal mimari örüntüsünde ardıl düzen işleme ile birbirlerine işledikleri verileri aktaran kanalların art arda bağlanması ile oluşturulur. Bu bağlamda girdiler ilk kanalın girdileri olurken çıktılar ise son kanalın ürettiği çıktılar olacaktır. Ara kanallarda üretilen girdi ve çıktılar ara verileri içerir. Dolayısıyla her bir kanalın kendisine ait verisi bulunur. Söz konusu veri kendisine gelen verinin ilgili kanalın işleminden geçtikten sonraki sürümünü kapsar. Kanal örüntüsü UML çizimleri

Şekil 3-5 ile gösterilmiştir [22].



Şekil 3-5: Kanal örüntüsü UML diyagramı

Kanal örüntüsü veri akışı olan sistemlerde sıkça kullanılan bir mimaridir ve gerçek zamanlı olmayan sistemlerde de tercih edilmektedir [22]. Bununla birlikte gerçek zamanlı veri akışı işleme sıkça ihtiyaç duyulan bir durumdur. Bu bağlamda önerilen örüntünün ağ akışı işlemleri, imge, ses ve görüntü işleme gibi gerçek zamanlı akış işlemlerinde kullanılması uygundur. Bununla birlikte bir grup alıcı tarafından üretilen verinin farklı katmanlarda işlenerek izlendiği sistemlerin (Elektrokardiyografi cihazları gibi) bu mimari ile kurulması tercih edilebilir.

3.1.3. Mikro çekirdek örüntüsü

Gerçek zamanlı sistemler (GZS), diğer bilgisayar sistemleri gibi, en temel düzeyde donanım, yazılım ve iletişim(ağ) katmanlarından oluşmaktadır. Donanım katmanı GZS için duyar ve uyarıcıların bulunduğu fiziksel iletişim katmanı olarak alt katmana ayrılabilir. Bu katmanların her biri sistemler arasında değişiklik gösteren mimari ve örüntü özellikleri barındıran birer alt sistemdir.

Yazılım katmanı işlev ve hizmete göre ayrılmış farklı katmanları kendi içinde barındırır. Bu katman temel düzeyde iletişim, işletim sistemi ve uygulama katmanları olarak tanımlanabilirler [22]. Gerçek zamanlı sistemler için işletim sistemi katmanı, tüm bilgisayar sistemlerinde ihtiyaç duyulan genelleştirilmiş kaynak yönetimi ve hizmetleri sağlayan hizmet katmanıdır. Tüm GZS işletim sistemi hizmetlerini sağlayan bir katmanı kendi içinde barındırır. Bu katman proje ölçeği ve ihtiyaçlar doğrultusunda geliştiriciler tarafından geliştirilebilir veya hazır ticari veya açık kaynak çözümlerle giderilebilir.

Mikro çekirdek örüntüsü, geliştirilen sistemin başta işletim sistemi olmak üzere bir grup hizmetin hazır sunulduğu bir katman üzerine tasarlanması ve kurulması çözümünü sunar [22]. Bu yaklaşımda bu hizmeti sağlayan katman mikro çekirdek olarak tanımlanır. Mikro çekirdek yaklaşımında geliştiricilere bir *platform* sağlanır. Sağlanan platform geliştiriciye sistem kaynakları ve görev yönetimi başta olmak üzere sağladığı hizmetleri bir ara yüz olarak sunar. Geliştirici, tasarladığı GZS bu platform üzerine uygulama geliştirme yaklaşımı ile inşa eder.

3.1.4. Bileşen tabanlı sistem örüntüsü

Sistemleri tasarlar ve geliştirirken, sistemi birbirini kapsayan/içeren bileşenler topluluğu olarak ele almak bilinen bir yaklaşımdır. Bu yaklaşımda sistem işlevlerine ve özelliklerine göre ayrılmış alt sistemlerden oluşur. Bu yapıda her alt sistem, belli bir işlev doğrultusunda bir araya getirilen alt bileşenlerin örgütsel olarak bir araya gelmesiyle oluştuğu özyineli bir yaklaşım ile oluşturulur.

Bileşen veya kapsayıcı yaklaşımının, sistem tasarımı, geliştirilmesi ve bakımı noktalarında geliştirici ve sistem mühendislerine sağladığı avantajları bulunmaktadır. Öncelikle sistemin alt bileşenlerinin bir araya gelmesi ile oluşması yaklaşımı, tasarım süreçlerini kolaylaştırır. Bununla birlikte bileşenler giriş/çıkış protokolleri bağlamında eş zamanlı olarak tasarlanarak son tasarımda bir araya getirilebilir. Dolayısıyla koşut olarak sürdürülen tasarım süreçlerinin kısalacağı da açıktır. Bu yaklaşımlar geliştirme süreçleri için de geçerlidir. Dolayısıyla benzer avantajlar bu süreçlere de yansıyarak daha yüksek bir toplu fayda sağlamaktadır.

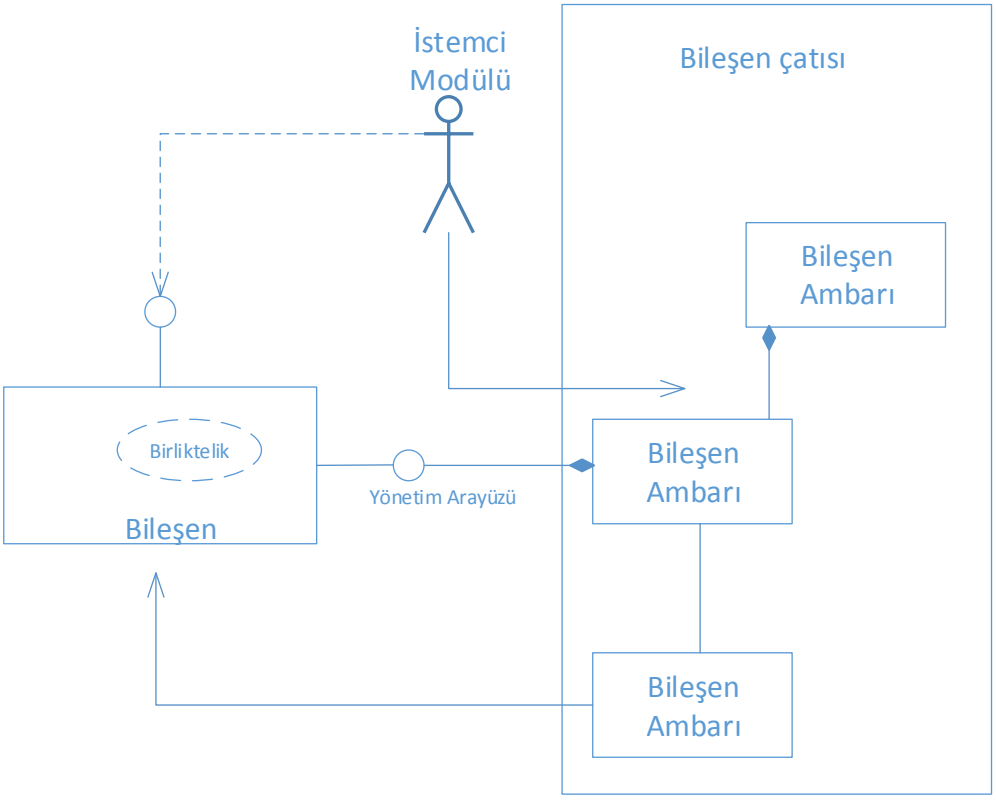
Sistemler devreye alındıktan sonra geliştirme ve bakım süreçleri devam etmektedir. Bakım süreci sistemin aktif ve istenilen davranışı sağlaması için gerekli teknik destekleri tanımlayan süreçtir. Alt sistemlerin bir araya gelerek oluşturduğu sistemlerin bakımı için süreçler için benzer avantajlar bulunmaktadır. Bu sistemlerin alt sistemleri için farklı personel grupları sorumlu atanarak bakım süreçleri dağıtılabilir. Bununla birlikte alt sistemlerde sorun oluştuğunda, ilgili alt sistem veya bileşenin değiştirilmesi veya düzeltilmesi sistemin ayağa kalkması için yeterli olacaktır.

Bileşen tabanlı sistem tasarımları standart haline gelmiş bir yaklaşımıdır. Günümüzde sadece bilgisayar sistemleri değil; çalışan veya tasarlanan sistemlerin bütününün bu kapsamda gerçekleştiği söylenebilir. Bileşen tabanlı sistemler bu yaklaşımı temel alan bir yazılım mimarisi örüntüdür. Bu örüntüde, sistem üzerinde belli işlevleri sağlayan yazılım

bileşenleri tanımlanmıştır. Bu bileşenler, çalışma zamanında ihtiyaç duyuldukça sistem tarafından oluşturulur veya devreye alınır. Bileşenin hizmeti sonlandığında ilgili bileşen sistem tarafından yok edilir veya devre dışı bırakılır. İlgili bileşenin yükleneceği veya oluşturulacağı bileşen maliyetleri doğrultusunda tasarım anında belirlenmektedir.

Bileşen tabanlı sistemlerde yazılım bileşenleri bileşen çatısı tarafından yönetilir [22]. Bileşen çatısı kendi içinde bileşen yükleyici, bileşen yöneticisi ve bileşen ambarı bileşenlerini içerir. Bileşen ambarı aktif bileşenlerin adres ve durum bilgilerinin tutulduğu yazılım bileşenidir. Ambardaki bileşenler için gelen isteklerin yönetilmesi, gerektiğinde yaratılması veya yok edilmesi bileşen yöneticisi tarafından sağlanır. Bir bileşene ihtiyaç olduğunda ilgili bileşen yüklenmesi veya yaratılması bileşen yükleyici tarafından gerçekleştirilir.

Bileşen tabanlı örüntü istek ve isteklere bağlı yazılım bileşenlerinin aperiyodik çalıştığı ve çalışma anında hizmet sürelerinin değişiklik gösterdiği senaryolar için kullanılabilir bir örüntü olduğu; Bu örüntünün özellikle yumuşak gerçek zamanlı sistemler için kullanılma alanlarını daha geniş olduğu söylenebilir. Bileşen tabanlı örüntü yaklaşımını tanımlayan UML çizimi Şekil 3-6 ile gösterilmiştir.

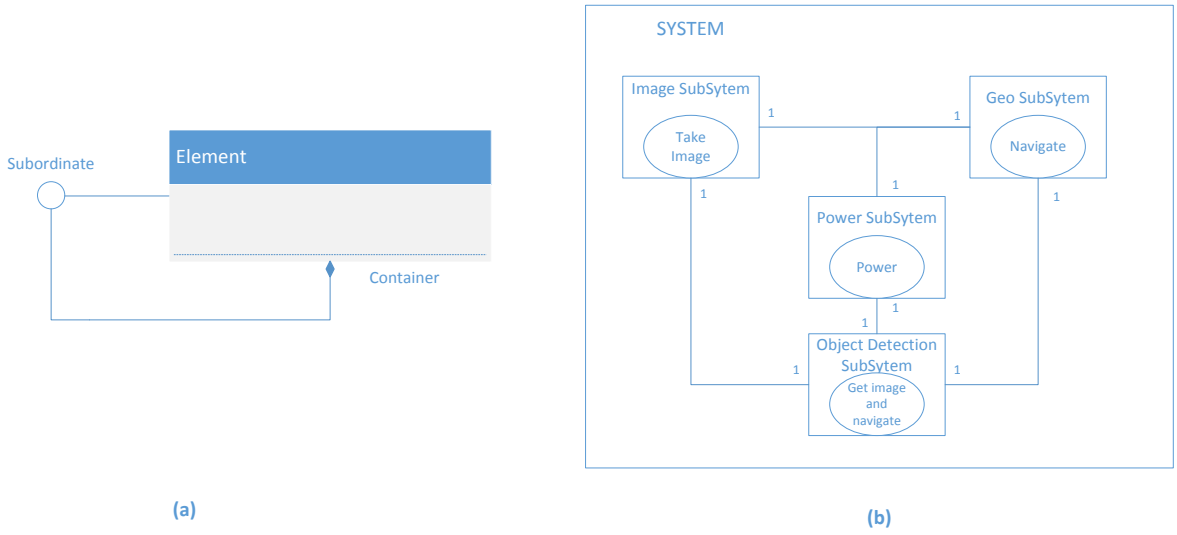


Şekil 3-6: Bileşen tabanlı mimari örüntü UML çizimi

3.1.5. Özyineli kapsayıcı örüntüsü

Sistemlerin karmaşıklıklarının artması, bu sistemlerin tasarımı ve gerçekleştirimi için süreçlerin geliştirilmesine sebep olmuştur. Karmaşık sistemlerin geliştirilmesinde en bilinen yaklaşımlardan biri sistemi daha az karmaşık alt sistemlere ayırarak, her alt sistemi kendi içinde ele almaktadır. Bu yaklaşımda her alt sistem ayrı bir sistem gibi düşünülür ve alt sistem bu sisteme ait girdi ve çıktılar bağlamında bağımsız bir sistem gibi tasarlanarak gerçekleştirilir. Bu süreç özyineli bir süreçtir ve ayrılan alt sistemler ihtiyaç doğrultusunda kendi içinde alt sistemlere ayrılarak süreç tekrarlanabilir. Özyineli geliştirme sürecinde her bir sistem tanımı basit tanımlı alt sistemlerin birleşiminden oluşur. Sistemlerin alt detaylara ilerledikçe karmaşıklaşır ve daha detaylı bileşenleri içeren soyut varlıklara evrilir.

Özyineli kapsayıcı örüntüsü temelinde özyineli geliştirme süreçlerini barındırır. Bu mimari örüntüde sistem alt sistemlere ayrılır. Her sistem birden fazla alt sistemin bir araya gelmesinden meydana gelir. Bu örüntüde alt sistemleri içeren her bir sistem kapsayıcı; sistemlerin kapsadığı alt sistemler ise element olarak tanımlanırlar [22]. Özyineli kapsayıcı örüntüsü UML çizimi ve örnek bir özyineli sistem çizimi Şekil 3-7 ile gösterilmiştir.



Şekil 3-7: (a) Özyineli kapsayıcı UML çizimi (b) Örnek özyineli kapsayıcı sistem

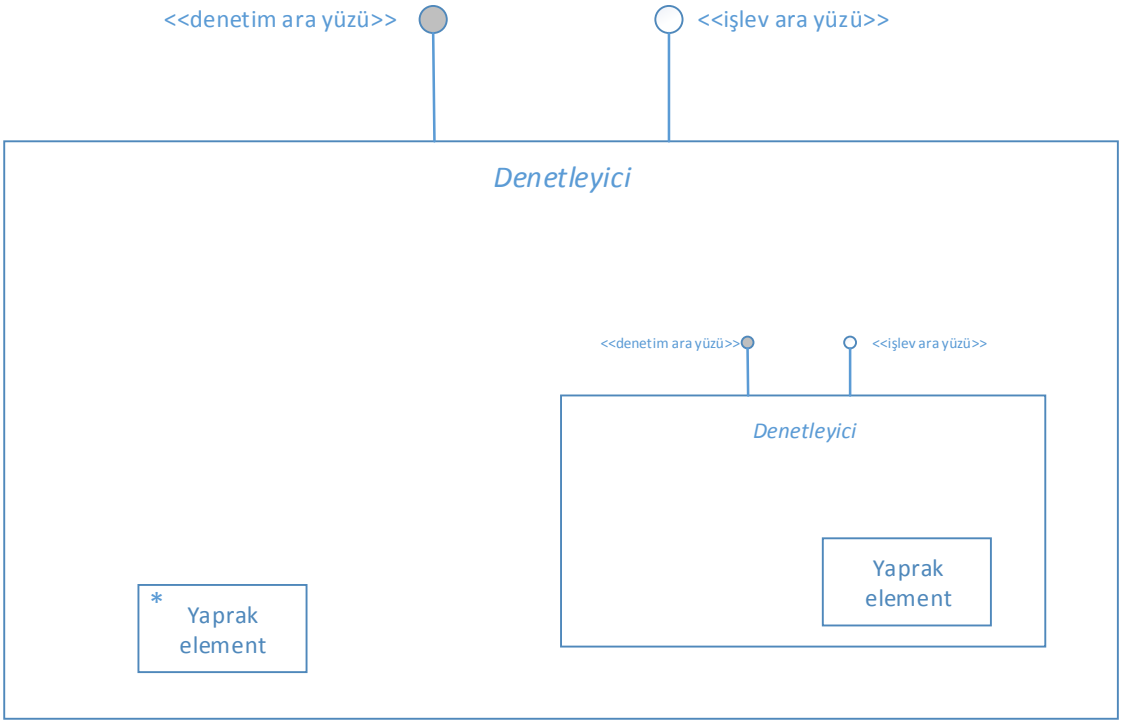
Özyineli kapsayıcı tasarımı bir sistem mimarisinden çok bir tasarım yaklaşımı örüntüsü olarak tanımlanabilir. Bu bağlamda bu örüntünün sistem tasarımında kullanılması ve sistem gerçekleştirilmesinde farklı örüntülerin kullanılması düşünülebilir.

3.1.6. Hiyerarşik denetim örüntüsü

Gerçek zamanlı sistemlerde (GZS) herhangi bir katmanda veri tutarsızlığı olması durumunda ciddi can ve mal kaybına neden olacak iş alanlarında kullanılmaktadırlar.

Dolayısıyla, GZS için verinin doğruluğu ve tutarlılığı, diğer bilgisayar sistemleriyle karşılaştırıldığında çok daha yüksek bir öneme sahiptir. Bu durum GZS için üst ve alt düzeyli bütün katmanlarda güvenlik yaklaşımlarının oluşmasını ve uygulanmasını zorunlu kılmaktadır. Kritik süreçlerde çalışan GZS, izledikleri çevreden kendilerine veri sağlayan alıcılar başta olmak üzere, veri üreten her bir bileşeni için tutarlılık denetimlerini sürekli olarak yapmak durumundadırlar.

Hiyerarşik denetim örüntüsü, bahsedilen güvenlik sorunlarına getirilmiş bir çözüm yaklaşımıdır. Bu mimari örüntüde, bileşen/kapsayıcı yaklaşımlara benzer biçimde, alt sistemler alt bileşenlerin bir araya getirilmesinden oluşur ve sistem tüm alt bileşenleri kapsayan en üst düzey alt sistemdir. Bu yaklaşımda her bir bileşen alt sistem, kendisini kullanan diğer sistemlere iki ara yüz sağlar: denetim ve işlev ara yüzleri[22]. Denetim ara yüzü, bileşen veya alt sistemin hizmet kalitesi doğrultusunda kesinlik, tutarlılık gibi denetimlerinin yapılmasını sağlar. Bu denetimler bileşeni kullanan üst sistemler tarafından sürekli olarak tekrarlanır. İşlev ara yüzü ise bileşenin kendisinden beklenen işlevleri sunmasını sağlar. Hiyerarşik örüntünün temel düzeyde UML çizimi Şekil 3-8 ile gösterilmiştir.



Şekil 3-8: Hiyerarşik Denetim UML Çizimi

3.2. Hizmet yönetimi yaklaşımına göre sistemler

Gerçek zamanlı sistemlerde (GZS) ve bu sistemlerde çalışan işletim sistemlerinin en temel görevi sistem kaynaklarını doğru şekilde yöneterek tüm görevlerin son zaman sınırı öncesi tamamlanmalarını sağlamak veya hedeflemektir. GZS bu süreci diğer bilgisayar işletim sistemleriyle benzer bir yaklaşımla, sistem kaynaklarını bekleyen görevler arasında atanma sürecini yöneterek sağlarlar. Çalışan ve bekleyen görevler arasında sistem kaynaklarının atanması sürecinin yönetimi görev yönetimi olarak tanımlanır.

GZS görev yönetimi yaparken periyodik veya aperiyodik olarak sistem kaynaklarını hangi görev(ler)e atayacağına karar verir. Bu süreç sistemin bütün çalışma zamanı boyunca devam eden tekrarlı bir süreçtir. Sistemlerin bu karar verme anları görev zamanlama olarak tanımlanır. Bu planlama anlarında sistem hangi görevlerin çalışma veya bekleme durumlarına geçeceğini belirler. Bu bağlamda sistem hali hazırda çalışan görev veya görevleri bekleme kuyruğuna alarak kaynaklarını başka görevlere tahsis edebilir.

GZS'de planlama tetiklemeleri temel olarak iki grupta toplanabilir: Olay odaklı sistemler ve zaman odaklı sistemler.

3.2.1. Olay odaklı sistemler

Gerçek zamanlı sistemlerin alıcıları vasıtasıyla izlediği çevre ve ortamlarda veya sistemin kendisinde oluşan kayda değer değişimler olay olarak tanımlanır. Bu değişimler iş akışını etkileyecek değişimler olarak tanımlanırlar ve olayların oluşumlarında sistemin bir tepki vermesi beklenir. Olay odaklı sistemlerde, sistem her olay oluştuğunda bir planlama süreci işleterek görevler arasındaki sistem kaynak atamalarını yeniden organize eder. Bu tür sistemlerde sistemin hangi olaya atanmış göreve öncelik vereceğinin belirlenmesi için olaylar arasında bir öncelik sistemi çalıştırılır. Öncelik belirleme mekanizması durağan veya devingen olabilir.

Olay odaklı sistemlerde, oluşan olaylarda sistem görevlerinin nasıl tetikleneceğinin belirlenmesinde kullanılan birçok strateji bulunmaktadır. Bu stratejiler aşağıda belirtilen grupta toplanabilir [22]:

- **Tekli olay yaklaşımı:** Bu yaklaşımda oluşan her bir olaya için bir iş parçası oluşturularak atanır ve olaya adanmış hizmet tamamlanınca bu görev yok edilir. Bu yaklaşım küçük boyutlu sistemlerde veya olayların sık olduğu ama son işletim zamanı kısa olduğu senaryolarda tercih edilebilirler.
- **Kaynak tabanlı yaklaşım:** Bu yaklaşımda her bir kaynak için bir görev atanır. Kaynaklar sistemin yönettiği çevrede bulunan veya sistem içinde konumlandırılmış, bir veya birden fazla alıcı üzerinden takip edilen bir olgudur. Kaynağa atana görev ilgili kaynaktan oluşan tüm olayları dinleyerek oluştuğunda ele alarak hizmet verir. Bu yaklaşımda, ilgili görevler sistem ile başlayarak bütün çalışma zamanı süresince hayatta kalırlar. Bu yaklaşım basit bir tasarımı olan ve az kaynak içeren sistemler için uygun bir modeldir[22].
- **Alan *(Domain)* tabanlı yaklaşım:** Kaynak tabanlı yaklaşıma benzer bir yaklaşımdır fakat bu stratejide sadece bir kaynağa değil, sistemin iş mantığında yer alan her bir alana bir görev atanır. Bu görevler atandıkları alana bağlı biçimde birden fazla kaynak /alıcıyı ve takip eder ve bu sürece bağlı uyarıcıları yönetirler. Bu yaklaşım çok sayıda kaynaktan sürekli bilgi geldiği fakat gelen bilgilerin tek bir görevce işlenebilir boyutta olduğu sistemler için tercih edilebilir.
- **Hedef nesne yaklaşımı:** Alan tabanlı yaklaşıma benzer bir yaklaşımdır. Fakat bu yaklaşımda sistemde çalışan kritik sürece atanmış özel nesnelere de ayrılmış görevler bulunur. Bu yaklaşım sunucu tabanlı sistemlerde de sıkça kullanılan bir yaklaşımdır. Alan tabanlı yaklaşıma göre daha soyut bir yaklaşımdır ve belli bir iş mantığı olan sistemlerde tercih edilebilir.

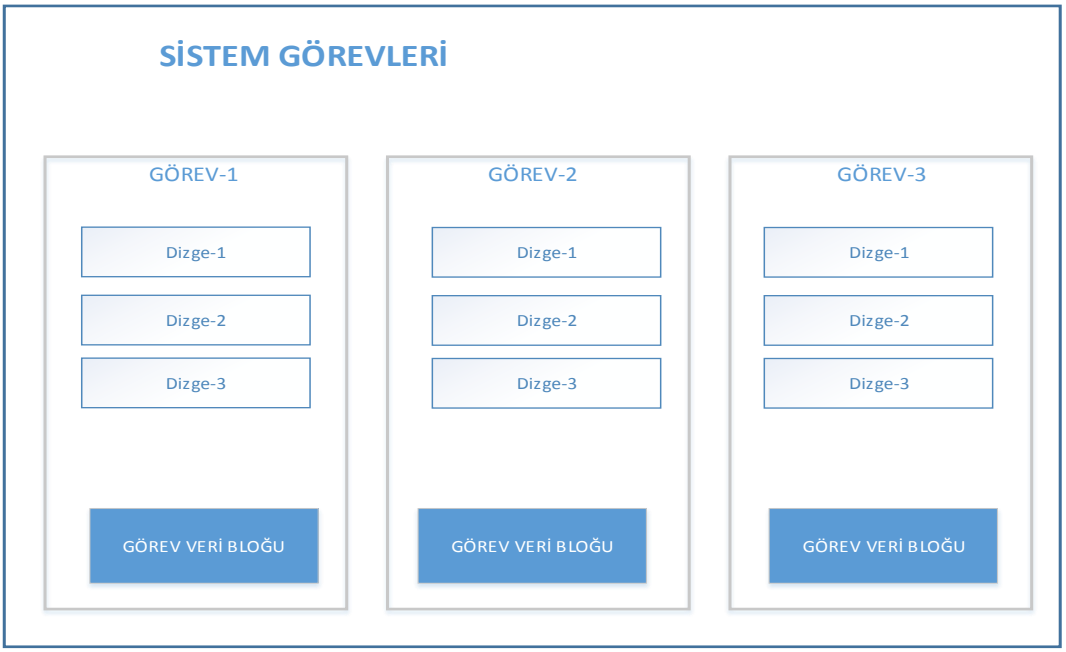
3.2.2. Zaman odaklı sistemler

Zaman odaklı sistemlerde görevlerin planlanması ve işletilmesi belli bir zaman rutini içinde gerçekleştirilir ve bu süreç bütün çalışma zamanı boyunca periyodik olarak tekrar eder. Bu sistemlerde sistemin saat vuruşları karar verme ve hesaplama süreçlerinde sistemdeki görevler için genelleştirilmiş temel zaman birimleri kullanılır. Söz konusu zaman birimleri çerçeve olarak tanımlanır [22]. Çerçeveler sistem saat vuruşlarını temel alırlar ve çerçeve başlangıç ve bitiş zamanları sistem saat vuruşlarıyla uyumludur. Sistemin görev planlamaları her çerçeveden veya toplu planlanan bir çerçeve grubundan önce gerçekleştirilir. Çalışan bir küçük veya büyük bir çerçeveden sonra yeni görevler anahtarlanabilir veya mevcut görevlere devam edilebilir.

3.3. Görev işletimi yaklaşımına göre

Gerçek zamanlı bir sistemde farklı süreçlerden sorumlu birden fazla görev bulunur ve bu görevler arasında başta işlemci olmak üzere sistem kaynakların etkin biçimde paylaştırılarak görevlerin sisteme zarar vermeden doğru biçimde çalışması sistemin temel hedefidir. Görevlerin zaman ve kaynak olarak yönetilerek doğru çalışmalarını sağlamaya çalışmak ve zamanında tamamlanamayan görevlerin sisteme zarar vermesini engellemek gerçek zamanlı sistemlerde görev işletimi katmanı tarafından sağlanır. Görev işletimi, gerçek zamanlı sistemin karmaşıklık düzeyine göre işletim sistemi düzeyinde veya sistem üzerinde işletim sistemi işlevini sağlayan bir çekirdek katman görev işletimi tarafından sağlanabilir [3].

Gerçek zamanlı sistemdeki görev, işletim sistemindeki görev tanımına benzer şekilde sistem üzerinde çalışan programları tanımlar ve görevler işletim sistemi düzeyinde zamanlanabilir mantıksal birimlerdir. Sistem üzerindeki her bir görevin kendisine ait bir kimliği, veri yapısı, yürüttüğü bir program bloğu ve sistem tarafından anlık işletim durumunu gösterir (çalışıyor, durdu, hazır vb.) bir durum bilgisi bulunur. Gerçek zamanlı sistem üzerindeki bir görevin bunlara ek olarak, öncelik tabanlı bir sistemde öncelik bilgisi ve zaman sınırı bulunur. Görevler kendi içinde eş zamanlı çalışan alt programlara ayrılabilirler ve görevlere göre daha hafif görev parçacıklar iş parçası olarak tanımlanırlar. İş parçalarının kendilerine ait veri blokları bulunmaz ve içinde konumlandıkları görevlerin veri yapılarını kullanırlar ve aynı görev üzerindeki kaynaklar üzerinden paylaşım yapabilirler. Bahsedilen görev yapısı ile gösterilmiştir.



Şekil 3-9: Sistem görevleri ve iş parçacıkları

Gerçek zamanlı sistemlerde görev işletimi sağlayan katmanlar, dört temel işlevi sağlarlar: görev zamanlama, görev yönlendirme, görevler arası iletişim ve eş zaman uyumlama [3]. Çok görevli bir gerçek zamanlı bir sistemin görev zamanlama ve yönlendirme / anahtarlama işlevlerini sağlaması gerekir. Bununla birlikte bahsedilen diğer işlevler sistem gereksinimlerine ve karmaşıklığına bağlı olarak farklı katmanlara veya çok katmanlı görev yönetimi yazılımlarına ihtiyaç duyar.

Gerçek zamanlı sistemlerde görev işletilmesi beş temel gruba ayrılmaktadır: Seçim tabanlı, sadece kesilmeli, öncelik ele geçirme tabanlı, hibrit ve zaman odaklı sistemler. Bu yaklaşımlar sırasıyla temel düzeyde ele alınacaklardır.

3.3.1. Seçim Tabanlı Sistemler

Seçim tabanlı sistemler, genel amaçlı işletim sistemlerinde giriş/çıkış programlamada kullanılan en eski yöntemlerden biridir. Bu yaklaşımda, işletim sisteminde periyodik olarak (örneğin 1sn periyotla) anahtarlanan bir görev bulunur. Seçim görevi tetiklendiğinde, sistemin bütün giriş/çıkış aygıt sürücülerini denetler ve işlenmesi gereken sürücülere hizmet verir. Verisi olmayan veya o denetim anında hizmete ihtiyacı olmayan sürücüler bir sonraki denetim anına kadar beklerler. Bu yaklaşım sistem kaynaklarının gereksiz yere tüketilmesinden günümüz sistemlerinde işlevini kesilme tabanlı yaklaşımlara bırakmıştır.

Seçim tabanlı gerçek zamanlı sistemler, işletim sistemlerinde bahsedilen G/Ç yönetimine benzer biçimde hizmet verirler. Bu sistemlerde, sistemin temel görevi diğer görev

verilerini denetler ve verisi bulunan veya hizmet için gerekli durumları tetiklenmiş olan görevleri anahtarlar / yönlendirir. Bu sistemler söz konusu denetim ve seçim görevinin sistem yaşam boyu süresince diğer görevleri denetlemesi, seçmesi ve ilgili görevi anahtarlaması yaklaşımına dayanır. Seçim tabanlı sistem yapısı orta ve büyük gerçek zamanlı sistemler için uygun olmayan bir yaklaşım olmasıyla birlikte, bu sistemler özellikle yoğun G/Ç işlemlerine dayalı küçük ve belirlenebilir sistemler için düşük maliyetli etkin bir çözüm yaklaşımı olarak görülmektedir [3].

3.3.2. Sadece Kesilmeli Sistemler

Kesilmeler günümüz bilgisayar sistemlerinin bütününde G/Ç yönetiminde kullanılan temel yaklaşımdır. Bu yaklaşımda, G/Ç aygıtı herhangi bir anda bir hizmete ihtiyaç duyduğunda, sistemde hizmet alan göreve ait program ve verisi yığıta atılır ve kesilme gelen G/Ç görevine hizmet verilir. Bu görevler genelde kısa süreli ve basit görevlerdir fakat sistemin devamlılığı ve kullanıcı ihtiyaçları doğrultusunda bu hizmetlerinin anlık olarak verilmesi gerekmektedir. İlgili kesilme görevi tamamlandıktan sonra, kesilme oluştuğunda hizmet alan görevin işlemine kaldığı yerden devam edilir. Sistem güvenliği ve bütünlüğü açısından bazı görevlerin kesintiye uğramaması gerekir ve bu görevler kesilemez görevler olarak tanımlanırlar. Bu görevler hizmet aldığında sistem kesilmelere kapatılır ve bu durumda kesilme hizmeti ilk kesilebilir hizmet anında anahtarlanır.

Sadece kesilme tabanlı gerçek zamanlı sistemlerde, tüm sistem görevleri birer kesilmeden oluşur ve kesilme oluştuğunda ilgili sistem görevine hizmet verilir. Kesilme görevleri zamana bağlı olarak oluşan periyodik görevler veya rasgele zamanlarda tetiklenen görevler olabilirler. Sistemin işlev sağlayan tüm görevleri birer kesilmeden oluştuğu için, sistemin temel görevi işlevi olmayan aptal bir görevdir ve bu görev aktif iken sistem kesilme görevleri için bekleme durumundadır.

Sistem üzerinde birden fazla kesilmenin olduğu bir sistemde herhangi bir kesilme hizmeti anında başka bir kesilme oluşabilir. Bu durumu yönetmek için kesilmeler arasında öncelik sıralaması bulunur ve herhangi bir kesilme hizmeti anında, hizmet alan kesilme görevinden daha yüksek öncelikli bir kesilme oluştuğunda, hizmet alan kesilme görevi kesintiye uğrar. Bu doğrultuda, işlemci yazmaçlarında ilgili hizmet görevine ait veriler yığıta atılarak yüksek öncelikli kesilme görevine hizmet verilir. Yüksek öncelikli kesilme hizmetinin tamamlanması sonrasında kesilen göreve verilen hizmet tamamlanır.

3.3.3. Öncelik ele geçirmeli sistemler (*Preemptive priority systems*)

Gerçek zamanlı görevler tetiklendikten sonra tamamlanmaları için azami bir zaman limitine sahip görevlerdir ve bu görevlerin zamanında tamamlanmaması sistemin bütünlüğünü bozulmasına hatta gerçek zamanlı sistemde ve çevresinde ciddi zararlar oluşmasına neden olabilir. Bu bağlamda gerçek zamanlı görevlerin zamanında tamamlanması sistem bütünlüğü için en kritik etmendir. Gerçek zamanlı görevlerin tamamlanma süresi ve bu süre aşıldığında oluşacak sistem zararı düzeyi sistem görevlerine bağlı olarak değişiklik gösterir. Bazı görevlerin son limitlerinin kaçırılması görev düzeyinde zarar verirken, bazı görevlerin kaçırılması felaket düzeyinde sonuçları ortaya çıkarabilir. Bu bağlamda, bir sistem içinde hizmet veren gerçek zamanlı görevlerin sisteme göreli olarak öncelikleri veya birbirlerine göre aciliyet düzeyleri bulunur.

