YONGA-ÜSTÜ-AĞLAR İÇİN UYGULAMAYA ÖZGÜ YENİDEN YAPILANDIRILABİLİR TOPOLOJİ TASARIMI

APPLICATION-SPECIFIC RECONFIGURABLE TOPOLOGY DESIGN FOR NETWORK-ON-CHIPS

PINAR KÜLLÜ

PROF. DR. SÜLEYMAN TOSUN

Tez Danışmanı

Hacettepe Üniversitesi

Lisansüstü Egitim-Öğretim ve Sınav Yönetmeliğinin

Bilgisayar Mühendisliği Anabilim Dalı için Öngördüğü

DOKTORA TEZİ olarak hazırlanmıştır.

Şubat 2020

Kızıma. . .

ÖZET

YONGA-ÜSTÜ-AĞLAR İÇİN UYGULAMAYA ÖZGÜ YENİDEN YAPILANDIRILABİLİR TOPOLOJİ TASARIMI

Pınar KÜLLÜ

Doktora, Bilgisayar Mühendisliği

Danışman: Prof. Dr. Süleyman TOSUN

Şubat 2020, 100 sayfa

Entegre devre boyutlarındaki hızlı azalma, her nesilde tek bir yongaya daha fazla bileşen yerleştirmeyi mümkün kılar. Bileşenlerdeki artış, geleneksel kablolama tabanlı iletişim yöntemlerinden daha iyi bir iletişim mekanizması gerektirse de, teknoloji boyutlarının daralması nedeniyle artan kalıcı arızaları tolere etmek için yeni tasarım algoritmalarına ihtiyaç vardır. Bu sorunu çözmek ve çok büyük sistemlerde iletişim taleplerine ayak uydurmak için Yonga-üstü-Ağ (YüA) paradigması geliştirilmiştir. Bir YüA mimarisi tasarlanırken, tasarımcılar bant genişliği, performans, enerji tüketimi, maliyet, yeniden kullanılabilirlik ve hataya dayanıklılık gibi çeşitli kriterleri dikkate almalıdır.

Tez kapsamında ilk olarak, belirli bir uygulama için tasarlanmış bir YüA mimarisinde tek kalıcı bağlantı hatasını tolere edebilen hataya dayanıklı bir topoloji oluşturma yöntemi sunulmuştur. Oluşturulan topolojiler, uygulamanın iletişim düğümleri arasında en az iki alternatif yol sağlayarak hataya dayanıklılık sağlar. Yöntem, halka topolojisine dayalı bir başlangıç popülasyonu üreten ve genetik operatörler aracılığıyla enerji tüketimi açısından daha iyi düzensiz topolojiler oluşturan genetik algoritma tabanlı bir yöntemdir. Önerilen yöntemin amaç işlevi, ağ iletişiminden kaynaklanan enerji tüketimini en aza indirmektir.

Uygulamaya özgü oluşturulan hata kaldırabilir topolojilerde sistem bileşenleri arasında en az iki yol bulunmaktadır. Ancak yonganın çalışma sürecinde bağlantılar üzerinde bir hata ortaya çıkarsa veriler doğru bir şekilde hedef yönlendiriciye iletilemez. Bu nedenle bu tarz topolojiler için, hataları sistemin çalışması sırasında dinamik olarak tespit edebilecek ve hatanın durumuna göre paketi alternatif yollardan hedef yönlendiriciye iletebilecek bir yapı tasarlanmıştır.

Yonga-üstü-Ağlarda örgü topolojisi tasarım için en yaygın olarak kullanılan topoloji olmasına rağmen, ağ tıkanıklığı ve enerji tüketimi gibi çeşitli sorunları vardır. Yeniden yapılandırılabilir ağ topolojisi, ağ tıkanıklığını azaltmak için daha fazla eşleme ve yönlendirme seçeneği sunduğu için geleneksel ağa iyi bir alternatiftir. Bununla birlikte, yeniden yapılandırılabilir örgü yapıları için verimli eşleme ve yönlendirme algoritmaları sayısı oldukça azdır. Tez çalışması kapsamında, eş zamanlı olarak, uygulama düğümlerini yeniden yapılandırılabilir ağ yapısı üzerinde eşleyen ve enerji minimizasyonu amacıyla iletişim çiftleri arasındaki yönlendirme yollarını belirleyen genetik algoritma (GA) tabanlı bir yöntem önerilmiştir.

Son olarak, düzenli ve düzensiz YüA topolojilerinin avantajlarını birleştiren iki aşamalı yeni bir yöntem sunulmuştur. İlk adımda, yöntem, alanı ve enerjiyi optimize etmek için en az miktarda yönlendirici ve bağlantı kullanan ve bu nedenle iletişim düğümleri arasında sadece bir yönlendirme yolu bulunan düzensiz topolojiyi, hataya dayanıklı hale getirmek için yeniden yapılandırılabilir ağ topolojisine eşlemektedir. İkinci aşamada ise, yerleştirilen düğümler arasındaki yollara karar verilmektedir.

Anahtar Kelimeler: Yonga-üstü-Ağ, Hata kaldırabilir Topoloji, Genetik Algoritma, Yeniden Yapılandırılabilir Örgü, Hata Mekanizması

ABSTRACT

APPLICATION-SPECIFIC RECONFIGURABLE TOPOLOGY DESIGN FOR NETWORK-ON-CHIPS

Pınar KÜLLÜ

Doctor of Philosophy, **Department of Computer Engineering**

Supervisor: Prof. Dr. Süleyman TOSUN

February 2020, 100 pages

The rapid reduction in integrated circuit dimensions makes it possible to place more components on a single chip in each generation. While the increase in components requires a better communication mechanism than traditional wiring-based communication methods, new design algorithms are needed to tolerate increasing permanent failures due to shrinking technology dimensions. To solve the problem, the network-on-chip (NoC) paradigm was developed to keep pace with the communication demands on these very large systems. When designing a Network-on-Chip (NoC) architecture, designers must consider various criteria such as bandwidth, performance, energy consumption, cost, re-usability, and fault tolerance.

Within the scope of the thesis, firstly we provide a fault-tolerant topology generation method that can tolerate single permanent link failure on a NoC architecture that is designed for a particular application. The generated topologies provide fault-tolerance by providing at least two alternative paths between the application's communicating nodes. The method is a genetic algorithm based method, which generates an initial population based on ring topology and

produces better irregular topologies in terms of energy consumption through genetic operators. The objective function of the proposed method is to minimize the energy consumption resulting from network communication.

There are at least two paths between system components in an application-specific fault tolerant topologies. However, if a fault occurs on the links during the chip's running time, the data cannot be correctly transmitted to the destination router. Therefore, we propose a mechanism that detects errors dynamically during the runtime and transmits packets to the destination routers by alternative paths according to the fault condition.

Although mesh topology is most commonly used topology for NoC design, it has several problems such as network congestion and energy consumption. Reconfigurable mesh topology is a good alternative to traditional mesh since it gives more mapping and routing options for reducing network congestion. However, design automation tools still lack efficient mapping and routing algorithms for reconfigurable meshes. In this study, we propose a genetic algorithm (GA) based method that simultaneously maps the application nodes on 2D reconfigurable mesh structure and determines the routing paths between communicating pairs with the objective of energy minimization.

Finally, we present a novel two-step method that combines the advantages of regular and irregular NoC topologies. In the first step, it maps the irregular topology, which uses the least amount of routers and links to minimize the area and energy and offers only one routing path between communicating nodes, to the reconfigurable network topology. In the second step, the method decides the paths between the placed nodes.

Keywords: Network-on-Chip, Fault-tolerant Topology, Genetic Algorithm, Reconfigurable Mesh, Fault Mechanism

TEŞEKKÜR

Tez çalışmamın her aşamasında bilgi ve deneyimleri ile bana yol gösteren, her durumda desteğini, yardımlarını esirgemeyen, sabrı ve motivasyonu ile yanımda olan danışman hocam Sayın Prof. Dr. Süleyman Tosun'a,

Tezimi inceyelerek değerli yorumları ile katkıda bulunan tez savunma sınavı jüri üyeleri Sayın Prof. Dr. Özcan Öztürk'e, Sayın Doç. Dr. Lale Özkahya'ya, Sayın Doç. Dr. Kayhan İmre'ye, ve Sayın Doç. Dr. Murat Hüsnü Sazlı'ya,

Akademik hayatım boyunca değerli fikirlerini benden esirgemeyen ve her türlü zorlukta yanımda olan sevgili mesai arkadaşlarım Sayın Sevgi Yiğit Sert'e ve Sayın Merve Özkan Okay'a,

Bu tezin tamamlanmasında 117E130 numaralı proje kapsamında maddi destek sağlayan TÜBİTAK'a, ve bu projede bilgileri ile çalışmalarımıza yön veren Sayın Prof. Dr. Suat Özdemir'e ve Sayın Dr. Öğr. Üyesi Yılmaz Ar'a,

Hem akademik çalışmalarımda hem de diğer bütün zoruklarda en büyük desteği gördüğüm sevgili eşim Kurtuluş'a, bu yolda maddi manevi desteklerini esirgemeyen aileme ve en önemlisi varlığıyla en büyük motivasyon kaynağım olan değerli kızım Arya'ya,

Sonsuz teşekkürlerimi sunarım.

İçindekiler

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR

Simgeler

B	Uygulama akış çizgesindeki kenarlar kümesi
D	Uygulama akış çizgesindeki düğümler kümesi
$E_{bağlantı}$	Bağlantıların enerji tüketim değeri
$E_{yönlendirici}$	Yönlendiricilerin enerji tüketim değeri
E_{toplam}	Topolojinin toplam enerji tüketim değeri
G	Düzensiz bağlantı çizgesindeki yollar kümesi
P_{boyut}	Popülasyon büyüklüğü
P_{min}	Evrim sürecinin gerçekleştirildiği popülasyon
P_{rast}	Genetik operatörlerin uygulanması için rastgele seçilen birey sayısı
$P_{yineleme}$	Genetik algoritma için yineleme sayısı
$t_{i,j}$	Uygulama akış çizgesinde d_i düğümü ile d_j düğümü arasında iletilen verinin miktarı
y	Yönlendirici
y_{min}	Topolojideki minimum yönlendirici sayısı
σ_d	Bağlantıyı güvenli olarak belirlemek için kullanılan eşik değeri
σ_g	Bağlantıyı geçici hatalı olarak belirlemek için kullanılan eşik değeri
σ_k	Bağlantıyı kalıcı hatalı olarak belirlemek için kullanılan eşik değeri

Kısaltmalar

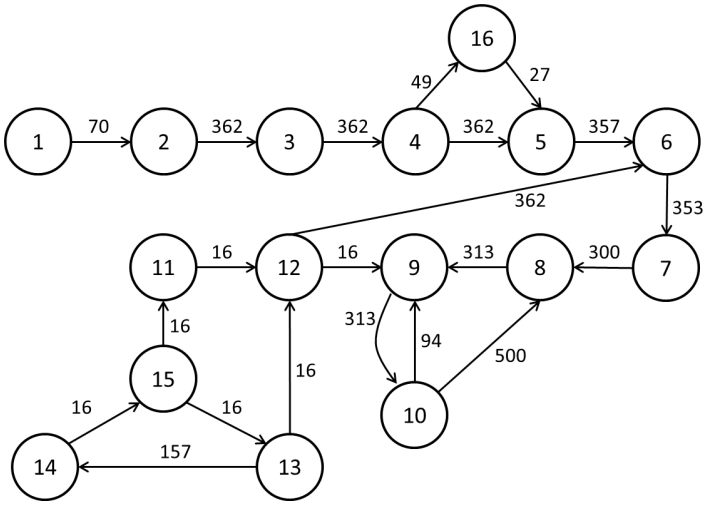
DBÇ	**D**üzensiz **B**ağlantı **Ç**izgesi
DMC	**D**epth **M**ap **C**omputation
DVOPD	**D**ual **V**ideo **O**bject **P**lane **Decoder**

FTTG	**F**ault **T**olerant **T**opology **G**eneration
GA	**G**enetic **A**lgorithm
GA-FTCT	**G**enetic **A**lgorithm-based **F**ault-tolerant **C**ustom **T**opology
GATGA	**G**enetic **A**lgorithm-based **T**opology **G**eneration **A**lgorithm
GHS	**G**eçici **H**ata **S**ayacı
H263 Dec.	H263 Decoder
H263 Enc.	H263 Encoder
HY	**H**edef **Y**önlendirici
KHS	**K**alıcı **H**ata **S**ayacı
KY	**K**aynak **Y**önlendirici
MARM-GA	**M**apping **A**pplication to **R**econfigurable **M**esh using **G**enetic **A**lgorithm
MMS	**M**ulti-**M**edia **System**
MP3 Enc.	MP3 Encoder
MWD	**M**ulti **W**indow **D**isplay
NoC	**N**etwork **O**n **C**hip
OAS	**O**rtalama **A**tlama **S**ayısı
UAÇ	**U**ygulama **A**kış **Ç**izgesi
VOPD	**V**ideo **O**bject **P**lane **Decoder**
YT	**Y**önlendirme **T**ablosu
YüA	**Y**onga **Ü**stü **A**ğ