Öncelik ele geçirmeli sistemler, gerçek zamanlı görevleri öncelikleri ve zaman sınırları doğrultusunda yönetirler ve kesilme tabanlı sistemler gibi davranırlar. Bu sistemlerde görevler kesilme tabanlı sistemlerden farklı olarak kesilme oluşturmazlar. Öncelik tabanlı görevler periyodik, aperiyodik veya düzensiz olarak oluşabilirler ve işlenmeye hazır görev öncelik sıralı bir hazır görev kuyruğuna eklenirler. Sistem üzerinde bir görev hizmet verirken, daha yüksek öncelikli bir görev oluştuğunda ise, sistem kesilme oluşmasına benzer bir davranış göstererek hali hazırda hizmet verdiği düşük öncelikli görev hizmetini yarıda keserek yüksek öncelikli göreve anahtarlanır. Bu görev sonrası yarım kalan göreve devam eder veya daha yüksek öncelikli başka bir göreve anahtarlanır.

Öncelik tabanlı görev yönetiminde düşük öncelikli görevlerin sürekli yarıda bırakılması bu görevlerin sürekli yarıda bırakılmasına ve sonuçta düşük öncelikli görevlerin sürekli zaman sınırlarını kaçırmalarına neden olabilmektedir. Bu durum istenmeyen bir durumdur ve bu sorunun üstesinden gelmek için birçok öncelik yönetim yaklaşımı getirilmiştir. Bu yaklaşımlar, 3.4.1 bölümünde ele alınmıştır

3.3.4. Melez sistemler (*Hybrid systems*)

Melez sistemler, yukarıda bahsedilen kesilme tabanlı ve öncelik ele geçirmeli sistemlerin bir arada kullanıldığı veya genişletildiği sistem yaklaşımlarıdır. Bu yaklaşımlardan ilkinde, kesilme tabanlı ve öncelik ele geçirmeli sistemler bir arada kullanılır. Bu bağlamda, sistemde kesilme ile oluşan görevler ve zamansal olarak periyodik, aperiyodik ve düzensiz görevler bir arada çalışırlar ve kesilmeler ve görevlerinin hepsinin sisteme göreli bir öncelik tanımı bulunur. Sistem zamanındaki herhangi bir görev hizmeti anında daha

yüksek öncelikli bir kesilme veya gerçek zamanlı görev oluştuğunda, sistem öncelik ele geçirme davranışı ile hizmet verdiği görevi yarıda keserek yüksek öncelikli göreve anahtarlanır. Burada öncelik yönetimi, öncelik tabanlı sistemler için geliştirilen yöntemlerden biriyle yönetilebilmektedir. Bu tür sistemlerde bir diğer yaklaşım round-robin görev zamanlama yapan bir görev yönetimi ile öncelik ele geçirmeli sistemleri bir arada çalıştırmaktadır. Burada normal sistem görevleri ve düşük öncelikli gerçek zamanlı sistemler için zaman odaklı bir yönetim ile yürütülürken daha yüksek öncelikli görevler oluştuğunda bu görevlere anahtarlanarak çalışan sistem modeli kurulmaktadır [3].

Diğer bir melez sistem yaklaşımında, sistem kesilme tabanlı bir sistem ile aynı davranışı gösterir fakat kesilme tabanlı sistemlerden farklı olarak kesilme beklendiğinde görev işlevsiz bir görev yerine en düşük öncelikli görev çalışır. Bu görev çalışırken oluşacak herhangi bir kesilme daha yüksek önceliğe sahip olacağı için bu görev yarıda kesilerek sistem kesilme görevine anahtarlanır. Bu sistemlerde seçilen en düşük öncelikli görev genellikle sistem izleme görevleri veya günlük görevleri olarak seçilmektedir [3].

3.3.5. Zaman odaklı sistemler

Zaman odaklı sistemlerde tüm görevler periyodik görevlerden oluşur ve sistem üzerindeki tek kesilme periyodik zaman kesilmeleridir. Söz konusu zaman kesilmeleri periyodik olarak oluşan tek bir zaman bir kesilmesi olabilir ve bu sistemde her bir göreve veya görev parçacığına ayrılmış zaman bloğu eşit süreye sahiptir. Bir diğer yaklaşımda her bir görev kendisine ayrılan süreye bağlı olarak sistem zamanlayıcı kesilmesini kurar ve bu süre tamamlandığında diğer göreve anahtarlanır. Bu süreç tüm görevler tamamlandığına tekrar baştan tekrarlanarak periyodik olarak devam eder. Bu sistemlerde tek kesilme mekanizması zaman sayaçları olduğu için öncelik belirteci hangi görevin hangi sırada işletileceğine bağlı olarak sistem tasarımında belirlenir ve bu sistemlerde öncelik tabanlı sistemlerdeki gibi görevler işletilirken birbirlerinin çalışmalarını kesmezler.

Bir diğer zaman odaklı sistem örneğinde ise tüm sistem görevleri sistem yaşam süresi boyunca tekrarlanan bir döngü bloğu gibi davrandığı yaklaşımdır. Bu yaklaşımda, her bir döngü anında sistem görevleri belirlenen sırada işletilir ve tüm görev hizmetleri tamamlandığında döngü tekrar başa döner. Bu yaklaşımda sistemde zaman kesilmesi de bulunmaz ve sistem belirlenebilir olarak tekrarlı olarak çalışır. Zaman odaklı sistemler karmaşık ve çeşitli sistem görevlerinin yönetiminde yetersiz kalabilmesine rağmen,

sistemin basitliği ve tahmin edilebilir davranış örüntüsü yaklaşımının özellikle yüksek güvenlik gerektiren sistemlerde yoğun biçimde tercih edilmesini sağlamaktadır.

3.4. Görev Zamanlama (*Scheduling*)

Bilgisayar sistemlerinde, her görevin sistemin yazılımsal ve donanımsal kaynaklarını kullanarak tamamlanması beklenmektedir ve bu durum çoklu görevli sistemlerde sistem kaynaklarının görevler arasında paylaştırılmasını gerektirir. Gerçek zamanlı görevler için en belirgin özelliğinin *zamanlılık* olduğu ve gerçek zamanlı görevlerin teslim zamanı öncesi tamamlanması gerektiği düşünüldüğünde, gerçek zamanlı görevlerin zamanında tamamlanması için işlemci kaynağının bu sistemlerde etkin yönetilmesi gerekir.

Görev zamanlama, başta işlemci olmak üzere sınırlı sistem kaynaklarının sistem görevleri arasında etkin biçimde paylaştırılmasıdır ve bu paylaşım yapılırken sistem görevlerinin kısıt ve amaçları doğrultusunda tamamlanması hedeflenir. Bir görev zamanlama algoritması, herhangi bir görevin hangi zamanda çalışacağının kurallarını belirler [21]. Bir görev zamanlama algoritması, sistemdeki tüm görevlerin çalışarak kısıtları ve amaçları doğrultusunda tamamlanacağını garanti ederse, bu görev zamanlama algoritması uygulanabilir görev zamanlama algoritması olarak tanımlanabilir [20]. Bu bağlamda, gerçek zamanlı sistemler için geliştirilmiş bir uygulanabilir görev zamanlama algoritması, tüm görevlerin teslim zamanları öncesi tamamlanacağını garanti edecektir.

Gerçek zamanlı sistemler, hali hazırda çalışan görevden daha kritik bir görev çalışmaya hazır olduğunda gösterdiği davranışa göre iki gruba ayrılır. Ele geçirmeli sistemlerde daha kritik bir görev hazır olduğunda çalışan görevin çalışmasına son verilerek kritik olan göreve anahtarlanır. Ele geçirmeli sistemlerde görevler kesilmeye uğramazlar ve anahtarlama kararları her bir görevin çalışması son bulduğunda gerçekleştirilir.

Görev anahtarlama algoritmaları, karar verme odağı bağlamında üç gruba ayrılmaktadır: Öncelik odaklı, zaman odaklı ve kaynak odaklı yönetim algoritmaları [24].

3.4.1. Öncelik Odaklı (*Priority Driven*)

Gerçek zamanlı sistemler için geliştirilmiş bir görev anahtarlama algoritmasının temel hedefi, başta zor gerçek zamanlı görevler olmak üzere tüm görevlerin son teslim zamanları öncesinde tamamlanmalarını sağlamaktır. Gerçek zamanlı sistemler farklı amaç ve özellikteki gerçek zamanlı görevlerin bir arada çalıştığı sistemlerdir ve işletim anında farklı görevlerin teslim zamanlarının çakıştığı durumlar gerçekleşebilir. Bir teslim anı çakışması anında, sistemin tutarlılığı açısından zor görevin işletilmesinin ve yumuşak görevin teslim

anını kaçırması tercih edilebilir. Bu bağlamda, birlikte çalışan görevler arasında işletim önceliği kararı verilmesinde öncelik kavramı ortaya çıkmıştır.

Öncelik gerçek zamanlı sistem içinde çalışan görevler arasında göreli olarak görevin önemini tanımlayan bir sistem parametresidir. Öncelik odaklı bir görev zamanlama algoritması, herhangi bir çalışma anında işletilecek görevin belirlenmesine bu parametre üzerinden karar verir. Seçilen algoritma yaklaşımına göre, görevlerin öncelikleri durağan veya devingen olabilir. Durağan öncelik yaklaşımında, tüm görevlerin öncelik değerleri görev zamanlama öncesinde belirlenmiştir ve sistem çalışma yaşamı süresince değişmez. Gerçek zamanlı sistemlerde en bilinen görev anahtarlama algoritması Liu ve Layland tarafından geliştirilen *Rate Monotonic* (RM) algoritmasıdır [21]. RM algoritmasında, tüm görevler birbirinden bağımsız periyodik görevler olarak tanımlanırlar ve aperiyodik görevler maksimum beklenen zaman sınırı sürelerine göre düzensiz görevlere benzetilirler [25]. Bu yaklaşımda, görevlerin periyotları son teslim zamanları olarak ele alınır ve en düşük periyota sahip göreve en yüksek öncelik verilir.

Devingen öncelikli algoritmalarda, öncelik görevlerin değil görevlerin çalışan işlerine ait parametrelerdir. Devingen öncelik algoritmaları, iş seviyesi devingen ve kısıtsız devingen olmak üzere iki grupta toplanır [26].İş seviyesi öncelik yaklaşımında, görevlere ait sabit öncelikler çalışma zamanındaki etmenler doğrultusunda karar anında değişiklik gösterir. En çok bilinen devingen görev zamanlama algoritması En yakın zaman sınırı (*EDF*) algoritmasıdır. Bu algoritmada, her görevin sabit bir zaman sınırı vardır ve sistemin o anki zamanına göreli olarak karar anında teslim zamanı en yakın olan göreve kaynaklar atanır. Sınırsız devingen yaklaşımda, sistemdeki öncelik parametresi tanımında sabit hiçbir sınır yoktur ve öncelikler tamamen devingen olarak belirlenir. En az esneklik öncelikli algoritmasında, esneklik bir öncelik parametresi olarak tanımlanır. Esneklik eşitlik (3) ile hesaplanmaktadır.

$$esneklik = teslim\ limiti - (şimdiki\ zaman\ + gereken\ işlemci\ zamanı) \qquad (3)$$

Anlatılan görev anahtarlama yaklaşımlarının yanında, sabit ve devingen önceliklerinin farklı seviyelerde bir arada kullanıldığı karma yaklaşımlar bulunmaktadır. [27]'de, sabit öncelik sorunlarına çözüm getirmek üzere, ilk seviyede sabit bir öncelik ve ikinci seviyede ise devingen öncelik yaklaşımı olmak üzere iki seviyeli bir karar mekanizması kullanılmıştır. Bu yaklaşım en yüksek aciliyet olarak tanımlanmaktadır. Burada bahsedilen

popüler algoritmaların yanında, gerçek zamanlı sistemler üzerinde öncelik tabanlı görev anahtarlama için yapılmış birçok çalışma literatürde mevcuttur [9], [21], [28].

3.4.2. Zaman odaklı (*Time Driven*)

Zaman odaklı görev zamanlama, öncelik odaklı görev zamanlama yaklaşımlarına alternatif olan bir gerçek zamanlı görev zamanlama yaklaşımıdır. Bu yaklaşımda işlemci zamanı, sistem görevleri arasında daha önceden belirlenen statik bir protokol doğrultusunda dağıtılır ve her görevin kendisine ayrılan zaman aralığında işlemci kaynaklarına erişmesi sağlanır.

Bu yaklaşımda, görevler arasında değişken bir öncelik yaklaşımı bulunmaz fakat görevlerin çalışma zamanı öncesinde belirlenen önem ve ihtiyaçları doğrultusunda öncelikleri veya diğer görevlere göreli olarak işlemci kaynaklarına daha yoğun erişmesi sağlanabilir.

Zaman odaklı görev zamanlama iki şekilde gerçekleştirilebilir. İlk yaklaşımda, tüm görevler verilen bir öncelik bağlamında tek bir döngü gibi çalıştırılır. Bu görev anahtarlamada, bütün görevleri sıralı olarak içeren statik görev, sistem yaşamı boyunca tekrarlı olarak işletilir. Bu yaklaşım, tüm görevlerin periyodik olduğu gerçek zamanlı sistemlere kolaylıkla uygulanabilir. Döngüsel çalıştırma [22] ve zaman çerçeve örüntüsü [3] bahsedilen yaklaşıma dayanan görev zamanlama algoritmalarıdır.

Diğer yaklaşımda, işlemci sabit zaman aralıklarına bölünür ve her görev kendisine ayrılmış zaman aralıklarında, belirlenen zaman kadar işlem yapar. Bu yaklaşım yapı olarak işletim sistemlerinde yaygın kullanılan *round-robin* yaklaşımına benzerlik gösterir. Minimal zaman bölütleme *(MTS)* algoritmasında [20], işlemci zamanı ΔL uzunluğundaki zaman aralıklarına bölünür ve sistemdeki her görevin bu ΔL zaman aralığında kendisine ayrılmış işlemci zamanı bulunur. Hangi görevin ne yoğunlukta çalışacağı, sistem tasarımında belirlenen bazı parametrelerle belirlenir. Bu parametrenin yoğunluk olduğu yaklaşımda, ΔL uzunluğundaki bir zaman aralığında i. görev ($p_i = (C_i \,/\, T_i)$) olduğu durumda $p_i \Delta L$ süre işlemci kaynaklarına erişebilir.

Gerçek zamanlı sistemler çözüm getirdikleri çevreler doğrultusunda önceden tahmin edilir şekilde denetlenmeleri gereken sistemlerdir ve bu sistemlerde bir görevin zaman kısıtını aşması felakete sebep olabilir. Bu bağlamda belirlenebilirlik gerçek zamanlı sistemlerde en çok beklenen davranış olacaktır. Zaman odaklı sistemlerin tamamen tahmin edilebilir görev yönetimini sağlamaları ve tek veya çoklu işlemcilere uygulanabilir olmaları, bu

yaklaşımı gerçek zamanlı sistemler için uygun kılmaktadır. Bununla birlikte, yaklaşımın tamamen önceden tahmin edilebilir bir model sunması, devingen süreçleri olan olay odaklı sistemlere uyarlanmasını zorlaştırmaktadır.

3.4.3. Paylaşım Odaklı (*Share Driven*)

Paylaşım odaklı yaklaşımda, tüm görev zamanlama sistem kaynaklarının görevler arasında paylaştırılması odağına dayanır. Kaynak odaklı görev zamanlamada, her görevin ulaşmak istediği kaynaklar için bir paylaşım ağırlığı bulunur ve görev yönetimi kaynak paylaşım kanalları üzerinden sağlanır. Bu yaklaşımda, sistem üzerinde görevlerin kaynaklara erişim önceliği olmasına rağmen görevler arasında bir öncelik bulunmaz. Bu bağlamda görev anahtarlamada ön-alımsal davranış göstermeyebilir. Bununla birlikte, görevlerin son teslim zamanlarından önce tamamlanacakları garanti edilmeyebilir. Kaynak odaklı yaklaşım günümüzde işletim sistemlerinde kullanılan görev ve kaynak yönetimi algoritmalarıyla benzerlik gösterir ve kaynak yönetiminin görev önceliklerinden daha baskın olduğu yumuşak gerçek zamanlı sistemler için uygundur [28]–[30].

3.5. Duyarga ve İşleticiler (*Sensors and Actuators*)

Gerçek zamanlı sistemlerin genelleştirilmiş yaşam döngüsü, bulunduğu çevre sürekli biçimde izleyerek, istenilen amaç doğrultusunda çevresini değiştirmesi olarak tanımlanabilir [3]. Bu bağlamda, sistemler giriş çıkış aygıtlarının türü ve kullanımı açısından diğer bilgisayar sistemlerinden ayrılırlar. Gerçek zamanlı sistemlerin, denilen ile bulunduğu çevrenin farklı ortamlarından örneklemler almasını sağlayan elektronik, elektromekanik cihazlar duyarga olarak tanımlanırlar. Bunun yanında sistemin etrafındaki çevreye etki etmesini sağlayan mekanik, elektronik ve elektromekanik bileşenler ise işleticiler olarak tanımlanırlar. Örneğin, gerçek zamanlı bir sistem örneği olarak bir droid düşünüldüğünde, bu sistem, *proximity detection* duyargalarını kullanarak etrafındaki engelleri tarar hesaplamaları doğrultusunda rotasını belirler ve işleticilerini kullanarak hareket etmesini sağlayan motorları tetikler. Bu bağlamda, gerçek zamanlı bir sistemin yaşam döngüsü duyargaların okunması, hesaplamaların yapılması ve işleticilerin tetiklenmesi ve/veya göstergelerin güncellenmesi olmak üzere üç temel kısma ayrılabilir.

3.6. Çok görevli gerçek zamanlı görev zamanlama

Gerçek zamanlı sistemlerde çoklu görev ve çok işlemcilere görev dağılımı, gerçek zamanlı sistemlerde görev zamanlama konusunda yapılmış ilk çalışmalardan beri tartışılmaktadır

[31]. Yapılan çalışmalar, tek işlemcili mimariler için geliştirilmiş görev zamanlama algoritmalarının çok işlemcili mimarilere doğrudan gerçekleştiriminin etkin sonuçlar vermediğini göstermiştir [23]. Bu bağlamda, bu konuda daha gelişmiş görev zamanlama yaklaşımlarına ihtiyaç duyulmuştur ve en temel düzeyde, çok işlemcili sistemler için geliştirilen gerçek zamanlı görev anahtarlama yaklaşımı iki temel aşama gerektirir: görevlerin işlemcilere atanması ve atanan görevlerin ilgili işlemcilerde işletimi.

Periyodik görevler için geliştirilen çoklu görev zamanlama yaklaşımları iki grupta toplanmaktadır: Genel şema/düzen ve bölümleme [26]. Genel şema yaklaşımında, sistemdeki tüm görevler tek bir kuyruk üzerinden yönetilirler ve belirlenen öncelik yaklaşımı doğrultusunda sırayla işlemcilere atanırlar. Yapılan çalışmalar, genel şema yaklaşımının EDF veya RM ile çok işlemcili sistemlerde birlikte kullanıldığı sistemlerin etkin bir görev zamanlama sağlamadığını göstermiştir. Bununla birlikte, görevlerin de kendi içinde kuantum alt görevlere bölünerek her alt görevin bağımsız bir görev gibi anahtarlandığı PFair görev zamanlama algoritması, genel şema yaklaşımında kabul edilebilir sonuçlar vermektedir [32].

Bölümleme yaklaşımında, her bir görev çalışma zamanı öncesi belli bir işlemciye atanır ve sistem yaşam zamanı boyunca aynı işlemcide çalışır. Çok işlemcili mimariler, daha çok birbirinden bağımsız tekil bağımsız işlemciler kümesine evirildiği için, bölümleme genel şema üzerinde yararları oluşmaktadır. Bununla birlikte, bölümlemenin ikili-paketleme (ve görev ayırma gibi olumsuz etkileri de vardır [26]. Bazı yaygın bölümleme algoritmaları arasında ilk uyan (*FF*) ve en iyi uyan (*BF*) algoritmalarıdır [1]. FF yaklaşımında, hazır durumda olan görevler, dizinlenmiş bir işlemci grubundaki ilk uygun işlemciye işletilmek için atanır. BF yaklaşımında, hazır olan bir görev, atama sonrası en az sığası kalacak uygun işlemciye atanır. Burada bahsedilen sığa uygunluğu, seçilen görev zamanlama yaklaşımı bağlamında hesaplanan göreli bir değerdir. Bölümleme yaklaşımı, tek işlemcili sistemler için geliştirilmiş gerçek zamanlı görev zamanlama algoritmaları ile birlikte kullanılarak gerçekleştirilir. Bu gerçekleştirmelere örnek olarak RM-FF veya EDF-BF verilebilir.

Çok işlemcili gerçek zamanlı görev anahtarlama çalışmaları, çok çekirdekli sistemler veya dağıtılmış sistemler için yapılmış çalışmalar olarak iki gruba ayrılmaktadır. İlk çalışmalar, dağıtılmış sistemler için geliştirilmiştir ve bu yaklaşımda her düğümün yerel zamanlayıcısı bulunur. Sistem sınırlarını aşarak zamanında bitirilemeyeceği öngörülen görevler, başka düğümler gönderili. Bu yapı genel zamanlama yapısı kullanılarak sağlanabilir veya

düğümler birbirlerinin zamanlayıcı kuyruklarına erişebilir [33], [34]. Çok izlekli işlemcilerin yaygınlaşmasından sonra, sanal koşutlaştırma üzerine [35] ve ön bellek yönetiminini bütünleştirilmesi kullanılarak gerçek koşutlaştırma üzerinde görev yönetimi sağlanan çalışmalar yapılmıştır [25], [36], [37].[25]'te, yük sınırlı iş koşutlaştırma yaklaşımları, gerçek zamanlı genel şema görev zamanlama kullanılarak, çok işlemcili bir sisteme uygulanması üzerine çalışılmıştır.

Belirtildiği üzere, Pfair görev zamanlama yaklaşımında her görev kendi içinde çalıştırılabilir alt kuantum görevlere bölünerek her alt görev bağımsız bir görev gibi çalıştırılır. Ayrılan alt görevlerin bir kısmı arasında bağımlılık olacağı gibi, gerçek zamanlı görevler genelde birbirinden bağımsız çalışan görevler içerdiği için, bu alt görevlerin birçoğu eş zamanlı olarak çalıştırılabilirler. Bu tür yaklaşımlar, gerçek zamanlı sistemler üzerinde görev zamanlama yaklaşımlarının, çok görevli anahtarlamadan çok izlekli anahtarlamaya evirilmesine olanak vermiştir. Doğrusal çevrimsiz çizge (DAG) tabanlı sistemler, çalışacak görevleri bir çizge üzerinde tanımlar. Bu yaklaşımda, dizgedeki düğümler alt görevleri, köşeler ise alt görevler arası sıralı veya mantıksal bağımlılıkları tanımlamaktadır. İzlek tabanlı bir görev yönetimi bağlamında, düğümler çalışan izlekler veya alt görevler olacaktır ve köşeler ile tanımlı bağımlılıklar sistem üzerinde çalıştırılacak bariyer katmanları ile sağlanacaktır. DAG tabanlı sistemler koşut sistemlerin geliştirilmesinde yoğun biçimde kullanılan bir yaklaşımdır, bununla birlikte günümüzde hava elektroniği, uyarlanır otomotiv sistemleri gibi birçok gerçek zamanlı sistemin geliştirilmesinde faydalanılmaktadır. Pfair görev zamanlamada kullanılan alt-görev yapısının, DAG tabanlı sistemlerle bir arada kullanılabileceği rahatlıkla söylenebilir ve yakın zamanda bahsedilen DAG tabanlı yaklaşımın homojen ve heterojen çok işlemcili sistemler üzerinde görev zamanlamaya uyarlandığı çalışmalardan umut veren sonuçlar vermiştir [38]–[41]. Görev zamanlama üzerine gerçekleştirilen teorik ve algoritmik çalışmaların yanında, görev zamanlama algoritmalarının gerçekleştirimi ve entegrasyonu üzerine çalışmalar mevcuttur [24], [42]–[47].Çalışma kapsamı dışında olmakla birlikte, güç sezimli sistemlere özel görev zamanlama ve kaynak yönetimi yaklaşımları da bulunmaktadır [48]–[55].[56], [57]'da, altyapı düzeyinde çalışan bulut hesaplama hizmetleri sağlayıcıları yumuşak gerçek zamanlı sistemler olarak tanımlanmıştır ve bu sistemler üzerinde görev zamanlama yaklaşımları tartışılmıştır.

4. FOTONİK YONGA ÜZERİ AĞLAR

Günümüzde kişisel bilgisayarlardan süper bilgisayarlara kadar bütün bilgisayar sistemi işlemcilerinde çok çekirdekli veya çok işlemcili mimarilere geçilmiştir. Koşut hesaplama kuramı, birçok işlemcinin bir problemi çözmek veya belirlenen sistemsel bir amaç doğrultusunda bir arada çalışmasına dayanır. Bu bağlamda işlemciler arası iletişim koşut sistemlerinin etkinliğini belirleyen etkenlerin başında gelir. Daha fazla işlemcinin aynı yonga üzerinde veya dağıtılmış olarak bir araya gelmesi artan işlem gücü potansiyeli, bu durumla doğru orantılı yükselen bir iletişim kapasitesi ihtiyacını getirmektedir.

Günümüzde genel amaçlı bir işlemci yongası üzerine onlarca işlemci sığdırılmaktadır ve gelecekte yüzlerce işlemcinin bulunduğu yonga mimarileri tasarlanmaktadır [58]. Çok işlemcilerin bir araya gelmesinde ortaya çıkan yüksek işlem gücü, iletişim kapasitesi ihtiyacını artırarak yonga üzeri iletişimde elektrik tabanlı iletişim yaklaşımlarının fiziksel sınırlarını zorlamaktadır.

Optik sistemlerin düşük enerji tüketimi ve yüksek bant genişliği sağlamaları, geliştiricileri yonga üzeri sistemlerde optik iletişim mimarilerinin kullanmaya yöneltmiştir. Yakın zamanda yapılan çalışmalar, optik sistemlerin sadece iletişim ve ağ alt yapılarında değil, yonga üzeri iletişim için kullanıldığı sistemleri ortaya çıkarmıştır. Bu sistemler bilgisayar üzerindeki birden fazla işlemcinin birbirleriyle optik alt yapı üzerinde veri iletişimi yapabilmesini sağlamaktadır [59]. Telekomünikasyon üretici ve araştırmacıları, özellikle dalga boyu bölmeli çoklama (WDM) kullanımı ile birlikte, elektronik alternatiflerine göre yüksek avantaj sağlayan fotonik sistemleri yonga sistemlere uyarlamaya başlamışlardır [8], [60].

Başta işlemciler arası iletişimi olmak üzere, yonga üzerinde optik iletişim sağlayan bu sistemler fotonik ağlar olarak tanımlanmaktadırlar. Fotonik ağlar teorikte ışık dalgaları üzerinde kodlanan verilerin, ağ bileşenleri üzerinde kurulan ışık yolları üzerinde yönlendirilerek hedef düğüme iletilmesi yaklaşımına dayanır.

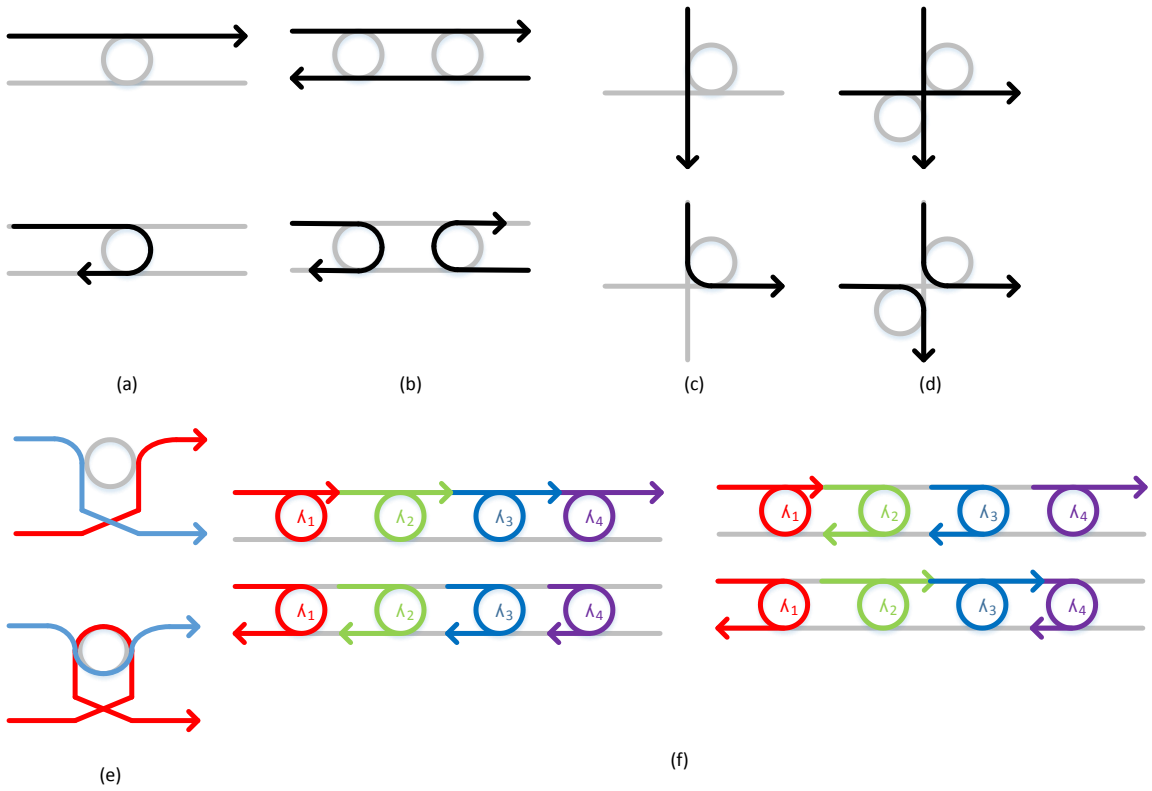
Fotonik sistemlerin yonga üzeri veri iletişiminde kullanılması ilk olarak Goodman tarafından önerilmiştir [61], [62]. Fotonik sistemlerin yonga üzeri üretime ve CMOS uygunluğu, bu sistemlerin yonga üzeri ağlarda elektronik ağ sistemleri için düşük gücü tüketimli ve yüksek bant genişliği sunan bir alternatif olmasını sağlamıştır [63].

Fotonik sistemlerin bir diğer avantajı, ikil oran şeffaflığı özellikleridir. Elektronik tabanlı ağ bileşenleri bit düzeyinde iletim yaparlar ve güç tüketimi ve iletim zamanı iletilen mesaj uzunluğuna bağlı biçimde artar. Fotonik sistemler ise iletim ileti uzunluğu ve mesafeden bağımsız bir güç tüketimi sağlamaktadırlar. İkil oran şeffaflığı enerji tüketimini düşürmekle birlikte, sistemin mesaj iletiminde mesafeden bağımsız şekilde belirlenebilir bir davranış sergilemesini sağlar [64].

Bununla birlikte, optik sistemlerin bir takım dezavantajları da bulunmaktadır. Optik tabanlı mantıksal kapıların ve saklama birimlerin ticari olgunluğa erişmemiş olmaları, optik düzeyde paket yönlendirme ve çakışma durumunda paketlerin yastıklarda saklanmasını olanaksız kılmaktadır. Bu durum optik sistemlerde yönlendirme ve ara düğümlerde saklama işlemlerin elektronik bileşenlerle yapılmasını gerektirmektedir [58]

Fotonik yonga üzeri sistemler; dalga kılavuz, optik filtreler, modülatör ve foton algılayıcı gibi karmaşık optik cihazların silikon düzeyinde bir arada kullanılması ile elde edilir [65]. Yonga üzeri fotonik ağlar, temelde 1x2 ve 2x2 optik anahtarların bir arada kullanılması ile oluşturulurlar. 1x2 ve 2x2 anahtarlar, girdilerden gelen ışık dalgalarını, sahip olduğu iki porttan istenilene yönlendirilmesini sağlarlar. Optik anahtarlar; teorik olarak Mach-Zehnder (MZ) interferometer, microring resonator yaklaşımlarından birine dayanmasına rağmen [65], microring tabanlı mimarilerin düşük maliyeti ve fiziksel olarak daha az yer kaplaması sebebiyle daha yaygın kullanıldığı söylenebilir. İlk tasarımlarda bu işlev birden fazla microring kullanılmasını gerektirirken, daha sonraki çalışmalarda bu işlev tek bir microring ile de sağlanmıştır.

Optik düzeyde anahtarlama iki şekilde yapılabilmektedir: uzamsal anahtarlama ve dalga boyu seçimli anahtarlama [66]. Uzamsal anahtarlama yaklaşımında, belli bir port üzerinden gelen bütün ışık dalgaları, fiziksel düzeyde bir arada yönlendirilir. Dalga boyu seçimli anahtarlama yaklaşımında, belirli bir dalga boyu aralığında olan ışık dalgaları grubu anahtar üzerinde seçilebilir ve belli bir porttan gelen farklı dalga boyu aralıklarındaki dalgalar fiziksel olarak farklı portlara yönlendirilebilir. Bu yaklaşımda, belirli bir port üzerinde seçim uygulanacak her bir dalga boyu aralığı için ayrı bir rezonatör ihtiyaç duyulur. Uzamsal ve dalga boyu seçimli anahtarlama yaklaşımlarının her ikisi için farklı iletim yaklaşımları Şekil 4-1 ile gösterilmiştir [65], [67].



Şekil 4-1: Fotonik anahtar bileşenleri: (a) 1×2 kesişmesiz (non-crossing), (b) 2×2 kesişmesiz, (c) 1×2 kesişmeli, 2×2 kesişmeli, (d) 2×2 tek halkalı (e) ve 4 dalga boyu seçmeli (f) [67]

1x2 2x2 anahtarlar microring resonator genelde fotonik anahtarlama bileşenleri (*PSE*) olarak tanımlanmaktadırlar [67]. PSE bileşenleri, optik anahtarlar için temel bir bileşen özelliği taşımaktadırlar. PSE bileşenleri daha yüksek kanal ve iletişim kapasitesine sahip anahtarların tasarlanmasının önünü açmıştır. Birden fazla PSE nin bir arada kullanılmasıyla, 3x3 4x4 5x5 ve 6x6 anahtar tasarımları gerçekleştirilmiştir. Tasarlanan bu anahtarlar çok kanallı optik anahtarlardır ve belli bir topoloji ile bir araya getirilmiş işlemciler arasında iletişim için kullanılabilmektedirler.