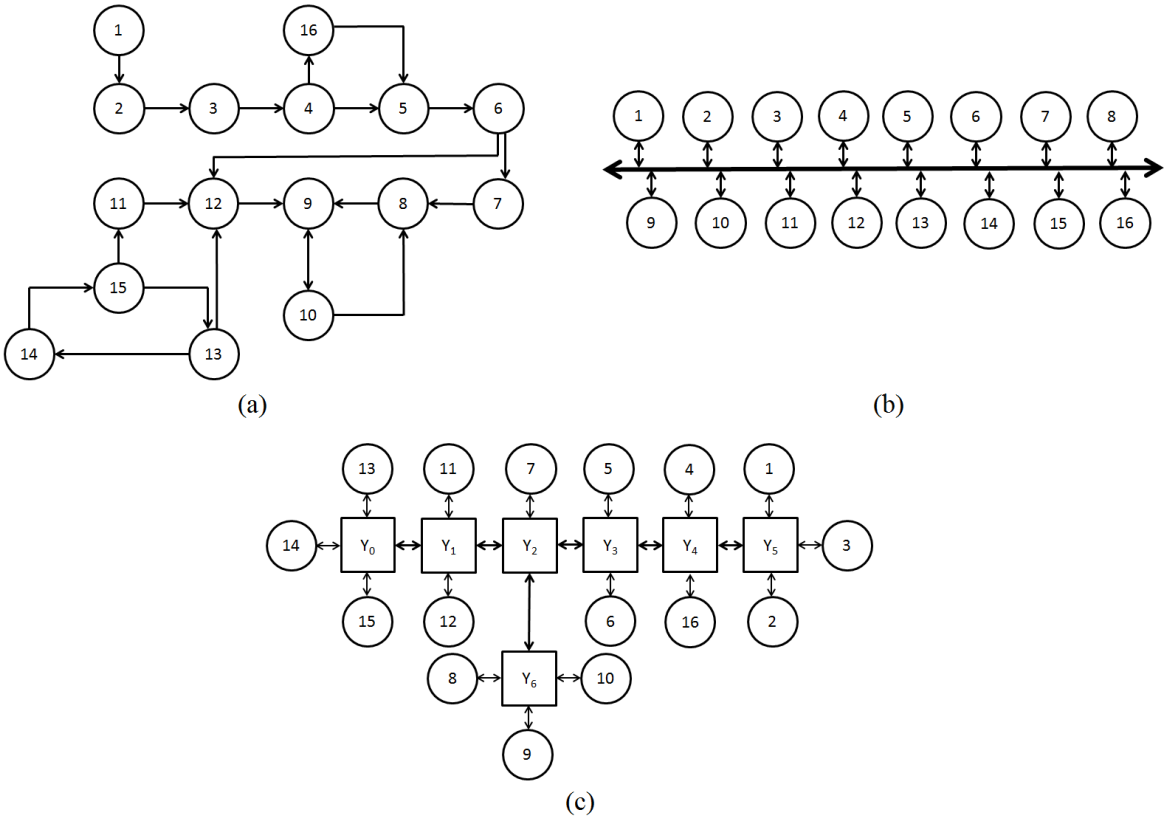
1. GİRİŞ

1.1. Motivasyon

Günümüzde, devre elemanlarının boyutlarının hızlı bir şekilde küçülmesi sonucunda bir uygulamaya ait çok sayıda çekirdek gömülü mimarilere yerleştirilebilmektedir. Şekil 1.1.'de örnek bir uygulama için akış çizgesi verilmiştir. Burada, yuvarlak şekiller bu gömülü uygulamanın sistem bileşenlerini göstermektedir. Bu şekiller arasındaki oklar ise uygulama çekirdekleri arasındaki bağlantıları temsil etmektedir. İki çekirdek arasında bağlantı olması, o bileşenlerin iletişim halinde olduklarını gösterir. Bir uygulamanın bileşenleri arasında haberleşmenin sağlanması için, gömülü sistem üzerinde temel olarak üç farklı iletişim mimarisi tanımlanabilir. En temel iletişim modeli Şekil 1.2.a'da verilmiştir. Burada her bir bileşenin doğrudan özel kablolar ile bağlandığı noktadan noktaya tabanlı haberleşme sistemi gösterilmektedir. Bu, haberleşme sistemi alternatifleri arasında en hızlı iletişim modelidir. Şekil 1.2.b'de ise klasik veri yolu tabanlı haberleşme örneği verilmiştir. Çekirdek sayısındaki artış, bileşenler arasında iletilen mesaj sayısını artırmaktadır. Bu iki haberleşme modelinde mesaj sayısının artması, performansın düşmesine ve artan enerji tüketimine neden olmaktadır. Dolayısı ile bu tarz yapılar ölçeklenemezler. Bu nedenle veriler, Şekil 1.2.c'de verildiği gibi Yonga-üstü-Ağ (YüA) mimarilerinde yönlendiriciler vasıtasıyla iletilebilirler.



Şekil 1.1. VOPD uygulamasının uygulama akış grafiği.



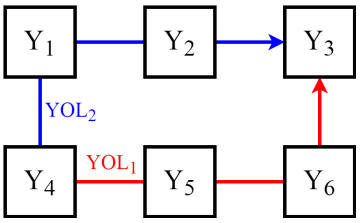
Şekil 1.2. Şekil 1.1.'de verilen VOPD uygulaması için olası yonga üstü haberleşme modelleri: (a) Noktadan noktaya, (b) klasik veri yolu tabanlı, (c) Yonga-üstü-Ağ.

Yonga-üstü-ağ haberleşmesinde ağın görevi, mesajların sistem bileşenleri arasında yönlendirici düğümler vasıtası ile iletimini sağlamaktır [4, 5]. Çok sayıda sistem bileşeni bir arada kullanıldığında, bu haberleşme sisteminde de işlemciler arası haberleşme ve veri aktarımı oldukça fazla olacaktır. Bileşenler arası haberleşme miktarı fazla olduğu zaman ağın trafik yoğunluğundan dolayı kilitlenme problemi olabilir. Bu problemin büyüklüğü, bileşenler arası yolların yapısı ve kullanılan yolun anlık veri gönderme hızıyla değişir. Bu nedenle farklı amaçlar için farklı topolojiler kullanılmaktadır. Topoloji terimi, ağdaki düğümlerin ve bu düğümler arasındaki kanalların dağılımını belirtir. Bu dağılımlar genellikle pek çok farklı şekilde tanımlanabilir. Bir ağ için topoloji seçimi, ağın mümkün olan en düşük maliyetle en iyi şekilde gösterebileceği performans ile ağın bant genişliği gibi kıstaslar göz önünde bulundurularak yapılır. Tasarlanacak ideal topoloji ile ağ mümkün olan en az gecikme ile en düşük enerjiyi tüketerek, olası karışıklıklarla en üst düzeyde başa çıkabilecek kapasiteye sahip olmalıdır. Aynı zamanda da performans açısından en iyi çıktıyı üretebilmelidir. Tüm

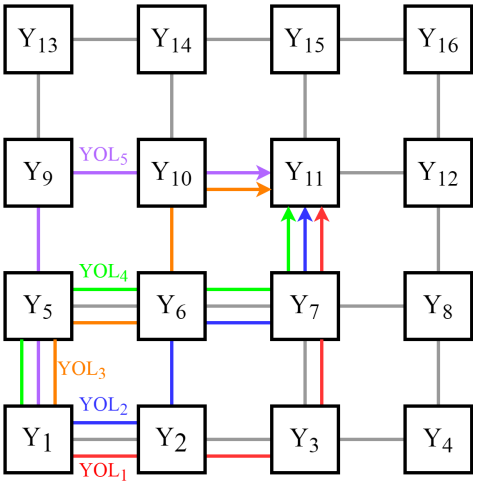
bu işlevlere sahip bir mimari geliştirmek genellikle mümkün değildir. Bu nedenle tasarımlar mümkün olduğunca çok avantajı kapsayacak şekilde seçilir. Bir tasarımcı ilk önce verilen uygulama için en uygun topolojiye karar vermelidir. Mimari seçimi, performans, maliyet, alan, sistemin toplam enerji tüketimi, hata toleransı veya yeniden kullanılabilirlik gibi çeşitli kriterler göz önünde bulundurularak, düzenli veya düzensiz topolojilerden yapılabilir. Her topoloji türünün kendi avantajları/dezavantajları vardır.

Topolojiler düzenli ve düzensiz olarak iki gruba ayrılır. İlk çalışmalar düzenli mimariler üzerine odaklanmış olsa da [6–8], zamanla enerji tüketimi ve performans gibi kriterleri daha iyi optimize edebilmek için düzensiz topolojiler üzerinde çalışmalar artmıştır [1, 9, 10]. Düzenli topolojilerin tasarımı kolaydır ve farklı uygulamalar için tekrar kullanılabilirler. Düzenli topolojilerin çoğu, iletişim düğümleri çifti arasında birden fazla yol sağladığı için hataya dayanıklıdır. Bunlar, birbiri ile iletişim kuran her düğüm çifti arasında birden fazla yol sağladığı için hataya dayanıklıdır. Yönlendirme yolu üzerindeki herhangi bir bağlantıda bir hata olması durumunda, paketler alternatif bir yoldan iletilebilir. Halka ve örgü topolojileri, düzenli topolojilerin temel örnekleridir. Şekil 1.3. ve Şekil 1.4. sırasıyla 5-portlu yönlendiriciler kullanılarak oluşturulan halka ve örgü topolojisi örneklerini göstermektedir. Halka topolojileri tek bir bağlantı hatasını tolere edebilirken, örgü topolojileri düğümler arasında birden fazla alternatif yola sahiptir. Örneğin, Şekil 1.3.'teki topolojide birinci yoldaki bir bağlantıda bir hata varsa, birinci yönlendiricideki kaynak düğümden dördüncü yönlendiricideki hedef düğüme gönderilen paket ikinci yol üzerinden iletilebilir. Şekil 1.4.'te gösterilen örgü yapısında ise, yönlendirici 1 ve yönlendirici 11 arasındaki veri birinci, ikinci, üçüncü, dördüncü veya beşinci yollardan biri ile aktarılabilir.Bununla birlikte, bu tür topolojiler uygulama, performans, maliyet, alan ve/veya enerji tüketimi açısından gerekli optimizasyonu sağlayamayabilirler. Uygulamaya özgü düzensiz topolojiler bu kriterler için daha büyük bir optimizasyon alanına sahip olduğundan, düzensiz topoloji tasarım yöntemleri son zamanlarda düzenli olanlardan daha fazla ilgi görmüştür. Bununla birlikte, düzensiz topoloji tasarım yöntemlerinin çoğu hata toleransını dikkate almamaktadır. Literatürde, hata toleranslı düzensiz topoloji tasarım yöntemleri çok nadirdir [11–15].

Düzensiz topolojiler, bir uygulama için özel olarak üretilir. Bu tür topolojilerde alan ve



Şekil 1.3. Halka topolojide Yönlendirici 1 ve Yönlendirici 4 arasındaki 2 alternatif yol örneği.



Şekil 1.4. Örgü topolojide Yönlendirici 1 ve Yönlendirici 11 arasındaki 5 alternatif yol örneği.

performans gibi kriterler daha iyi optimize edilebilir. Şekil 1.2.c, VOPD uygulaması için oluşturulan uygulamaya özgü düzensiz topolojinin bir örneğini göstermektedir. Böyle bir topoloji oluştururken amaç, minimum sayıda bileşen kullanarak sistemin toplam enerji tüketimini en aza indirmektir. Bununla birlikte, hata toleransı yeteneği göz ardı edilir. Topolojinin hataya dayanıklı olmasını sağlamak için sistemin her bir bileşeni arasında en az iki yol olmalıdır. Hata kaldırabilir bir topolojide, bileşenler arasındaki yollardan birinde bir hata varsa, paketler alternatif yol/yollardan hedef düğüme iletilebilir. Her ne kadar verilen örnekteki topoloji alan, maliyet ve performans gibi kriterler dikkate alınarak oluşturulmuş olsa da, sistem, iletişim düğümleri arasındaki yolda bir hata olduğunda çalışmaya devam edemeyecektir.