Işık dalgaları, parçacık ve dalga davranışları birlikte gösterirler. Işığın fiziksel özellikleri, yonga üzeri ağlarda bant genişliği artırımı ve aynı ortamdan birbirlerinden farklı verilerin iletilmesi için birden fazla yaklaşımın geliştirilmesine olanak vermiştir. Bu yaklaşımlardan en temeli dalga boyu bölmeli çoğullama yaklaşımıdır. WDM aynı dalga kılavuzunda farklı dalga boyundaki ışık dalgalarının birlikte iletimine olanak verir. Yapılan çalışmalarda 32 adet dalga boyunun eş zamanlı kullanıldığı çalışmalar bulunmaktadır [68]. WDM aynı verinin eş zamanlı olarak iletilerek bant genişliğinin artırılmasında kullanılmakla birlikte,

aynı zamanda farklı verilerin çakışmadan aynı dalga kılavuzundan iletilmesine de olanak verir.

Optik sistemlerde ışığın fiziksel davranışlarından faydalanılarak geliştirilmiş bir diğer çoklama yaklaşımı mod bölmeli çoklama yaklaşımdır. Bu yaklaşım, ışığın farklı sıklıktaki dalga kılavuzlarında farklı elektromanyetik örüntü göstermesi durumunu kullanan bir yaklaşımdır [69]. Bu yaklaşımın avantajı WDM ile birlikte kullanılarak bant genişliği ve eş zamanlı veri iletişim olanaklarını daha da artırabilmesidir.

Optik sistemlerde kullanılan diğer çoklama yaklaşımı zaman bölmeli yaklaşımıdır. Bu yaklaşım birçok bilgisayar sisteminde kullanılan, kaynakların zaman ekseninde kullanıcılar arasında paylaştırılması yaklaşımına dayanır. Bu yaklaşımda dalga kılavuzları farklı zaman dilimlerinde düğümler arasında paylaştırılır. Bu yaklaşım belirlenebilir yönlendirme algoritması geliştirmek için kullanılmıştır [60].

Çok işlemcili bir mimari üzerindeki iki düğüm arasında bir ileti göndermek için, hedef düğüm ve kaynak düğüm arasında iletiye ayrılmış bir yol kurulması gerekir. Bu iletim yolu birden fazla düğümden geçilmeyi gerektirebilir ve sistemin ağ mimarisi bağlamında her düğümde bir yönlendirme süreci gerektirebilir ve ara düğümlerde belirlenen yol için gerekli portlar başka iletimlere atanmış olabilir ve iletinin ara düğümde beklemesi gerekebilir. Ara bağlantılı bir ağ üzerindeki bir iletinin iletim maliyeti genelleştirilmiş olarak eşitlik (4) ile gösterilebilir:

$$T = H(\propto + \delta + t_r + L\tau) \tag{4}$$

Bu eşitlikte $\propto$ iletim başlangıcı için gerekli maliyetleri, H iki düğüm arasındaki düğüm/bağlantı sayısını, δ her bir düğümdeki ağ bileşeni/anahtar gecikmelerini, L ileti boyunu, t_r yönlendirme gecikmesini ve $(1/\tau)$ bağlantı bant genişliğini tanımlamaktadır. Devre anahtarlamalı iletişim yaklaşımında, iletim boyunca gerekli olan tüm düğümler ve ilgili portları iletime atandıktan sonra, ileti kurulan tek bir devre üzerinden iletilir. Devre anahtarlamalı iletişim için iletim maliyeti eşitlik (5) gibi gösterilebilir [12].

$$\begin{aligned} T_{devre} &= \propto + H\delta + L\tau \\ \propto &= H(t_r + 2(\delta + (1/\tau)) \end{aligned} \tag{5}$$

Eşitlikte görüleceği üzere bu iletimdeki maliyet, kurulum maliyeti $(\propto)$ ve iletim maliyeti olarak iki grupta toplanacaktır. İletim maliyeti gerekli yolun kurulması için tüm düğümlere kurulum iletisinin iletilmesi ve ayrılan tüm düğümlerden kaynak düğüme geri bildirim için

gerekli iletim maliyetlerini kapsar. Günümüzdeki modern anahtarların cihazlarının yeterince hızlı olması ve anahtar beklemeleri asgari düzeye indiği için, eşitlikteki δ diğer ∝ ve Lτ maliyetlerine göre oldukça göz ardı edilebilir olacaktır. Bu yaklaşımla eşitlik (6)'daki gibi sadeleştirilebilir [67].

$$T_{devre} = \propto + L\tau \tag{6}$$

Optik veriler için geliştirilmiş mantık kapıları ve yastıklama birimlerinin ticari olarak yetersiz olması sebebiyle, fotonik iletişimde ara düğümlerde doğrudan optik veri üzerinde yönlendirme yapılmasının veya çakışma anında ara düğümlerde verinin optik formatta saklanmasının olanaksız olduğunu hatırlayalım. Bu durum, fotonik ağlarda yastıksız veri iletimi yapılmasını gerektirmektedir ve fotonik ağlarda ancak devre anahtarlamalı iletişim yöntemleri uygulanabilmektedir. Bahsedildiği üzere, devre anahtarlama yaklaşımında iletim maliyeti kurulum ve iletim maliyetlerinden oluşur ve bu iletimin maliyeti eşitlik (6) ile hesaplanabilir. Optik veri üzerinde yapılabilir işlem kısıtları, optik verinin sadece kurulan devreden iletilmesine olanak verir ve devre kurulumu için yazılımsal ve/veya donanımsal bir üst katmana ihtiyaç duyulmaktadır.

Optik verinin yönlendirme ve yastıklanmasındaki yetersizlik, fotonik iletişim öncesi iletinin elektriksel biçimden optik biçime ve her iletim adımı sonunda optik biçimden elektriksel biçime çevrilmesini zorunlu kılar. Dolayısıyla optik devre anahtarlama maliyetinde elektriksel devre anahtarlama maliyetine ek olarak Elektro-optik (EO) ve Opto-elektrik (OE) dönüşüm maliyetleri bulunacaktır ve iletişimde verinin optik ve elektriksel dönüşüm maliyetleri de bulunmaktadır. Bu yaklaşımla, optik devre iletişim maliyeti eşitlik (7) ile hesaplanabilir.

$$T_{optik-devre} = \propto + L\tau_{EO} + L\tau_{O} + L\tau_{OE} \tag{7}$$

Burada τ_O fotonik sistem bant genişliği, τ_{EO} verini Elektro-optik dönüşümü için bant genişliği ve τ_{OE} opto-elektrik dönüşüm bant genişliğini tanımlar.

Yonga üzeri iletişim için geliştirilen fotonik sistemler, kurulum katmanı yaklaşımı ve mimarileri doğrultusunda gruplara ayrılmaktadır. Son dönemde özellikle CMOS üretiminin uygunluğu sonrası, fotonik yonga üzeri ağlar ve bu ağlar üzerindeki kurulum üst katmanı hakkında birçok çalışma yapılmıştır. Bu konuda ilk yapılan çalışmalar, veri-yolu tabanlı sistemler olarak gruplanabilir ve bu mimaride geliştirilen *Corona* ve *Firefly* sistemleri fotonik yonga üzerinde ağ kullanan ilk çok işlemcili mimariler olarak gösterilebilir [5], [70], [71]. İkinci ve en çok çalışmaya konu olan yaklaşım, üst katman için basit bir elektrik

tabanlı iletim katmanının kullanıldığı karma sistemlerdir. Bu yaklaşımda sistemde biri elektrik ve biri optik tabanlı olmak üzere iki iletişim altyapısı bulunur ve optik veri iletimi için devre kurulması elektrik tabanlı sistem üzerinden sağlanır [13], [58], [59], [64], [72]–[76]. Bu yaklaşımda genel olarak devre kurulumunda paket anahtarlama ve veri aktarımında devre anahtarlama tercih edilmektedir. [74]'te alındı iletileri için bir dalga boyu ayrılarak, bu mesajların da optik sistem üzerinden iletilmesi sağlanmıştır. Benzer şekilde *Iris* çalışmasında uzun iletiler ve toplu iletişim için optik katman, kısa iletiler için ve devre kurulumu için elektrik tabanlı katman kullanılmıştır [77].

Fotonik iletişim için geliştirilmiş üçüncü yaklaşım, zaman bölmeli çoklama yaklaşımıdır [60], [78]. Bu yaklaşımda, kurulum için elektrik tabanlı ayrı bir katman kullanılmamıştır ve her bir işlemci çifti, kendisine ayrılmış zaman aralığında *round-robin* yaklaşımıyla iletişim kurar. Bu yaklaşım temel olarak sürekli tekrarlanan bir hepsinden hepsine iletim olarak tanımlanabilir.

Son yaklaşım belirlenebilir ve ölçeklenebilir çoklu gönderme yaklaşımıdır ve TDM yaklaşımına benzer şekilde fiziksel olarak üst bir katman bulunmamaktadır. Bu yaklaşımda, iletişimde dinamik yönlendirme yaklaşımları kullanılmamaktadır ve tüm iletim rotaları önceden belirlenmiş iletişim örüntüleri ve toplu iletişim algoritmaları doğrultusunda gerçekleştirilmektedir. Bu bağlamda, bütün rotalar önceden belirlendiği için, iletim öncesi yolun ayrılması gerekmemektedir ve iletim sırasında herhangi bir çakışmanın olması engellenmektedir. [67]'de İmre, söz konusu yaklaşımı çift mod fotonik mimari için toplu iletişim algoritmalarına uyarlamıştır.

5. GZS İÇİN GÖREV ZAMANLAMA VE İLETİŞİM YÖNETİM ALGORİTMALARI

Bu tez çalışmasında, çok işlemcili dağıtılmış bellekli bir yonga mimarisi tasarlanmış ve önerilen mimari üzerinde özgün bir iletişim ve görev yönetimi yaklaşımı önerilmiştir. Tasarlanan bilgisayar mimarisinin iletişim düzeyinde, elektronik tabanlı sistemlere alternatif olan gelişmiş fotonik yonga üzeri ağ mimarilerinden faydalanılmıştır. Önerilen donanım mimarisi üzerinde görev ve iletişim süreçlerinin yönetimi için belirlenebilir, ölçeklenebilir ve örüntüsel algoritmaların tasarlanması sağlanmıştır. Geliştirilen görev ve iletişim yönetimi yaklaşımı, sistem kaynaklarının zaman odaklı biçimde iletişim ve hesaplama görevleri arasında tahmin edilebilir ve etkin biçimde paylaşılmasını sağlamıştır. Bununla birlikte, gerçek zamanlı sistemler için iletişim ve görev yönetimini tek noktadan yapması yönüyle yenilik içermektedir. Bu bölümde önerilen iletişim ve yönetim örüntüleri detaylandırılmıştır.

5.1. İletişim

Önerilen ağ ve iletişim mimarisi, *all-port* seçmeli dalga boyu tabanlı fotonik anahtarlar üzerinden iletişim sağlayan düğümlerin, hasır doku topolojisinde birbirlerine bağlanması ile oluşturulmuştur. Önerilen mimarinin dağıtılmış bellekli MIMD bir koşut sistemdir ve her düğüm bir fotonik anahtarla birlikte bir işlemci ve bir bellek bileşeni içermektedir. Bununla birlikte, mimari birden fazla işlemci içeren bir işlemci öbeğinin bir fotonik anahtara bağlanabilmesini desteklemektedir. Önerilen mimaride, iletişim tahmin edilebilir ve belirlenebilir iletişim örüntülerine dayanmaktadır ve bir yönlendirme katmanına ihtiyaç duymaz. Bu bağlamda, sistem iletişim katmanında anahtarlama ve akış denetimi katmanlarından oluşmaktadır. Sistem mimarisi yazılım tabanlı bir iletişim katmanı sağlamaktadır ve anahtarların iletişim öncesi ayarlanmaları ve dalga boylarının atanması bağlı olduğu düğümdeki işlemci tarafından yapılmaktadır. Sistem koşturduğu gerçek zamanlı sistem senaryosunun aşamaları, zamana dayalı bir eş-zaman uyumlama mekanizması ile programlanmaktadır, böylece yazılım, iletişim yollarını ve bu yollara atanan dalga boylarını tanımlamaktadır.

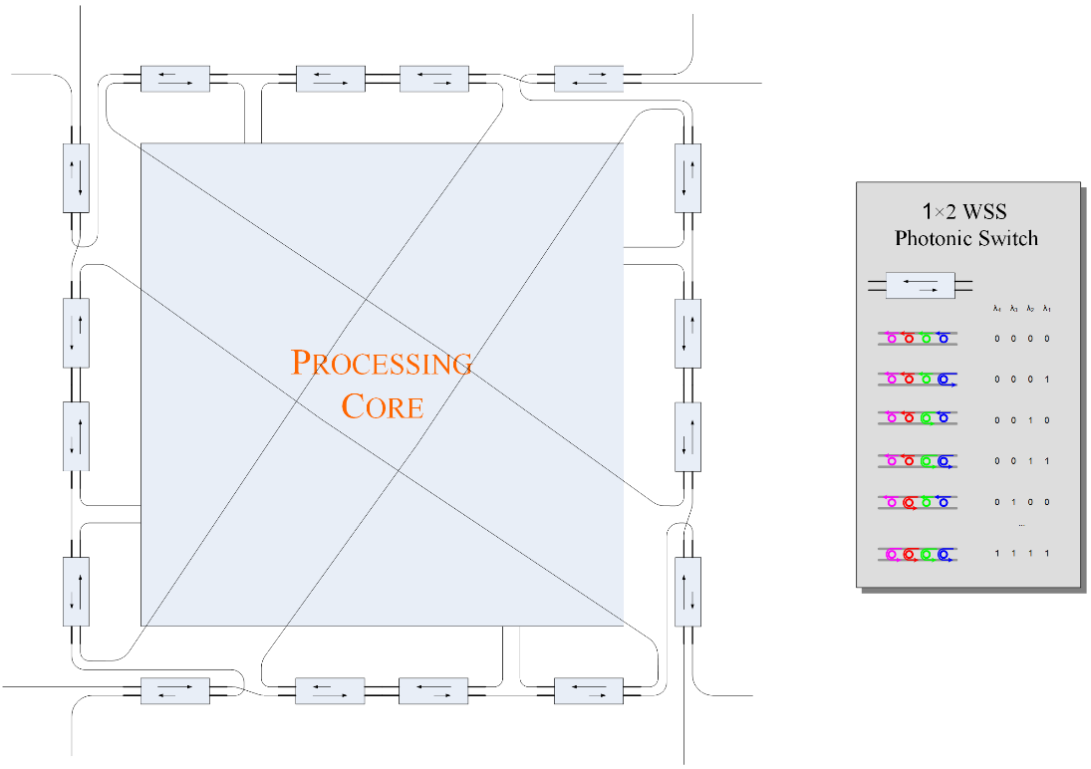
Önerilen sistemdeki bir fotonik anahtarda, her bir yönde dört adet olmak üzere 16 adet Fotonik Anahtarlama Bileşeni (FAB) bulunmaktadır. Sistemde dört eş zamanlı iletişim için dört dalga boyu grubu kullanmaktadır ve her bir FAB içinde, her bir microring bileşeninin bir dalga boyu grubuna atandığı dört adet microring bulunmaktadır. Dolayısıyla, bir fotonik anahtarda toplamda 64 adet microring bileşeni bulunur. Önerilen sistemde herhangi

bir microring bileşeni üzerinden geçen ışık dalgalarının başka yöne yönlendirileceği veya doğrusal devam edeceği ilgili microring in aktif edilip edilmemesi ile bağlanır. Dolayısıyla, herhangi bir iletişim adımında FAB bileşenleri içindeki microring bileşenlerinin aktiflik durumunu programlamak için bir bit kurulum bilgisi (microring ON/OFF) yeterli olacaktır. Bu durumda, her işlemci bir iletişim adımından önce fotonik anahtarın kurulması için, her bit bitin bir microring aktiflik durumunu kurduğu 64-bit uzunluğunda bir yazmaca ihtiyaç duyar. Bu durumda sistem üzerinde herhangi bir iletişimi adımının başlamasından önce, işlemcilerin anahtar bileşeninin sürücü yazılımına 64-bit uzunluğunda bir komut sözcüğü göndererek anahtarın kurulması gerekmektedir.

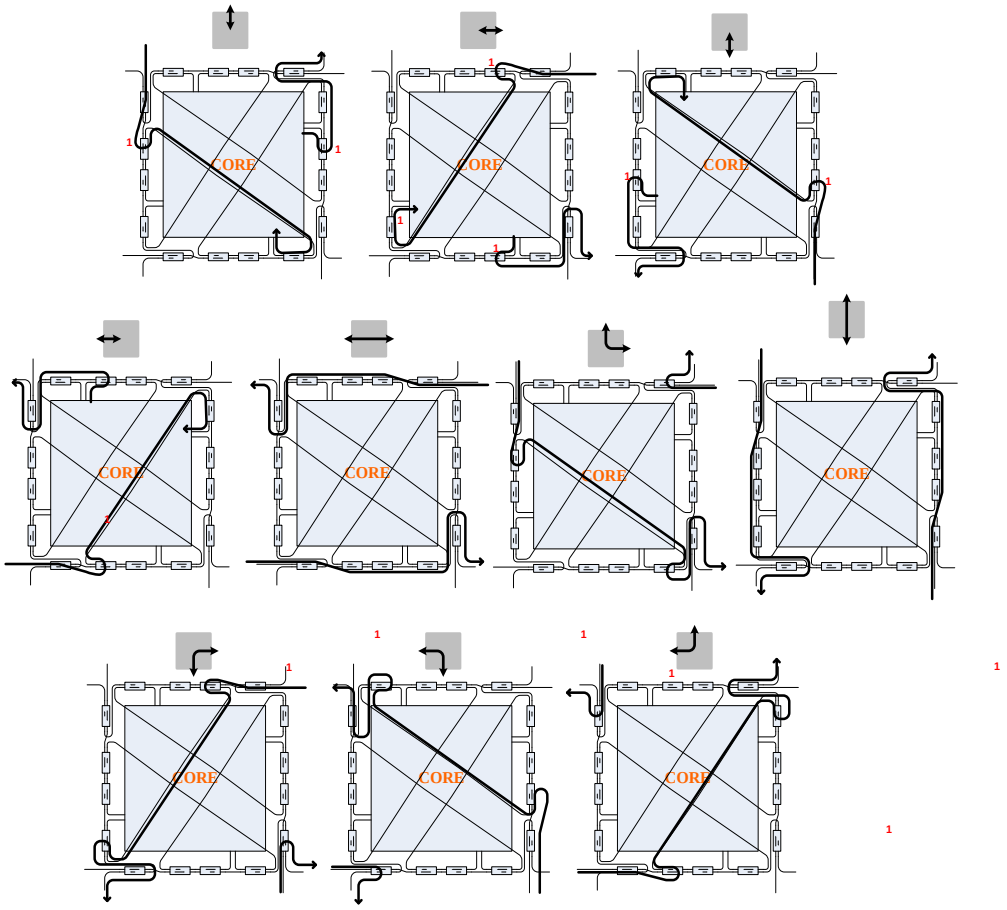
Söz konusu anahtar tasarımı Şekil 5-1 ile gösterilmiştir. Tasarlanan anahtar 8x8 *all-port* bir anahtar tasarımıdır her bir dalga boyu için ve eş zamanlı olarak dört farklı yönde iki yönlü olmak üzere iletişime izin vermektedir. Dolayısıyla her düğüm eş zamanlı olarak dört komşusu ile eş zamanlı olarak iletişim kurabilir. Sistemde dört dalga boyu kullanılmıştır bu sayede bir anahtarda her bir yönde dört tane olmak üzere eş zamanlı olarak 16 iki yönlü iletişime olanak vermektedir. Bu durum her bir anahtarın birbirinden bağımsız çalışan 4 adet 8x8 tam kapı anahtar gibi davranabilmektedir.

Önerilen iletişim örüntüleri çakışmadan bağımsız temel örüntülerin bir araya getirilmesinden oluşmaktadır ve bu temel örüntüler de anahtar düzeyinde tanımlamış basit ve eş zamanlı olarak birbirleriyle çakışmayan iletişim yolları grubu üzerine kurulmuştur. Bahsedilen çakışmasız basit iletişim yolları Şekil 5-2 ile gösterilmiştir.

Önerilen anahtar microring tabanlı bir fotonik altyapıyı kullanmaktadır ve iletişimde ışık dalgalarının farklı yönlere iletilmesi microring bileşenleri ile sağlanmaktadır. Sistemde dört dalga boyu kullanıldığı için her iletişim yönünde dört adet microring bileşeni kullanılmıştır. Bu bağlamda, her bir anahtarda 64 adet microring bileşeni kullanılmıştır. Microring bileşenleri kendilerine atanmış dalga boyu grubunu birbirlerinden bağımsız olarak yönlendirebildiği için, önerilen basit iletişim yolları dört dalga boyu için eş zamanlı olarak ve birbirlerini etkilemeden uygulanabilmektedir.



Şekil 5-1: Dalga boyu seçmeli anahtar yapısı ve fotonik iletişim element üzerinde dalga boyu seçim yapısı



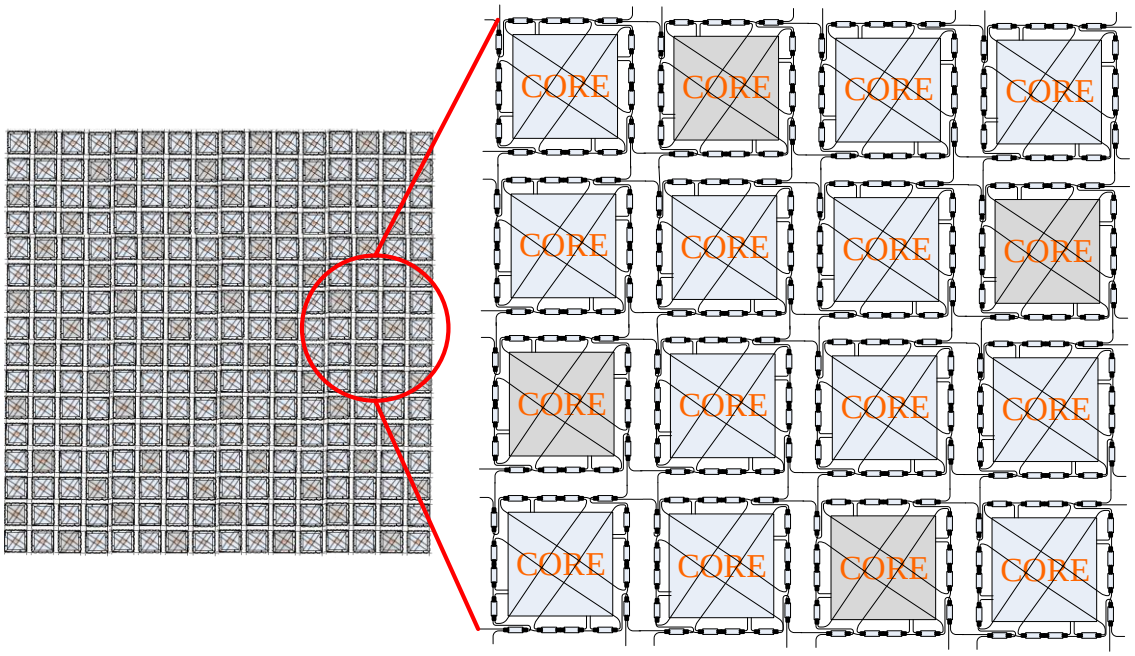
Şekil 5-2: Basit iletişim yolları ve iletişim yolları için gerekli anahtar yapılandırmaları

Önerilen sistemin iletişim katmanı yazılım tabanlı ağlar temel alınarak kurulmuştur. Bu ağlar üzerindeki veri düzlemi; hasır dokusu topolojisi üzerinden birbirlerine bağlanmış 8x8 fotonik anahtarlar bileşenleri tarafından oluşturulmaktadır ve denetim düzlemi anahtar yapılandırılmaları sağlayan işlemcilere dağıtılmıştır. İşlemciler üzerine dağıtılmış şekilde çalışan denetim düzlemi yazılımı, belirlenen örüntüleri kullanarak ilgili iletişim adımına bağlı olarak toplu olarak iletişim rotalarını veya yollarını fotonik anahtarlar üzerinden düzenler. Bununla birlikte, belirlenen çakışmadan bağımsız toplu iletişim örüntüleri yanında, yazılım geliştiricisi uygulamaya özel kendi iletişim yol ve örüntüleri tanımlayabilir.

Önerilen sistemde hasır dokusu topolojisinde birbirlerine bağlanmış küçük boyutta işlemci grupları işlemci öbeği olarak tanımlanmaktadır. İşlemci öbekleri 2×2, 3×3, 4×4 veya 5×5boyutlarında olabildikleri düşünülmektedir ve sistem topolojisi öbeklerin özyineli olarak bir araya getirilmesiyle oluşturulmaktadır. İşlemci öbekleri, önerilen temel iletişim örüntülerinin uygulanabildiği en küçük işlemci topolojisi boyutunu tanımlamaktadır ve sistem düzeyindeki iletişimler öbekler üzerinde uygulanan örüntülerin öbekler arasında

doğrusal olarak ölçeklenerek genişletilmesi ile sağlanmaktadır. 256 adet işlemci içeren 16x16 hasır doku bir sistem topolojisini düşünelim. Bu sistem 4x4 işlemci öbeklerinin bir araya gelmesiyle kurulmuştur ve sistemde 16 adet işlemci öbeği bulunmaktadır. Her bir işlemci öbeğinde 4 adet baskın düğüm bulunmaktadır ve bu düğümler tanımlanan temel örüntülerde öbek içi iletişimde verilerin toplanması ve dağıtılmasından sorumludur. Baskın düğüm tanımı ilk olarak [79] çalışmasında kullanılmıştır ve bu düğümler öbek içindeki düğümlere tek adımlık bir iletimi çakışma olmadan sağlayabilen düğümlerdir. Önerilen mimaride, 4x4 bir işlemci öbeğinde 4 adet baskın düğüm bulunmaktadır ve her bir baskın düğüm diğer baskın düğümlere göreli olarak öbekteki farklı bir satır ve sütun adresinde konumlanmaktadır. 256 işlemci içeren bir sistem topolojisi

Şekil 5-3 ile gösterilmiştir ve şekil üzerindeki gri renkli düğümler her bir öbek içindeki baskın düğümleri betimlemektedir.



Şekil 5-3: 16-işlemcili öbeklerin bir araya gelerek oluşturduğu 256-işlemcili sistem

5.1.1. Temel iletişim örüntüleri

Bu çalışmada, ölçeklenebilir bir grup temel iletişim örüntüsü tanımlandı. Önerilen iletişim örüntülerinin hepsi önceden belirlenebilir olarak çalışırlar ve her zaman aynı iletişim davranışlarını gösterirler. Bununla birlikte, bu iletişim örüntüleri aynı topolojiye sahip daha yüksek ölçekli ağlara, logaritmik olarak yükselen adım sayısı ile uygulanabilmektedir. Bu temel algoritmalar örüntü yaklaşımı ile geliştirilmiştir ve daha büyük toplu iletişim

algoritmaları bir veya birden fazla temel örüntüyü sıralı veya tekrarlı olarak uygulayarak elde edilebilir.

Bahsedildiği üzere, seçmeli dalga boyu bir fotonik iletişimde etkin sonuçlar ancak fotonik düzeyde devre anahtarlama yaklaşımı ile elde edilmektedir. Bu yaklaşımla, temel örüntülerin bütünü devre anahtarlama ile çalıştırılmaktadır. Bununla birlikte, örüntüler yönlendirme fazına ihtiyaç duymadığı için devre anahtarlamanın en büyük maliyeti olan kurulum maliyetlerini de düşürmektedir.

Önerilen iletişim algoritmaları iletişim davranışları doğrultusunda iki temel gruba ayrılmaktadır: Baskın düğüm iletişimi (BDİ) ve öbek iletişimi (Öİ). BDİ aynı işlemci öbeğinde bulunmayan baskın düğümler arasında yapılan veri toplama/dağıtım algoritmalarıdır. Bu algoritmalar, toplanacak veri işlemci öbekleri düzeyinde toplandıktan sonra öbekler arası veri alış verişi adımlarını veya dağıtılacak veri öbek seviyesine kadar dağıtılması adımlarını kapsar. Öİ algoritmaları ise işlemci öbeği seviyesindeki veri toplama veya dağıtım iletişim adımlarını kapsar. Öİ algoritmaları işlemci öbeği seviyesinde uygulanırlar ve bu adımlar tüm işlemci öbeklerinde eş zamanlı olarak çalışırlar.

Önerilen yaklaşımda iletişim, çoklu dalga boyu tabanlı fotonik ağlar için geliştirilmiş temel örüntülerine dayanmaktadır ve tüm toplu iletişim algoritmaları, önerilen temel örüntülerin tekrarlı veya sıralı olarak birlikte kullanılması ile elde edilir. Önerilen temel örüntüler aşağıda belirtilen özelliklere sahiptir:

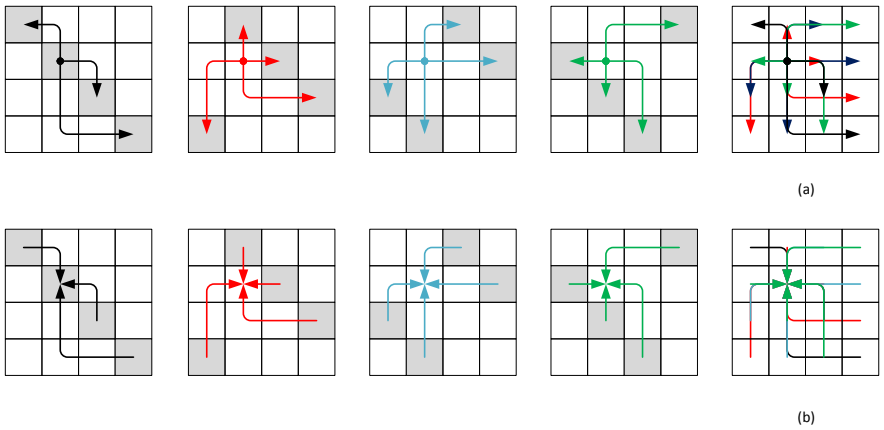
- Tüm temel örüntüler tek bir iletişim adımında tamamlanırlar.
- Tüm örüntüler ölçeklenebilir algoritmalardır. Bu örüntüler baskın düğümler arasında ve öbek seviyesindeki veri alışverişi adımlarına, değişik rotalama boyutlarıyla uygulanabilir.
- Fotonik tabanlı iletişim mimarisinde, bir temel iletişim örüntüsü en fazla dört adet dalga boyunun eş zamanlı kullanılmasını gerektirir.

Bu çalışma kapsamda, altı adet temel iletişim örüntüsü tasarlanmıştır:

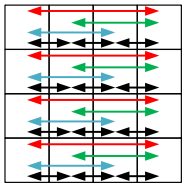
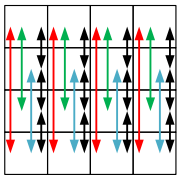
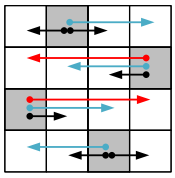
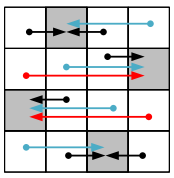
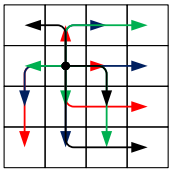
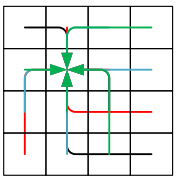
- Satır veri değişimi: Aynı satırda konumlanmış olan, öbek veya baskın düğümler düzeyindeki dört düğüm arasındaki veri değişimi adımıdır. Bu adım sonunda, değişimi yapan dört düğümde aynı veriye sahip olur.

- Sütun veri değişimi: Aynı sütunda konumlanmış olan, öbek veya baskın düğümler düzeyindeki dört düğüm arasındaki veri değişimi adımıdır. Bu adım sonunda, değişimi yapan dört düğümde aynı veriye sahip olur.
- Satır çoklu gönderme: Baskın düğümdeki veri aynı satırdaki aynı öbekteki veya farklı öbeklerdeki işlemci grubuna çoklu olarak dağıtılır.
- Satır veri toplama: Düğümlerdeki veri baskın düğümde toplanır. Bu veri baskın düğümlerde veya diğer düğümlerde olabilir.
- Merkezi baskın düğümden veri dağıtımı: Adres olarak öbeğin ortasında bulunan merkezi baskın düğümdeki veri, öbekteki diğer tüm düğümlere dağıtılır.
- Merkezi baskın düğümde veri toplama: Öbekte bulunan düğümlerdeki veriler adres olarak öbeğin ortasında bulunan merkezi baskın düğümde toplanır.

Tanımlanan temel örüntüler ve 4x4 bir işlemci öbeğinde uygulanması Çizelge 5-1 ile gösterilmiştir. Anlaşılırlığı artırmak üzere, merkezi düğüm tabanlı dağıtım ve toplama işlemleri dalga boyuna göre ayrılmış olarak Şekil 5-4 ile gösterilmiştir.



Şekil 5-4: Merkezi düğüm örüntülerinin dalga boylarına göre ayrılmış gösterimi (a) merkezi düğümden veri dağıtma (b) merkezi düğümde veri toplama

Satır veri değişimi (E_R)	
Sütun veri değişimi (E_C)	
Satır çoklu gönderme (M_R)	
Satır veri toplama (G_R)	
Merkezi baskın düğümden veri dağıtımı (S_{CDN})	
Merkezi baskın düğümde veri toplama (G_{CDN})	

Çizelge 5-1: Temel örüntüler ve 4x4 öbek üzerinde uygulanması

5.1.2. Birden hepsine yayın (*Broadcast*) Algoritması

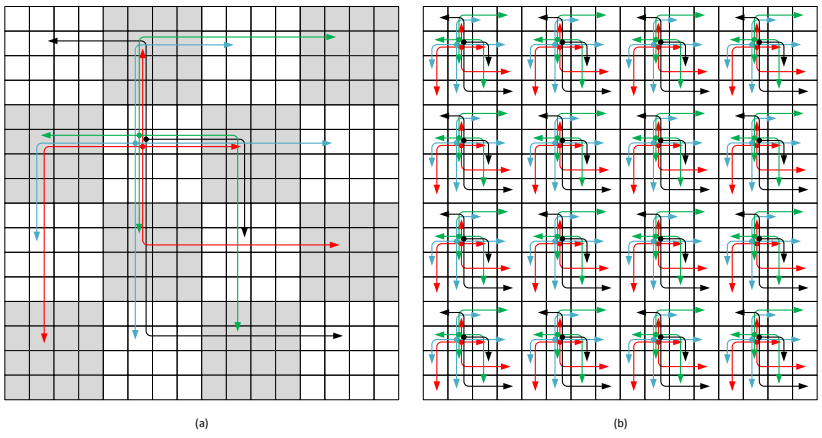
Önerilen birden hepsine yayın ve hepsinden bire algoritmaları, tanımlanan temel örüntülerden merkezi baskın düğümden veri toplama (G_{CDN}) ve dağıtma (S_{CDN}) örüntülerini temel almaktadır. Bu yaklaşımda, birden hepsine yayın algoritması, tüm düğümlere ilgili veriler dağıtılana kadar S_{CDN} adımı özyineli olarak tekrarlanır. Önerilen

algoritma, tek bir işlemci öbeğinde 4 dalga boyu kullanılarak tek bir iletişim adımında tamamlanabilmektedir. Bu iletim algoritması devre anahtarlama ile işletildiği için, bu iletişimin maliyeti eşitlik (6) ile hesaplanabilir. Yayın algoritması Şekil 5-4 (a)' daki temel örüntü iletişimi ile gösterilebilir.

Önerilen yaklaşımın sistem düzeyinde uygulandığında, yayın mesajı tüm öbeklerdeki merkezi baskın düğümlere iletilene kadar özyineli olarak devam eder. Bu adımdan sonra tüm öbeklerde eş zamanlı olarak merkezi baskın düğümden öbekteki tüm düğümlere tek adımda dağıtım yapılır. $4^k \times 4^k$ hasır doku bir ağ topolojisi üzerinde birden hepsine yayın algoritması maliyeti eşitlik (10) ile hesaplanabilir.

$$T_{yayın} = k(t_{S_{CDN}})$$
$$T_{yayın} = k(\propto_o + L\tau o) \quad (10)$$

Burada $\propto_o$ ileti başlangıç gecikme ve Lτo ise L uzunluğundaki mesajın iletim maliyetlerini tanımlar. Eşitlik (10), $4^k \times 4^k$ hasır doku bir ağ topolojisinde sistemde yayın algoritmasının $(k-1)$ baskın düğüm iletişim adımına ihtiyaç duyulduğunu ve toplam k adet iletişim adımında tamamlandığını göstermektedir. Bu durumda algoritmanın ihtiyaç duyduğu iletişim adım sayısı, ağ boyutuna bağlı olarak logaritmik olarak büyüdüğü görülmektedir. 256-çekirdek sistem üzerinde birden hepsine yayın algoritması Şekil 5-5 ile gösterilmiştir.

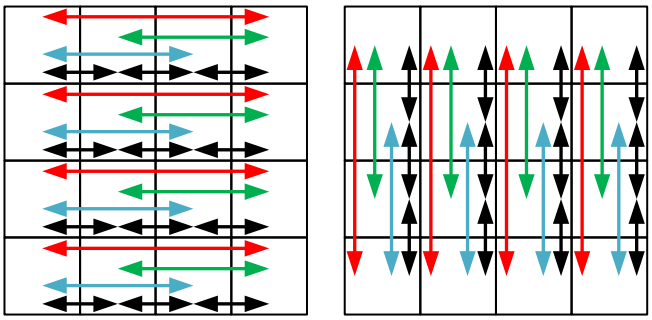


Şekil 5-5: Birden hepsine yayın (a) baskın düğüm adımı (b) öbek adımı

5.1.3. Hepsinden hepsine yayın Algoritması

Önerilen hepsinden-hepsine iletişim örüntüsü, dört temel örüntünün topoloji boyutuna bağlı olarak farklı sayılarda kullanılması ile uygulanmaktadır ve burada satır veri toplama, satır çoklu gönderme, sütun veri değişimi ve satır çoklu gönderme örüntüleri

kullanılmaktadır. Önerilen algoritmada, tek bir öbek üzerinde hepsinden hepsine algoritmanın işletilmesi için satır veri değişimi ve sütun veri değişimi olmak üzere iki iletişim adımına ihtiyaç duyar. Bu işlem Şekil 5-6 ile gösterilmiştir.



Şekil 5-6: 4x4 tek öbek üzerinde hepsinden-hepsine algoritması

Tek öbek üzerinde işletim için satır veri değişimi ve sütun veri değişimi adımları yeterli iken, daha büyük ağlar üzerinde işletim ayrıca satır veri toplama ve sütun çoklu gönderme adımlarına da ihtiyaç duyar. Daha geniş algoritmalarda hepsinden hepsine yayın algoritması $(k-1)$ adım satır veri toplama adımıyla başlar ve işlem sonunda düğümlerdeki veriler satır düzeyinde aynı satırda konumlanmış 4 baskın düğüme indirgenmiş olur. Bu adımdan sonra, tek öbek üzerindeki iletişime benzer biçimde satır veri değişimi ve sütun veri değişimi adımları uygulanır. Bu iki adımın tamamlanmasından sonra, sütün veri değişimi ve satır çoklu gönderme adımları ardıl biçimde uygulanır ve bu işlem baskın düğümlerdeki veri tüm düğümlere aktarılana kadar özyineli olarak tekrarlanır. $4^k \times 4^k$ hasır doku bir ağ topolojisinde söz konusu ardıl adımların $(k-1)$ kez uygulanması gerekmektedir. Her düğümün diğer tüm düğümlere iletilmesi gereken L uzunluğunda bir mesajı olduğu ve fotonik altyapının $1/\tau_o$ bant genişliğine sahip olduğu bir senaryoyu ele alalım. $M = L\tau_o$ için $4^k \times 4^k$ hasır doku bir ağ topolojisinde hepsinden hepsine yayın algoritması maliyeti eşitlik (11) ile hesaplanabilir:

$$T_{hepsindenhepsine} = (k-1)G_R + (E_R + E_C) + (k-1)(M_R + E_C)$$

$$(k-1)(t_{G_R}) = (\propto_o + 4^0 M) + (\propto_o + 4^1 M) + \cdots + (\propto_o + 4^{k-2} M)$$

$$(k-1)(t_{G_R}) = \frac{(4^{k-1}-1)}{3} M + (k-1) \propto_o$$

$$t_{E_R} = \propto_o + 4^{k-1} M$$

$$t_{E_C} = \propto_o + (4^{2k-1} M)$$

$$(k-1)(t_{E_C}) = (\propto_o + 4^0 4^k M) + (\propto_o + 4^1 4^k M) + \cdots + (\propto_o + 4^{k-2} 4^k M)$$

$$(k-1)(t_{E_C}) = \frac{4^k M (4^{k-1}-1)}{3} + (k-1) \propto_o$$

$$(k-1)(t_{M_R}) = (\propto_o + 4^1 4^k M) + (\propto_o + 4^2 4^k M) + \cdots + (\propto_o + 4^{k-1} 4^k M)$$

$$(k-1)(t_{M_R}) = \frac{4^k M (4^k-4)}{3} + (k-1) \propto_o \quad (11)$$

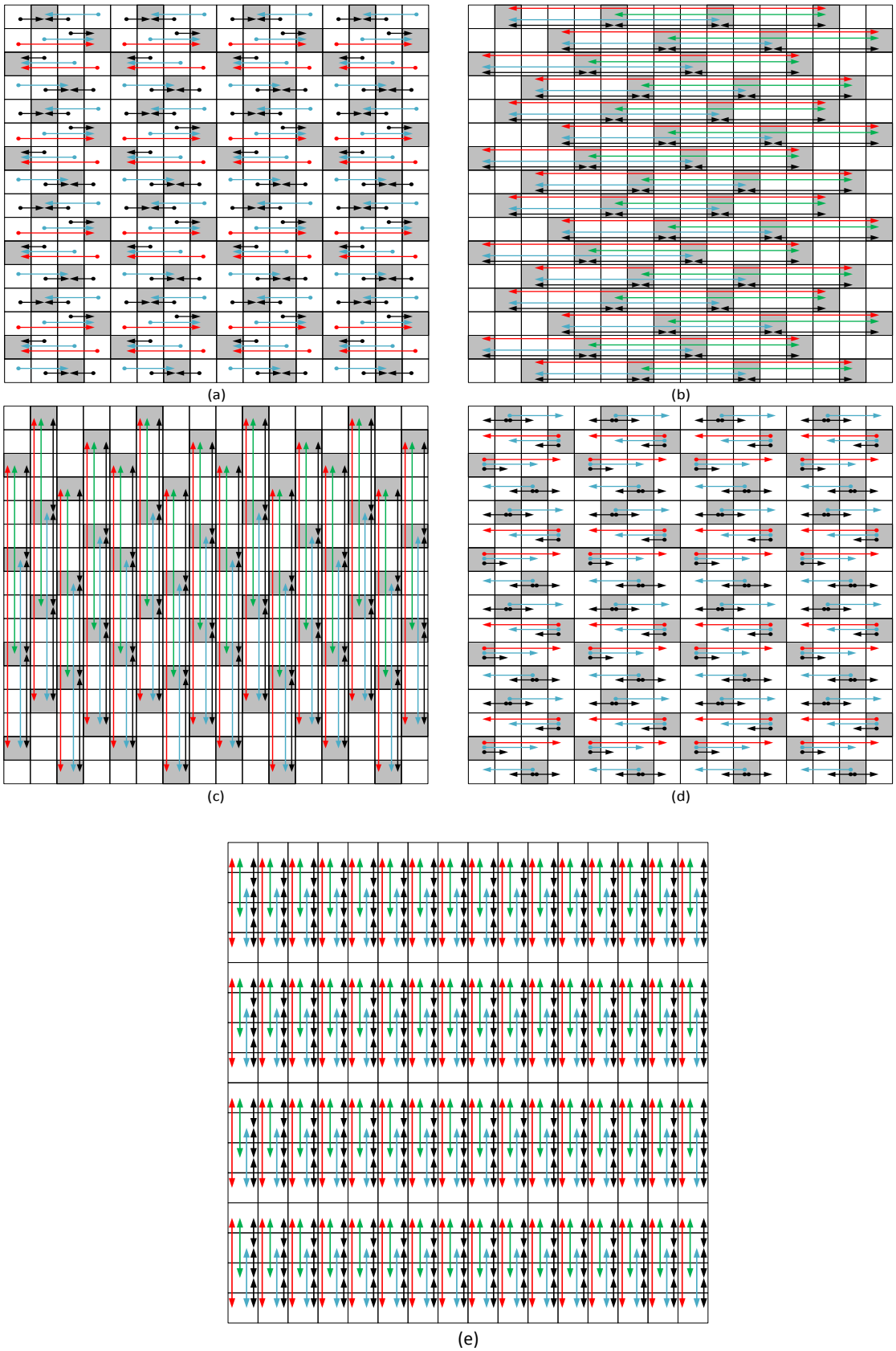
$$T_{hepsindenhepsine} = T_{toplam-baslangic} + T_{toplam-iletisim}$$

$$T_{toplam-baslangic} = (3k-1) \propto_o$$

$$T_{toplam-iletisim} = \frac{2(4^{2k}) - (4^{k+1}+1)}{3} M$$

$$T_{hepsindenhepsine} = (3k-1) \propto_o + \frac{2(4^{2k}) - (4^{k+1}+1)}{3} M$$

Eşitlik (11), $4^k \times 4^k$ hasır doku bir ağ topolojisinde toplam hepsinden hepsine iletişiminin $(3k-1)$ adıma ihtiyaç duyduğunu göstermektedir ve birden-hepsine algoritmasına benzer biçimde bu algoritmada tamamlanmak için gerekli adım sayısı ağ boyutuna bağlı olarak logaritmik büyüme göstermektedir. 256-çekirdek sistem üzerinde işletilmiş hepsinden-hepsine algoritması Şekil 5-7 ile gösterilmiştir.



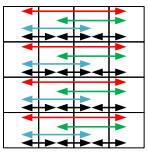
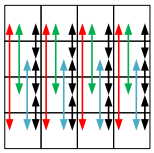
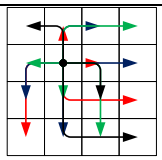
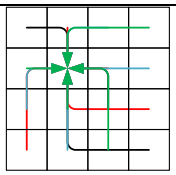
Şekil 5-7: Hepsinden hepsine algoritması (a) Satır veri değişimi, (b) satır veri değişimi (c) sütun veri değişimi (d) satır çoklu gönderme (e) sütun veri değişimi

5.1.4. Genelleştirilmiş Toplu iletişim yaklaşımı

Tez kapsamında, altı adet temel iletişim örüntüsü önerilmiştir ve daha büyük toplu iletişim algoritmaları söz konusu temel örüntülerin farklı örüntülerde özyineli ve tekrarlı olarak uygulanması ile elde edilir. Önerilen algoritmalar belirlenebilir olarak tanımlanmıştır ve adımların hiçbiri herhangi bir yönlendirme işlemine ihtiyaç duymazlar ve bu durum örüntülerin, en büyük sorunu yönlendirme ve ara düğümlerde verinin saklanması olan fotonik sistemlere uygulanmasını elverişli hale getirmektedir. Önerilen algoritmalar seçmeli dalga boyu fotonik sistemler kullanılarak tasarlanmıştır bununla birlikte örüntüler geleneksel elektrik tabanlı sistemlere de uygulanabilir.

Çalışma kapsamında tanımlanan temel örüntülerin bütünü ölçeklenebilir ve etkin örüntülerdir ve standart toplu iletişim algoritmalarının birçoğuna uygulanabilmektedir. Bununla birlikte, geliştiriciler temel örüntüleri tekil veya bir arada kullanarak yazılım sorununa özelleşmiş iletişim algoritmalarını tanımlayabilmektedirler.

Koşut sistemlerde sık kullanılan bir grup toplu iletişim algoritmasının tek öbek üzerinde ($k = 1$) uygulanması Çizelge 5-2 ile gösterilmiştir. Bu algoritmaların daha geniş ağ topolojilerinde ($k > 1$) uygulanması için gerekli temel örüntü adımları, sayıları ve maliyetleri Çizelge 5-3 ile gösterilmiştir.

	ADIM-1	ADIM-2
All-to-all broadcast, Sparse/Complete exchange, All-reduce		
One-to-all broadcast, scatter		
Reduce,Gather		

Çizelge 5-2: Temel örüntüler kullanılarak tanımlanmış toplu iletişim algoritmaları ve 4x4 öbek üzerinde uygulanmaları

All-to-All Broadcast, Sparse/Complete Exchange, All-Reduce,	$(k-1)G_R + (E_R + E_C) + (k-1)(M_R + E_C)$
One-to-All Broadcast, Scatter	$(k)S_{CDN}$
Reduce, Gather	$(k)G_{CDN}$

Çizelge 5-3: Temel örüntüler kullanılarak tanımlanmış toplu iletişim algoritmaları için gerekli adımlar (k > 1)

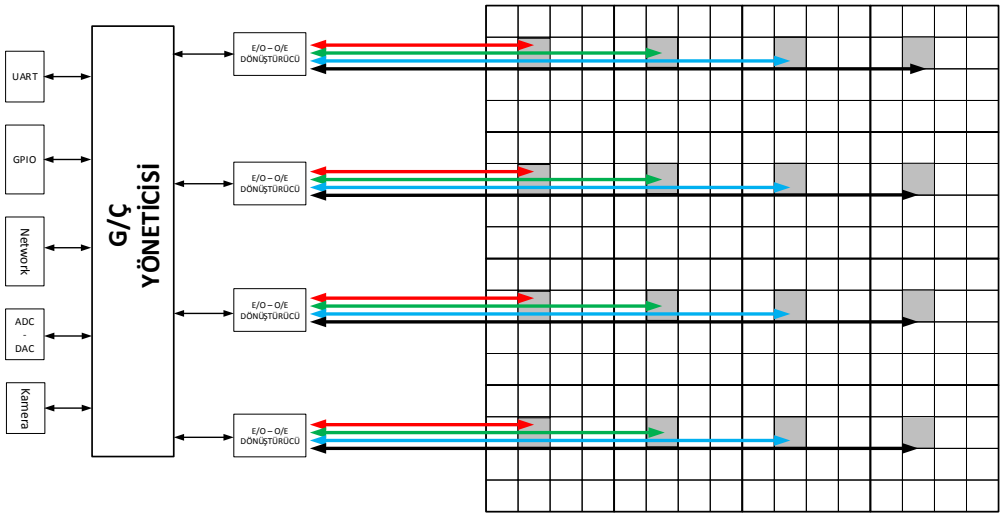
5.2. G/Ç Yönetimi

Yapılan çalışmada, duyarga ve eyleyiciler ile birlikte çalışan ve bunlardan toplanan veriyi, periyodik olarak dağıtan bir bileşen katmanı tasarlanmıştır. Bu katman, giriş/çıkış (G/Ç) yöneticisi olarak tanımlanmıştır. G/Ç yöneticisinin temel işlevi duyargalardan gelen verileri sayısal ve elektronik olarak topladıktan sonra, bu verileri fotonik olarak işlemci yongasına dağıtmaktır. Bu bileşene veri sağlayan duyargaların çalışması periyodik olmamakla birlikte veriler işlemci kümelerine periyodik olarak dağıtılmaktadır.

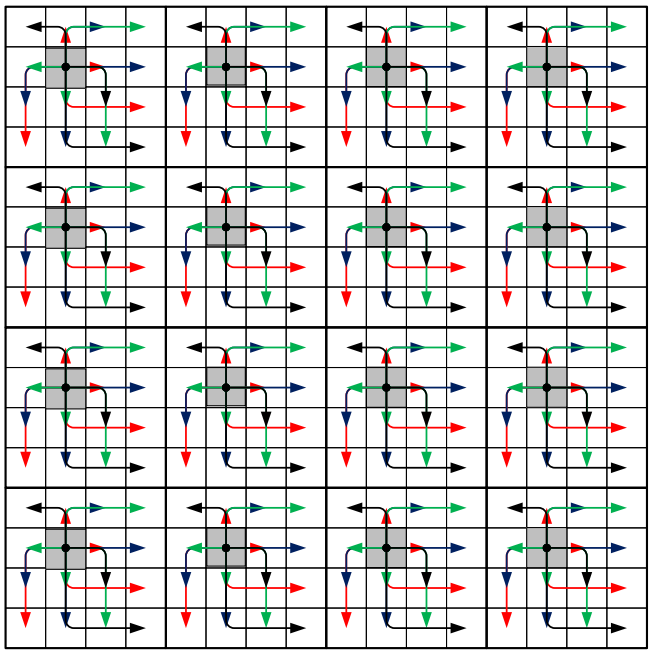
Hasır dokusu topolojisinde, tüm düğümler homojen değildir ve dış düğümlerin her bir boyutunda en az bir kullanılmayan kanal bulunur (x ve y boyutlarının herhangi birindeki en küçük indise sahip düğümler). Tasarımda dış düğümlerin boşta olan kanalları harici G/Ç iletişimi için kullanılmıştır. Bu protokolü sağlamak için, G/Ç yöneticisi elektronik/optik veri dönüşümü sağlayan elektronikten optiğe ve optikten elektroniğe (E/O-O/E) dönüştürücü bileşenlere ihtiyaç duymaktadır. Bu bileşenler, duyarga verilerinin optik sinyallere kodlanarak dalga kılavuzlarına iletilmesini sağlarlar. Önerilen yaklaşımda, her bir işlemci kümesi satırı için bir E/O-O/E dönüştürücüye ihtiyaç duyulmaktadır. $4^k x\ 4^k$ hasır doku için $4k-1$ adet dönüştürücü kullanılmaktadır.

Önerilen dağıtım modeli iki adımdan oluşmaktadır. İlk adımda, G/Ç yöneticisi, her bir işlemci kümesi üzerindeki merkezi düğüme duyarga verisini iletir. Burada WDM ile dalga boyu anahtarlama yaklaşımından faydalanılarak, bir işlemci kümesi satırı üzerindeki tüm merkezi düğümlere eş zamanlı iletim sağlanabilmektedir. Bu bağlamda, satır üzerindeki

her bir $4kx\ 4k$ bir hasır doku için her satırda $4k-1$ adet dalga boyu grubuna gerek duyulmaktadır. İşlemci kümesi düzeyinde farklı dalga boylarının kullanılması, aynı zamanda görüntü verisi gibi dağıtılabilir verilerde, duyarga verisinin işlemci kümesi düzeyinde dağıtılmasına olanak sağlar. İletişimin bir sonraki adımı, verinin işlemci kümesi düzeyinde dağıtılması adımıdır. Bu adımda, kullanıma bağlı olarak birinden hepsine gönderme veya hepsinden hepsine seyrek gönderme seçilebilir. 256 işlemci için duyarga verisi iletimi veri yayılımı Şekil 5-8 ile ve verinin işlemcilere dağıtımı Şekil 5-9 ile gösterilmiştir.



Şekil 5-8: Fotonik yonga üzerinde duyarga verisi yayılımı



Şekil 5-9: Fotonik yonga üzerinde duyarga verisi dağıtımı

5.3. Görev Zamanlama

Bu çalışmada, G/Ç iletişimi, düğümler arası iletişim ve işlem süreçlerini bir arada ele alarak yöneten zaman odaklı bir görev yönetim yaklaşımı geliştirilmiştir. Geliştirilen yaklaşım zaman odaklı ve ele geçirmesiz bir algoritmadır ve döngüsel kod işletimi görev zamanlama algoritmasına da uygulanabilmektedir [3]. Bu yaklaşımda, işlemcinin çalışma zamanı eşit zaman çerçevelerine bölünerek, iletişim ve işlem görevlerine belli kurallar doğrultusunda atanması veya dağıtılmasını temel almaktadır.

Önerilen iletişim yaklaşımında tüm iletişim örüntüleri belirlenebilir ve önceden tahmin edilebilir tasarlandıkları için, görev yönetimi katmanı, bir iletişim işleminin en kötü koşullarda ne zaman tamamlanacağını bilmektedir. Bu bağlamda, önerilen görev yönetimi yaklaşımında döngüsel işletim yaklaşımına uygun biçimde G/Ç ve iletişim görevleri de birer periyodik görev olarak tasarlanıp yönetilebilmektedir. İşleme ayrılmış zaman aralıkları çevresel verilerin işlenmesi ve hesaplanması gibi bilinen işlem görevlerinin işletilmesine ayrılmıştır. Bu zaman aralıklarındaki görev yönetimi, bir koşut programlama veya çoklu-görev yapısı örüntüleriyle yapılabilir.

G/Ç, iletişim ve işlem süreçlerinin bir arada yönetilmesi, tasarlanan gerçek zamanlı sistemin işlem ve iletişim katmanlarında zamansal olarak tamamen tahmin edilebilir ve yönetilebilir bir sistem olmasını sağlamaktadır. Bununla birlikte, bu yapı iletişim ve işlem görevlerinin zamansal düzeyde üst üste gelmesine de olanak verir. Geliştirilen yaklaşımda, iletişim sürecinde kullanılan bileşenler seçilen örüntü doğrultusunda yapılandırıldıktan sonra, işlemci iletişim sürecinin tamamlanmasını beklemeden diğer işlemlere anahtarlanabilir. Bu bağlamda, sistemin zaman boyutunda üç adet eksen bulunur: İşlem, İletişim ve G/Ç zaman eksenleri. Önerilen yaklaşımda, tüm zaman eksenleri aynı tür ve uzunluktaki zaman eksenlerine bölünmüşlerdir ve bu üç zaman ekseni zaman boyutunda üst üste gelmiş olarak çalışırlar. Görev yapılandırmaları her bir zaman aralığının başında gerçekleştirilir ve önerilen yapıda bir görev birden fazla zaman aralığında çalışabilir.

Bilindiği üzere, gerçek zamanlı bir görev eşitlik (1) ile tanımlandığı üzere zamansal davranış olarak periyodik, aperiyodik veya düzensiz olabilir. Geliştirilen görev zamanlama algoritması, her bir zaman aralığının başında tetiklenen zamansal kesilmelere göre görevlerin yönetilmesine dayanır. Bu bağlamda, periyodik gerçek zamanlı görevlerin yönetilmesi, zaman aralıklarının ve periyodik görev zamanlarıyla eş zaman uyumlaması sağlanarak bu yapıya kolaylıkla uygulanabilir. Bununla birlikte, aperiyodik görevlerin

kullanıldığı olay odaklı bir sistemin önerilen görev yönetimine uyarlanabilmesi için bir örneklendirme yaklaşımı önerdik. Bu yaklaşımda, sistem üzerinde çalışan bir olay mekanizması bulunmaz, fakat çevresel duyargalar periyodik olarak sistem tarafından taranarak olay oluşmasına sebep verecek eşik değerindeki veriler ilgili zaman aralığında sistem tarafından ele alınır.

Gerçek zamanlı bir sistemde, sisteme gelecekte yeni görev ve işlevlerin eklenebilmesi ve diğer güvenlik sorunları doğrultusunda boş zaman aralıkları bırakılır [3]. Bu doğrultuda geliştirilen sistem yönetimi algoritmasında bazı zaman aralıkları boş bırakılmıştır. Bununla birlikte, sistemde görev anahtarlamalar sadece ilgili zaman aralıklarının başında tetiklendiği için, ilgili zaman aralığı bitmeden tamamlanan görevler de ayrıca zaman payları oluşturabilirler.

Bir gerçek zamanlı göreve ait zamanlama bileşenleri temel düzeyde eşitlik (2) ile tanımlanabilir. Bu tanım hem periyodik ve düzensiz hem de aperiyodik görevleri tanımlamaktadır. Tasarlanan görev zamanlama algoritması tüm görevleri periyodik olarak ele almakta ve aperiyodik görevleri örnekleme yaklaşımı ile belli bir periyotla çalışmalarını sağlamaktadır. Bu doğrultuda eşitlik (2) ile verilen gerçek zamanlı görev tanımı periyodik görevler için eşitlik (8)'deki gibi sadeleştirilebilir.

$$t_i = \{r_i , e_i, D_i, p_i\} \tag{8}$$

Önerilen görev yönetimi işlem ve iletişim görevlerini bir arada yönetmektedir ve görevler birden fazla zaman ekseninde bağımsız veya eş zamanlı olarak işletilmektedirler. Bu bağlamda, eşitlik (8) ile gösterilen gerçek zamanlı görev tanımı, herhangi bir T görev grubu için ($t_i \in T,\ i \in [1, n]$) , önerilen algoritma eşitlik (9) 'daki gibi genişletilebilir.

$$\begin{gathered} t_i = \{r_i , e_i, D_i, p_i, E_i\} \\ i \in \{iletişim, gç, işlem\}, E_i \neq \emptyset \end{gathered} \tag{9}$$

Burada E_i eksen bilgisini gösterir ve önerilen sistemde üç adet zaman ekseni tanımı bulunur: $E_{iletişim}$, $E_{gç}$, $E_{işlem}$. Burada tanımlanan herhangi bir görev en az bir eksende hizmet vermek üzere üç görev zamanını etkileyebilir. Önerilen görev zamanlama algoritmasında iki çeşit görev bulunmaktadır: işlem, iletişim görevleri. İletişim görevleri G/Ç veya düğümler arası iletişim görevleri olarak ayrılmalarıyla birlikte görev zamanlama davranışlarının benzerlikleri doğrultusunda burada tek bir görev tanımı üzerinden ele alınmışlarıdır.

İşlem görevleri bilinen gerçek zamanlı görevlere karşılık gelir ve sadece $E_{işlem}$ ekseninde çalışırlar. Bu doğrultuda işlem görevi ($pt_i \in T,\ i \in [1,n]$) eşitlik (12) ile tanımlanabilir.

$$pt_i = \{r_i\ , e_i, D_i, p_i, E_{işlem\ i}\} \tag{12}$$

İletişim görevleri aynı anda üç eksende çalışan görevler olabilirler. Bilindiği üzere önerilen sistem mimarisi iletişim düzeyinde sadece fotonik tabanlı ağ yapısını kullanmaktadır. Fotonik tabanlı bir iletişim üç fazdan oluşmaktadır: kurulum, iletişim ve veri dönüşüm/aktarım işlemleri. Önerilen yapıdan iletişim görevleri bu alt yapı doğrultusunda tanımlanmıştır. Görev zamanlama algoritmasında her bir iletişim adımı için, birbirlerine bağımlı üç görev tanımlanmıştır ve bu görevler birbirlerini sıralı olarak takip ederek çalışırlar: kurulum görevi, veri iletişimi görevi ve veri aktarım görevi ve birbirleriyle ilişkili üç iletişim görevi bir iletişim adımını oluşturur (CS_i). Bu yapıda bu görevlerin zamansal davranışları çağrılan görev işleri arasında dağıtılan veriye bağlı olarak farklılıklar gösterebilir. Dolayısıyla bu yapıda alt görevlerin değil iletişim adımının periyodu bulunur. Önerilen iletişim görevleri eşitlik (13) ile tanımlanmıştır.

$$\begin{aligned} CS_i &= \{ct_{k_i}, ct_{vi_i}, ct_{va_i}, p_i\} \\ ct_{k_i} &= \{r_{k_i}\ , e_{k_i}, D_{k_i}, p_{k_i}, E_{k_i}\},\ \ E_{k_i} \in E_{işlem} \cap \{E_{iletişim}, E_{gç}\} \\ ct_{vi_i} &= \{r_{k_i} + e_{k_i}\ , e_{vi_i}, D_{vi_i}, p_{vi_i}, E_{vi_i}\},\ \ E_{vi_i} \in \{E_{iletişim}, E_{gç}\} \\ ct_{va_i} &= \{r_{vi_i} + e_{vi_i}, D_{va_i}, p_{va_i}, E_{va_i}\},\ \ E_{va_i} \in E_{işlem} \cap \{E_{iletişim}, E_{gç}\} \end{aligned} \tag{13}$$

Burada anlaşılacağı üzere birbirlerine bağımlı üç gerçek zamanlı görev bir iletişim adımını tanımlar ve birden fazla iletişim adımı bir araya gelerek bir toplu iletişimi oluştururlar. Bir kurulum görevi tamamlandıktan sonraki herhangi bir zamanda iletişim süreci başlatılabilir, yalnız bu görevin nasıl başlayacağı görev zamanlama düzeyindeki öncelik kararlarına bağlı olarak daha geç bir sistem zamanında başlatılabilir.

Bir iletişim adımı birden fazla alt iletişim adımının sıralı olarak çalıştığı bir toplu iletişim adımı veya yalın bir iletişim adımı olabilir. Yalın bir adım olan iletişim adımı eşitlik (13) ile gösterilen bileşenlerden oluşurken bir toplu iletişim birden fazla alt iletişim adımının sıralı işletiminden oluşur. Bu doğrultuda yukarıda belirtilen iletişim adımı eşitlik (14)'teki gibi genişletilebilir.

$$CS_i = \begin{cases} \{ct_{k_i}, ct_{vi_i}, ct_{va_i}, p_i\}, CS\ yalın\ iletişim\ adımı \\ \{CS_{i0}, CS_{i1}, \dots, CS_{in}\},\ \ CS\ toplu\ iletişim\ adımı \end{cases} \tag{14}$$

Döngüsel kod işletimi tabanlı bir gerçek zamanlı görev zamanlama ile yönetilen bir sistem için görevlerin işlemlerini yapmalarını için belirlenen zaman çerçevesi boyutu (f) ve bir periyottaki tüm işlemlerin tamamlanması için gerekli olan temel zaman çerçevesi periyodu (p_{hyper}) periyotların ortak katlarının en küçüğü (OKEK) kullanılarak eşitlik (15) ile hesaplanabilir [3].

$$p_{hyper} = OKEK(p_1, p_2, \dots, p_n)$$
$$f \geq \max(e_i), i \in [1, n] \quad (15)$$
$$\lfloor p_{hyper}/f \rfloor - (p_{hyper}/f) = 0$$

Önerilen zamanlama yaklaşımı döngüsel kod işletiminin, işlem iletişim ve G/Ç süreçlerini birlikte yürütüldüğü genişletilmiş bir haline uygulanabilmektedir. Önerilen zamanlama algoritmasında, zaman çerçevesi boyutu (f), işlem ekseninde işletilen tüm görevlerinin işletim zamanlarının en büyük değerinden küçük olmayacak bir değer olarak belirlenmektedir ve tüm eksenlere bu çerçeve uygulanmaktadır. İşlem ekseni için temel periyot değeri ($p_{hyper_{işlem}}$) ise işlem ekseninde hizmet veren işlem görevi (pt_i) periyotlarının OKEK değeriyle hesaplanacaktır. İletişim eksenlerindeki görevlerden kurulum ve veri aktarım görevleri işlem ekseninde de çalıştıkları için hesaplanan çerçeve boyutuyla uyumlu olacaklardır. Veri iletim görevleri ise çerçeve boyutuyla uyumlu çalışarak bu çerçevelerin tam katı bir zamanda tamamlanmaktadırlar.

Sistemdeki iletişim adımları tek bir görev gibi ele alınarak her bir iletişim adımı için bir periyot belirlenmektedir. İletişim görevleri işlem süreçlerini doğrudan etkilemedikleri için, bu görevlerin iletişim eksenlerine bağlı olarak ayrı bir temel zaman çerçevesi olacaktır. Bu bağlamda sistemin temel periyodu işlem ve iletişim için hesaplanan temel periyotlardan büyük olanına eşit olacaktır.

Açıklanan yaklaşımlar doğrultusunda, önerilen sistem için zaman çerçevesi boyutu (f) ve temel periyot (p_{hyper}) eşitlik (16)'daki gibi genişletilebilir.

$$\forall t_i \in T_{işlem}, E_{işlem} \in E_i$$

$$\mathrm{CT} = \{CS_0, CS_1, CS_2, \dots, CS_n\}, \mathrm{PT} = \{pt_0, pt_1, pt_2, \dots, pt_n\}$$

$$f \geq \max(e_i), i \in [1, n],\ t_i \in T_{işlem}$$

$$p_{hyper_{işlem}} = OKEK(p_1, p_2, \dots, p_n), \qquad t \in PT \tag{16}$$

$$p_{hyper_{iletişim}} = OKEK\left(p_{CS_0}, p_{CS_1}, \dots, p_{CS_n}\right), \qquad CS \in CT$$

$$p_{hyper} = maks(p_{hyper_{işlem}}, p_{hyper_{iletişim}})$$

$$\left\lfloor p_{hyper}/f \right\rfloor - \left(p_{hyper}/f\right) = 0$$

Başarım çalışması

Burada bahsedilen görev yapısı aşağıda belirtilen görev kümesi için örneklenmiştir.