Birçok çalışma, çip üzerindeki düğümlerin dağılımını ve çalışma zamanı boyunca paketlerin nasıl iletileceğini, yani; her bir iletişim düğüm çifti arasındaki yolları, tasarım sürecinde belirler. Başka bir deyişle, tek bir uygulama için yonga üstü ağı optimize ederler. Literatürdeki bazı çalışmalar, çip üzerine aynı anda birden fazla uygulamayı entegre edebilecek yapılar önermiştir [2, 3, 16]. Bu yapılarda, yönlendiriciler arasına yapılandırma anahtarları yerleştirilir. Uygulamaların tüm düğümleri yapıya eşlendiğinde, anahtarlar çalışan uygulamaya göre yapılandırılabilir. Paketlerin, daha fazla güç kullanan yönlendiriciler yerine anahtarların üzerinden iletilmesi, sistemin toplam enerji tüketimini de azaltmktadır. Ayrıca, yönlendiriciler arasındaki mesafe kısaltılır ve uzak yönlendiriciler arasındaki paketler bu yapılarda daha hızlı iletilebilir. [2, 3, 16] numaralı çalışmalar, yeniden yapılandırılabilir ağ yapıları için haritalama algoritmaları önermektedir. Bu çalışmalar eşleme işlemi tamamlandıktan sonra, iletişim düğümleri arasındaki yolları belirlemek için yönlendiriciler arasındaki anahtarları yapılandırmaktadırlar.

Düzenli topolojilerde tek bağlantı hatalarını saptamak ve tolere etmek için daha önce yapılmış birçok çalışma olsa da [17–23], düzensiz topolojiler üzerine yapılan çalışma sayısı oldukça azdır [24]. Bunların çoğu, sisteme yeniden yapılandırılabilirlik ekleyerek geçici hataları çalışma zamanında algılamaya ve düzeltmeye çalışır. Örneğin, [23] hataların tespiti için aynı verinin iki kopyasını hedefe gönderir. Benzer bir şekilde [22], verinin üç kopyasını gönderir ve aynı olan iki kopyanın doğru olduğunu varsayar. Bu doğru belirlenen kopya bir sonraki hedefe gönderilir ve böylece sistemin çalışmaya devam etmesine olanak sağlanır. Bu tarz sistemler, maliyet ve enerji gibi tasarım parametrelerini olumsuz yönde etkileyen miktarda, temel tasarıma fazladan donanım eklemektedirler. [24] numaralı çalışmada hata kaldırabilir uygulamaya özgü topoloji yönlendiricilerine sistemde ortaya çıkabilecek bağlantı hatalarını tolere edebilecek yönlendirme tabloları eklenmiştir. Yonganın dışına eklenen pinler sayesinde paketlerin izleyeceği yolların belirlenebildiği yöntemin dezavantajı çalışma sırasında hataları kendi kendine belirleyebilecek ve yönlendirme tablolarını güncelleyecek bir mekanizmanın bulunmamasıdır.

1.2. Tez Çalışmasının Katkıları

Tez çalışması kapsamında ilk olarak uygulamaya özgü düzensiz topolojiler üzerine çalışmalar gerçekleştirilmiştir. Önceki çalışmalar ya hataya dayanıklı düzenli topolojiler için optimizasyon yöntemleri sunar ya da hata toleransı dikkate alınmadan enerji verimli düzensiz topolojiler üretmeye odaklanır. Literatürde, sistemde ortaya çıkabilecek hatalara dayanıklı uygulamaya özgü topoloji üretme yöntemi oldukça azdır. Tez kapsamında ilk olarak, önceki çalışmaların aksine, YüA topolojisi tasarımı için genetik algoritma (GA) tabanlı, hataya dayanıklı uygulamaya özgü topoloji oluşturma yöntemi sunulmaktadır. Yöntem diğer YüA optimizasyon çalışmalarından iki yönden ayrılmaktadır: 1) Sonuç tasarıma hata toleransı kazandırmak için temel topolojiye ihmal edilebilir miktarda fazladan ağ kaynağı ekleyen yeni bir hataya dayanıklı topoloji oluşturma fikri önerilmiştir ve 2) GA tabanlı yöntemimiz çeşitli topolojiler oluşturabilmekte ve uygulamayı aynı anda oluşturulan topolojiye eşleyebilmektedir. Yöntemin çaprazlama ve mutasyon operatörleri topoloji bağlantılarının çeşitliliğini arttırır ve verilen uygulama düğümlerini minimum enerji tüketimi ile topolojilere eşler. Ayrıca, sunulan GA tabanlı yöntem, tamsayılı doğrusal programlama tabanlı [15] ve sezgisel [11] yöntemlerin aksine fazla sayıda düğüm içeren uygulamalar için ölçeklenebilmektedir.

Hata kaldırabilen uygulamaya özgü topolojiler kullanılarak üretilen YüA tasarımlar, bir link veya verinin aktığı yoldaki başka bir ağ elemanı üzerinde bir hata varsa bunu tolere edebilebilirler. Bu özelliği, her iki yönlendirici düğüm arasında birden fazla alternatif yol olmasına borçludurlar. Ancak bu tarz sistemler, paketlerin iletiminde, en kısa yol kullanılarak oluşturulmuş yönlendirme tablosu tabanlı yönlendirme kullanmaktadırlar. Hataları tolere edebilmek için ise alternatif yönlendirme tablosuna ihtiyaç duyulmaktadır. Alternatif yönlendirme bilgisi bulunan sistemlerde, eğer uygulama için oluşturulmuş temel yönlendirme yollarında hata tespit edilirse, bu alternatif yollar kullanılabilir. Ama öncelikle hatanın tespit edilmesi ve daha sonra paketin alternatif yol üzerinden hedef yönlendiriciye yeniden yönlendirilmesi gerekmektedir. Bu amaçla, tez kapsamında sunulan bir çalışma da, uygulamaya özgü oluşturulan hata kaldırabilir topolojilere hata tespit ve yol güncelleme birimlerinin eklenmesidir. Tasarlanan yöntem ile öncelikle paketler için varsayılan yollar ve sistemde

bir ve iki hata olması durumunda paketlerin iletileceği alternatif yollar yönlendiricilerin yönlendirme tablolarına eklenmiştir. Tasarlanan hata ünitesi, iki yönlendirici düğüm üzerindeki hatanın kalıcı mı yoksa geçici mi olduğunu tespit ederek, ona göre karar alınmasını sağlamaktadır. Yönlendiricinin aldığı veride hata varsa, veriyi gönderen yönlendiriciye bilgi verilir. Eğer art arda hata olmaya devam ederse hatanın sayısına göre geçici mi kalıcı mı hata olduğuna karar verilir. Bu durumda yönlendiricilerin yönlendirme tablolarını güncelleyerek yeni bir rota oluşturmaları gerekmektedir.

Tez kapsamındaki ikinci genel başlık ise [2] numaralı çalışmada sunulan yeniden yapılandırılabilir örgü topoloji üzerinde gerçekleştirilen çalışmalardır. Bu kapsamda ilk olarak uygulama düğümlerini yeniden yapılandırılabilir ağ yapısına eşleyen ve yönlendirme yollarını ayarlamak için anahtarları yapılandıran genetik algoritma tabanlı bir yöntem önerilmiştir. Önceki yöntemlerden farklı olarak, sunulan yöntem eşleme ve yönlendirme sorunlarını aynı anda çözer. Bu, aynı zamanda hem eşleme hem de yönlendirme prosedürleri için en uygun çözümleri belirleme şansını artırır. Algoritmanın amacı, yönlendiriciler arasında iletilen paketlerin toplam dinamik enerji tüketimini en aza indirmektir.

Son olarak sunduğumuz yöntemde amacımız, örgü topolojilerinin yeniden kullanılabilirlik ve ölçeklenebilirlik özellikleri ile uygulamaya özgü topolojilerin enerji ve maliyet optimizasyon özelliklerini birleştirmektir. Hedefe ulaşmak için iki aşamalı bir yöntem önerilmiştir: (1) düzensiz topolojiyi yeniden yapılandırılabilir ağ topolojisine eşlemek ve (2) yönlendirmeler oluşturmak için yönlendiriciler arasındaki anahtarları yapılandırmak. Sunulan yöntemde düzensiz topoloji oluşturmak için [9] numaralı çalışmada sunulan GA tabanlı düzensiz topoloji oluşturma yöntemi kullanılmıştır. Ardından, oluşturulan düzensiz topoloji, [2] numaralı çalışmada önerilen yeniden yapılandırılabilir örgü topolojiye eşlenmiştir. Eşleme prosedürü için, minimum yayılan ağaç algoritması ve tasarımın enerji tüketimini dikkate alan sezgisel bir yöntem kullanılmıştır. Yöntemin ikinci adımında, eşlenen düğümler arasındaki yönlendirmelerin belirlenmesi için yönlendiriciler arasına yerleştirilen anahtarların konfigürasyonlarına karar verilmiştir.

1.3. Tez Metninin Organizasyonu

Tez metninin devamı şu şekilde tasarlanmıştır:

İkinci bölümde Yonga-üstü-Ağlar hakkında kısa bilgiler verilmiş ve bu alanda yapılan çalışmalar ele alınmıştır. Uygulama düğümlerini düzenli topolojiler üzerine eşleyen çalışmaları takiben uygulamaya özgü hata kaldırma özelliği olmayan/olan topoloji oluşturma yöntemlerine kısaca değinilmiştir. Son olarak ağın yaşam ömrü boyunca ortaya çıkabilecek hatalar ile başetme teknikleri sunan düzenli ve düzensiz topolojiler üzerinde çalışmalarını gerçekleştirmiş yöntemlerden bahsedilmiştir. Bunlara ek olarak da yeniden yapılandırılabilir örgü topolojilere eşleme ve yönlendirme belirleme algoritmaları üzerinde durulmuştur.

Üçüncü bölümde geliştirilen uygulamaya özgü hata kaldırabilir topoloji tasarım yöntemlerinden bahsedilmiştir. Hem uygulamaya özgü hata kaldırabilir topoloji tasarım yönteminin hem de bu tarz topolojilere eklenebilecek hata tespit ve yeniden yönlendirme algoritmalarının problem tanımları, yöntem detayları ve deneysel sonuçları bu kısımda paylaşılmıştır.

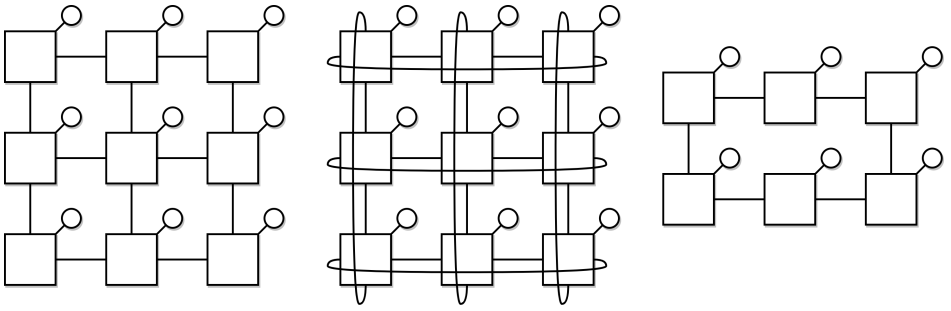
Dördüncü bölümde yeniden yapılandırılabilir örgü topoloji üzerine eşleme problemlerine getirilen çözümler üzerinde durulmuştur. Yeniden yapılandırılabilir örgü topoloji yapısının detayları ve problem tanımına değinilmiştir. Bu kısımda bu yapılar üzerinde gerçekleştirilen iki yöntemin detaylarından ve deneysel sonuçlarından bahsedilmiştir. Bu yöntemlerden ilki uygulama akış diyagramından uygulama düğümlerinin direk bu yapıya eşlenmesi problemidir. Diğeri ise hata kaldırma özelliği bulunmayan uygulamaya özgü topolojilerin alınarak bu yapı üzerine taşınması için geliştirilmiş bir yöntemdir.

Son bölümde tezin genel sonuçları ve literatüre katkıları kısaca özetlenmiştir.

2. KAYNAK ÖZETLERİ

Entegre devre üretiminde kullanılan devre elemanlarının boyutlarının nanometre seviyelerinde her birkaç yılda bir küçülmesi sonucunda milyonlarca transistor tek bir yonga üstüne dizilebilmektedir. Bunun sonucu olarak daha karmaşık ve yoğun tasarımlar tek bir yonga üstünde gerçekleştirilebilmektedir. Tek bir yonga üstüne tüm sistem bileşenlerinin gerçekleşmesiyle oluşan tasarım Yonga-üstü-Sistem (YüS) olarak adlandırılmaktadır. Fakat bu kadar çok sistem bileşenlerinin yongalar üstüne gerçekleşmesi sonucu klasik veri yolu tabanlı ve noktadan noktaya tabanlı haberleşme yöntemleri düşük performans ve senkronizasyon problemleri nedeniyle kullanışsız hale gelmektedir. Özellikle geleceğin mimarilerinin çok sayıda yüksek kapasiteli işlemcilerden oluşacağı ve işlemciler arası haberleşmenin ve veri aktarımının oldukça fazla olacağı düşünüldüğünde, daha etkin bir haberleşme yönteminin yongalar üstünde gerçekleştirilmesi kaçınılmaz olmuştur. Bu nedenle bu yüzyılın başında Yonga-üstü-Ağ (YüA) adı verilen yeni bir haberleşme yöntemi sunulmuştur [25, 26].

Yonga-üstü-Ağ teknolojisinde düğümler arası haberleşme, yönlendiriciler arasında paketler iletilerek gerçekleşmektedir. Bu haberleşme türünde bileşenler arası haberleşme miktarı fazla olduğu zaman, ağın trafik yoğunluğundan dolayı kilitlenme problemi vardır. Bu problem, bileşenler arası yol ve yolların yapısıyla (topoloji) ve kullanılan yolun anlık veri gönderme hızıyla değişir. Bu nedenle YüAlarda farklı amaçlar için farklı topolojiler kullanılmaktadır. YüA tasarım sürecinin ilk adımı, maliyet, enerji, performans ve alan gibi optimizasyon parametreleri göz önünde bulundurularak, verilen uygulama için en uygun topolojiyi seçmektir. Eşleme, yönlendirme ve topoloji seçim mekanizmaları, bir uygulamanın trafik akış modelini bilerek, mesajların ağda ne kadar zaman harcadığını gösteren, paketlerin ağdaki atlama miktarlarını azaltmaya çalışır. YüA topolojileri, düzenli ve düzensiz olmak üzere iki gruba ayrılmaktadır.



Şekil 2.1. (a) Örgü, (b) torus, ve (c) halka topoloji örnekleri.

Düzenli topolojiler, mimarileri düzenli bir şekle dayanan yonga üstü ağ topolojileridir. Şekil 2.1.'de verilen örgü, torus ve halka topolojiler düzenli topolojilere örnek olarak gösterilebilir. Düzenli topolojilerin oluşturulması kolaydır ve farklı uygulamalar için yeniden kullanılabilme özelliğine sahiptirler. Bu tarz topolojilerde, tasarım basittir, kolay imal edilebilir. Tasarımdan dolayı hataların tespiti kolaydır ve arızalı bileşenler kolaylıkla tespit edilebilir. Fakat düzenli topolojiler uygulama için gerekli optimizasyon imkanını sağlamayabilirler. Ayrıca düzenli topolojilerde enerji tüketimi minimizasyonu da kısıtlıdır. Düzenli topolojilerde, özellikle örgü topolojide, bütün sistem bileşenleri arasında farklı yollar bulunmaktadır. Sistemde bir hata olması durumunda, paketlerin kaynak yönlendiriciden hedef yönlendiriciye iletimini alternatif yollar kullanarak sağlamak mümkündür. İlk yıllarda, birçok çalışma, 2 boyutlu düzlemde düzenlenebilen düzenli topolojiler üzerine eşleme ve topoloji oluşturma işlemleri üzerinde durmuşlardır [6–8, 27, 28]. [7] numaralı çalışmada yazarlar, örgü yapısı üzerine, uygulama düğümlerinin eşlenmesi ve yönlendirmelerinin belirlenmesi için dal ve sınır algoritması önermişlerdir. Ayrıca aynı çalışmada sistemin toplam enerji tüketimini hesaplamak için yeni bir enerji modeli de sunmuşlardır. [8] numaralı çalışmada, YüA'daki bant genişliği kısıtlamalarını karşılayan ve ortalama gecikmeyi en aza indiren hızlı bir örgü topoloji üzerine eşleme algoritması sunulmuştur. YüA üzerindeki trafiği bölerek, diğer çalışmalara kıyasla önemli oranda bant genişliği ve maliyet tasarrufu etmektedirler. [6] numaralı çalışmada yazarlar düşük enerjili ağ tasarımı için bir teknik sunmuşlardır. Bu teknik uygulama düğümlerini yönlendiricilere eşlemek için uygulamanın akış diyagramından ağaçlar oluşturan yinelemeli bir algoritmadır. Algoritma her aşamada uygulama diyagramını ve örgü yapısını, dallarında aynı sayıda düğüm bulunan ve dallarındaki kenarların kümülatif

ağırlıklarının en aza indirildiği iki ağaca böler. Algoritma tamamlandığında, yaprak düğümler yönlendiricilere eşlenecek uygulama düğümleridir. Daha sonra yönlendiriciler arasındaki yollar belirlenir.

Sonraki yıllarda, düzensiz topolojiler, enerji, alan ve performans gibi farklı parametreler için daha geniş bir optimizasyon alanına sahip olduklarından dolayı ilgi görmeye başlamıştır [1, 9, 10]. Düzensiz topolojiler uygulamaya özel olarak üretilir. [9] numaralı çalışmada yazarlar, uygulamaya özgü düzensiz topolojileri oluşturmak için sezgisel ve genetik algoritmaya dayalı yöntemler önermişlerdir. Sezgisel yöntem (TopGen), sistemin enerji tüketimini en aza indirmeyi hedefleyen, iki aşamalı bir uygulamaya özgü topoloji oluşturma algoritmasıdır. Bu yöntem önce düğümler arasında aktarılan veri miktarına göre uygulama düğümlerini kümelendirir. Ardından, bu kümeleri ağın iletişim maliyetini en aza indirmeyi hedefleyerek [10]'daki gibi yönlendiricilere atar. [9]'daki ikinci yöntem genetik algoritma tabanlı topoloji oluşturma algoritmasıdır (GATGA). Başlangıçta oluşturulan popülasyon üzerinde genetik operatörler uygulayarak, enerjiyi en aza indirgemek için farklı topolojiler üretmeyi amaçlamaktadır. [1] numaralı çalışmada yazarlar, iki aşamalı doğrusal programlama tabanlı bir yöntem önermektedirler. İlk adımda, verilen uygulama için uygun bir düzen belirlenir. Ardından yönlendiriciler ve düğümler arasındaki bağlantılar ikinci adımda atanır.

Bahsedilen düzensiz topolojilerde, amaç en az sayıda yönlendirici düğüm kullanarak enerji tüketimini mümkün olduğunca azaltmaktır. Bu özellikleri sağlamak için üretilen topolojilerde her bir sistem bileşeni çifti arasına yalnızca bir yol yerleştirilmektedir. Topolojiyi bu şekilde tasarlamanın dezavantajı, hata tolerans yeteneğinin olmamasıdır. Eğer herhangi bir link üzerinde fabrikasyondan kaynaklanan bir kalıcı hata varsa, tasarlanan devrenin çalışamaz hale gelmesine neden olabilir ve sonuçta imal edilen YüA tabanlı entegre kullanılamaz. Uygulamaya özgü topolojiyi bağlantı hatalarına karşı dayanıklı hale getirmenin bir yolu, tüm bağlantıları yedeklemek olabilir. Fakat bu durumda yönlendirici düğümlerdeki donanım da iki katına çıkacağından, YüA maliyeti artacak ve hatta donanımın artması sonucu enerji tüketimi de artacaktır. Hataya dayanıklı düzensiz topolojilerin üretilmesine yönelik çalışmalar literatürde çok nadirdir ([11–15]). [11]'de yazarlar, bağlantı hatası durumunda alternatif bir yol kullanan hataya dayanıklı düzensiz topolojiler üretmek için FTTG olarak adlandırılan bir

yöntem önermektedirler. Bu yöntemde yazarlar yinelemeli bir algoritma sunarlar. İlk olarak, algoritma, hataya dayanıklılığı dikkate almayan rastgele bir topoloji üretir. Her yinelemede, hataya dayanıklı hale getirmek için bu topolojiye fazladan bağlantılar eklenir. Sunulan algoritma, çeşitli yönlendirici sayısıyla topolojiler üretir ve bunları bir kitaplığa depolar. Bu, uygulama için en iyi topoloji üretme olasılığını artırır ve kullanıcının, optimizasyon parametreleri için en uygun topolojiyi seçmesine olanak tanır. Bununla birlikte, bu işlem zaman alıcıdır ve daha fazla düğüm sayısı olan uygulamalar için uygun değildir. [15]'te sunulan yöntem tamsayılı doğrusal programlamaya dayanan bir yöntemdir. Bu yöntemde, [11]'deki çalışmadan farklı olarak K tane hata tolere edilebilir. [14] numaralı çalışmada ise, yazarlar, hata kaldırabilir düzensiz topolojiler üretebilmek için açgözlü bir algoritma kullanmışlardır. [12] ve [13] numaralı çalışmalarda ise yazarlar biyolojiden esinlendikleri yöntemler ile topolojiler üretmektedirler. Buna ek olarak [13]'te klasik grafik-teorik algoritmalara ve optimizasyon yaklaşımlarına yönelik iki farklı yöntem sunmuşlar ve sonuçları karşılaştırmışlardır.

Topolojiye yeniden yapılandırma özelliği ekleyerek, alternatif yol bulma algoritmaları kullanarak veya sistemde bulunan diğer bileşenleri kullanarak bağlantı hataları tolere edebilir. Uygulamaya özgü ve hata toleransı olan yeni bir topoloji üretmek yerine, çalışmaların çoğu var olan hataları tespit etmek için algoritmalara ve hatanın tespit edilmesi durumunda ne yapılabileceği konusuna değinmektedirler. Bu yüzden çalışmalarını düzenli topolojiler üzerinde gerçekleştirmektedirler. Çünkü herhangi bir düğüm çifti arasında birden fazla yol sağladıkları için düzenli topolojiler hata kaldırabilir topolojilerdir. Bu topolojilerde herhangi bir bağlantıda bir hata olması durumunda, veriler alternatif güzergaha kolayca yönlendirilebilir. İlk çalışmaların bir kısmı [17–20], topoloji üzerinde hatalı bileşenler olduğunda yeni yönlendirme yolunu belirlemeye çalışmaktadır. Ancak bu tarz yöntemler her hatalı bileşen için yönlendirme tablosu alternatifi depoladıkları için fazladan belleğe ihtiyaç duyarlar. Buna ek olarak, alternatif yönlendirmenin varsayılan yönlendirmeden daha uzun yollara sahip olması nedeniyle performans kaybı yaşanır. [22, 23], çalışmalarında N-Modüler Yedekleme kavramını kullanmışlardır. Bu tarz sistemlerde kaynak yönlendirici hedef yönlendiriciye verinin N tane kopyasını gönderir. [23] 2-Modüler bir sistem önermişken, [22] verinin üç

kopyasını göndermektedir. İki kopya sadece hata olup olmadığını tespit etmek için kullanılmaktadır. Üç kopyada ise aynı olan iki kopyanın doğru olduğu varsayılır ve bir sonraki hedef yönlendiriciye bu kopya gönderilerek sistemin çalışmaya devam etmesi sağlanır. Çalışmaların bazıları, çalışma zamanında topolojinin yeniden yapılandırılmasına odaklanmaktadır ([2, 29–34]). Örneğin, [29], arızaları tolere edebilmek için kullanılmayan ağ bileşenlerini dinamik olarak etkinleştirir. Her yönlendirici üzerinde bulunan, kendi kendini sınama (BIST) birimi arızaları bulur ve arıza türüne göre ne yapılması gerektiğine karar verir.

Bazı çalışmalar ise hata toleranslı topolojiler üretmek yerine, uygulama düğümlerini yeniden yapılandırılabilir yeni bir örgü yapısı üzerine eşlerler ([2, 3, 16]). Bu yeni ağ yapısında, her düğüm arasına fazladan anahtarlar yerleştirilir. Bu çoklayıcı anahtar yapısı sayesinde, bir düğümün diğer düğümlere olan bağlantısını dinamik olarak belirlemek mümkündür. Bir diğer deyişle bu yapıda, o anda çalıştırılan uygulamanın iletişim modeline göre, o uygulamaya özgü ağ topolojisi değiştirilebilmektedir. Bir mimarideki toplam güç tüketimine yönlendiricilerin fazlasıyla etkisi olduğu düşünüldüğünde, sistemin, paket iletiminde yönlendiriciler yerine bu anahtarları kullanması toplam güç tüketimini oldukça azaltmaktadır. [2]'de yazarlar uygulama düğümlerini aralarındaki trafik miktarına bağlı olarak haritalarlar. Maksimum trafik miktarına sahip olan düğüm, örgü yapısı üzerinde en fazla komşu düğüme sahip olan yere yerleştirilir. Daha sonra, algoritma tüm düğümler yapıya yerleştirilene kadar devam eder. [3] numaralı çalışmada, uygulama düğümlerinin yerleştirilmesi için sistematik partikül filtrelemesini kullanan bir algoritma önerilmiştir. Sunulan algoritma, rastgele üretilen topolojileri başlangıç parçacıkları olarak kabul eder ve iyi parçacıkların sayısını arttırmak için yeniden örneklemeye devam eder. [16] numaralı çalışmada yazarlar, düğümleri eşlemek ve aralarındaki bağlantıları belirlemek için tamsayılı doğrusal programlamaya ve parçacık sürü optimizasyonuna dayalı iki farklı yaklaşım önermektedir.