$\mathrm{T} = \{t_1, t_2, t_3, ct_1, ct_2, ct_3, ct_4, ct_5\}$

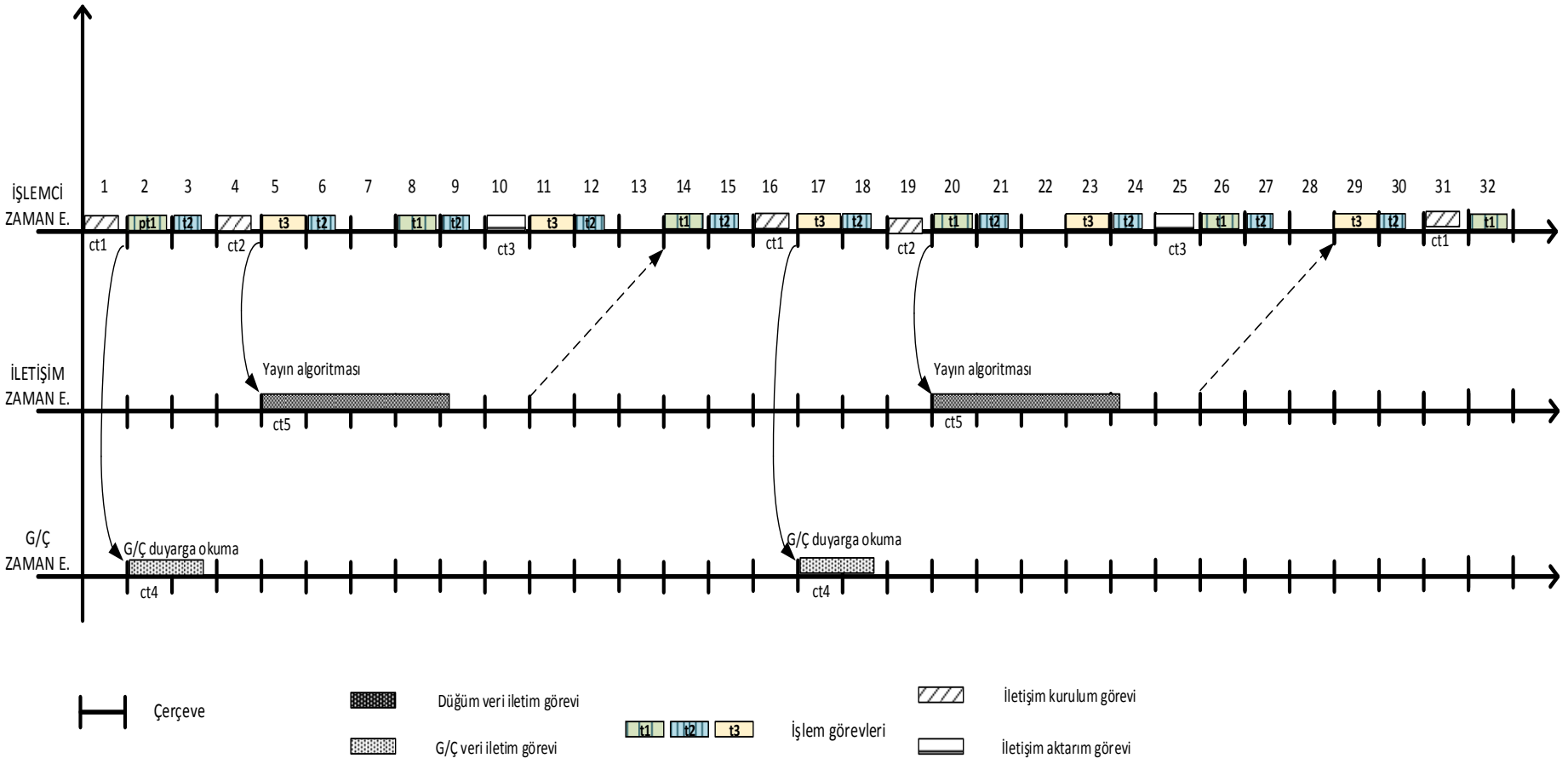
$\mathrm{T}_{işlem} = \{t_1, t_2, t_3, ct_1, ct_2, ct_3\}$

$PT = \{t_1, t_2, t_3\}$

$CT = \{CS_{gç} + CS_{yayin}\}$

$CS_{gç} = \{ct_k = ct_1, ct_{vi} = ct_4\}$

$CS_{yayin} = \{ct_k = ct_2, ct_{vi} = ct_5, ct_{va} = ct_3\}$



Şekil 5-10: Önerilen görev zamanlama yaklaşımı örneği

Verilen başarım çalışması Şekil 5-10 ile örneklenmiştir. Bu senaryoda, sistem için çerçeve boyu (f), işlem ekseninde çalışan görevler içinden ($T_{işlem}$) en yüksek işletim zamanına sahip olana eşit olduğu düşünülmüştür. Bu örnekte bu görevin t_3 olduğu varsayılmıştır.

Sistemde üç adet periyodik görev bulunmaktadır (t_1, t_2, t_3) ve bu görevlerden t_1 ve t_2 görevlerinin periyodu 6 zaman çerçevesi uzunluğunda ve t_3 görevinin periyodunun 3 çerçeve boyu uzunluğunda olduğu varsayılmıştır. Bununla birlikte, sistem 15 çerçeve uzunluğunda periyotlarla G/Ç duyarga verisi okuması yaparak bu veriyi birden-hepsine yayın algoritması ile tüm düğümlere dağıtımını yapmaktadır. Verilen bilgiler doğrultusunda temel çerçeve boyu eşitlik (16) kullanılarak aşağıdaki gibi hesaplanacaktır.

$$p_{hyper_{işlem}} = OKEK(t_1, t_2, t_3) = 6$$

$$p_{hyper_{iletişim}} = OKEK\left(CS_{gç} + CS_{yayin}\right), = 15$$

$$p_{hyper} = maks\left(p_{hyper_{işlem}}, p_{hyper_{iletişim}}\right) = 15$$

Bu senaryoda, G/Ç verisi bellek alanına kopyalanmadan, anahtar yastığından yayın algoritması ile dağıtıldığı düşünüldüğü için $CS_{gç}$ için ct_{va} görevi bulunmamaktadır.

6. BAŞARIM ÇALIŞMALARI VE ANALİZİ

Bu bölümde ortaya konulan fotonik yonga üzeri iletişim tabanlı dağıtılmış bellekli MIMD mimari için ortaya konulan iletişim ve görev zamanlama yaklaşımlarını destekleyici başarım çalışmaları gerçekleştirilmiş ve incelenmiştir. Buradaki çalışmalar iki gruba ayrılmaktadır:

- Hesaplamalara dayalı teorik çalışmalar
- Benzetim çalışmaları

Matematiksel modele dayalı teorik çalışmalarda, tanımlanan iletişim örüntüleri, bu örüntüler için ortaya konulan matematiksel modelleri bağlamında ele alınarak maliyet hesaplamaları yapılmış ve sonuçları incelenmiştir. Benzetim çalışmalarında ise bu kapsamda belirlenen başarım çalışması senaryolarının çalışma kapsamında gerçekleştirilen benzetim sistemi üzerinde gerçekleştirilmiştir ve ortaya çıkan benzetim sonuçları analiz edilmiştir.

6.1. Teorik Çalışmalar

Gerçek zamanlı bir sistemin yaşam döngüsü çevresel aygıtlardan durum okuması yapılması, bu verilerin ve diğer sistem verilerinin işlenmesi ve işlenen veriler doğrultusunda sistemin bulunduğu çevrede veya sistemde yapılacak değişiklerin belirlenmesi olarak genelleştirilebilir. Bu doğrultuda, çok işlemcili bir gerçek zamanlı sistem için düğümler arası ve G/Ç aygıtlarla yapılacak iletişimlerin etkinliği, sistemin başarımını doğrudan belirleyecektir.

Tez kapsamında, gerçek zamanlı sistemler için etkin ve tahmin edilebilir bir iletişim örüntüsü grubu önerilmiştir ve önerilen iletişim örüntüsü G/Ç verilerinin okunması ve tekrar G/Ç sürücülerine dağıtılması süreçlerine uyarlanmıştır. Önerilen iletişim mimarisinin başarımını değerlendirmek üzere, iki teorik çalışma belirlenmiştir:

- G/Ç aygıtlarından düğümlere veri iletişimi
- Düğümler arasında hepsinden-hepsine yayın algoritmasının işletilmesi

G/Ç aygıt iletişimi, duyarga sürücülerinden verinin okunması ve sistemdeki düğümlere yayın algoritması ile dağıtılması işlemlerini kapsar. Burada G/Ç aygıtlarından fotonik ortam üzerinden verinin okunması süreci aynı veri boyutu için sabit olduğu için, bu adımın etkinliğini yayın algoritmasının performansı belirlemektedir. Hepsinden hepsine yayın algoritması sistemdeki her bir düğümün sistemdeki diğer tüm düğümlere iletilmek üzere bir iletisinin olduğu senaryodur. Bu algoritma koşut sistemlerdeki en maliyetli iletişim protokollerinden biridir ve bu sebeple düğümler arası iletişimde sistem performansının ölçülmesi için bu algoritma tercih edilmiştir.

Bu kısımdaki teorik hesaplamalar analitik düzeyde genel devre anahtarlama yaklaşımı üzerinden ele alınmıştır. Bu bağlamda optik ve elektriksel dönüşümlerin iletim başlangıç maliyetleri içinde olduğu varsayılmıştır ve hesaplamalar eşitlik (7) üzerinden yapılmıştır.

Performans çalışmalarının yapıldığı sistemin, önerilen fotonik NoC iletişim altyapısına sahip düğümlerden oluşan ve hasır doku ağ topolojisine sahip çok işlemcili bir sistem olarak kabul edilmektedir. Bu sistemdeki tüm işlemciler homojen işlemcilerdir ve her bir işlemcinin 1 GFLOPS işlem kapasiteye sahip olduğu varsayılmaktadır. Önerilen sistem senaryosunda fotonik anahtarlar eş zamanlı olarak dört dalga boyunun kullanılmasına olanak veren 4-port fotonik anahtar olduğu düşünülmektedir. Önerilen sistemde her bir dalga boyunun 64 Gb/s bant genişliğine sahip olduğu var sayılmaktadır. İleti başlatma maliyetlerinin, her adımda alınan iletinin işlemcilerde işlenmesi ve iletinin başka bir

düğüme aktarılması durumunda ilgili giriş yastıklarından çıkış yastıklarına iletilmesi maliyetlerine büyük oranda bağımlı olduğu bilinmektedir. Bu bağlamda, ileti başlatma maliyetleri optimal bir yaklaşımla yaklaşık 50 nano saniye (ns) olarak varsayılmıştır.

Çalışma için tanımlanan sistem parametreleri Çizelge 6-1 ile gösterilmiştir.

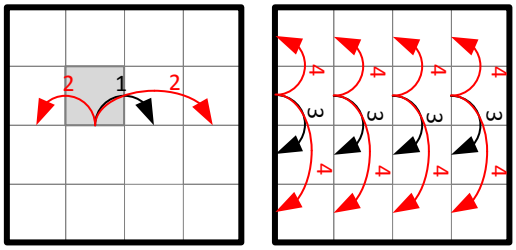
Parametre	**Değer**
Sistem topolojisi	2D Hasır doku
Anahtar tipi	4-port fotonik anahtar
Fotonik bant genişliği	64Gb/s (τ = 64bit veri iletimi için 1ns)

Çizelge 6-1: Başarım çalışması için tanımlanan sistem parametreleri

6.1.1. Yayın algoritması ile veri dağıtımı

Belirtildiği üzere, önerilen yaklaşımda, G/Ç verisinin hedef düğüme iletilmesi, tüm dalga boylarının tek bant genişliği olarak kullanılarak G/Ç yöneticisi tarafından tek seferde iletilmesi işlemine dayanır. Dolayısıyla bu adımın maliyeti ileti boyuna bağlı biçimde sabittir ve G/Ç verisinin düğümlere dağıtılması adımının başarımını duyarga verisinin hedef düğümden diğer tüm düğümlere iletilmesini sağlayacak olan yayın algoritmasının başarımı belirlemektedir.

Yapılan başarım çalışması kapsamında, önerilen yayın algoritması bilinen boyut sıralı yayın algoritması ile karşılaştırılmıştır. Boyut sıralı iletimde, ileti önce X boyutundaki tüm düğümlere dağıtılır ve her adımda mesaj ikiye bölünerek logaritmik sayıda adımda yatay eksendeki tüm düğümlere iletinin dağıtılması sağlanır. Bir sonraki fazda, X boyutundaki her bir düğüm tarafından tüm Y boyutundaki düğümlere eş zamanlı olarak X dağıtımındaki örüntünün dikey eksende uygulanması ile sağlanır ve yatay eksendeki dağıtımla eşit sayıda adım sonucunda tüm düğümlere iletinin ulaştırılması sağlanır. Bahsedilen boyut sıralı yayın algoritmasının 16-çekirdekli bir sistem üzerinde tek bir kanal (dalga boyu) kullanılarak uygulanması Şekil 6-1 ile gösterilmiştir. Şekil üzerindeki sayılar ilgili dağıtım adımını göstermektedir.



Şekil 6-1: 16 düğüm tek kanal kullanılarak boyut sıralı yayın algoritmasının uygulanması. (a) yatay eksende dağıtım (b) dikey eksende dağıtım

Bilindiği üzere fotonik iletişimde kullanılan dalga boyları eş zamanlı farklı verilerin iletilmesinde (ayrı iletişim kanalları) veya aynı verinin daha yüksek bant genişliği ile iletilmesi (aynı kanalın koşut veri yolları) olarak kullanılabilmektedir. Bu bağlamda kullanılan dalga boyları ihtiyaca göre iletim kanalları veya bant genişliği olarak kullanılabilirler.

Önerilen fotonik anahtar eş zamanlı dört yönde trafiği destekleyebilmektedir. Boyut sıralı algoritmada önce yatay eksen sonra dikey eksen boyutunda iletim yapıldığı için, yatay ve dikey eksende eş zamanlı yürütülen iletimler çakışmadan iletilebilirler. Bu doğrultuda, boyut sıralı olarak dağıtılan ileti ikiye bölünerek yarısı yatay eksende dağıtılırken, diğer yarısı eş zamanlı olarak dikey eksende dağıtılabilir. Bir sonraki adımda bölünen iletinin ilk yarısı dikey eksen kanallarını kullanırken, ikinci yarı yatay eksen kanallarını kullanacaktır ve iletiler çakışmadan düğümler iki parça olarak toplanabilecektir.

Bu iki durum doğrultusunda, boyut sıralı algoritma önerilen fotonik anahtar üzerinde dört farklı yolla uygulanabilir ve başarım çalışması kapsamında önerilen yayın algoritması ile birlikte toplam beş farklı yayın algoritması tanımlanmıştır:

- Uyg.1 (1Kanal/KG-4): Tüm dalga boylarının tek bir iletiye dalga boyu olarak atandığı boyut sıralı yayın algoritması
- Uyg-2 (4Kanal/KG-1): Her dalga boyunun ayrı bir kanal gibi davrandığı boyut sıralı yayın algoritması
- Uyg.3 (1Kanal/KG-4/BS): Tüm dalga boylarının tek bir iletiye dalga boyu olarak atandığı boyut sıralı yayın algoritmasında iletinin ikiye bölünerek eş zamanlı olarak yatay ve dikey eksende iletildiği yaklaşım.

- Uyg.4 (4Kanal/KG-1/BS): Her dalga boyunun ayrı bir kanal gibi davrandığı boyut sıralı yayın algoritmasında iletinin ikiye bölünerek eş zamanlı olarak yatay ve dikey eksende iletildiği yaklaşım.
- Uyg.5(Önerilen): Bölüm 5.1.2'de önerilen yayın algoritması

Önerilen algoritmalar için hesaplanan iletim maliyetleri Çizelge 6-2 ile gösterilmiştir.

Uyg.1 (1Kanal/KG-4)	$2\lceil 2k \log_3 2\rceil(\propto_o + {L\tau_o}/{4})$
Uyg-2 (4Kanal/KG-1)	$2\lceil k\ log_3\ 2\rceil\ (\propto_o + L\tau_o)$
Uyg.3 (1Kanal/KG-4/BS)	$2\lceil 2k\ log_3\ 2\rceil\ (\propto_o + {\frac{L}{2}\tau_o}/{4})$
Uyg.4 (4Kanal/KG-1/Yİ)	$2\lceil k\ log_3\ 2\rceil(\propto_o + \frac{L}{2}\tau_o)$
Uyg.5(Önerilen)	$\lceil k\rceil\ (\propto_o + L\tau_o)$

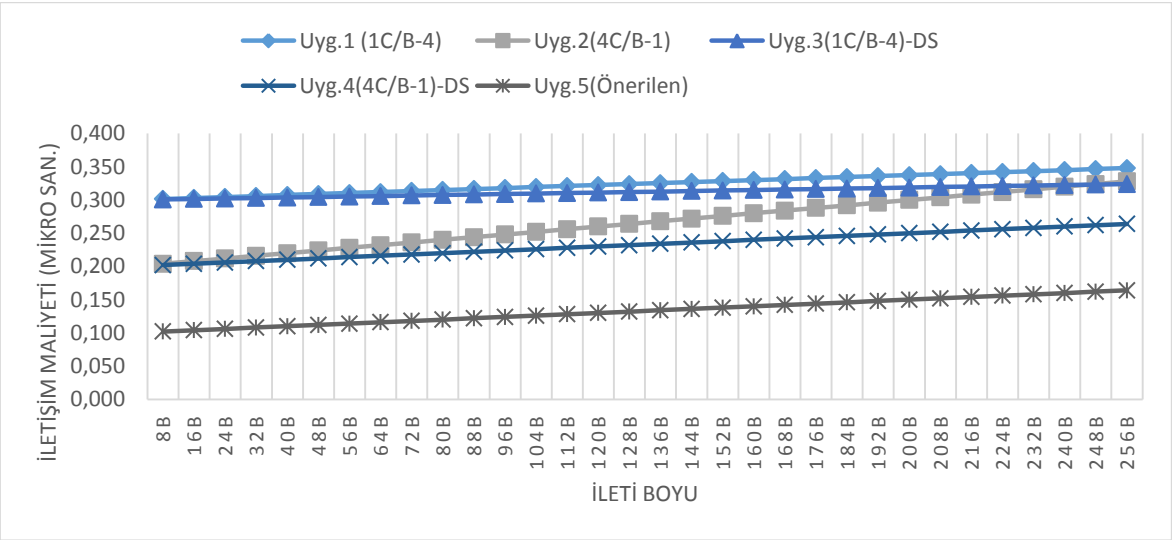
Çizelge 6-2: Yayın algoritmaları ve maliyetleri

Yapılan başarım çalışması, çalışma kapsamında 1 KB ve daha düşük boyuttaki verilerin dağılımında, önerilen algoritmanın seçilen algoritmalar arasında en iyi sonucu verdiğini göstermektedir. 1KB'dan daha büyük verilerin iletiminde sistem başarımı için bant genişliği, iletim için gerekli adım sayısından daha belirleyici olmaya başlamaktadır dolayısıyla bu ve daha büyük veriler için en yüksek bant genişliğine sahip olan Uyg.3 daha iyi sonuçlar vermektedir. 256 çekirdekli bir sistem üzerinde farklı ileti boyutları için yayın algoritması maliyetleri Çizelge 6-3 ile verilmiştir. 256 çekirdekli bir sistem için 8B ile 256B veriler için yayın algoritması maliyet grafiği Şekil 6-2 ile 512B ile 16KB arasındaki ileti boyutları için yayın algoritması maliyet grafiği Şekil 6-3 ile ve 64KB-4MB arasındaki ileti boyutları için yayın algoritması maliyet grafiği Şekil 6-4 ile verilmiştir. İletişim maliyeti μs olarak sunulmuştur.

Veri boyu	G/Ç Maliyeti	Uyg.1 (1Kanal/KG-4)	Uyg.2 (4Kanal/KG-1)	Uyg.3 (1Kanal/KG-4)-BS	Uyg.4 (4K/KG-1)-BS	Uyg.5 (Önerilen)
8B	0.000	0.302	0.204	0.301	0.202	0.102
16B	0.001	0.303	0.208	0.302	0.204	0.104
24B	0.001	0.305	0.212	0.302	0.206	0.106
32B	0.001	0.306	0.216	0.303	0.208	0.108
40B	0.001	0.308	0.220	0.304	0.210	0.110
48B	0.002	0.309	0.224	0.305	0.212	0.112
56B	0.002	0.311	0.228	0.305	0.214	0.114
64B	0.002	0.312	0.232	0.306	0.216	0.116
72B	0.002	0.314	0.236	0.307	0.218	0.118
80B	0.003	0.315	0.240	0.308	0.220	0.120
88B	0.003	0.317	0.244	0.308	0.222	0.122
96B	0.003	0.318	0.248	0.309	0.224	0.124
104B	0.003	0.320	0.252	0.310	0.226	0.126
112B	0.004	0.321	0.256	0.311	0.228	0.128
120B	0.004	0.323	0.260	0.311	0.230	0.130
128B	0.004	0.324	0.264	0.312	0.232	0.132
136B	0.004	0.326	0.268	0.313	0.234	0.134
144B	0.005	0.327	0.272	0.314	0.236	0.136
152B	0.005	0.329	0.276	0.314	0.238	0.138
160B	0.005	0.330	0.280	0.315	0.240	0.140
168B	0.005	0.332	0.284	0.316	0.242	0.142
176B	0.006	0.333	0.288	0.317	0.244	0.144
184B	0.006	0.335	0.292	0.317	0.246	0.146
192B	0.006	0.336	0.296	0.318	0.248	0.148
200B	0.006	0.338	0.300	0.319	0.250	0.150
208B	0.007	0.339	0.304	0.320	0.252	0.152
216B	0.007	0.341	0.308	0.320	0.254	0.154
224B	0.007	0.342	0.312	0.321	0.256	0.156
232B	0.007	0.344	0.316	0.322	0.258	0.158
240B	0.008	0.345	0.320	0.323	0.260	0.160
248B	0.008	0.347	0.324	0.323	0.262	0.162
256B	0.008	0.348	0.328	0.324	0.264	0.164
512B	0.016	0.396	0.456	0.348	0.328	0.228
1KB	0.032	0.492	0.712	0.396	0.456	0.356
2KB	0.064	0.684	1.224	0.492	0.712	0.612
4KB	0.128	1.068	2.248	0.684	1.224	1.124
8KB	0.256	1.836	4.296	1.068	2.248	2.148
16KB	0.512	3.372	8.392	1.836	4.296	4.196

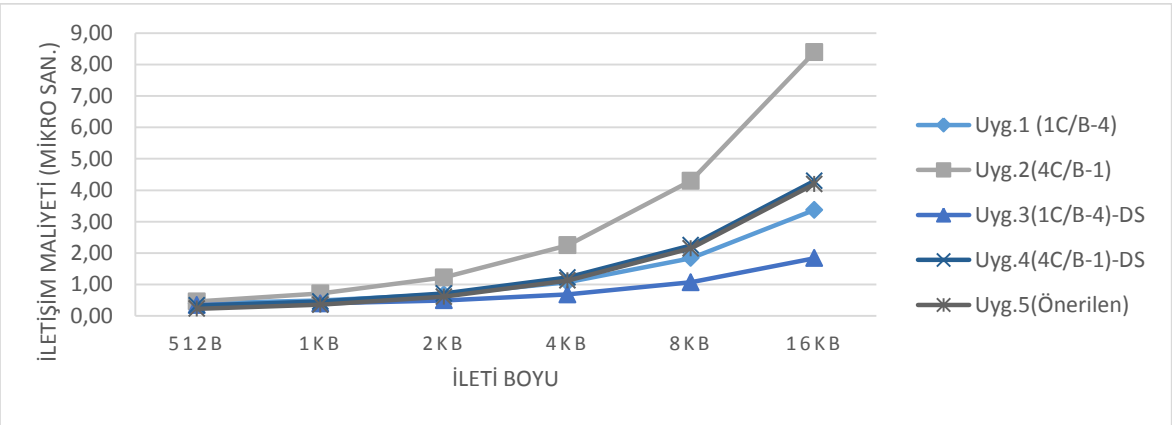
64KB	2.048	12.588	32.968	6.444	16.584	16.484
256KB	8.192	49.452	131.272	24.876	65.736	65.636
512KB	16.384	98.604	262.344	49.452	131.272	131.172
1MB	32.768	196.908	524.488	98.604	262.344	262.244

Çizelge 6-3: 8B-1MB veri boyutları için yayın algoritması sonuçları



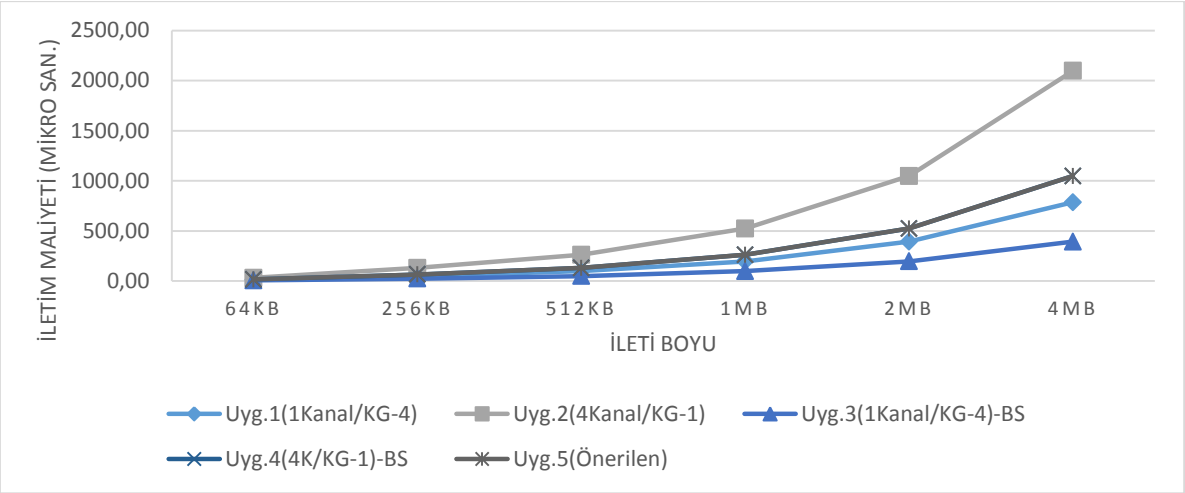
Şekil 6-2: 8B-256B arası veriler için iletişim maliyetleri

(k = 2, $\propto_o$= 50ns)



Şekil 6-3: 512B-16KB ileti boyları için yayın algoritması iletim maliyetleri

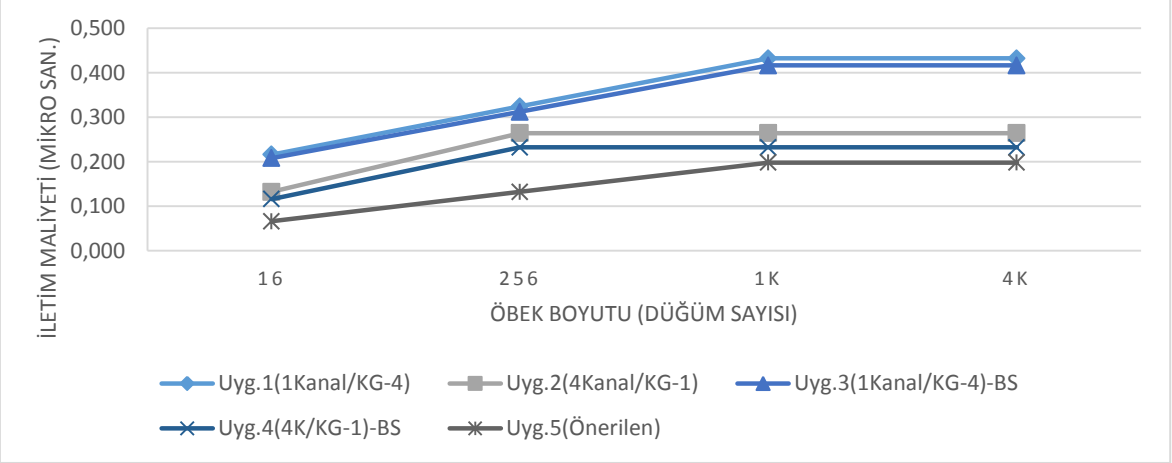
(k = 2, $\propto_o$= 50ns)



Şekil 6-4: 64KB-4MB ileti boyları için yayın algoritması iletim maliyetleri

(k = 2, $\propto_o = 50ns$)

Değişken öbek boyutları için, yapılan çalışma 256B verinin dağıtımı için önerilen yayın algoritması seçilen uygulamalar arasında en iyi sonucu vermektedir. 16-4K çekirdek sayılı sistemler için 256B verinin yayın algoritması ile dağıtım maliyetleri Şekil 6-5 ile gösterilmiştir.

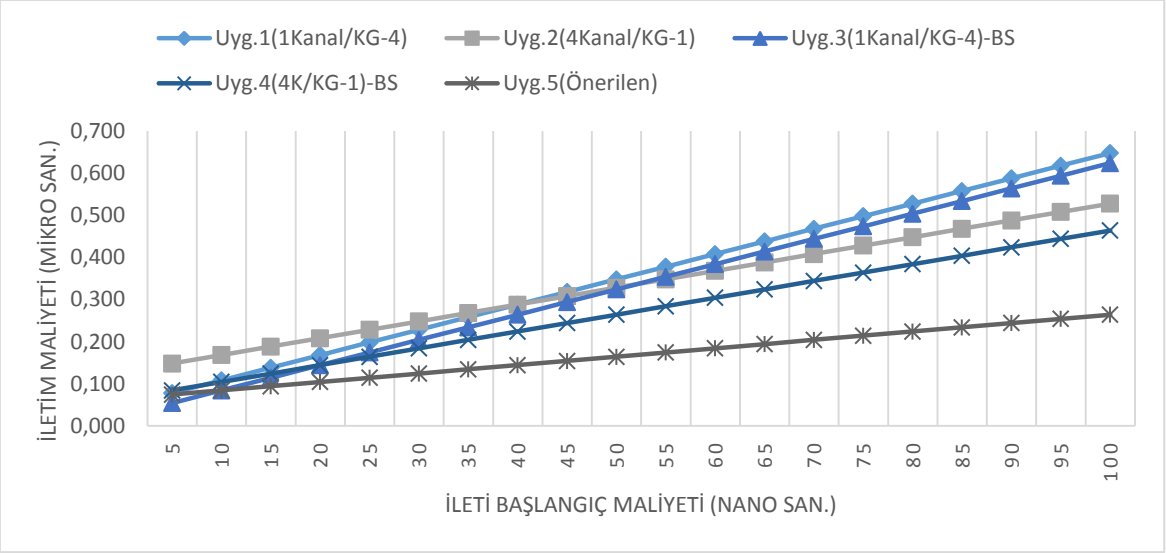


Şekil 6-5: Farklı öbek boyutları için yayın algoritması maliyetleri

(L = 256B, $\propto_o = 50ns$)

Belirtildiği üzere, ileti başlangıç maliyeti büyük ölçüde iletilerin işlemcide işlenmesi ve yastıklama maliyetlerin oluşmaktadır. Bu bağlamda, ileti maliyetinin yüksek hızlı bilgisayarlar kullanılarak düşürülebilir, ama bu durum sistemin enerji tüketimini artıracaktır. Bu açıdan, 256B verinin seçilen yayın algoritmaları kullanılarak iletilmesi değişken iletim maliyetleri için incelenmiştir. Yapılan çalışma, 20ns ve yukarı iletim

maliyetleri için önerilen algoritmanın en iyi sonucu verdiğini göstermektedir. Değişken iletim maliyetlerine göre iletim maliyetleri Şekil 6-6 ile gösterilmiştir.

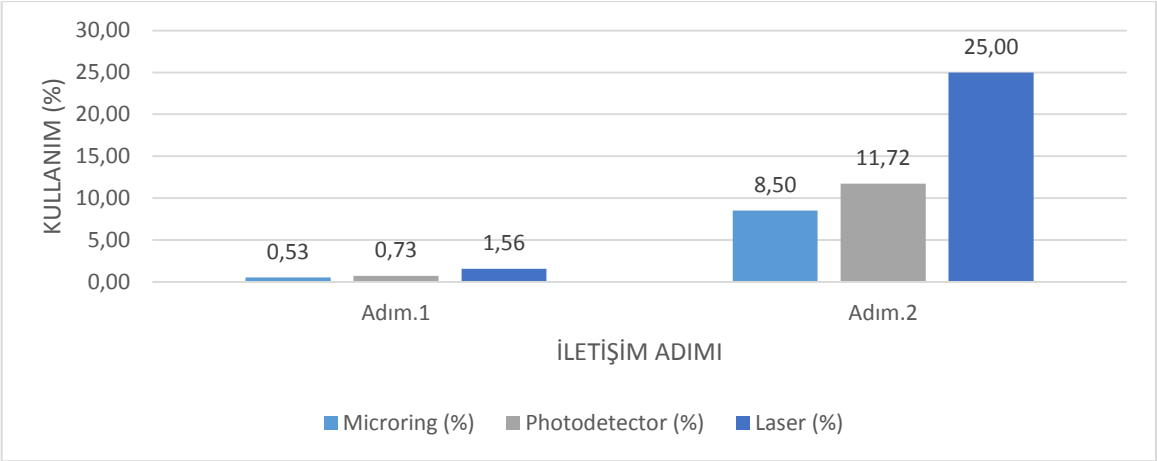


Şekil 6-6: Farklı iletim başlangıç maliyetlerine göre yayın algoritması iletim maliyetleri (L = 256 B)

Enerji tüketimi, gerçek zamanlı sistemler için temel faktörlerden biridir. Önerilen çok işlemcili sistem mimarisi için, toplu iletişiminde tüketilen enerji, sistemin enerji tüketimi için belirleyici bir faktör olacaktır. Enerji tüketimi göstergelerini göstermek için 256 çekirdekli bir sistemdeki yayın algoritması işletimi için, her adımdaki iletişim bileşenlerinin toplamı, kullanılan sayısı ve kullanım yüzdeleri Çizelge 6-4 ve Şekil 6-7 ile gösterilmiştir.

	Microring (Toplam:16384)		**Foton algılayıcı (Toplam:4096)**		**Lazer (Toplam:1024)**	
	Kull.	**Kullanım (%)**	**Kull.**	**Kullanım (%)**	**Kull.**	**Kullanım (%)**
Adım.1	87	0.53	30	0.73	16	1.56
Adım.2	1392	8.50	480	11.72	256	25.00

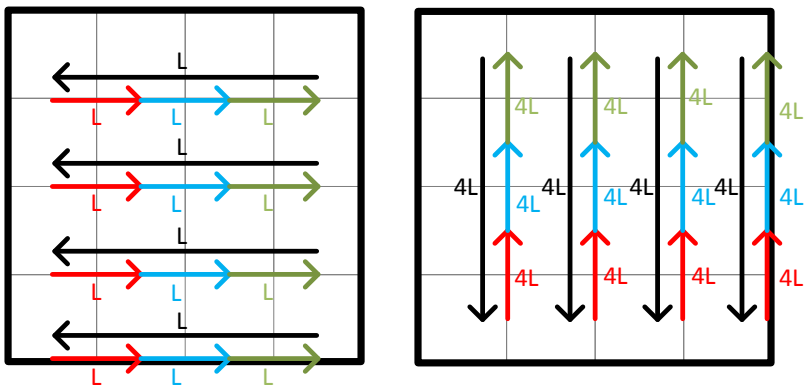
Çizelge 6-4: Önerilen yayın algoritması için fotonik ağ bileşenleri ve bileşenlerinin kullanım oranları



Şekil 6-7: 256 çekirdek üzerinde yayın algoritması işletiminde fotonik ağ bileşenlerinin kullanım oranları

6.1.2. Hepsinden hepsine veri dağıtımı

Hepsinden hepsine algoritması, belirtildiği üzere çok işlemcili bir sistemdeki her düğümün diğer tüm düğümlere iletilmek üzere bir iletisinin bulunduğu toplu iletişim senaryosudur. Hepsinden hepsine algoritması birden fazla yöntemle gerçekleştirilebilir. En bilinen yaklaşımlardan biri boyut sıralı iletim yaklaşımıdır. Bu yaklaşımda, her düğüm, verilerini kendisiyle aynı sıradaki tüm düğümlerine dağıtır; daha sonra her düğüm tüm satır verilerini ilk adımdakiyle benzer bir örüntüyle sütun boyunca dağıtır. Dolayısıyla, bu iletişimin adım sayısı her boyuttaki iletim atlama adımlarının sayısına eşittir. Bu iletişim yaklaşımı 4x4 hasır doku topolojideki 16 düğüm için uygulanması Şekil 6-8 ile gösterilmiştir.