Bu konuda düzensiz topolojiler üzerinde gerçekleştirilen çalışma sayısı oldukça düşüktür. [24] numaralı çalışmada iletişim kuran her bir yönlendirici çifti arasında en az iki alternatif yol bulunan düzensiz topoloji tabanlı YüA için, çeşitli alternatif yönlendirme seçeneklerini belirleyen, hata-kaldırabilir yönlendirme oluşturma algoritması sunulmuştur. Algoritmanın

amacı, enerji tüketimini en aza indiren rota alternatiflerini belirlemektir. Algoritma, en az sayıda alternatif rotayı belirler ve yönlendiriciye yönlendirme tablosu olarak ekler. Bir bağlantıda hata olması durumunda ilgili yönlendirme tablosu yonganın dış bacakları aracılığıyla güncellenebilir. Hatalı bağlantı tespit edildikten sonra, yeni yola karar verilmeli ve dışardan yonganın bacak ayarı buna göre yapılmalıdır. Bu yöntemin en önemli problemi veya kısıtı, kullanılabilme alanının az olmasıdır. Bu yöntemde sadece bir bağlantı hatası olan YüA tasarımları hedeflenmiştir. Ayrıca, yonga üstündeki link hatalarının tespiti, imalattan sonra yapılmakta ve eğer bir link hatası var ise hatayı tolere edebilecek yönlendirme tablosu aktif hale getirilmektedir. Yani, yonga üstündeki hatanın, devrenin çalışma ömrü boyunca olması durumunda, dinamik olarak hataları tespit edecek ve yeni rotayı kendi kendine güncelleyecek bir mekanizma bulunmamaktadır. Bir başka problem ise, yongadaki yönlendirici tabloları aktif hale getirmek için yongaya dış pinlerin eklenmesi zorunluluğudur.

3. UYGULAMAYA ÖZGÜ HATA KALDIRABİLİR TOPOLOJİ TASARIMI

3.1. Uygulamaya Özgü Hata Kaldırabilir Topoloji Tasarımı

YüA, yonga üzerindeki işleme elemanlarının sayısı çok fazla olduğunda iletişimi kolaylaştıran bir yaklaşımdır. Belirli bir uygulama için YüA mimarisi tasarlarken performans, maliyet, yonga alanı, hata toleransı ve enerji tüketimi gibi çeşitli kriterler dikkate alınmalıdır. Topoloji türü, nihai tasarımda yukarıda belirtilen kriterlerin gereklerini yerine getirmek için önemli bir rol oynamaktadır. Örneğin, performans, enerji ve maliyet için büyük optimizasyon alanı nedeniyle düzensiz bir topoloji tercih edilebilirken, farklı uygulamalar için ölçeklenebilirliği, hata toleransı ve tekrar kullanılabilirliği için düzenli bir topoloji tercih edilebilir.

Uygulamaya özgü topoloji oluşturmada önceki yöntemlerin çoğu çok uzun yürütme süreleri gerektirir ve genellikle çok sayıda düğümü olan uygulamalar için kullanılabilir değildir. Artan uygulama düğümü sayısına bağlı olarak katlanarak artan karmaşıklık göz önünde bulundurulduğunda bu problemi makul bir sürede çözebilecek algoritmalara ihtiyaç duyulmaktadır.

Tezin bu kısmında, bu probleme getirilen çözüm hakkında bilgiler bulunmaktadır. Öncelikle bazı temel bilgiler ve problem tanımı verilmiştir. Daha sonra sunulan yöntem, ayrıntıları ile açıklanmıştır. Son olarak da yöntemin performansını değerlendirmek için yapılan deneyler anlatılmış ve sonuçlar değerlendirilmiştir.

3.1.1. Problem Tanımı

Düzensiz topoloji tasarımı için önceki çalışmalar, genellikle, performans ve bant genişliği kısıtlamaları altında enerji tüketimini en aza indirmeye odaklanmıştır. Bu çalışmada amacımız, uygulama akış çizgesi (UAÇ) verilen uygulamaya uygun, hataya dayanıklı özel bir topoloji oluşturmaktır. Bir UAÇ, $U(D, B)$ ile temsil edilir; burada D, uygulamanın düğümlerini gösterir ve B, bu düğümler arasındaki bağlantıların kümesidir. Her $b_{i,j} \in B$, $d_i \in D$ düğümü ile $d_j \in D$ düğümünün bağımlılığını gösterir. $t_{i,j}$, $b_{i,j}$, bağlantısı üzerinden d_i ve d_j düğümleri arasındaki veri aktarım miktarını belirtir. Bir örnek UAÇ, Şekil 3.1.'de verilmiştir. Hedefimiz, UAÇ'nin uygulama düğümlerini, oluşturulan topoloji ile eşleştirmek ve iletişimin enerji tüketimini en aza indirgemek olacaktır. Bu tanımlar ışığında problem aşağıdaki gibi tanımlanabilir.

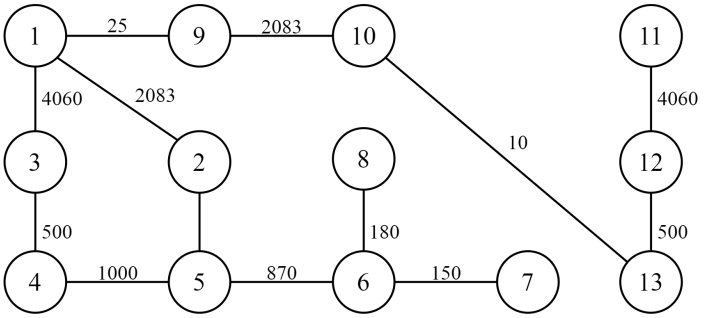
Oluşturulacak topoloji, aşağıdaki ölçütleri karşılayan p bağlantı noktasına sahip y yönlendiriciden oluşacaktır.

1. Her yönlendirici y_i, en az iki bağlantı ile ağa bağlı olmalıdır. Bu koşul, oluşturulan topoloji için hataya dayanıklılık sağlar. Bir bağlantıda hata olması durumunda varsayılan yol yerine alternatif yol kullanılabilir.

2. Her uygulama düğümü d_i bir yönlendiriciye atanmış olmalıdır.

3. Topoloji tamamen bağlı olmalıdır. Yani herhangi iki yönlendirici arasında en az bir yol bulunmalıdır.

4. Oluşturulan topolojideki iletişim maliyeti en aza indirilmelidir. İletişim maliyetini hesaplamak için düğümler arasındaki minimum mesafeler kullanılır. Dijkstra algoritması, bu minimum mesafeleri, diğer bir deyişle, yönlendiriciler arasındaki asgari atlama mesafelerini bulmak için kullanılır. Topoloji üretiminin amaç fonksiyonu, sistem bileşenleri arasındaki veri akışı miktarını en aza indirgemeyi hedeflemektedir. Amaç fonksiyonu Denklem 1'deki gibi yazılabilir. Burada n, uygulama düğümlerinin sayısını



Şekil 3.1. MP3 Encoder için uygulama akış çizgesi [1].

ve $m_{min}(i, j)$, d_i ve d_j düğümleri arasındaki en kısa mesafeyi yani aralarındaki yönlendirici sayısını göstermektedir.

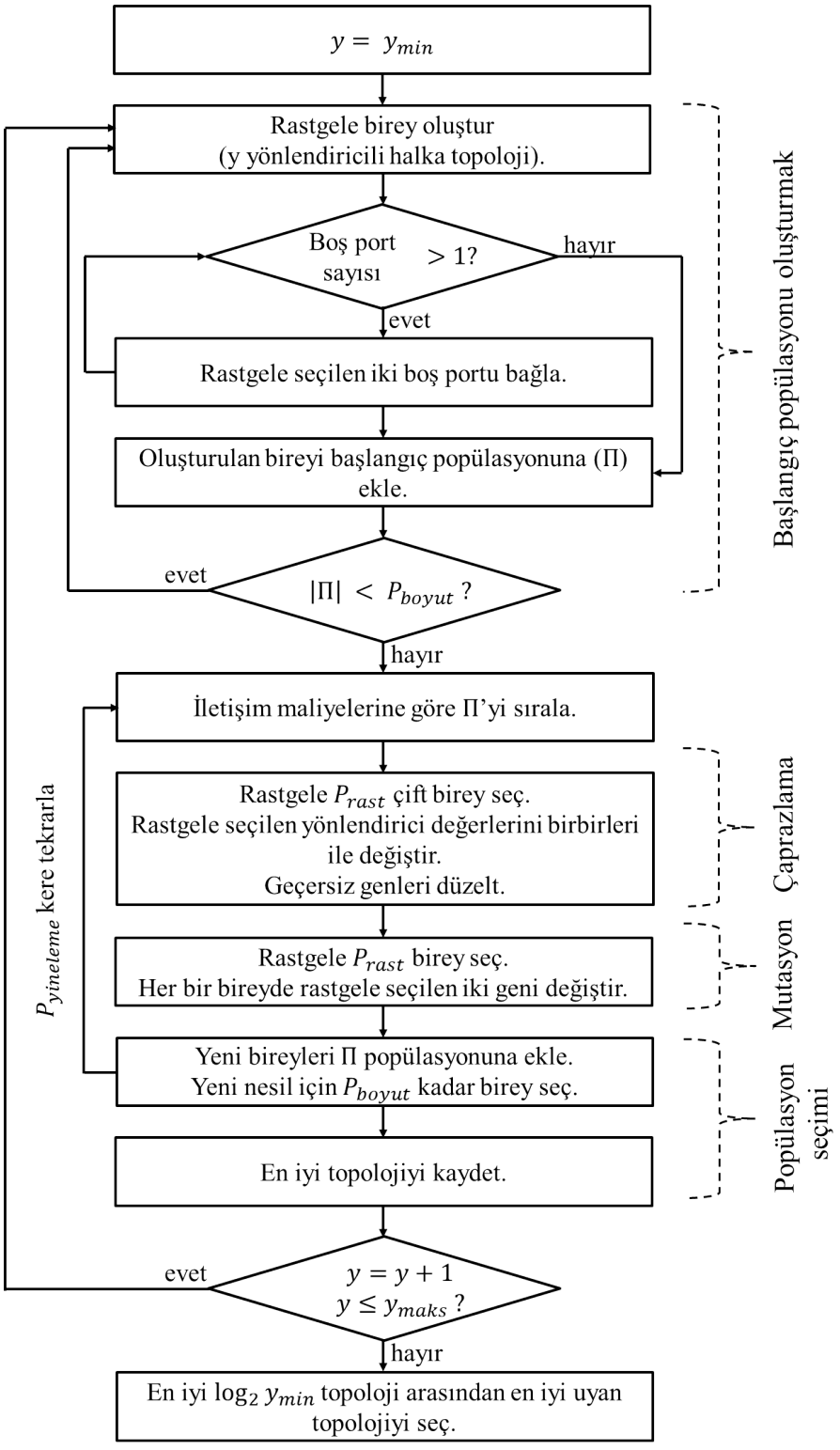
$$\text{min:} \sum_{i=1}^{n} \sum_{j=i+1}^{n} t_{i,j} \times m_{min}(i, j), \tag{1}$$

3.1.2. Sunulan Yaklaşım

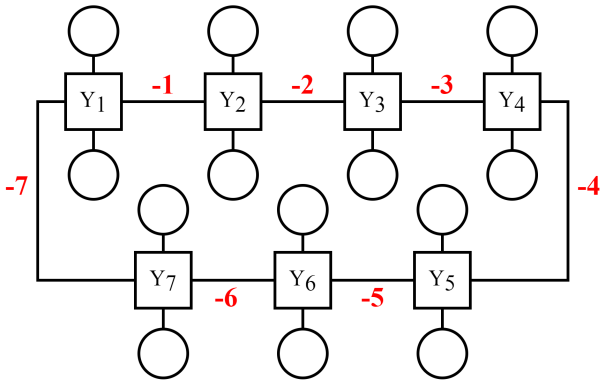
Uygulamaya özgü hata kaldırabilir topoloji oluşturmak için, belirtilen problemi makul bir zamanda çözmeyi hedefleyerek genetik algoritma tabanlı bir yaklaşım sunulmuştur. Bu tip algoritmalar, genellikle, arama alanında rastgele hareketler yaparak, en iyi çözüme ulaşmayı hedeflerler. Genetik algoritma, optimizasyon problemlerini çözmek için bilgisayar ortamındaki evrimsel süreçleri taklit eder. Her adımda, algoritma, mevcut popülasyondaki bireylerden rastgele çiftleri seçer ve bunları bir sonraki nesil oluşturmak için kullanır. Daha iyi bireylerle devam etmek, yeni nesilleri en uygun çözüme götürür. Tezin bu kısmında GA-FTCT olarak adlandırdığımız genetik algoritma tabanlı hata kaldırabilir topoloji üretme metodu anlatılmıştır.

GA-FTCT algoritmasına genel bakış

Şekil 3.2.'de verilen algoritma akış şemasında gösterildiği gibi, algoritma üç temel aşamadan oluşmaktadır: ilk popülasyonu üretmek, genetik operatörleri uygulamak (çaprazlama ve mutasyon), ve sonraki iterasyon için yeni popülasyonu seçmek. İlk popülasyonu oluşturma



Şekil 3.2. Sunulan yaklaşımın akış şeması.



(a)

Yönlendirici1				Yönlendirici2				Yönlendirici3				Yönlendirici4				Yönlendirici5				Yönlendirici6				Yönlendirici7			
0	0	-7	-1	0	0	-1	-2	0	0	-2	-3	0	0	-3	-4	0	0	-4	-5	0	0	-5	-6	0	0	-6	-7

(b)

Şekil 3.3. (a) Uygulama düğümleri atanmadan sadece bağlantıların belirlendiği (negatif değerler) topoloji örneği ve (b) kromozom temsili.

aşamasında, rastgele halka tipi topolojiler üretilmektedir. Ardından, bu kümedeki en düşük iletişim maliyeti olan bireylerden rastgele seçilenler üzerinde çaprazlama işlemi gerçekleştirilmektedir. Bu işlem ile topoloji çeşitliliğini arttırmak amaçlanmıştır. Bir sonraki genetik operatör mutasyon, yine rastgele seçilen bireyler üzerinde uygulanır ve bu işlem ile daha iyi eşleme sonuçlarına ulaşmak hedeflenmektedir. Son olarak, eski nesil ve yeni oluşan bireyler birleştirilerek bunlar arasından en düşük maliyete sahip olanlar ile iterasyonlara devam edilir. Bu adımlar, önceden tanımlanmış bir sabit tekrarlama sayısı olan $P_{yineleme}$ kez tekrarlanmıştır. Akış şemasının dış döngüsünde görüldüğü gibi bütün bu aşamalar farklı yönlendirici sayıları ile (y_{min} - y_{maks} $=$ y_{min} $+$ $\lceil \log_2 y_{min} \rceil$) denenmiştir. y_{min} yönlendirici sayılı topolojiler halka topoloji iken, yönlendirici sayısı artırıldıkça, yönlendiriciler arasındaki ortalama atlama mesafesini azaltmak için topolojiye yeni bağlantılar eklemek mümkün olmaktadır. Algoritma son olarak en iyi topolojiyi döndürmektedir. Aşağıda bu adımlar daha detaylı bir şekilde anlatılmıştır.

Kromozom temsili

Oluşturulan geçerli topolojiler, kromozomlar tarafından temsil edilen bireyler olarak kabul edilirler [9]. Kromozomlar yönlendirici bağlantı noktalarını temsil eden genlerden oluşur.

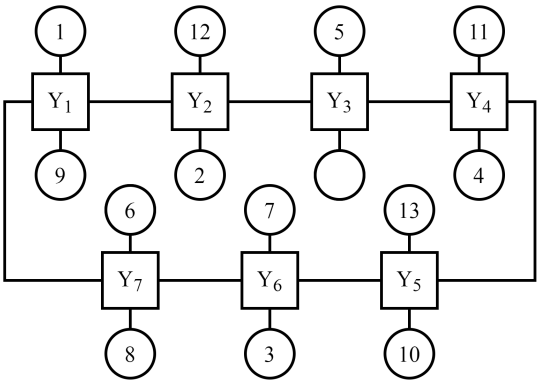
Kromozom yaratmada ilk görev, sistem tarafından kullanılan yönlendirici sayısını belirlemektir. Yönlendirici bağlantı noktalarının boyutu ile birlikte yönlendirici sayısı, kromozomun uzunluğunu belirtir. Sunulan yöntem en az sayıda yönlendirici kullanarak oluşturulan halka topoloji ile işlemlere başlamaktadır. Halka topolojisinde her bir yönlendiriciyi komşu yönlendiricilere bağlamak için yönlendiricilerin iki portu kullanılacaktır. Homojen yönlendiriciler kullanıldığı için, her bir yönlendiricide $p-2$ boş port uygulama düğümlerine ayrılmıştır. Yönlendiricileri birbirine bağladıktan sonra, n uygulama düğümlerini eşlemek için en az n boş porta sahip olunmalıdır. Topolojide y_{min} kadar yönlendirici olduğunda, boş portların sayısı Denklem 2'de verilen n değerinden daha büyük veya eşit olmalıdır. Dolayısıyla, halka topolojisi için minimum yönlendirici sayısı Denklem 3 ile bulunabilir.

$$y_{min}.(p-2) \geq n \tag{2}$$

$$y = \left\lceil \frac{n}{p-2} \right\rceil \tag{3}$$

Şekil 3.4.a'da, bir halka topolojisi örneği ve üzerinde MP3 Encoder eşleştirmesi gösterilmiştir. Ayrıca, topolojiyi ve haritayı temsil eden kromozom yapısı Şekil 3.4.b'de görülmektedir. Kromozomlar aşağıdaki kurallar göz önünde bulundurularak oluşturulmaktadır.

- Bir kromozom tam $p \times y$ gen içerir.
- Bir yönlendiricinin y_i port numaraları, kromozomdaki $((i-1) \times p) + 1$ ve $i \times p$ konumlarındaki genler arasında bulunur.
- Pozitif sayılar uygulama düğümlerini gösterir.
- Yönlendiriciler arasındaki bağlantılar negatif sayılarla ifade edilir. Şekil 3.4.b'de görüldüğü gibi, negatif bir sayı, belirli bir bağlantıyı temsil eder ve tam olarak iki farklı yönlendirici bağlantı noktasında bulunur. İki yönlendirici bağlantı noktasındaki aynı negatif değer, bu iki yönlendirici arasında bağlantı olduğu, yani; direkt bağlı oldukları anlamına gelir.



(a)

Yönlendirici1				Yönlendirici2				Yönlendirici3				Yönlendirici4				Yönlendirici5				Yönlendirici6				Yönlendirici7			
1	9	-7	-1	12	2	-1	-2	5	0	-2	-3	11	4	-3	-4	13	10	-4	-5	7	3	-5	-6	6	8	-6	-7

(b)

Şekil 3.4. (a) Şekil. 3.1.'de verilen uygulama akış çizgesinin eşlenmesi ile oluşan bir topoloji örneği ve (b) kromozom gösterimi.

- Negatif bir sayı tam olarak iki bağlantı noktasına yazılmalıdır. Bu numaralar, kromozom üzerindeki farklı yönlendirici bölümlerinde olmalıdır.
- Her bir yönlendiricinin y_i değerinin son iki portu ($(i \times p) - 1$ ve $i \times p$) negatif link değerleri için ayrılmıştır. Bu iki port sayesinde, her yönlendirici, topolojinin geri kalanı ile en az iki port üzerinden bağlıdır ve bu da topolojiyi hataya dayanıklı hale getirir.
- Boş bağlantı noktaları sıfırlarla gösterilir.

Başlangıç popülasyonu oluşturmak

Popülasyondaki her kromozom, eşlenmiş uygulama düğümlerine sahip geçerli bir topolojiyi temsil eder. Her kromozom, $y \times p$ genlerinden oluşan bir dizi olarak oluşturulmaktadır. Daha sonra, her bir yönlendiricinin son iki genine, bağlantı yapısı için ayrılmış bağlantı belirleyici değerler atanır. Bu prosedür için bir önceki bölümde verilen kurallar göz önünde bulundurulmaktadır.

İlk kuşakta, bağlantı yapısı, Şekil 3.3.'te görüldüğü gibi tüm bireyler için aynıdır. Minimum sayıda yönlendirici, y_{min}, topolojide kullanılan yönlendiriciler sayısına (y) eşit olduğunda,

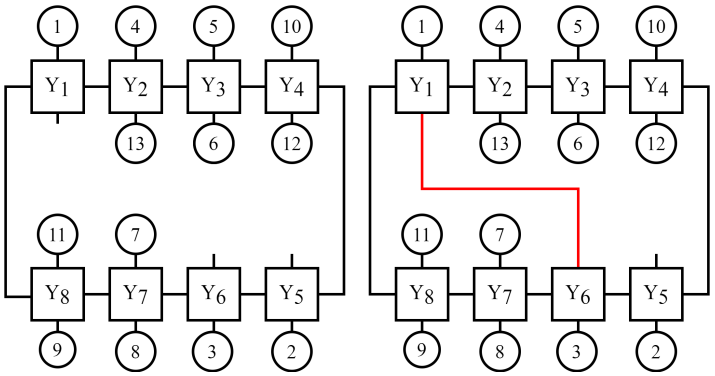
(yani, $y_{min} = y$), yalnızca halka topolojisi oluşabilir. Bununla birlikte, tasarımdaki yönlendirici sayısı arttığında, topolojiye daha fazla bağ eklenebilir. Oluşturulan topolojiye daha fazla bağ eklemenin arkasındaki mantık, paketlerin ortalama atlama sayısını azaltmak için yönlendiriciler arasındaki mesafeyi azaltmaktır. Bu, performansı artıracak ve paketlerin iletimi sonucunda tüketilen enerjiyi azaltacaktır.

Şekil 3.4.a'da gösterildiği gibi, y_{min} yönlendiricileri ile oluşturulan halka topolojisinde sadece 7 yönlendirici vardır. Topolojide kalan yalnızca boş bir bağlantı noktası ($Boş = 1$) var, bu da ek bir bağlantı eklemek için yeterli değildir. Topolojideki boş portların sayısı aşağıdaki gibi hesaplanabilir.

$$Boş = py - n - 2l \tag{4}$$

Burada py toplam kullanılabilir bağlantı noktası, n, uygulama düğümlerinin sayısı ve l geçerli topoloji yapısında kullanılan bağlantıların toplamıdır. Yönlendirici sayısını artırdığımızda, daha fazla boş porta sahip olabiliriz. Yeni bir bağlantı oluşturabilmek için iki port birbiri ile bağlanacağından, topolojiye en fazla $\lfloor Boş/2 \rfloor$ yeni bağlantı eklenebilir. Şekil 3.5.'te, 8 yönlendiricili bir halka topolojiye fazladan bir bağlantı ekleme örneği gösterilmektedir. Bu ekstra bağlantıyı ekleyerek, örnek uygulama için iletişim maliyeti 52.419'dan 39.163 bit/s'ye düşürülür. Bu örnekte y_1 ve y_6 yönlendiricileri boş portlar üzerinden birbirine bağlanmıştır. Ancak, farklı topoloji yapısı oluşturacak başka olası bağlantılar da bulunmaktadır. Bu küçük örnek için, yönlendiricinin kendisinin ve iki komşu yönlendiricisinin bağlantıları göz ardı edildiğinde, eklenebilecek $(y-3)\times(y-2)/2$ olası bağlantı bulunmaktadır. Her bağlantı eklemesinde, bağlantı kombinasyonu y^2 ile orantılı olarak artar. Sonuç olarak, olası bağlantıların sayısı, l_e ekstra bağlantı için y^{2l_e} ile orantılıdır. Tüm bu katlanarak artan sayıda topolojiyi makul bir zamanda denemek mümkün olmadığından, Şekil 3.2.'de gösterildiği gibi başlangıçta rastgele P_{boyut} topoloji üretilmiştir.

Yukarıda belirttildiği gibi, ilk topoloji y_{min} yönlendiricili halka topolojisidir. Daha sonra topolojilere daha fazla bağlantı ekleyebilmek için yönlendirici sayısı birer birer artırılmıştır.



Şekil 3.5. Yönlendiriciler arasındaki uzaklığı azaltmak için fazladan bağlantı ekleme örneği.