Şekil 6-8: 4x4 hasır doku üzerinde Boyut sıralı hepsinden hepsine iletim gösterimi

(L= ileti boyu)

Bu yaklaşımda yayın algoritması hesaplamalarındaki gibi veri dağıtımı önce yatay eksen sonra dikey eksen boyutunda yapıldığı için, yatay ve dikey eksende eş zamanlı yürütülen

iletimler çakışmadan iletilebilirler ve bu yaklaşımla her düğümdeki gönderilen iletiler ikiye bölünerek yatay ve dikey eksen eş zamanlı dağıtılabilirler. Burada kullanılan hepsinden hepsine algoritmasının yapısı doğrultusunda sistemdeki dört dalga boyu farklı verilerin eş zamanlı iletimi sağlayan kanallar olarak kullanılmıştır.

Bu çalışmada boyut sıralı hepsinden hepsine algoritmasının iki durumu, önerilen hepsinden hepsine algoritması ile karşılaştırılmıştır. Önerilen algoritma adımları benzer şekilde yatay ve dikey eksende boyut sıralı çakışmayan adımlardan oluşmaktadır. Dolayısıyla bu algoritmada da verinin iki parça halinde yatay ve dikey eksende eş zamanlı gönderilmesi sağlanabilir.

Bu bağlamda, Bu çalışmada dört algoritma kullanılmıştır:

- Uyg.1: Boyut sıralı hepsinden hepsine algoritması
- Uyg.2 (BS): Boyut sıralı hepsinden hepsine algoritmasında iletinin ikiye bölünerek eş zamanlı olarak yatay ve dikey eksende iletildiği yaklaşım.
- Uyg.3 (Önerilen): Bölüm 5.1.3'de önerilen hepsinden hepsine algoritması
- Uyg.4 (Önerilen-BS): Bölüm 5.1.3'de önerilen hepsinden hepsine algoritmasında iletinin ikiye bölünerek eş zamanlı olarak yatay ve dikey eksende iletildiği yaklaşım.

Bu algoritmalar için iletim maliyetleri Çizelge 6-5 ile hesaplanabilir.

Uyg.1	$2^{2k+1} \propto_o + (4^{k-1} + 4^{2k-1})M$
Uyg.2 (BS)	$2^{2k+1} \propto_o + (4^{k-1} + 4^{2k-1})\frac{M}{2}$
Uyg.3 (Önerilen)	$(3k-1) \propto_o + \frac{2(4^{2k}) - (4^{k+1}+1)}{3} M$
Uyg.4 (Önerilen-BS)	$(3k-1) \propto_o + \frac{2(4^{2k}) - (4^{k+1}+1)}{3} \frac{M}{2}$

Çizelge 6-5: $4^k \times 4^k$ hasır doku üzerinde hepsinden hepsine algoritmalarının iletim maliyetleri $(M = L\tau_o)$

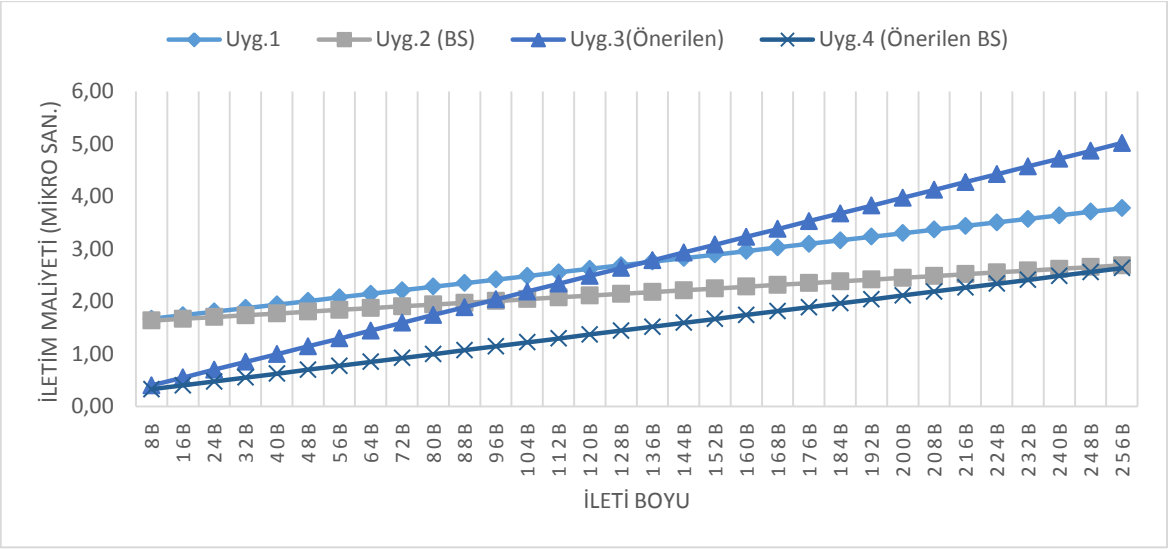
Yapılan çalışma, önerilen algoritmanın 256B ve daha kısa ileti boylarında verilen algoritmalar arasında en iyi sonucu vermektedir. 256B üzeri uzunlukta iletiler için Uyg.2 daha iyi sonuçlar vermektedir. 256 çekirdekli bir sistem üzerindeki farklı ileti boyutları için G/Ç maliyeti ve yayın algoritması maliyetleri Çizelge 6-6 ile verilmiştir. 256 çekirdekli bir sistem için 8B ile 256B veriler için yayın algoritması maliyet grafiği Şekil

6-9 ile 512B ile 16KB arasındaki ileti boyutları için yayın algoritması maliyet grafiği Şekil 6-10 ile ve 64KB-4MB arasındaki ileti boyutları için yayın algoritması maliyet grafiği Şekil 6-11 ile verilmiştir. İletişim maliyeti μs olarak sunulmuştur.

İleti Boyu	Uyg.1	Uyg.2 (BS)	Uyg.3 (Önerilen)	Uyg.4 (Önerilen-BS)
8B	1,67	1,63	0,40	0,32
16B	1,74	1,67	0,55	0,40
32B	1,87	1,74	0,85	0,55
48B	2,01	1,80	1,14	0,70
64B	2,14	1,87	1,44	0,85
80B	2,28	1,94	1,74	1,00
96B	2,42	2,01	2,04	1,14
112B	2,55	2,08	2,34	1,29
128B	2,69	2,14	2,63	1,44
144B	2,82	2,21	2,93	1,59
160B	2,96	2,28	3,23	1,74
176B	3,10	2,35	3,53	1,89
192B	3,23	2,42	3,83	2,04
208B	3,37	2,48	4,12	2,19
224B	3,50	2,55	4,42	2,34
240B	3,64	2,62	4,72	2,49
256B	3,78	2,69	5,02	2,63
512B	5,95	3,78	9,79	5,02
1KB	10,30	5,95	19,32	9,79
2KB	19,01	10,30	38,39	19,32
4KB	36,42	19,01	76,54	38,39
8KB	71,23	36,42	152,83	76,54
16KB	140,86	71,23	305,40	152,83
64KB	558,66	280,13	1220,86	610,55
256KB	2229,82	1115,71	4882,68	2441,47
512KB	4458,05	2229,82	9765,11	4882,68
1MB	8914,50	4458,05	19529,98	9765,11
2MB	17827,39	8914,50	39059,71	19529,98
4MB	35653,18	17827,39	78119,16	39059,71
16MB	142607,94	71304,77	312475,90	156238,07

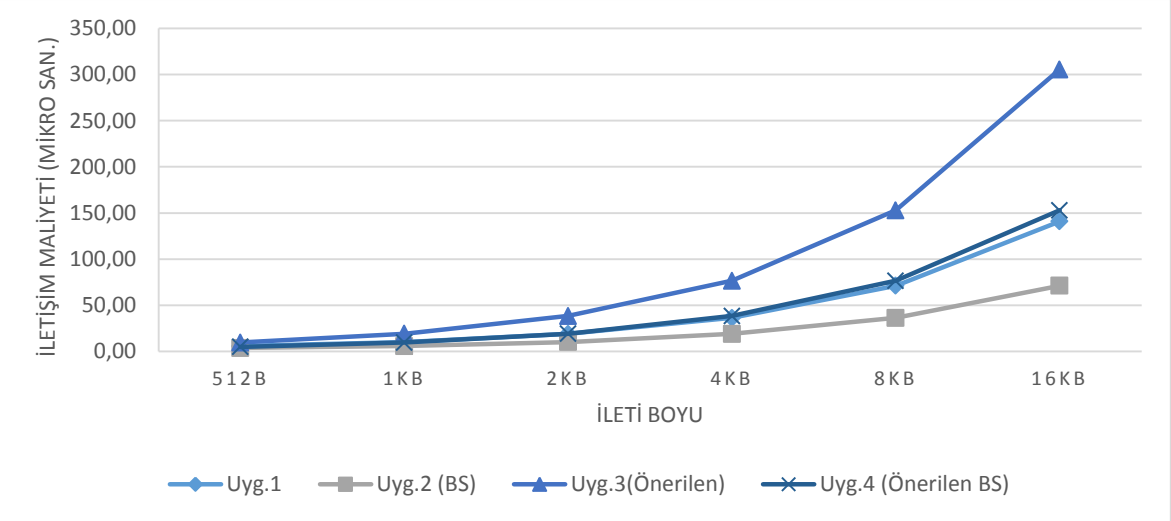
Çizelge 6-6: 8B-16MB ileti boyları için hepsinden hepsine iletim maliyetleri

(k = 2, $\propto_o = 50ns$)



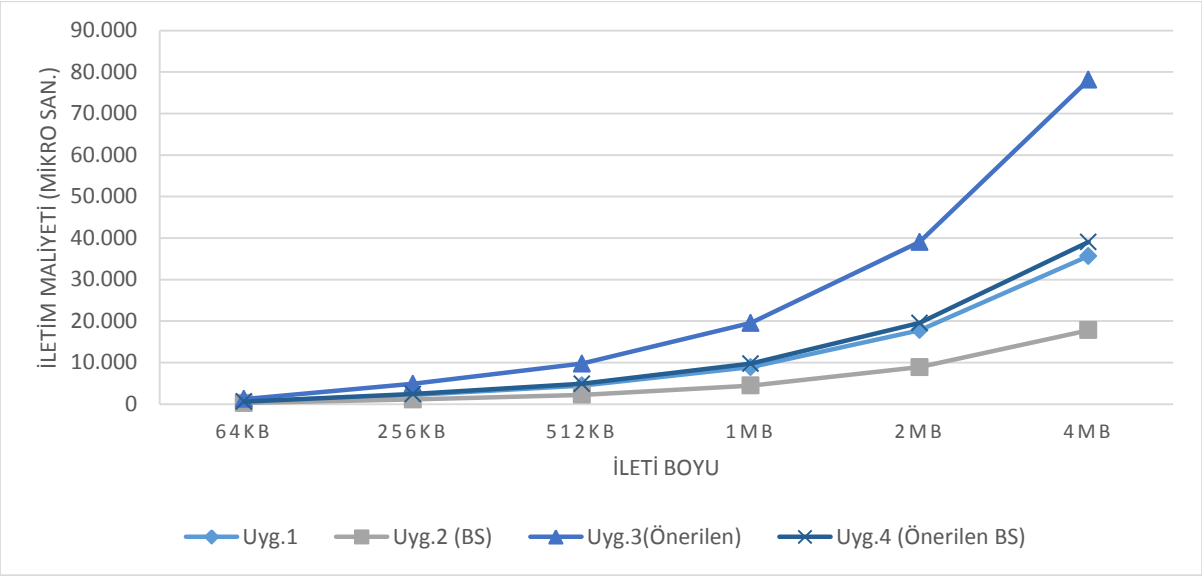
Şekil 6-9: 8B-256B arası ileti boyları için hepsinden hepsine algoritma maliyetleri

(k = 2, $\propto_o = 50ns$)



Şekil 6-10: 512B-16KB arası ileti boyları için hepsinden hepsine algoritma maliyetleri

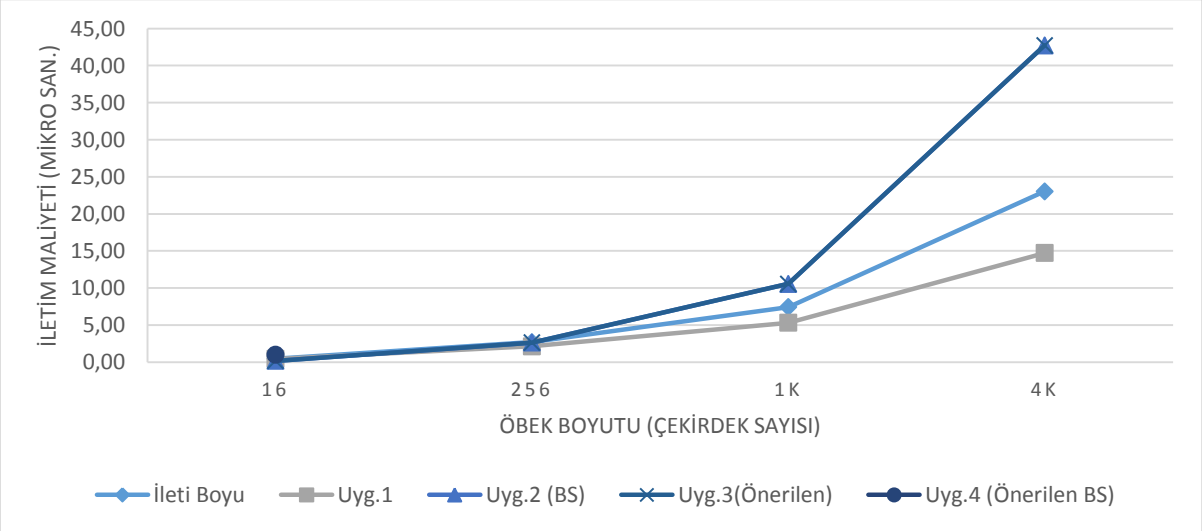
(k = 2, $\propto_o = 50ns$)



Şekil 6-11: 64KB-4MB arası ileti boyları için hepsinden hepsine algoritma maliyetleri

(k = 2, $\propto_o = 50ns$)

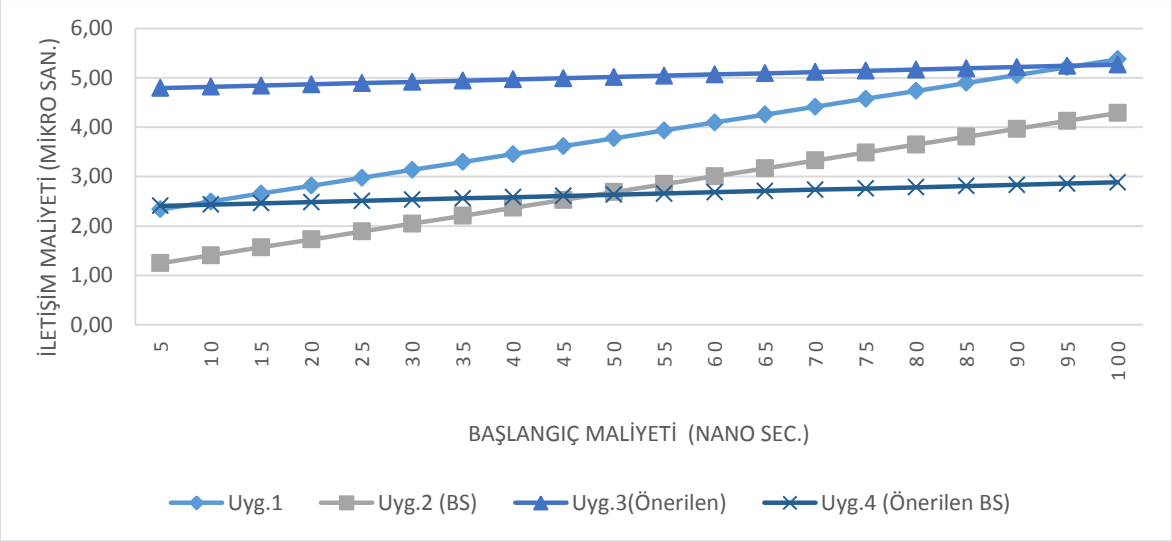
Değişken öbek boyutları için, yapılan çalışma 256B verinin dağıtımı için önerilen algoritma seçilen algoritmalar için 256 çekirdek öbek sayısına kadar en iyi sonucu vermektedir. Daha büyük öbekler için Uyg.2 daha iyi sonuç vermektedir. 16-4K çekirdek sayılı sistemler için 256B verinin yayın algoritması ile dağıtım maliyetleri Şekil 6-12 ile gösterilmiştir.



Şekil 6-12: Farklı öbek boyutları için hepsinden hepsine algoritması maliyetleri

(L = 256B, $\propto_o = 50ns$)

Değişen iletim başlangıç maliyetlerine göre 256B verinin dağıtımı Şekil 6-13 ile gösterilmiştir. Yapılan çalışma, 50ns ve yukarı iletim maliyetleri için önerilen algoritmanın en iyi sonucu verdiğini göstermektedir.

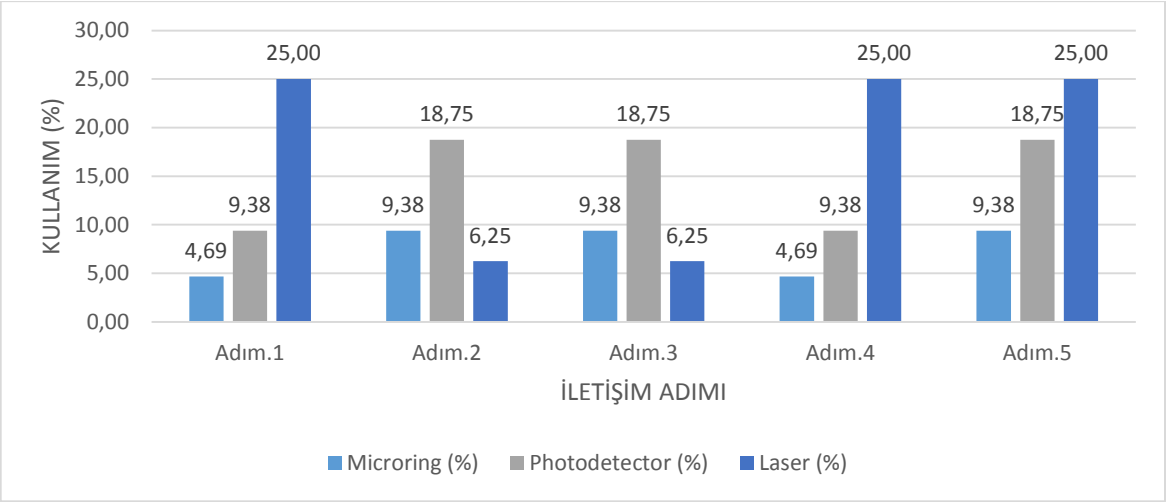


Şekil 6-13: Farklı iletim başlangıç maliyetlerine göre hepsinden hepsine algoritması iletim maliyetleri (L = 256 B)

Enerji tüketimi göstergelerini göstermek için 256 çekirdekli bir sistemdeki algoritmasının işletimi için, her adımdaki iletişim bileşenlerinin toplamı, kullanılan sayısı ve kullanım yüzdeleri Çizelge 6-7 ve Şekil 6-14 ile gösterilmiştir.

	Microring (Toplam:16384)		**Foton Algılayıcı (Toplam:4096)**		**Lazer (Toplam:1024)**	
Adım	**Kull.**	**Kullanım (%)**	**Kull.**	**Kullanım (%)**	**Kull.**	**Kullanım (%)**
Adım.1	768	4,69	384	9,38	256	25,00
Adım.2	1536	9,38	768	18,75	64	6,25
Adım.3	1536	9,38	768	18,75	64	6,25
Adım.4	768	4,69	384	9,38	256	25,00
Adım.5	1536	9,38	768	18,75	256	25,00

Çizelge 6-7: Önerilen hepsinden hepsine algoritması için fotonik ağ bileşenleri ve bileşenlerinin kullanım oranları



Şekil 6-14: 256 çekirdek üzerinde yayın algoritması işletiminde fotonik ağ bileşenlerinin kullanım oranları

6.2. Benzetim Çalışmaları

Tez kapsamında, önerilen mimarinin çalışma ve etkinlik ölçümleri için benzetim çalışmaları yapılmıştır ve benzetim sonuçları analiz edilerek yorumlanmıştır. Önerilen sistem ve yapıların gerçek zamanlı sistemler için geliştirildiği düşünülerek, bu çalışmalar için gerçek dünyada geçerli gerçek zamanlı sistem senaryoları tercih edilmiştir.

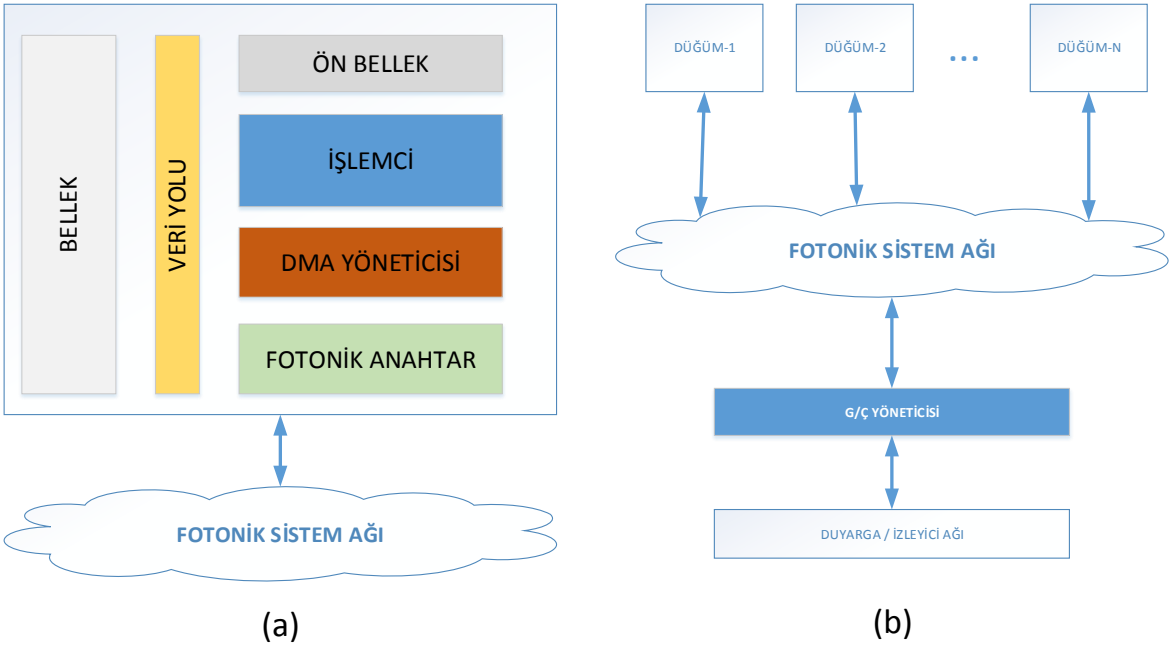
Bu çalışmada, iki boyutlu hasır doku topolojisinde dağıtılmış bellekli çok işlemcili bir mimari önerilmiştir ve bu sistemin ağ katmanı için 8x8 all-port fotonik anahtar tasarımı yapılmıştır. Daha sonra önerilen sistem üzerinde belirlenebilir iletişim protokolleri tanımlanmış ve bu protokollerden faydalanılarak işlem ve iletişimin bir arada yönetildiği bir görev zamanlama algoritması tanımlanmıştır.

Yonga üzeri ağ performansını ölçmek için hali hazırda geliştirilmiş birçok yonga üzeri ağ benzetim aracı bulunmaktadır. Fakat hem yeni bir mimari önerilmesi hem de önerilen mimari üzerinde yenilikçi bir ağ yönetimi önerilmesi durumu, benzetim olay yönetimi katmanı üzerinde hem ağ katmanı hem de uygulama katmanlarının gerçekleştirilmesini gerektirmektedir ve bu durum benzetim çalışmalarını mevcut benzetim araçlarıyla gerçekleştirilmesi maliyetlerini fazlasıyla artırmaktadır. Bu bağlamda, tez çalışması kapsamında bir benzetim aracı gerçekleştirilmiş ve bu benzetim aracı üzerinde benzetim çalışmaları gerçekleştirilmiştir.

Çalışma kapsamında Bölüm 6.1'de kullanılana benzer bir sistem tercih edilmiştir. Benzetim çalışmalarında kullanılan sistem $4^k x\ 4^k$hasır doku topolojisine sahip bir sistem

kullanılmıştır. Bu sistemde ağ iletişim katmanında tasarlanan 8x8 all-port fotonik anahtar kullanılmıştır ve her düğümün kendine adanmış bir fotonik anahtarı bulunmaktadır. Sistem dağıtılmış bellekli MIMD mimarisindedir ve her düğümün bir işlemci çekirdeği ve kendine ayrılmış bir bellek alanı bulunmaktadır. Dolayısıyla düğümler arası veri alışverişi sadece sistem alan ağı üzerinden düğümler arası veya toplu iletişim yaklaşımlarıyla ileti geçerek sağlanabilmektedir.

Yapılan benzetim çalışmasında kullanılan sistemde, her düğümde beş temel bileşen bulunmaktadır: işlemci, ön bellek, sistem belleği, fotonik ağ anahtarı ve DMA yöneticisi. Düğüm seviyesindeki bileşenler kendi aralarında bilinen elektrik tabanlı iletişim ve hesaplama yapılarını kullanırlar. İşlemcinin DMA iletişimlerinde bellek erişimi kısıtlanacağı için ilgili verilerini ön bellekte saklayarak burası üzerinden eriştiği düşünülmektedir. Sistem yonga dışındaki G/Ç bileşenleri ve düğümler arası veri iletişimi fotonik tabanlı sistem alan ağı üzerinden iletişime geçmektedir. G/Ç bileşenleri elektrik tabanlı olduğu için bu bileşenlerden gelen verinin toplanarak fotonik ağ üzerinden yongaya iletilmesi G/Ç yöneticisi (Bölüm 5.2) bileşeni tarafından sağlanır. Benzetilen sistem mimarisi Şekil 6-15 ile gösterilmiştir.



Şekil 6-15: Sistem mimarisi (a) Tek düğüm yapısı (b) genel sistem yapısı

Benzetimde kullanılan sistemdeki herhangi bir ağ iletimi başlamadan önce, bellek alanındaki verilerin ağ anahtarı yastıklarına iletilmesi için DMA kullanıldığı düşünülmüştür. Bu yapıda işlemci DMA sürecini tetikledikten sonra, diğer işlemlerine

devam etmektedir ve veri aktarımı tamamlandıktan sonra, verinin sistem belleğini alınması için ilgili iletim adımını tetiklemektedir. Hesaplamaları sadeleştirmek için DMA bileşeninin *burst* modunda kullanıldığı, yani verilerin ağ katmanına aktarımı tamamlanana kadar bellek veri yoluna erişemediği düşünülmektedir. Yapılan benzetim çalışmasında, DMA üzerinden her saat vuruşunda 64bit bir sözcük iletildiği ve bellek veri yolu tahsis ve bırakma işlemlerinin de dörder saat vuruşu sürdüğü varsayılmıştır [80].

Yapılan çalışmada, işlem ve iletişim birlikte yönetilmektedir ve bu süreçlerdeki işlemler zaman ekseni üzerinde üst üste bindirilmiştir. Bu sayede sistem başarımının artırılması ve iletim süreci boyunca işlemci başka görevleri işletebilmektedir. Bu bakış açısıyla, sistem zaman ekseni İşlemci, İletişim ve G/Ç olmak üzere 3 ayrı zaman eksenine ayrılmıştır. G/Ç ve İletişim işlemleri aynı sistem alan ağını kullandığı için bu iki zaman ekseni birbirleriyle eş zaman uyumlu olarak yürümektedir. İşlemci zaman ekseni diğer iki zaman ekseninden bağımsız yürümektedir fakat bir iletim adımı başlamadan önce, bu süreç işlemci üzerinden tetiklendiği için, iletişim ve işlemci zaman eksenleri eş zaman uyumlu hale gelmektedirler. Bahsedilen zaman ekseni yapısı benzetim aracı üzerinde gerçekleştirilmiş ve benzetimde yapılan çalışma maliyetleri üç zaman ekseni üzerinde ayrı olarak takip edilmiştir.

Burada bahsedilen yapılar doğrultusunda, benzetim üzerinde işletilen bir senaryoda iki ayrı süreç bulunmaktadır: işlem ve iletim süreçleri. İşlem süreci doğruda işlemci zaman eksenini etkiler ve sisteme sadece işlem zaman maliyeti uygulanmaktadır. İletim sürecinin benzetim üzerinde işletilmesi işlem sürecine göre daha karmaşıktır. Bu süreç ilgili verinin ağ katmanına aktarılması, iletim adımları ve iletim sonunda gelen iletinin tekrar düğüm belleğine iletilmesi adımlarından oluşmaktadır. İletim adımı fotonik düzeyde yapıldığı için, Bölüm 4'te bahsedilen sebeplerden dolayı sadece devre anahtarlama yapısında iletilebilmekte ve her ara iletim adımı öncesi elektro-optik dönüşüme ve her adım sonunda opto-elektrik dönüşümlere ihtiyaç duyar. Bu doğrultuda sistem üzerindeki bir ağ işletimi aşağıdaki adımlardan oluşmaktadır:

- İletilecek verinin DMA üzerinden bellek alanından ağ yastıklarına aktarılması
- Bir toplu iletimdeki her bir adım için
    - Verinin ilgili başlık verileri eklenerek ağ üzerinden iletilebilir paket yapısına çevrilmesi.
    - İşlemcinin fotonik anahtar bileşenini ilgili iletim için kurması ve fotonik anahtar yayılım gecikmeleri

    - Verinin elektro-optik çevrimi
    - Paketin *flit* verilerine bölünerek devre anahtarlama ile verinin aktarılması
    - Verinin opto-elektrik dönüşümler sonrası ağ yastıklarında saklanması
- Toplu iletişim sonunda verinin DMA üzerinden ağ yastıklarından bellek alanına aktarılması

Kullanılan sistem sadece fotonik iletişim altyapısını kullandığı için, iletişimde sadece devre anahtarlama kullanılabilmektedir. Devre anahtarlamada, önce başlık flit veya flit grupları ağ üzerinden gerekli kanalların hepsini tahsis ettikten sonra, iletinin kurulan kanallar üzerinden tek kanal üzerinden aktarılır gibi dağıtılması ile sağlanır. Bu yapıda devre anahtarlama yolun kurulması, geri bildirim ve iletinin devre üzerinden aktarılması olmak üzere üç katmandan oluşur.

Önerilen iletişim alt yapısı fotonik iletişim özelliklerinden de faydalanarak devre anahtarlamaya yenilikçi bir yaklaşım getirmektedir. Bu yapıda, iletişim adımı öncesi hangi yolların kullanılacağı belli olduğu için sistem çakışma olmadan tüm devrelerin belirlenen zamanda tamamlanacağını garanti eder. Dolayısıyla bu yapıda devre kurulması ve geri bildirim adımlarına ihtiyaç duyulmaz. Bu bağlamda, iletilerin yol kurulum için gerekli başlık bilgilerine de ihtiyacı yoktur ve bu durum sistemdeki ileti yapısını sadeleştirmektedir. Önerilen sistemde iletilen veri ile birlikte kaynak düğüm kimliği için 16bit başlık alanı eklenmiştir ve ağ paket yapısı (veri boyu + 16-bit başlık) yapısından oluşacaktır.

Modern işlemcilerde özellikle komut ardıl düzeni yapılarının gelişmesi bir saat vuruşunda yapılan matematiksel işlemler için doğrudan teorik var sayımların koyulmasını olanaksız hale getirmiştir. Bu doğrultuda, kullanılan benzetim yapısındaki işlemci hızını tanımlamak için saat frekansı ve saniyede yapabileceği kayan noktalı işlem sayısı olmak üzere iki ayrı parametre kullanılmıştır. Burada parametre değerleri belirlenirken, işlemcilerdeki kayan noktalı işlem kapasitesi üzerine yapılmış gerçek başarım testleri temel alınmıştır [80].

Fotonik iletişimde yastık kullanımı, optik yastık düzeylerinin yetersiz olması dolayısıyla olanaksızdır. Bu doğrultuda yastıksız bir akış denetimi tercih edilmektedir. Yapılan fotonik tabanlı benzetim işlemlerinde, ACK-NACK akış denetimi tercih edilmiştir

Yapılan çalışmalar, optik kayıpların fotonik sistemlerdeki hatalı bit oranının elektronik sistemlerden yüksek olmasına neden olduğunu belirtmektedir [81]. Bu bağlamda veri tutarlılığı için hata düzeltici yapıların kullanılması gerekmektedir. Benzetim çalışmalarında

CRC yapısı kullanılmış ve CRC boyu olarak 16-bit tercih edilmiştir. Bu yapı sisteme CRC denetimlerini de ekleyeceği için, sistemdeki iletim senaryosu aşağıdaki gibi güncellenecektir.

- İletilecek verinin DMA kullanılarak bellek alanından ağ yastıklarına aktarılması
- Toplu iletimdeki her bir adım veya tek adımlık bir iletişim için
  - Verinin ilgili başlık ve denetim verileri eklenerek ağ üzerinden iletilebilir paket yapısına çevrilmesi
  - İşlemcinin fotonik anahtarı ilgili iletim için kurması ve fotonik anahtar yayılım gecikmeleri
  - Verinin elektro-optik çevrimi
  - Devre anahtarlama ile verinin aktarılması
  - Verinin opto-elektrik dönüşümler sonrası ağ yastıklarında saklanması
- DMA üzerinden verinin bellek alanına aktarılması
- CRC denetimleri
- Hatalı flit'lerin tekrar iletimi için zaman ekseninde ayrılan zamanda tekrar iletilmesi

Benzetim üzerindeki fotonik bileşenlerinin bekleme ve iletim maliyetlerinde gerçek yonga üzeri ağ çalışmalarındaki [82] ve bu konuda yapılmış benzetim çalışmalarından [81], [83] faydalanılmıştır. Bu doğrultuda benzetim sistemi için belirlenen parametreler ve değerleri Çizelge 6-8 ile gösterilmiştir.