Topoloji için kullanılan maksimum yönlendirici sayısı [11] numaralı çalışma ile adil bir karşılaştırma yapmak adına aşağıdaki gibi hesaplanmaktadır:

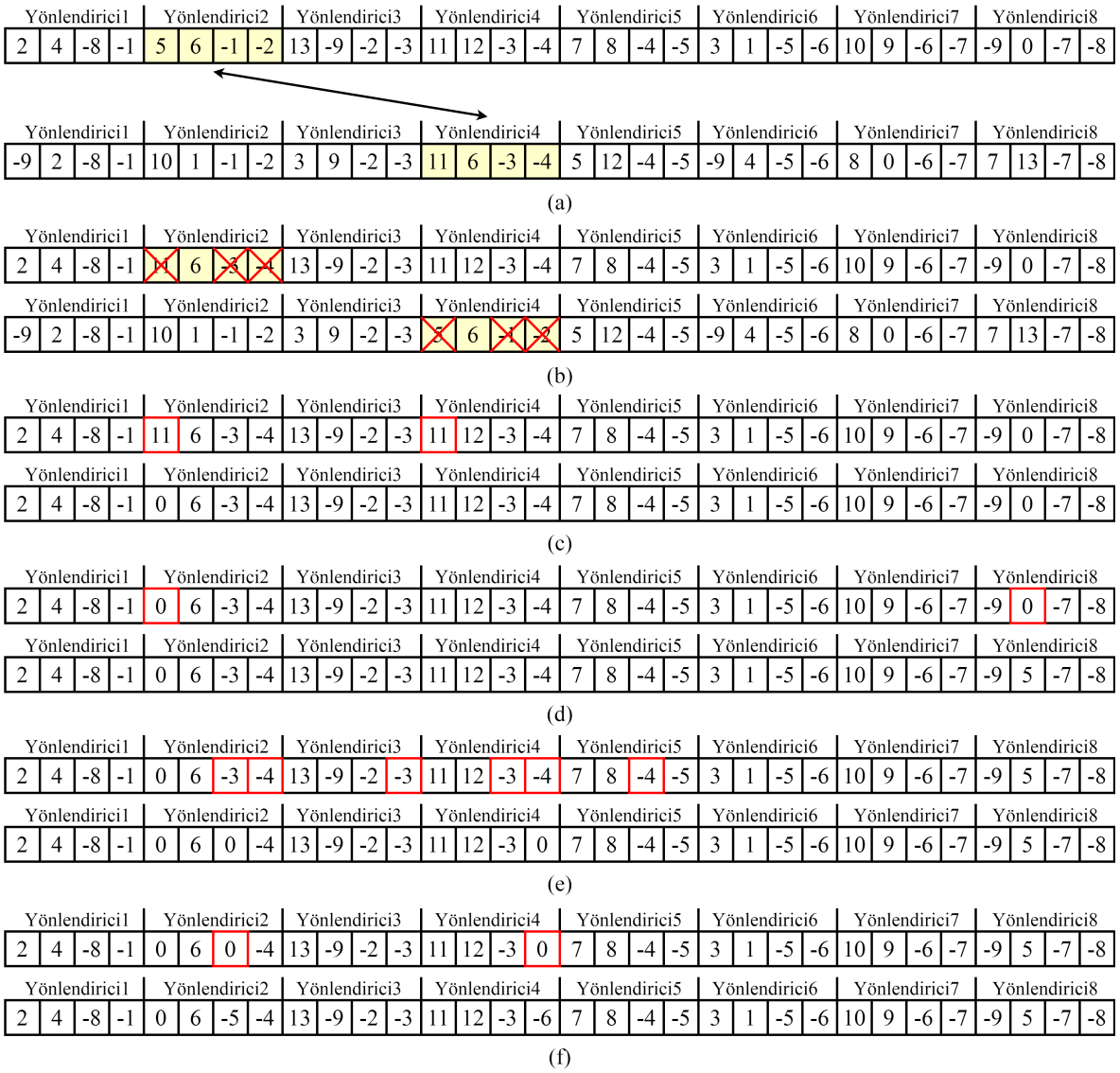
$$y_{maks} = y_{min} + \log_2 y_{min} \tag{5}$$

Her yönlendirici sayısı için genetik operatörler uygulanarak en iyi eşlemeler elde edilir. Böylece, $\log_2 y_{min}$ çözüm ile algoritma sonuçlanır. Nihai çözüm olarak bu eşleme sonuçları arasından en iyisi seçilmektedir.

Çaprazlama

İlk kromozomlar üretildikten sonra, GA-FTCT, seçilen bireylerde çaprazlama ve mutasyon gibi genetik operatörler kullanarak popülasyonu çeşitlendirir. Algoritma, bu operatörleri uygulamak için, önce kromozomları iletişim maliyetlerine göre artan düzende sıralamaktadır. Ardından, bu ardışık listenin ilk P_{min} öğelerinden seçilen bireylere genetik operatörleri uygular.

Çaprazlama gerçekleştirmek için, GA-FTCT, rastgele iki kromozomu ve bunların her birinden rastgele bir yönlendiriciyi seçer. Ardından, seçilen yönlendiricilerin değerlerini birbiri ile değiştirir. Bu işlem sonucunda iki yavru kromozom oluşmaktadır. Şekil 3.6.'da MP3 Encoder uygulaması üzerinde çaprazlama operatörünün uygulama adımları gösterilmiştir.



Şekil 3.6. Çaprazlama işlemi adımları. (a) Ebeveyn bireyler ve değişim işlemi. (b)-(f) Geçersiz kromozomların onarılması.

Şekil 3.6.a'da gösterildiği üzere algoritma ilk kromozomdan ikinci yönlendiriciyi, ikinci kromozomdan ise dördüncü yönlendiriciyi değiştirmek için rastgele seçmiştir. Bu işlem sonucunda oluşan yeni nesil bireyler analiz edildiğinde geçersiz genlerin olduğu görülmektedir (Şekil 3.6.b). Örneğin, düğüm 11, ilk bireyin hem ikinci hem de dördüncü yönlendiricilerinde bulunmaktadır. Bu geçersiz genlerin onarılması gerekmektedir. Aşağıdaki paragraflarda, bir kromozomu geçersiz hale getiren sorunların listesi ve onarmak için gerekli olan prosedürler sıralanmıştır.

- **Problem 1:** Uygulama düğümleri yalnızca bir yönlendiriciye eşlenebilir. Kromozom içinde aynı uygulama düğümüne sahip iki gen varsa, bu düğümün farklı yönlendiricilere bağlı olduğunu gösterir. Bu nedenle, oluşturulan birey geçersiz olur.
  Çözüm: Rastgele seçilen genlerden birine sıfır atanır. Şekil 3.6.c'de görüldüğü gibi 11 numaralı uygulama düğümü, çaprazlama operatöründen sonra iki farklı gende görülmektedir. Bu geçersiz durumu düzeltmek için rastgele seçilen ikinci yönlendiricinin genine 0 atanır.

- **Problem 2:** Uygulama düğümlerinden bir veya daha fazlası kromozomda bulunamayabilir. Bu, eksik olan düğümün yönlendiricilere atanmadığı anlamına gelir. Dolayısıyla, eşleme tamamlanmadığından, oluşturulan birey geçersiz sayılacaktır.
  Çözüm: Kromozomda olmayan uygulama düğümü, rastgele seçilen boş genlerden (yani, sıfır değerli genler) birine atanır. Şekil 3.6.d'de, uygulama düğümü 5 başlangıçta kromozomda eksiktir. Algoritma, kromozomun iki boş geninden biri olan sekizinci yönlendiricinin ikinci portuna 5 değerini atar.

- **Problem 3:** Geçerli bir topolojide, bağlantıları temsil eden negatif değerler, iki farklı yönlendiricide tam olarak iki gende bulunmalıdır. Kromozom dizesinde ikiden fazla aynı negatif değer varsa, oluşturulan birey geçersiz sayılacaktır.
  Çözüm: Rastgele seçilen genlerden birine sıfır atanır. Şekil 3.6.e'de gösterildiği gibi, -3 ve -4 numaralı bağlantılar, kromozomda üç kez atanmıştır. Oluşan kromozomun geçerli olması için rastgele seçilen bir tane -3 içeren ve bir tane de -4 içeren gen 0 olarak atanır.

- **Problem 4:** Linkleri gösteren negatif değerler tam olarak iki farklı yönlendiricinin geninde bulunmalıdır. Bir bağlantı değeri kromozomda yalnızca bir kez görünürse, bu birey geçersizdir.
  Çözüm: Kromozomda yalnızca bir gende bulunan bağlantı belirteci, kromozomun

Şekil 3.7. Mutasyon işlemine bir örnek.

bağlantı bölümündeki rastgele seçilen boş genlerden birine atanır. Şekil 3.6.f'de, -5 ve -6 bağlantıları, dizgenin son durumunda sadece bir gende görülmektedir. Bu sorunu çözmek için, yönlendiricilerin bağlantı kısımlarındaki rastgele seçilen boş genlere -5 ve -6 numaraları atanmaktadır.

GA-FTCT, eski kuşağın en düşük maliyetli ilk P_{min} bireyinden rastgele seçilen P_{rast} çiftinde çaprazlama işlemlerini tekrarlar. Sonunda bu iki kuşak yeni bir listeye birleştirilir. Bu operatör, topolojileri ve eşlemeleri çeşitlendirmek için çok etkilidir. Bu nedenle, uygulama eşlemesi için en iyi topolojiyi belirleme olasılığını artırır.

Mutasyon

Çaprazlama operatöründen sonra, GA-FTCT rastgele seçtiği P_{rast} birey üzerinde mutasyon operatörü uygular. Seçilen her bireyden rastgele iki pozitif veya bir pozitif bir sıfır gen değeri seçer. Sonra, onları değiştirir. Mutasyon operatöründe, yalnızca uygulama düğüm değerleri ve sıfır değerleri değiştirilir. Bağlantı belirteçleri aynı kalır. Başka bir deyişle, mutasyon operatörü bireyin topolojisini değiştirmez. Yalnızca eşleme sonucunu iyileştirmeye çalışır.

Şekil 3.7.'de örnek bir gen mutasyonu gösterilmiştir. Bu örnekte, düğüm değerleri 13 ve 1 rastgele seçilir ve takas edilir. Sonuç olarak, yeni bir birey oluşur. Tüm mutasyona uğramış bireyler yeni nesil listesine eklenir.

Çizelge 3.1. Deneylerde kullanılan denektaşları ve rastgele oluşturulan çizgelerin özellikleri.

Akış Çizgesi	Adı	Düğüm sayısı	Kenar sayısı
MWD [10]	G1	12	12
263 Encoder [1]	G2	12	12
MP3 Encoder [1]	G3	13	13
VOPD [10]	G4	16	20
DVOPD [35]	G5	32	41
Rastgele Çizge 1 [11]	G6	30	33
Rastgele Çizge 2 [11]	G7	40	43
Rastgele Çizge 3 [11]	G8	50	53
Rastgele Çizge 4	G9	60	78
Rastgele Çizge 5	G10	70	101

Popülasyon seçimi

Çaprazlama ve mutasyon işlemleri sonrasında nüfusun büyüklüğü artar. Hesaplama süresini azaltmak için, nüfus büyüklüğü Şekil 3.2.'de de gösterildiği gibi sabit bir değer (P_{boyut}) ile sınırlandırılmaktadır. Bu amaçla, Elitist seçimi adlı tanınmış ve etkili seçme mekanizmasını kullanılmaktadır. Bu yöntemde, bireyler toplam iletişim maliyetine göre artan düzende sıralanır. Ardından, bir sonraki nesil için bu listeden en iyi P_{boyut} birey seçilir.

3.1.3. Deneysel Sonuçlar

Bu bölümde, önerilen yöntemin performansını değerlendirmek için yapılan deneyler açıklanmıştır. İlk olarak kurulum ayarlarından bahsedilmiştir. Daha sonra elde edilen sonuçlar üç farklı topoloji oluşturma yöntemiyle karşılaştırılmıştır.

Sistem ayarları

Bu bölümde, deneysel kurulum yapılandırmaları ve performans değerlendirmesinde kullanılan denektaşları açıklanmaktadır. Denemelerimizde hem gerçek multimedya denektaşları hem de rastgele oluşturulmuş veri akış çizgeleri kullanılmıştır. Bu çizgeler ile ilgili ayrıntılar Çizelge 3.1.'de gösterilmiştir.

Deneylerde, genetik algoritma ve diğer mimari parametreler için aşağıdaki değerler uygulanmıştır:

- Varsayılan GA-FTCT parametreleri Çizelge 3.2.'de gösterilmiştir. İlk nüfusun boyutu P_{boyut} 1000'dir. Evrim süreci, en düşük iletişim maliyeti olan ilk 200 kişi üzerinde gerçekleştirilir. Algoritma bu 200 bireyden rastgele 50 çift seçer ve çaprazlama işlemini gerçekleştirir. Ayrıca aynı kümeden rastgele seçilen 50 birey üzerinde mutasyon operatörü uygulanır. Yeni popülasyonlar üretmek için yineleme sayısı 1000 olarak seçilmiştir. Bu parametrelerin bazılarını artırmak veya azaltmak, işlemci zamanı ve doğruluğunu dengelemektedir. Yapılan bazı deneyler göz önünde bulundurularak parametrelere karar verilmiştir. Bir diğer neden de adil bir karşılaştırma gerçekleştirebilmek için GATGA [9]'da kullanılan aynı GA parametreler tercih edilmiştir.

Çizelge 3.2. Genetik algoritma parametreleri.

Parametreler		Değer
Başlangıç popülasyon büyüklüğü	P_{boyut}	1000
Operatörlerin uygulanacağı popülasyon büyüklüğü	P_{min}	200
İterasyon sayısı	$P_{yineleme}$	1000
Seçim metoodu	-	Elitist
Çaprazlama olasılığı	-	0,1
Mutasyon olasılığı	-	0,05

- Aksi belirtilmediği sürece 4 portlu yönlendiriciler kullanılmıştır.

- GA-FTCT algoritması, yonga üstünde kullanılabilecek maksimum yönlendirici sayısını Denklem 6'teki gibi sınırlar. FTTG [11] çalışması da bu sınırı kullanmaktadır.

$$y_{maks} = y_{min} + log_2(y_{min}) \tag{6}$$

- [36] numaralı çalışmada 45-nm teknolojisi için verilen güç modeli benimsenmiştir. Bir yönlendirici için tahmini enerji tüketimi değerleri 3,20 pJ/Kb iken, bağlantılar için 4,78 pJ/Kb/mm'dir. FTTG'de olduğu gibi bağlantıların uzunluğunun, sabit 1 mm olduğu varsayılmıştır.