Parametre	**Değer**
İşlemci kapasitesi	2 GHz / 50 MFLOPS
Sistem topolojisi	2D Hasır doku
DMA iletim hızı	Her saat vuruşunda 64 bit (per cycle)
Anahtar tipi	4-port fotonik anahtar
Fotonik bant genişliği	64Gb/s (τ = 64bit data iletimi için 1ns)
Foto detektör hızı [83]	40 Gb/s
Fotonik Anahtar kurulum hızı [82]	1 ns
PSE ve Fotonik yayılım gecikmeleri [82]	30ps ve 220ps
Fotonik Hata Bit Oranı	10^{-9}

Çizelge 6-8: Benzetim sistem parametreleri

6.2.1. Optik akış algoritmasının benzetimi

Optik akış algoritması optik farelerde fare hareketlerinin saptamasından, video akışı üzerinden hareket eden nesnelerin belirlenmesi gibi birçok alanda kullanılmaktadır. Bu algoritma akış görüntüsü üzerinden işlenerek nesne belirlenmesini sağlar. Bu doğrultuda başta savunma sistemleri gibi gerçek zamanlı sistemler için uygun olan bir algoritma olduğu için, çalışma kapsamında tercih edilmiştir.

Benzetim çalışmasında, Lucas-Kanade optik akış yöntemi tercih edilmiştir. Bu çalışma kapsamında, seçilen optik akış algoritması önerilen benzetim sistemi üzerinde benzetilmiştir. Lucas-Kanade optik akış algoritması, ardışık görüntülerde ilginç özelliklerin hareketi tahminini sağlayabilen, iyi bilinen ve basit bir tekniktir. Bu algoritma, iki ardışık görüntüyü karşılaştırıp ve iki sahnedeki her "ilginç" piksele bir hareket vektörü (u, v) atar. Lucas-Kanade yöntemi, birkaç komşu pikselden gelen bilgileri birleştirerek optik akış denkleminin hesaplanmasına dayanır. Bu çalışmada kullanılan bu yöntemin pseudo kodu aşağıda gösterilmiştir.

```
OpticalFlow (img_1,img_2):

    //Image cropping operations
    // m*n*3 floating point operations
    x_c= img_1(1:end-1,2:end) - img_2(1:end-1,1:end-1);
    y_c= img_2 (2:end,1:end-1)- img_1( (1:end-1,1:end-1);
    t_c= img_2 (1:end-1,1:end-1)- img_1( (1:end-1,1:end-1);
    //Memory allocation
    u = init(size(x_c)); // create a  zero matrix
    v = u;
    for x = 1:size(x_c,1)-kernelSize
        for y = 1:size(x_c,2)-kernelSize

        win_x=imcrop(x_c,[x y kernelSize kernelSize]);
        win_y=imcrop(y_c,[x y kernelSize kernelSize]);
        win_t=imcrop(t_c,[x y kernelSize kernelSize]);
        A = [win_x(:) win_y(:)];

                th=min(eig(A'*A));
                rank =2;

                if th<0.011 || rank(A)~=2
                        unk=[0 0];
                else
                        unk = pinv(A'*A)*A'*win_t(:);
                end

                u(x,y)=unk(1);
end
```

Optik akış algoritmasının çok işlemcili bir sistemde çalıştırılmasında, seçilen algoritma akışından bağımsız olarak bir veri dağıtımı problemi bulunur. Bu yaklaşımda işlenecek resim verisinin işlemcilere bölünmesi ve işlenmesi sonrası her piksel için hesaplanmış kayan noktalı sayı değerinin tekrar toplanması gerekir. Bununla birlikte, her düğüme atanmış alt resim çerçevesi, hesaplamalar için resmin dört tarafından komşu piksellere ihtiyaç duyar. İhtiyaç duyulan piksel sayısı hesaplama kuramına göre bir veya üç uzaklıktaki komşu piksel olabilir. Bu bağlamda $4^k x 4^k$ hasır dokudaki bir işlemci yongasında, her düğümün eşit miktarda veri işlediği düşünülerek WxH boyutundaki verinin işlenmesi için her düğüme iletilmesi gereken veri miktarı ile (12) gösterilmiştir.

$$\left[\left(\frac{W}{16}\right) x \left(\frac{H}{16}\right) + \left[2x\left(\left(\frac{W}{16}\right) + \left(\frac{H}{16}\right)\right) x\, ENS\right]\right] x\, \text{Görüntü}\, Veri\ Boyutu \tag{12}$$

Benzetim kapsamında kullanılan görüntü verisinin RGB yapısında olduğu ve her renk için 8-bit değer içerdiği düşünülmüş, bu doğrultuda Görüntü veri boyutu olarak 24-bit tercih edilmiştir. Burada ENS hesaplama için gerekli ekstra nokta sayısını belirtmektedir ve çalışma kapsamında 1 nokta değeri tercih edilmiştir.

Tez çalışması kapsamında ortaya konulan optik akış algoritması benzetim ortamında gerçekleştirilmiştir. Benzetim ortamında 16x16 hasır dokuda bir işlemci yongası üzerinde çalışıldığı varsayılmıştır ve sistem için Çizelge 6-8'de verilen parametreler kullanılmıştır. Burada verilen problem verinin okunarak dağıtılması ve Lucas-Kanade algoritması ile işlenmesi olmak üzere iki gruba ayrılmış ve görev grupları önerilen görev zamanlama yaklaşımı ile yönetilmiştir.

Optik akış algoritmasının önerilen sistem mimarisi ile işletilmesi aşağıdaki belirtilen adımlardan oluşmaktadır:

- WxH boyutundaki resim verisinin G/Ç yöneticisi üzerinden okunması
- Verinin dağıtım algoritması ile dağıtılması
- Her düğümün kendi resim çerçevesindeki veriyi işlemesi
- Sonuçların toplama algoritması ile toplanması

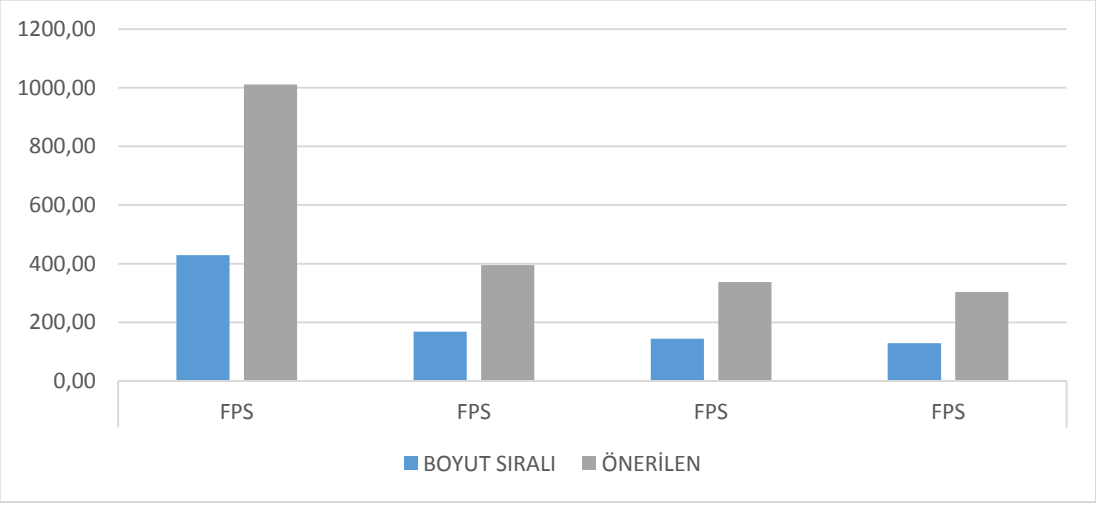
Burada belirtilen algoritmanın işletilmesi için geliştirilen benzetim aracı kullanılmıştır. Burada dağıtım ve toplama algoritmaları için iki yöntem kullanılmış ve karşılaştırılmıştır:

- Boyut sıralı dağıtım algoritması (Bölüm 6.1.1)
- Önerilen dağıtım algoritması (Bölüm 5.1.4)

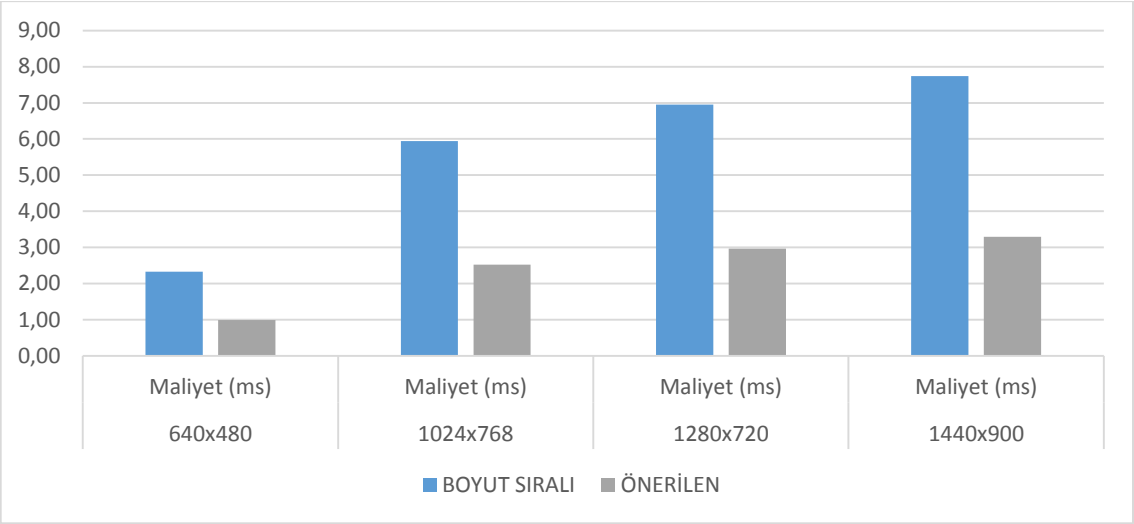
Burada benzetim testleri için dört görüntü boyutu kullanılmıştır: 640x480, 1024x768, 1280x720 ve 1440x900. Yapılan benzetim çalışmasında, görüntü dağıtımında önerilen algoritmanın boyut sıralı algoritmaya göre iki katın üstünde daha iyi sonuç vermiştir. Her iki algoritma için, sistemin sadece tek bir görev yaptığı ve sürekli olarak bu görevi tekrarladığı senaryo için görüntü okuma ve dağıtım sonuçları ve FPS değerleri ile belirtilmiştir. Bu değerler Çizelge 6-9, Şekil 6-16 ve Şekil 6-17 ile gösterilmiştir.

		Maliyet (ms)	FPS
BOYUT SIRALI	640x480	2.33	428.55
	1024x768	5.94	168.39
	1280x720	6.95	143.93
	1440x900	7.74	129.21
ÖNERİLEN	640x480	0.99	1011.29
	1024x768	2.53	395.39
	1280x720	2.96	337.42
	1440x900	3.30	303.12

Çizelge 6-9: Optik akış algoritması için görüntü dağıtım ve sonuç toplama zaman ve FPS karşılaştırılması



Şekil 6-16: Optik akış algoritması için görüntü dağıtım ve sonuç FPS karşılaştırılması



Şekil 6-17: Optik akış algoritması için görüntü dağıtım ve sonuç toplam zaman karşılaştırılması

Yapılan benzetim çalışmasında üç zaman ekseni için işlemler bir arada ve işlemci üzerinden yürütülmüştür. Bu bağlamda her zaman ekseni için farklı işlem maliyetleri oluşmaktadır. Her iki algoritma için üç zaman ekseni üzerindeki işlem maliyetleri Çizelge 6-10 ile gösterilmiştir. Burada bazı eksenlerdeki işlemler eş zamanlı olarak diğer eksenlerde de olabileceği için (örneğin DMA kurma komutu hem işlemci hem iletişim eksenini etkilemektedir), bu maliyet toplamları verilen toplamlarla eşleşmemektedir.

		İLETİŞİM (ms)	G/Ç (ms)	İŞLEMCİ (ms)
BOYUT SIRALI	640x480	1.49	0.84	0.68
	1024x768	3.79	2.15	1.73
	1280x720	4.43	2.52	2.03
	1440x900	4.93	2.81	2.26
ÖNERİLEN	640x480	0.15	0.84	0.30
	1024x768	0.38	2.15	0.78
	1280x720	0.44	2.52	0.91
	1440x900	0.49	2.81	1.02

Çizelge 6-10: Verilen algoritmalar için eksen tabanlı maliyetler

Optik akış algoritmasının diğer fazı alınan çerçeve verisinin ilgili düğümde işlenmesidir. Yapılan hesaplamalarda, Lucas-Kanade algoritmasına göre, bir noktanın hesaplanması için 2307 adet kayan noktalı işlem hesabı gerekmektedir. Bu bağlamda bir işlemci üzerinde çerçeve hesabı için $(W/16) * (H/16) * 2307$ adet kayan noktalı nokta hesabı yapılması gerekmektedir. Yapılan benzetim çalışmasında, hesaplamanın yapıldığı sistemde bir çerçeve hesabı farklı boyuttaki veriler için benzetim üzerinden hesaplanmıştır Önerilen

sistemde iletişim ve işlem bağımsız olarak yürütülebildiği için, gelen görüntülerin dağıtılması, ağ ortamından gelen görüntü verilerinin işlenmesi ve sonuçları hedef düğümde toplanması ile eş zamanlı olarak işletilebilir. Fakat görüntü dağıtılması ve sonuçların toplanması süreçlerinin ikisi de ağ ortamını kullandığı için koşullandırılamaz. Bu durumda, süreç birbirinden bağımsız iki faza ayrılabilir:

- Yeni gelen görüntünün işlemcilere dağıtılması ve İşlenen görüntü verisinin hedef düğümde toplanması
- Görüntü çerçevesinin işlemcide işlenmesi

Verilen iki faz için bağımsız maliyet ve FPS değerleri her iki algoritma için Çizelge 6-11 ile gösterilmiştir. Burada maliyetler mili saniye (ms) olarak gösterilmiştir.

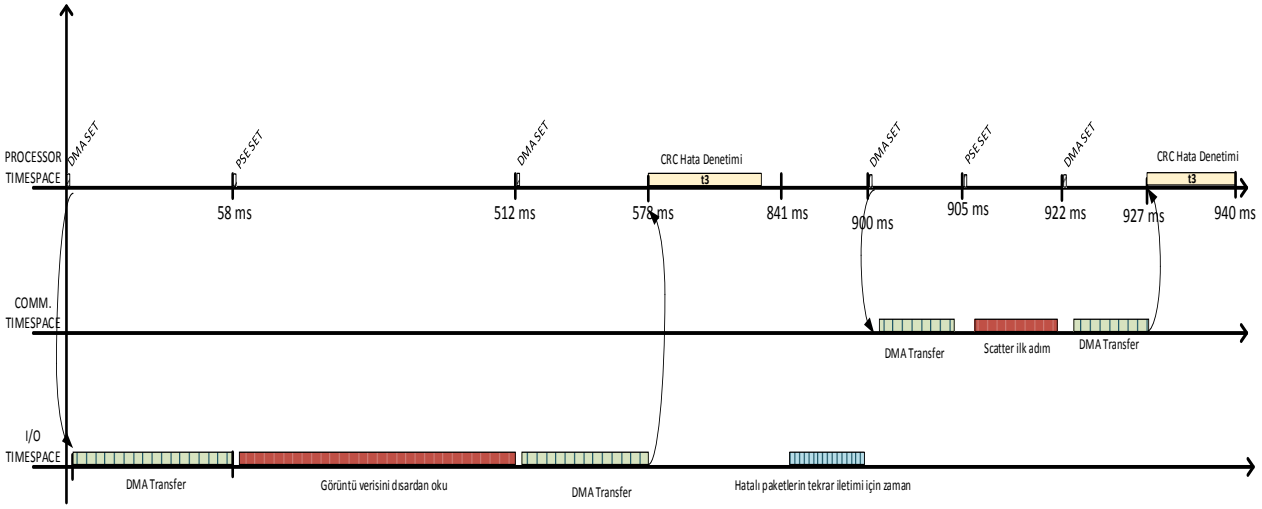
		Maliyet (ms)	FPS
BOYUT SIRALI	640x480	2.33	428.55
	1024x768	5.94	168.39
	1280x720	6.95	143.93
	1440x900	7.74	129.21
ÖNERİLEN	640x480	0.99	1011.29
	1024x768	2.53	395.39
	1280x720	2.96	337.42
	1440x900	3.30	303.12
VERİ İŞLEME	640x480	55.38	18.06
	1024x768	141.74	7.06
	1280x720	166.10	6.02
	1440x900	233.58	4.28

Çizelge 6-11: İki bağımsız faz olarak optik akış algoritmasının maliyet ve FPS değerleri

Benzetim sonuçları, sistemde en maliyetli işlemin, verinin işlenmesi olduğu ve bu işlemin diğer hesaplamalara göre oldukça düşük olduğu görülmektedir. Görüntü dağıtımı ve sonuçların toplanması FPS değerleri işlem maliyetine göre oldukça yüksektir. İşlem maliyetlerinin düşürülmesi için, daha çok işlemci içeren mimariler tercih edilmesi veya algoritmanın iyileştirilmesi gerekmektedir.

Bu çalışma kapsamında, önerilen dağıtım algoritmasını kullanan bir gerçek zamanlı sistem için, çok görevli bir yapıda Bölüm 5.3'de önerilen görev zamanlama algoritması G/Ç okuma ve dağıtım ilk adımları için uygulanmıştır. 640x480 boyutundaki görüntü ile çalışan sistem için uygulanan algoritmanın zaman eksenine uygulanması Şekil 6-18 ile gösterilmiştir. Burada kurgulanan sistem 5ns'lik zaman dilimlerine sahiptir ve görevler

periyotlarına bağlı biçimde bu dilimlere yerleştirilmektedirler. Tasarlanan sistemi daha basit düzeyde anlatmak için şekilde sadece belirlenen senaryodaki kritik anlar betimlenmiştir. Boş alanlarda, sistemin diğer sistem görevleri ve optik akış verilerinin hesaplanması sağlanmaktadır. Görüldüğü üzere, G/Ç, iletişim ve işlem görevleri bir arada ve birer periyodik görev olarak yönetilmektedir.



Şekil 6-18: G/Ç adımı ve dağıtım ilk adımları için görev zamanlama kurgusu

6.2.2. G/Ç verisi toplanması, işlenmesi ve dağıtımı algoritmasının benzetimi

Belirtildiği üzere, bir gerçek zamanlı sistemin genelleştirilmiş yaşam döngüsü çevresi dinlemesi veya izlemesi, gelen çevre verileri ve kendi sistem verilerinin kullanarak hesaplamalar yapması ve çevresini bu doğrultuda değiştirmesi veya etkilemesi olarak üç faza ayrılabilir. Burada gerçek zamanlı sistemlerin çevresini izlemesi duyarga bileşenlerinden gelen verilerle ve çevresini etkilemesi veya değiştirmesi ise işleticileri üzerinden yönettiği araç veya makineleri tetiklemesi ile sağlanır. Bu yapıda gerçek zamanlı bir sistem için duyarga ve işletici verilerini etkin biçimde yönetmesi hayati önem taşımaktadır.

Tez kapsamında belirlenen bir duyarga işletici modeli için, önerilen G/Ç yöneticisi üzerinden duyarga verilerinin toplanması, ilgili hesaplamaların yapılması ve G/Ç yöneticisi üzerinden verilerin dağıtılması senaryosu ele alınmıştır. İşletilen senaryoda, 4x4 hasır doku üzerinde çalıştığı ve diğer sistem veri parametrelerin için Çizelge 6-8 ‘deki değerlerin kullanıldığı var sayılmıştır. Bu sistem 32 adet duyarga ile 16 adet işleticinin olduğu bir G/Ç ağı üzerinde çalışmaktadır. Her düğümün bütün duyargalardaki veriye ihtiyaç

duyduğu ve düğümlerin duyarga verileri üzerinde maksimum 25 kayan noktalı işlem yapıldığı varsayılıyor.

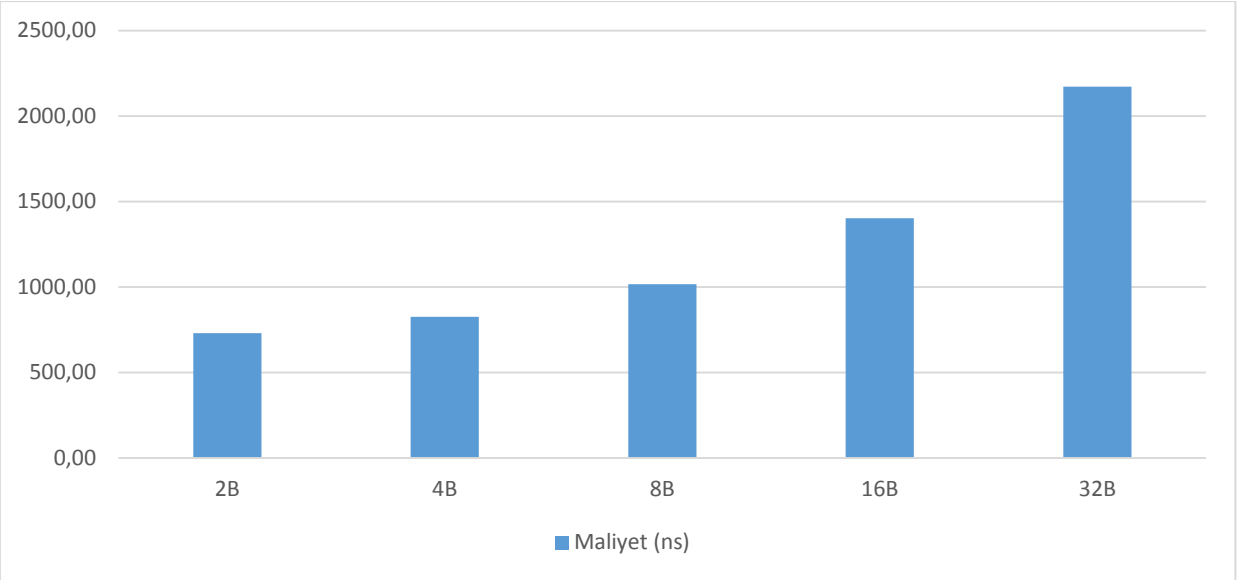
Bu doğrultuda, önerilen sistem aşağıdaki adımların sıralı ve sürekli olarak işletilmesinden oluşacaktır.

- Duyarga verilerinin G/Ç yöneticisinden okunması
- Yayın algoritması ile verilerin işlemcilere dağıtılması
- İşlemcilerin hesaplamaları yapmaları
- İndirgeme ile verilerin toplanması
- İşletici verilerinin G/Ç yöneticisine yazılması

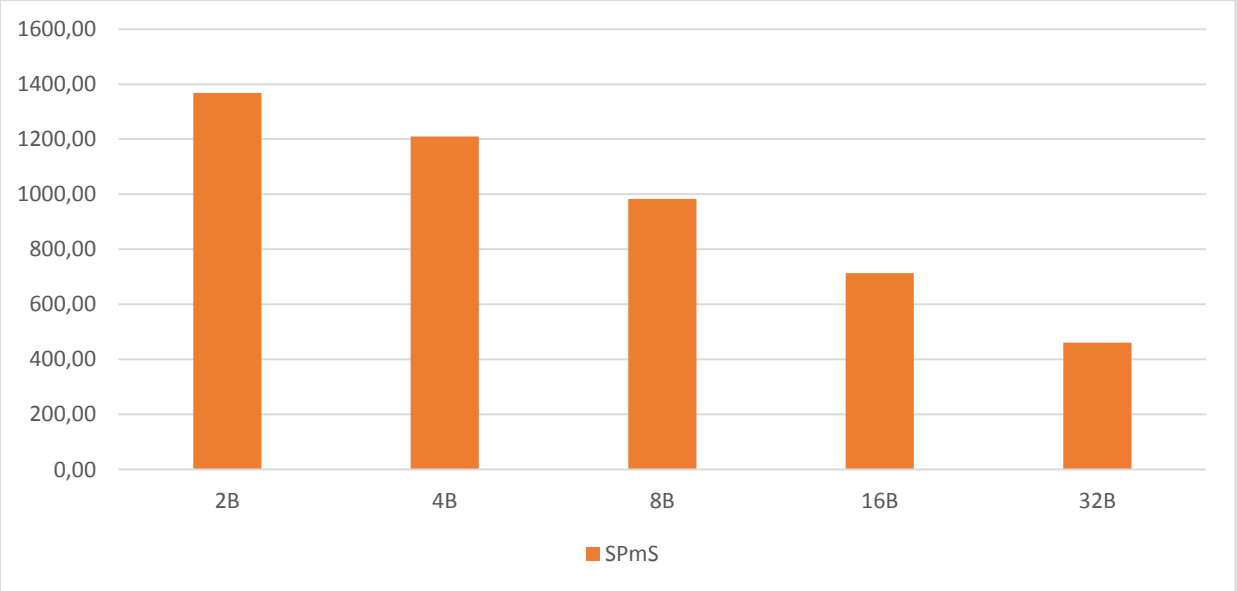
Burada indirgeme ve yayın adımlarında önerilen yayın algoritması kullanılmıştır. Daha önce yapılan çalışmada, önerilen yayın algoritması diğer algoritmalara göre en yüksek başarıma sahip olduğu gösterildiği için, bu adımda sadece önerilen yayın algoritması için benzetim çalışması yapılmıştır. Farklı duyarga verisi boyları için benzetim sonuçları ile gösterilmiştir. Sonuçlardaki maliyetler ve ms'deki örneklem miktarı (SPmS) Çizelge 6-12, Şekil 6-19 ve Şekil 6-20 ile gösterilmiştir. İşleticiler için her adımda 2B veri aktarıldığı düşünülmektedir.

Duyarga veri-boyu	Maliyet (ns)	SPmS
2-byte	730.91	1368.16
4-byte	826.43	1210.02
8-byte	1017.47	982.83
16-byte	1403.31	712.60
32-byte	2171.23	460.57

Çizelge 6-12: Farklı veri boyları için benzetim maliyetleri ve SPmS değerleri



Şekil 6-19: Farklı veri boyları için benzetim maliyetleri

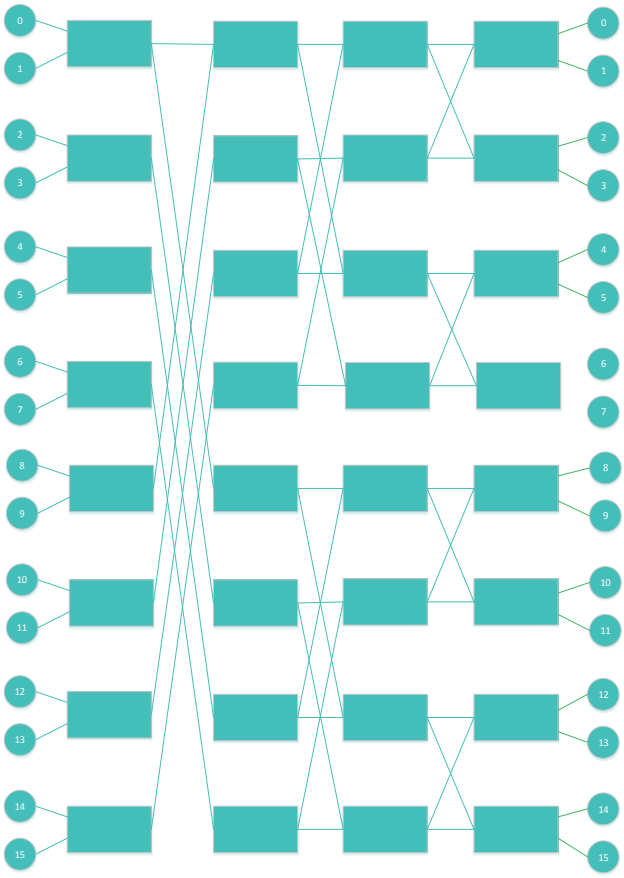


Şekil 6-20: Farklı veri boyları için SPmS değerleri

Yapılan çalışma, önerilen mimarinin duyarga verisinin yoğun biçimde okunarak işleticilerin tetiklendiği gerçek zamanlı sistemlerin için uygun olduğunu göstermiştir. Verilen parametreler doğrultusunda, 1024B veri için 4x4 hasır doku kullanılarak ms'de 460 kere hesaplama yapılabilmektedir.

6.2.3. Kelebek ağlar için iletim algoritmasının benzetimi

Kelebek ağlar sistem alan ağlarının ilk yıllarından beri kullanılan bir dolaylı ağ topolojisidir. Bu topolojide düğümler birbirlerine çok katmanlı bir anahtarlar grubu ile bağlıdırlar. Bu yapıda ikili komşular birbirlerine doğrudan anahtar erişebilecekleri bir anahtara bağlıdırlar ve daha uzaktaki komşuya erişmek için, logaritmik düzeyde anahtar üzerinden iletişim sağlanması gerekir. N adet düğüm içeren bir kelebek ağda, δ = 2k anahtar derecesi için, maksimum uzaklık adımı $log_k N + 1$ ile hesaplanır. Standart 16 düğümlü bir kelebek ağı Şekil 6-21 ile gösterilmiştir.



Şekil 6-21: 16 düğümlü bir kelebek ağı (2-ary 4-fly)[12]

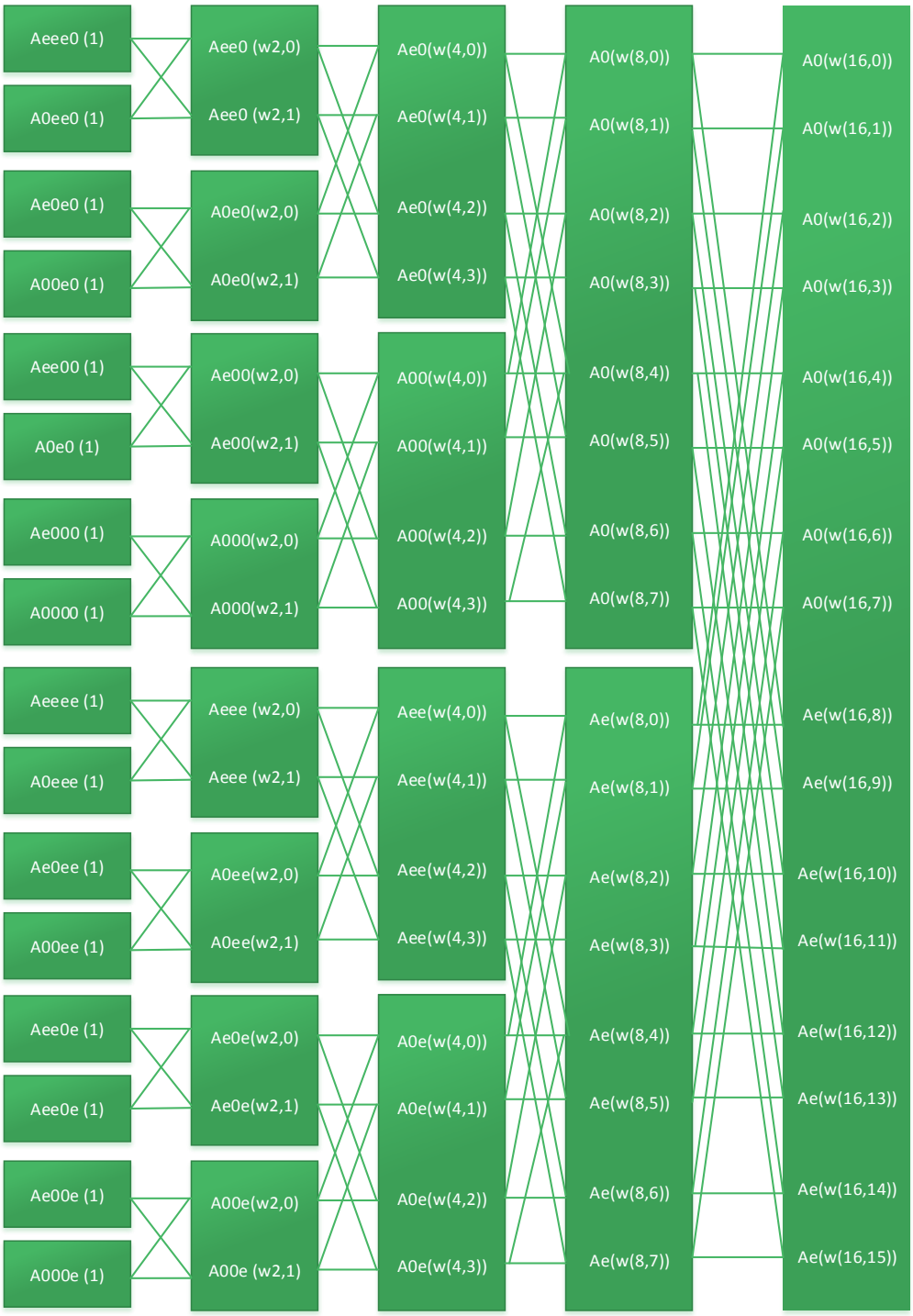
Kelebek ağları, düğümler için belirlenebilir ve optimal bir ulaşım derecesi sunmasına rağmen, bu ağların bir takım dezavantajları bulunur. Öncelikle bu ağlarda herhangi iki düğüm arasında sadece tek bir yol seçeneği bulunur ve bu yoldaki anahtarların başka iletişimlere atanması durumunda çakışmalar meydana gelir [12]. Bir diğer dezavantaj iletişim için geçilmesi gereken devre ve anahtar sayılarının çoğalması iletişim etkinliğini düşürmektedir.

Kelebek ağları fiziksel ağlarda tercih edilmese de, günümüzde özellikle çoklu ortam programlamada yoğun biçimde kullanılmaktadır [84]. Bu kuramlar başında, Fast Fourier Transform (FFT) gelmektedir [85]. FFT özellikle sinyal işleme ve sayısal filtrelemede yoğun biçimde kullanılan temel bir algoritmadır ve özellikle görüntü ve ses işleme yaklaşımlarının birçoğu bu algoritmaya dayanır. FFT üzerine hem GPU hem de çok işlemcili sistemler için yapılmış gerçekleştirim ve kütüphane bulunmaktadır. Bu çalışma kapsamında kelebek ağ algoritması ile gerçekleştirilen çok düzeyli bir veri değişimi adımı, önerilen mimari üzerine uygulanmıştır.

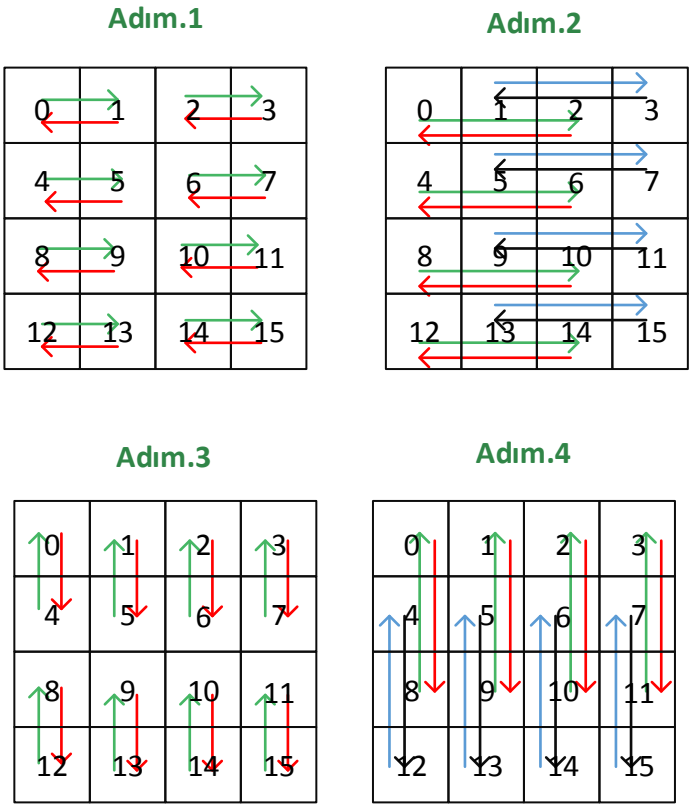
Kelebek ağları üzerinde bir veri değişimi adımında, her düğüm bulunduğu adım uzaklığındaki anahtar üzerinden erişebildiği komşusuyla veri değişimi yapar ve bu süreç ağ üzerindeki anahtar seviyesi kadar devam eder. Her adım sonunda, düğüm komşusundan aldığı veriyi kullanarak bir hesaplama yapar. Bu bağlamda bu hesaplama 4 adım iletişim ve 4 adım hesaplama içerir. Örneklenen kelebek ağ örüntüsü 16 düğüm için Şekil 6-22 ile gösterilmiştir. Burada sekiz düğümdeki veriler dört düzeyli bir hesaplama sonucunda 16 adet ağırlık değerine indirgenmektedir.

Yapılan çalışma kapsamında, verilen 16 düğümlü kelebek ağ üzerinde ağırlık hesaplama örüntüsü önerilen mimarideki 4x4 hasır doku üzerine uyarlanmıştır. Bu uyarlama işleminde, diğer çalışmalardan farklı olarak önerilen iletişim örüntülerinden farklı bir yaklaşım bu algoritma için geliştirilmiştir.

Şekil 6-22 ile verilen iletişim örüntüsünde dört düzey bulunmaktadır ve her düzeyde 32 adet veri değişim işlemi bulunmaktadır. Burada ilk seviye de dahil olmak üzere her seviyede aynı düzlemdeki değerlerin aynı düğüm olduğunu düşünelim. Bu durumda her adımda ilgili düğüm kendisine ve bir başka düğüme ileti göndermekte ve almaktadır. Bununla birlikte örüntü incelendiğinde, her düğüm kendi düzlemindeki düğüm dışında, her i. Adımda kendisinden $2^{(i-1)}$ uzaklıktaki komşusu ile mesajlaşmaktadır. Verilen örüntü 4x4 hasır dokuya uygulandığında, ilk adımda yatay eksendeki komşusu, 2.adımda iki uzağındaki ve 3.adımda dikey eksendeki komşusu ile mesajlaşacaktır. Bahsedilen örüntü Şekil 6-23 ile gösterilmiştir.



Şekil 6-22: 4x4 16 düğüm içeren kelebek ağ ağırlık hesaplama örüntüsü



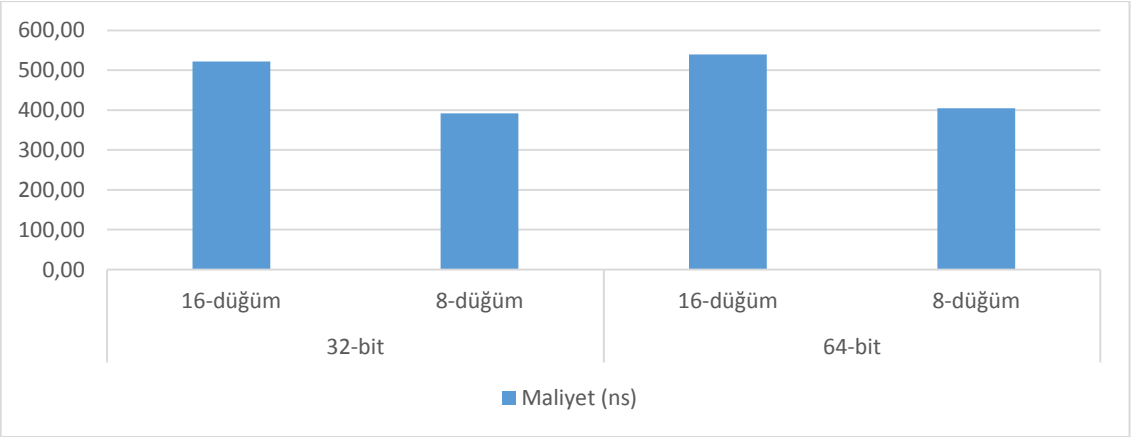
Şekil 6-23: 16 düğümlü kelebek ağı iletişiminin önerilen mimariye uyarlanması

Burada önerilen yapının çalışması için 3 farklı dalga boyuna ihtiyaç duyulmaktadır ve önerilen dört dalga boyu destekleyen fotonik anahtar ile ilgili iletişim sağlanabilir. Benzer şekilde, hasır dokusu alt ve sekiz düğümün ayrı gruplar halinde işletildiği yapıda iki 8 düğümlü kelebek ağı hesaplamasını da yapabilmektedir. Bu durumda 4.adıma ihtiyaç duyulmamaktadır.

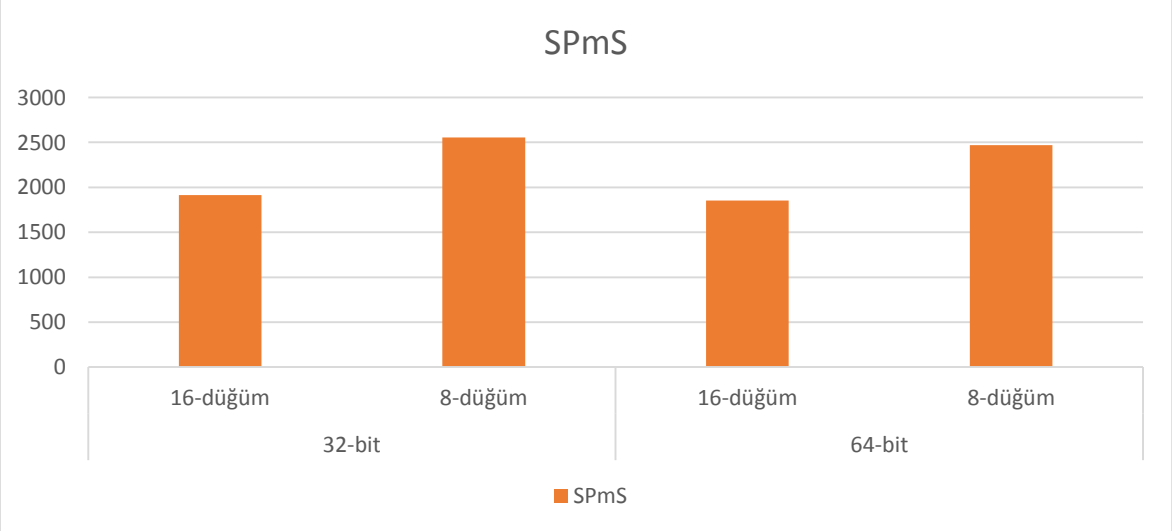
Önerilen tasarım, benzetim ortamında gerçekleştirilmiştir. Burada 16 ve 8 düğümlü kelebek ağlarının uyarlandığı iki durum verilen benzetim mimari ve parametreleri ile test edilmiştir. Sistemlerin 32 ve 64 bit veri iletişimi sağladığı ve her adımda 5 kayan noktalı sayı işlemi yaptığı varsayılmıştır. Benzetim maliyet sonuçları ve mikro saniyede hesaplanabilecek örneklem (SPmS) miktarı Çizelge 6-13 ile gösterilmiştir. Maliyet değerleri Şekil 6-24 ile ve SPmS Şekil 6-25 ile karşılaştırılmıştır.

	Satır Etiketleri	Maliyet (ns)	SPmS
32-bit	16-node	522.08	1915.415
	8node	391.56	2553.887
64bit	16node	539.83	1852.435
	8node	404.87	2469.929

Çizelge 6-13: Kelebek ağ hesaplamaları benzetim sonuçları



Şekil 6-24: Kelebek ağ hesaplamaları benzetim maliyetleri (ns)



Şekil 6-25: Kelebek ağ hesaplamaları benzetim SPmS değerleri

Benzetim sonuçları 16 ve 8 düğüm üzerinde başarılı sonuçlar vermiştir. Önerilen yapıda, 16 düğüm içeren bir kelebek ağ için 64 bit veri kullanılarak ms'de 1800 üzerinde örneklem hesaplaması yapılabilmektedir.

7. TARTIŞMA VE SONUÇ

Tez kapsamında, iki boyutlu hasır doku topolojisinde dağıtılmış bellekli çok işlemcili bir mimari önerilmiştir ve bu sistemin ağ katmanı için 8x8 *all-port* fotonik anahtar tasarımı yapılmıştır ve fotonik dalga boyu çoklamalı bir iletişim yapısı tasarlanmıştır. Daha sonra önerilen sistem üzerinde belirlenebilir ve tahmin edilebilir toplu iletişim protokolleri tanımlanmıştır ve sistem üzerindeki tüm iletişimler bu protokoller üzerinden sağlanmıştır. Ağ işlemleri ve G/Ç işlemlerinin tamamen tahmin edilebilir olarak gerçekleştirilmesi, sistem üzerinde ağ işlemlerinin de gerçek zamanlı görev yapısı olarak periyodik veya aperiyodik yönetilebilmesini sağlamıştır. Bu bağlamda sistem üzerinde G/Ç, ağ ve işlem görevlerinin bir arada yönetildiği bir görev zamanlama algoritması tanımlanmıştır ve hem iletişim hem işlemin bir arada yönetildiği yenilikçi bir yapı sunulmuştur. Önerilen sistem benzer fotonik tabanlı yonga üzeri sistem tasarımlardan farklı olarak rotalama için ek bir elektrik tabanlı bir üst ağ katmanı bulunmamaktadır ve ağ katmanındaki yönlendirme işlemci düzeyinde tanımlanmış yazılım tabanlı ağ yapısı ile sağlanmıştır.

Önerilen donanım ve yazılım mimarilerinin başarımını ölçmek için teorik ve benzetim çalışmaları yapılmış ve sistem başarımı test edilmiştir. Yapılan teorik ve benzetim çalışmaları, önerilen sistemin seçilen gerçek problemler üzerinde başarım sağladığını gösterilmiş, ayrıca önerilen iletişim örüntüleri bilinen boyut sıralı algoritmalara göreli olarak birçok veri boyutunda daha iyi sonuç vermiştir. Çalışma kapsamında kelebek ağlar için önerilen iletişim örüntülerinden bağımsız olarak uygulamaya özel bir iletişim örüntüsü tanımlanmış ve düşük işlemci boyutlarında bile başarım sağlanmıştır.

Önerilen sistem mimarisi gerçek zamanlı sistemler için uygun bir sistem olmasıyla birlikte yüksek bant genişliği ve düşük güç tüketimi vaat eden yeni nesil yonga iletişimi bileşenleri olan fotonik sistemlerden faydalanılmıştır.

Önerilen yaklaşımda iletişim süreçleri tahmin edilebilir ve belirlenebilir yapıda oldukları için görev zamanlama yönetimi ile gerçek zamanlı periyodik görevler olarak yönetilmişlerdir. İletişim yapıları periyodik görev yapısında olduğu için, sistem mimarisinin periyodik diğer gerçek zamanlı görev zamanlama yaklaşımlarına uyarlanmasına da olanak vermektedir.

Önerilen belirlenebilir ve öngörülebilir temel örüntü grubu, fotonik ağlar üzerinde yönlendirme ve dalga boyu atama (RWA) problemini çözmek için yenilikçi bir yaklaşım getirmektedir. RWA sorununun çözülmesine ek olarak, iletişimin işlemci ve sistem

tasarımdan belirlenebilir yöntemler ile denetimi, gerçek zamanlı sistemler için kritik olan güç tüketimi sorunlarına önlem alınabilmesine ve güç sezimli sistemler için protokoller geliştirilebilmesini sağlayabilir.

İleriki çalışmalarda, önerilen mimarinin, çok işlemcili gerçek zamanlı işletim sistemleri için bir katman olarak ele alınması ve bu bağlamda bellek yönetimi gibi diğer çekirdek katmanların tasarlanması hedeflenmektedir. Bununla birlikte, önerilen fotonik alt yapının mod bölmeli çoklama yaklaşımları ile birleştirilmesi düşünülmektedir. Bununla birlikte, mimarinin çalışma dışındaki gerçek zamanlı sistem senaryolarına uyarlanarak geliştirilmesi ve analizlerine devam edilecektir.

KAYNAKLAR

[1] G. E. Moore, "Cramming more components onto integrated circuits (Reprinted from Electronics, pg 114-117, April 19, 1965)," 1965.

[2] P. Gargini, "ITRS past, present and future," in *ITRC-RS Workshop*, 2015.

[3] P. A. Laplante and S. J. Ovaska, *Real-Time Systems Design and Analysis: Tools for the Practitioner*. Wiley-IEEE Press, 2012.

[4] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection networks: An engineering approach*. 2003.

[5] D. Vantrease *et al.*, "Corona: System Implications of Emerging Nanophotonic Technology," in *2008 International Symposium on Computer Architecture*, 2008, vol. 36, no. 3, pp. 153–164.

[6] B. Hoefflinger, *Chips 2020: A Guide To the Future of Nanoelectronics*. Springer Berlin Heidelberg, 2012.

[7] A. Shacham, K. Bergman, and L. P. Carloni, "Photonic Networks-on-Chip for Future Generations of Chip Multiprocessors," *IEEE Trans. Comput.*, vol. 57, no. 9, pp. 1246–1260, Sep. 2008.

[8] R. G. Beausoleil, "Large-scale integrated photonics for high-performance interconnects," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 7, no. 2, pp. 1–54, Jun. 2011.

[9] H. Chetto, M. Silly, and T. Bouchentouf, "Dynamic scheduling of real-time tasks under precedence constraints," *J. Real-Time Syst.*, vol. 2, no. 3, pp. 181–194, Sep. 1990.

[10] M. J. Flynn, "Some computer organization and their effectiveness," *IEEE Trans. Comput.*, vol. C-21, no. 9, pp. 948–960, 1972.

[11] H. Temuçin, "Torus Ağı Benzetiminin Çok işlemcili mimariler için gerçekleşitirimi," Hacettepe Üniversitesi, 2011.

[12] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. 2004.

[13] M. Petracca, B. G. Lee, K. Bergman, and L. P. Carloni, "Photonic NoCs: System-

Level Design Exploration," *IEEE Micro*, vol. 29, no. 4, pp. 74–85, Jul. 2009.

[14] S. Poddar, P. Ghosal, and H. Rahaman, "Design of a High-Performance CDMA-Based Broadcast-Free Photonic Multi-Core Network on Chip," *ACM Trans. Embed. Comput. Syst.*, vol. 15, no. 1, pp. 1–30, Jan. 2016.

[15] J. Duato, I. Johnson, J. Flich, F. Naven, and P. Garca, "A New Scalable and Cost Effective Congestion Management Strategy for Lossless Multistage Interconnection Networks," *Proc. Int. Symp. High Perform. Comput. Archit.*, pp. 559–564, 2005.

[16] P. McKinley, T. Yih-jia, and D. Robinson, "A Survey of Collective Communication in Wormhole-Routed Massively Parallel Computers," no. June, 1994.

[17] P. McKinley, Y. Tsai, and D. Robinson, "Collective communication in wormhole-routed massively parallel computers," *Computer (Long. Beach. Calif).*, 1995.

[18] K. M. Imre, C. Baransel, and H. Artuner, "Efficient and scalable routing algorithms for collective communication operations on 2D all-port torus networks," *Int. J. Parallel Program.*, vol. 39, no. 6, pp. 746–782, Dec. 2011.

[19] P. Vernon, "Systems in engineering," *IEE Rev.*, vol. 35, no. 10, p. 383, 1989.

[20] L. E. Jackson and G. N. Rouskas, "Deterministic preemptive scheduling of real-time tasks," *Comput. IEEE*, vol. 35, no. 5, 2002.

[21] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973.

[22] B. P. Douglass, *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems*. Addison-Wesley Professional, 2002.

[23] A. Burns, "Scheduling hard real-time systems: a review," *Softw. Eng. J.*, vol. 6, no. 3, p. 116, 1991.

[24] Y.-C. Wang and K.-J. Lin, "Implementing a general real-time scheduling framework in the RED-Linux real-time kernel," in *Proceedings 20th IEEE Real-Time Systems Symposium (Cat. No.99CB37054)*, 1999, pp. 246–255.

[25] S. Collette, L. Cucu, and J. Goossens, "Integrating job parallelism in real-time scheduling theory," *Inf. Process. Lett.*, vol. 106, no. 5, pp. 180–187, 2008.

[26] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. Anderson, and S. Baruah, "A

categorization of real-time multiprocessor scheduling problems and algorithms," *Handb. Sched. Algorithms Methods Model.*, pp. 1–30, 2004.

[27] D. Stewart and P. Khosla, "Real-Time Scheduling of Dynamically Reconfigurable Systems," *Inst. Softw. Res.*, 1991.

[28] L. Sha *et al.*, "Real time scheduling theory: A historical perspective," *Real-Time Systems*, vol. 28, no. 2–3 SPEC. ISS. Kluwer Academic Publishers, pp. 101–155, Nov-2004.

[29] T. P. Baker, "Stack-based scheduling of realtime processes," *Real-Time Syst.*, vol. 3, no. 1, pp. 67–99, Mar. 1991.

[30] R. Jain, C. J. Hughes, and S. V. Adve, "Soft Real-Time Scheduling on Simultaneous Multithreaded Processors," *Proc. 23rd IEEE Real-Time Syst. Symp.*, no. December, pp. 134–145, 2002.

[31] S. K. Dhall and C. L. Liu, "On a Real-Time Scheduling Problem," *Oper. Res.*, vol. 26, no. 1, pp. 127–140, Feb. 1978.

[32] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, "Proportionate progress: A notion of fairness in resource allocation," *Algorithmica*, vol. 15, no. 6, pp. 600–625, Jun. 1996.

[33] K. Ramamritham, J. A. Stankovic, and P.-F. Shiah, "Efficient scheduling algorithms for real-time multiprocessor systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 1, no. 2, pp. 184–194, Apr. 1990.

[34] Mok and A. K., "Fundamental design problems of distributed systems for the hard-real-time environment." Massachusetts Institute of Technology, 1983.

[35] J. H. Anderson, J. M. Calandrino, and U. C. Devi, "Real-Time Scheduling on Multicore Platforms," *Proc. 12th IEEE Real-Time Embed. Technol. Appl. Symp.*, pp. 179–190, 2006.

[36] S. Kato and N. Yamasaki, "Real-time scheduling with task splitting on multiprocessors," in *Proceedings - 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 2007*, 2007, pp. 441–450.

[37] J. M. Calandrino, J. H. Anderson, and D. P. Baumberger, “A Hybrid Real-Time Scheduling Approach for Large-Scale Multicore Platforms,” *19th Euromicro Conf. Real-Time Syst.*, pp. 247–258, 2007.

[38] A. Saifullah, J. Li, K. Agrawal, C. Lu, and C. Gill, “Multi-core real-time scheduling for generalized parallel task models,” in *Real-Time Systems*, 2013, vol. 49, no. 4, pp. 404–435.

[39] A. Saifullah, D. Ferry, J. Li, K. Agrawal, C. Lu, and C. Gill, “Parallel Real-Time Scheduling of DAGs,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3242–3252, Dec. 2014.

[40] G. Xie, G. Zeng, L. Liu, R. Li, and K. Li, “High performance real-time scheduling of multiple mixed-criticality functions in heterogeneous distributed embedded systems,” *J. Syst. Archit.*, vol. 70, pp. 3–14, 2016.

[41] G. Xie, G. Zeng, L. Liu, R. Li, and K. Li, “Mixed real-time scheduling of multiple DAGs-based applications on heterogeneous multi-core processors,” *Microprocess. Microsyst.*, vol. 47, pp. 93–103, 2016.

[42] H. Chu and K. Nahrstedt, “A soft real time scheduling server in UNIX operating system,” Springer Berlin Heidelberg, 1997, pp. 153–162.

[43] Z. Deng and J. W.-S. Liu, “Scheduling real-time applications in an open environment,” in *Proceedings Real-Time Systems Symposium*, 1997, pp. 1–25.

[44] P. A. Dinda, “A prediction-based real-time scheduling advisor,” in *Proceedings - International Parallel and Distributed Processing Symposium, IPDPS 2002*, 2002, pp. 88–95.

[45] D. Ferry, Jing Li, M. Mahadevan, K. Agrawal, C. Gill, and Chenyang Lu, “A real-time scheduling service for parallel tasks,” in *2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2013, pp. 261–272.

[46] Y. Yan *et al.*, “Real-time android with RTDroid,” *MobiSys ’14*, pp. 273–286, 2014.

[47] F. Malandrino, C. Casetti, C.-F. Chiasserini, and S. Zhou, “Real-Time Scheduling for Content Broadcasting in LTE,” *2014 IEEE 22nd Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst.*, pp. 126–131, Sep. 2014.

[48] F. Gruian, “Hard real-time scheduling using stochastic data and {DVS} processors,” *Proc. Int. Symp. Low-Power Electron. Des.*, pp. 46–51, 2001.

[49] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, “Dynamic and aggressive scheduling techniques for power-aware real-time systems,” *22nd IEEE Real-Time Syst. Symp. 2001. (RTSS 2001). Proc.*, pp. 95–105, 2001.

[50] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, “Power-aware scheduling for periodic real-time tasks,” *IEEE Trans. Comput.*, vol. 53, no. 5, pp. 584–600, May 2004.

[51] M. Digalwar, S. Mohan, and B. K. Raveendran, “Energy aware real time scheduling algorithm for mixed task set,” in *Proceedings of the 2013 International Conference on Advanced Electronic Systems, ICAES 2013*, 2013, pp. 325–327.

[52] J. Li, M. Qiu, J.-W. Niu, L. T. Yang, Y. Zhu, and Z. Ming, “Thermal-aware task scheduling in 3D chip multiprocessor with real-time constrained workloads,” *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 2, pp. 1–22, Feb. 2013.

[53] M. Chetto, “Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems,” *IEEE Trans. Emerg. Top. Comput.*, vol. 2, no. 2, pp. 122–133, Jun. 2014.

[54] M. Bambagini, M. Bertogna, and G. Buttazzo, “On the effectiveness of energy-aware real-time scheduling algorithms on single-core platforms,” in *19th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2014*, 2014, pp. 1–8.

[55] M. Bambagini, M. Marinoni, H. Aydin, and G. Buttazzo, “Energy-Aware Scheduling for Real-Time Systems,” *ACM Trans. Embed. Comput. Syst.*, vol. 15, no. 1, pp. 1–34, Jan. 2016.

[56] R. Santhosh and T. Ravichandran, “Pre-emptive scheduling of on-line real time services with task migration for cloud computing,” in *2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering*, 2013, pp. 271–276.

[57] L. Yang, X. Zhu, H. Chen, J. Wang, S. Yin, and X. Liu, “Real-Time Tasks Oriented Energy-Aware Scheduling in Virtualized Clouds,” *IEEE Trans. Cloud Comput.*, vol.

2, no. 2, pp. 1–14, Apr. 2014.

[58] M. Cianchetti and D. Albonesi, “A low-latency, high-throughput on-chip optical router architecture for future chip multiprocessors,” *ACM J. Emerg. Technol.*, 2011.

[59] A. Shacham, B. G. Lee, A. Biberman, K. Bergman, and L. P. Carloni, “Photonic NoC for DMA Communications in Chip Multiprocessors,” in *15th Annual IEEE Symposium on High-Performance Interconnects (HOTI 2007)*, 2007, pp. 29–38.

[60] G. Hendry *et al.*, “Time-division-multiplexed arbitration in silicon nanophotonic networks-on-chip for high-performance chip multiprocessors,” *J. Parallel Distrib. Comput.*, vol. 71, no. 5, pp. 641–650, 2011.

[61] J. W. Goodman, F. I. Leonberger, S. Y. Kung, and R. A. Athale, “Optical Interconnections for VLSI Systems,” *Proc. IEEE*, vol. 72, no. 7, pp. 850–866, 1984.

[62] S. Bahirat and S. Pasricha, “A Software Framework for Rapid Application-Specific Hybrid Photonic Network-on-Chip Synthesis,” *Electronics*, vol. 5, no. 2, p. 21, May 2016.

[63] A. Biberman *et al.*, “Broadband Silicon Photonic Electrooptic Switch for Photonic Interconnection Networks,” *IEEE Photonics Technol. Lett.*, vol. 23, no. 8, pp. 504–506, Apr. 2011.

[64] S. Koohi and S. Hessabi, “Hierarchical opto-electrical on-chip network for future multiprocessor architectures,” *J. Syst. Archit.*, vol. 57, no. 1, pp. 4–23, 2011.

[65] Q. Li, D. Nikolova, D. Calhoun, and Y. Liu, “Single Microring-Based 2x2 Silicon Photonic Crossbar Switches,” *IEEE Photonics*, 2015.

[66] A. Gazman *et al.*, “Software-defined control-plane for wavelength selective unicast and multicast of optical data in a silicon photonic platform,” *Opt. Express*, vol. 25, no. 1, pp. 232–242, Jan. 2017.

[67] K. İmre, “Dual-mode routing approach for photonic network on chip platforms,” *J. Supercomput.*, vol. 72, no. 3, pp. 904–925, Mar. 2016.

[68] H. Jia *et al.*, “5-port optical router based on Si microring optical switches for photonic networks-on-chip,” *IEEE Photonics Technol. Lett.*, pp. 1–1, 2016.

[69] L.-W. Luo *et al.*, “WDM-compatible mode-division multiplexing on a silicon chip,”

Nat. Commun., vol. 5, pp. 50–65, Jan. 2014.

[70] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly," in *Proceedings of the 36th annual international symposium on Computer architecture - ISCA '09*, 2009, vol. 37, no. 3, p. 429.

[71] M. A. I. Sikder, A. K. Kodi, M. Kennedy, S. Kaya, and A. Louri, "OWN: Optical and Wireless Network-on-Chip for Kilo-core Architectures," in *2015 IEEE 23rd Annual Symposium on High-Performance Interconnects*, 2015, pp. 44–51.

[72] H. Gu, K. Gao, Z. Wang, Y. Yang, and X. Yu, "New path-setup method for optical network-on-chip," *ETRI J.*, 2014.

[73] M. C. Meyer, A. Ben Ahmed, Y. Tanaka, and A. Ben Abdallah, "On the Design of a Fault-Tolerant Photonic Network-on-Chip," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, 2015, pp. 821–826.

[74] A. Ben Ahmed and A. Ben Abdallah, "Hybrid silicon-photonic network-on-chip for future generations of high-performance many-core systems," *J. Supercomput.*, vol. 71, no. 12, pp. 4446–4475, Dec. 2015.

[75] D. Dang, B. Patra, and R. Mahapatra, "A 2-layer laser multiplexed photonic network-on-chip," in *Sixteenth International Symposium on Quality Electronic Design*, 2015, pp. 397–401.

[76] K. Zhu, H. Gu, Y. Yang, W. Tan, and B. Zhang, "A 3D multilayer optical network on chip based on mesh topology," *Photonic Netw. Commun.*, vol. 32, no. 2, pp. 293–299, Oct. 2016.

[77] Z. Li *et al.*, "Iris: A Hybrid Nanophotonic Network Design for High-Performance and Low-Power on-Chip Communication," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 7, no. 2, pp. 1–22, Jun. 2011.

[78] G. Hendry, J. Chan, S. Kamil, and L. Oliker, "Silicon nanophotonic network-on-chip using TDM arbitration," *2010 18th IEEE*, 2010.

[79] Yih-Jia Tsai and P. K. McKinley, "Broadcast in all-port wormhole-routed 3D mesh networks using extended dominating sets," in *Proceedings of 1994 International Conference on Parallel and Distributed Systems*, pp. 120–127.

[80] Nicolas LIMARE, “Integer and Floating-Point Arithmetic Speed vs Precision,” 2014. [Online]. Available: http://nicolas.limare.net/pro/notes/2014/12/12_arit_speed/.

[81] J. You and N. C. Panoiu, “Calculation of bit error rates in optical systems with silicon photonic wires,” *IEEE J. Quantum Electron.*, vol. 51, no. 4, pp. 1–8, 2015.

[82] K. . C. L. P. Shacham A.; Bergman, “On the Design of a Photonic Network-on-Chip T2 - Networks-on-Chip, 2007. NOCS 2007. First International Symposium on,” *Networks-on-Chip, 2007. NOCS 2007. First Int. Symp.*, pp. 53–64, 2007.

[83] V. K. Narayana, S. Sun, A. Mehrabian, V. J. Sorger, and T. El-Ghazawi, “HyPPI NoC: Bringing Hybrid Plasmonics to an Opto-Electronic Network-on-Chip,” 2017.

[84] D. N. Kanellopoulos, *Intelligent multimedia techologies for networking applications : techniques and tools*. Information Science Reference, 2013.

[85] N. Ahmed and K. R. Rao, “Fast Fourier Transform,” in *Orthogonal Transforms for Digital Signal Processing*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 54–84.

ÖZGEÇMİŞ

Kimlik Bilgileri

Adı Soyadı : Hüseyin Temuçin

Doğum Yeri : Ankara

Medeni Hali : Evli

E-posta : htemucin@gmail.com

Adresi : Ata Mahallesi Panorama Gold Sitesi B-24, Yenimahalle, Ankara

Eğitim

Lise : Malatya Süper Lisesi

Lisans : Hacettepe Üniversitesi Bilgisayar Mühendisliği

Yüksek Lisans : Hacettepe Üniversitesi Bilgisayar Mühendisliği

Doktora : Hacettepe Üniversitesi Bilgisayar Mühendisliği

Yabancı Dil ve Düzeyi

İngilizce- ÜDS 82.5 (2008-Güz)

İş Deneyimi

Hacettepe Üniversitesi : Araştırma Görevlisi, 2008 - (Devam ediyor)

Deneyim Alanları

Koşut ve Dağıtılmış Sistemler, Koşut ve Dağıtılmış Benzetim, Fotonik Tabanlı Yonga üzeri ağlar, Makine öğrenmesi, Nesnelerin İnterneti, Sağlık Bilişim Sistemleri

Tezden Üretilmiş Projeler ve Bütçeleri

Tezden Üretilmiş Yayınlar

Tezden Üretilmiş Tebliğ ve/veya Poster sunumu ile Katıldığı Toplantılar

- A photonic-based sensor data distribution approach for a real-time multi-processor system, IEEE 25. Sinyal İşleme ve İletişim Uygulamaları Kurultayı, 2017.
- Fotonik tabanlı yonga üzeri ağ sistemleri ve torus topolojisi için dalga boyu seçmeli fotonik bir anahtar tasarımı, 5. Ulusal Yüksek Başarımlı Hesaplama Konferansı, 2017.

HACETTEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
YÜKSEK LİSANS/DOKTORA TEZ ÇALIŞMASI ORJİNALLİK RAPORU

HACETTEPE ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI BAŞKANLIĞI'NA

Tarih: 14/03/2018

Tez Başlığı / Konusu: Çok çekirdekli mimari üzerinde gerçek zamanlı görev zamanlama ve iletişim yönetim algoritmalarının birlikte tasarımı / Gerçek zamanlı çok işlemcili sistemler

Yukarıda başlığı/konusu gösterilen tez çalışmamın a) Kapak sayfası, b) Giriş, c) Ana bölümler d) Sonuç kısımlarından oluşan toplam 133 sayfalık kısmına ilişkin, 14/03/2018 tarihinde tez danışmanım tarafından *Turnitin* adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % 2'dir.

Uygulanan filtrelemeler:

1- Kaynakça hariç
2- Alıntılar dâhil
3- 5 kelimeden daha az örtüşme içeren metin kısımları hariç

Hacettepe Üniversitesi Fen Bilimleri Enstitüsü Tez Çalışması Orjinallik Raporu Alınması ve Kullanılması Uygulama Esasları'nı inceledim ve bu Uygulama Esasları'nda belirtilen azami benzerlik oranlarına göre tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Gereğini saygılarımla arz ederim.

14/03/2018

Adı Soyadı: HÜSEYİN TEMUÇİN
Öğrenci No: N10349084
Anabilim Dalı: BİLGİSAYAR MÜHENDİSLİĞİ
Programı: DOKTORA
Statüsü: ☐ Y.Lisans ☐ Doktora ☐ Bütünleşik Dr.

DANIŞMAN ONAYI

UYGUNDUR.

DOÇ.DR. KAYHAN M. İMRE

HACETTEPE UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING
THESIS/DISSERTATION ORIGINALITY REPORT

HACETTEPE UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING
TO THE DEPARTMENT OF COMPUTER ENGINEERING

Date: 14/03/2018

Thesis Title / Topic: Design of real-time scheduling and communication management algorithms on multicore architecture / Real-time multicore systems
According to the originality report obtained by my thesis advisor by using the *Turnitin* plagiarism detection software and by applying the filtering options stated below on 14/03/2018 for the total of 133 pages including the a) Title Page, b) Introduction, c) Main Chapters, d) Conclusion sections of my thesis entitled as above, the similarity index of my thesis is 2%.

Filtering options applied:

1. Bibliography/Works Cited excluded
2. Quotes included
3. Match size up to 5 words excluded

I declare that I have carefully read Hacettepe University Graduate School of Sciene and Engineering Guidelines for Obtaining and Using Thesis Originality Reports; that according to the maximum similarity index values specified in the Guidelines, my thesis does not include any form of plagiarism; that in any future detection of possible infringement of the regulations I accept all legal responsibility; and that all the information I have provided is correct to the best of my knowledge.

I respectfully submit this for approval.

14/03/2018

Name Surname: HÜSEYİN TEMUÇİN
Student No: N10349084
Department: COMPUTER ENGINEERING
Program: Ph.D.
Status: ☐ Masters ☐ Ph.D. ☐ Integrated Ph.D.

ADVISOR APPROVAL

APPROVED.

ASSOC.PROF. KAYHAN M. İMRE