- Sunulan algoritma karşılaştırıldığı diğer yöntemler ile aynı ortamda çalıştırılmıştır. Deneylerde Windows 7 üzerinde çalışan Intel Core i7-4510U 2.00GHz 64 bit CPU, 8 GB bellekli bir dizüstü bilgisayar kullanılmıştır.

Bu deneylerde, sunulan algoritma ile üretilen topolojiler FTTG yöntemi ile üretilen topolojiler ve halka tipi topolojiler ile karşılaştırılmıştır. Bu iki farklı yöntem ile elde edilen topolojiler hata kaldırabilir topolojilerdir. Bu yöntemlerle beraber temel topolojilere hata toleransı ekleme maliyetini göstermek için hata kaldırabilirlik özelliğini dikkate almayan genetik algoritma tabanlı GATGA yöntemi ile elde edilen sonuçlar da değerlendirilmiştir. Karşılaştırmalar enerji tüketimi, donanım maliyeti (yönlendirici sayısı), algoritmaların çalışma süreleri ve gecikme ölçütü olarak ortalama atlama sayısı göz önünde bulundurularak gerçekleştirilmiştir.

Energy karşılaştırması

Çizelge 3.3. Yöntemlerin enerji karşılaştırmaları.

	GATGA		Halka		FTTG		GA-FTCT		GA-FTCT'nin enerji gelişimi (%)		
Çizge	Enerji (mJ)	r#	Enerji (mJ)	r#	Enerji (mJ)	r#	Enerji (mJ)	r#	GATGA	Halka	FTTG
G1	19,11	5	19,14	6	19,14	7	19,14	6	-0,16	-	-
G2	3,44	5	3,75	6	3,74	7	3,74	7	-8,74	0,32	-
G3	0,23	6	0,254	7	0,254	8	0,254	7	-9,58	-	-
G4	58,92	7	64,61	8	65,15	9	64,15	9	-8,89	0,72	1,53
G5	15,36	15	16,89	16	22,45	19	15,56	19	-1,3	7,88	30,69
G6	22,48	14	23,19	15	29,15	16	22,94	16	-2,02	1,08	21,3
G7	34,41	19	31,19	20	50,86	21	29,85	23	13,25	4,29	41,31
G8	41,79	24	47,08	25	43,62	26	35,91	30	14,06	23,72	17,68
G9	78,68	29	64,75	30	124,66	34	54,97	35	30,13	15,11	55,9
G10	99,52	34	87,64	35	146,49	39	70,36	39	29,3	19,72	51,97
Ortalama enerji gelişimi (%)									5,60	7,28	22,04

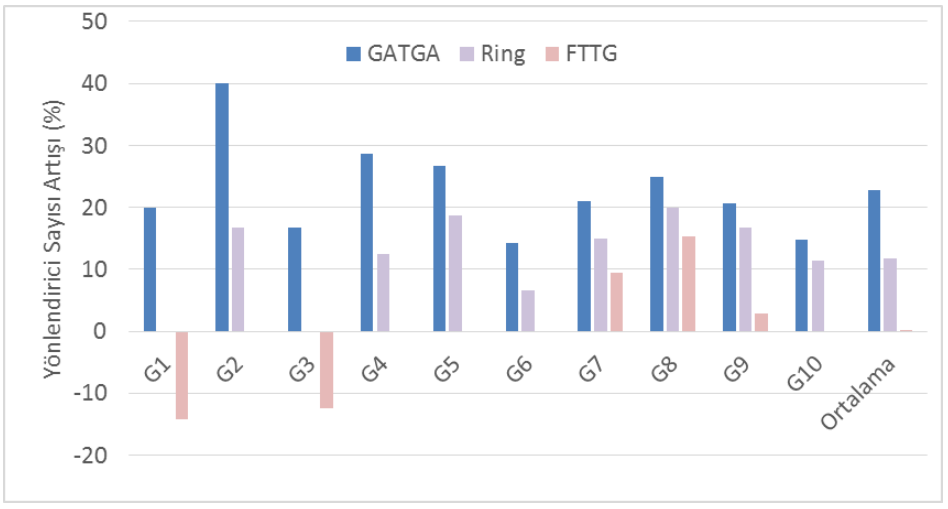
Çizelge 3.3.'te dört yöntemle elde edilen sonuçlar gösterilmektedir. İlk sütunda çizgelerin isimleri verilmiştir. Sonraki sekiz sütunda sırasıyla GATGA, Halka, FTTG ve GA-FTCT tarafından elde edilen topolojiler için enerji tüketimleri ve yönlendirici sayıları görülmektedir. Çizelgede görüldüğü üzere ilk dört denektaşı için (G1'den G4'e) enerji kazanımı çok düşüktür veya hiç gelişme bulunmamaktadır. Yönlendiricilerin sayısı da bu kıyaslamalar için çok benzerdir. Bu küçük farkların nedeni, bu uygulamaların az sayıda düğüme sahip olmasıdır. Diğer bir deyişle, tüm yöntemler, küçük ölçekli denektaşları için benzer veya aynı

sonuçları verir. Öte yandan, GATGA ilk altı karşılaştırmada diğer üç yöntemle karşılaştırıldığında en az yönlendiriciyi kullanarak en iyi enerji değerlerine ulaşmıştır. Bu, GATGA'nın hata toleransını göz önünde bulundurmadan en az donanım gereksinimi ile topoloji üretmesinden kaynaklanmaktadır. Hataya dayanıklı topoloji üretme algoritması GA-FTCT'yi, GATGA ile karşılaştırmamızın nedeni, tasarımlara hata toleransının dahil edilmesinden kaynaklanan donanım ve enerji yükünü göstermektir. Buna rağmen yöntem son dört çizge için GATGA'dan daha iyi enerji değerlerine ulaşmıştır. Daha fazla yönlendirici kullanmanın sonucu olarak daha fazla eşleme seçeneği ortaya çıkmış ve bu on denektaşı için yöntemimiz GATGA'ya göre ortalama %5,60 iyileşme göstermiştir.

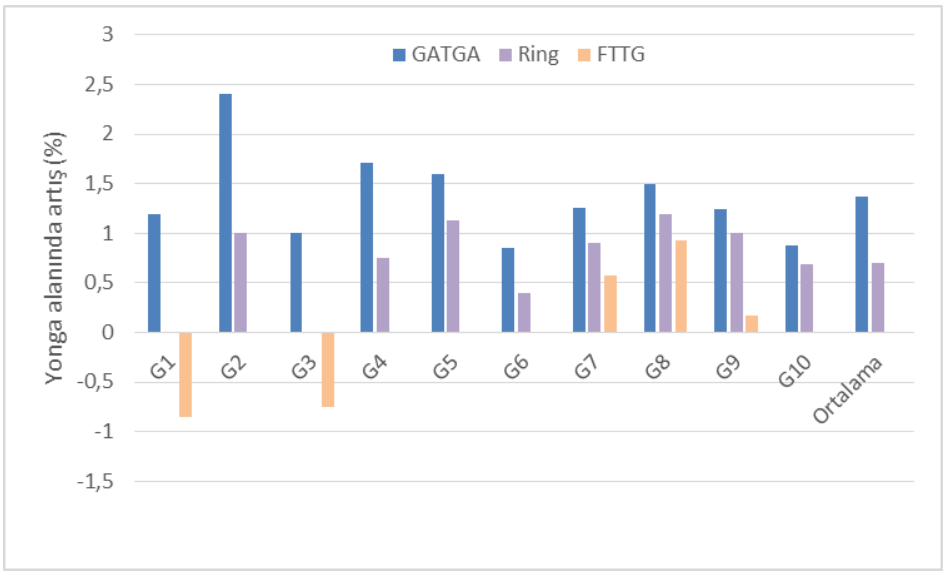
Hataya dayanıklı topolojileri üreten halka ve FTTG yöntemlerine göre sunulan yaklaşım özellikle fazla düğümlü uygulamalarda büyük fark yaratmaktadır. Yukarıda da bahsedildiği gibi, GA-FTCT, uygulama düğümlerinin sayısı küçük olduğunda ise benzer sonuçlar elde etmektedir. Çizelge 3.3.'teki son iki sütun enerji iyileşmelerini göstermektedir. Son altı çizge (G5 - G10) göz önünde bulundurulduğunda, sunulan yöntem yüksek enerji gelişimi göstermektedir. Ortalama enerji tüketiminde halkaya karşı %23,72 ve FTTG'ye karşı %55,9 oranında azalma sağlanmıştır.

Alan karşılaştırması

Uygulamaya özgü düzensiz topolojilere hataya dayanıklılık özelliği eklemek istendiğinde, yönlendirici düğümlerinin sayısındaki artış kaçınılmazdır. İletişim kuran düğümler arasında alternatif yollar oluşturabilmek için topolojiye fazladan bağlar eklenmelidir. Bunun içinde boş portlara ihtiyaç duyulmaktadır. Boş port için ise topolojiye fazladan yönlendirici eklenmelidir. Çizelge 3.3.'te her topoloji oluşturma yöntemi için gerekli olan minimum yönlendirici sayısı belirtilmiştir. Şekil 3.8.'de ise GA-FTCT algoritmamızın diğer yöntemlerin kullandığı yönlendirici sayılarına göre yüzdelik artışı grafiksel olarak bulunmaktadır. Beklendiği ve grafikte de gösterildiği gibi, GA-FTCT, GATGA ve halka topolojilere göre daha fazla yönlendirici kullanmaktadır. Nitekim, on denektaşı için ortalama yönlendirici



Şekil 3.8. GA-FTCT'de kullanılan yönlendirici sayısının GATGA, Halka ve FTTG yöntemlerine göre % artışı.



Şekil 3.9. GA-FTCT ile oluşturulan topolojilerdeki GATGA, Halka ve FTTG yöntemlerine göre toplam yonga alanı artışı.

artışı, GATGA'ya oranla ortalama %22, halka topolojilere karşı ortalama %12 civarındadır. FTTG'ye karşı ise ortalama yönlendirici artışı sadece %0,1'dir ve bu önemsizdir.

Bir topolojiye hata kaldırabilirlik dahil edildiğinde alan artışı düşünülecekse, ağ bileşenlerinin sayısından çok tasarımın toplam alan artışı değerlendirilmelidir. Önceki çalışmalarda, temel bir NoC uygulamasının ağ bileşenlerinin sonuç tasarımın yaklaşık %6'sını kapsadığı

varsayılmıştır [25]. Bu varsayımı göz önünde bulundurduğumuzda, GA-FTCT topolojilerinin alanının diğer üç topoloji türüne karşı toplam artışı çok azdır. Şekil 3.9.'da yöntemimizin diğer yöntemlere göre yonga alanındaki artışı gösterilmiştir. Uygulamaya özgü topolojiye hata kaldırabilirlik özelliği eklemek ve enerji tüketimini düşürmek için bu değerler göz ardı edilebilir.

Çalışma süresi karşılaştırması

Ölçeklenebilirlik göz önüne alındığında bir algoritmanın çalışma zamanı son derece önemlidir. Bu nedenle, raporun bu kısmında, yöntemin diğer üç yöntem ile çalışma süresi karşılaştırmalarına yer verilmiştir. Adil bir karşılaştırma yapmak için, dört yöntemin tümü aynı ortamda çalıştırılmıştır.

Çizelge 3.4. Yöntemlerin çalışma sürelerinin saniye türünde karşılaştırması.

Çizge	GATGA	Halka	FTTG (1)	GA-FTCT(2)	Hız(1/2)
G1	133	31	1521	31	49,06
G2	140	30	1525	34	44,85
G3	151	50	1742	50	34,84
G4	180	34	2523	36	70,08
G5	162	53	9211	62	148,56
G6	166	49	8244	52	158,53
G7	202	66	14427	80	180,33
G8	196	89	22238	125	177,90
G9	232	177	32460	219	148,21
G10	296	219	43993	284	154,90

Çizelge 3.4.'te, sonuç topolojileri elde etmek için dört yöntemin çalışma süreleri saniye türünde verilmiştir. Halka topolojisi üretme GA-FTCT'nin bir parçası olduğu için yürütme süreleri çok yakındır. GA-FTCT'nin ana rakibi olan FTTG, [11]'de belirtildiği gibi çok uzun yürütme süreleri almaktadır. Çizelge 3.4.'ün son sütununda, GA-FTCT'nin FTTG'ye karşı elde ettiği hız gelişimi gösterilmektedir. Bu sütundan görüleceği üzere, FTTG'den 180 kat daha hızlıdır. Bu sütun ayrıca, UAÇ'deki düğüm sayısı arttıkça, sunulan yöntem ile elde edilen hızın, küçük uygulamalara göre çok daha yüksek olduğunu göstermektedir. İlk dört

küçük denektaşlarında (yani G1 ila G4) bile hızlı büyüme dikkat çekicidir. GA-FTCT'nin FTTG'ye kıyasla en belirgin yararı hızı olmuştur.

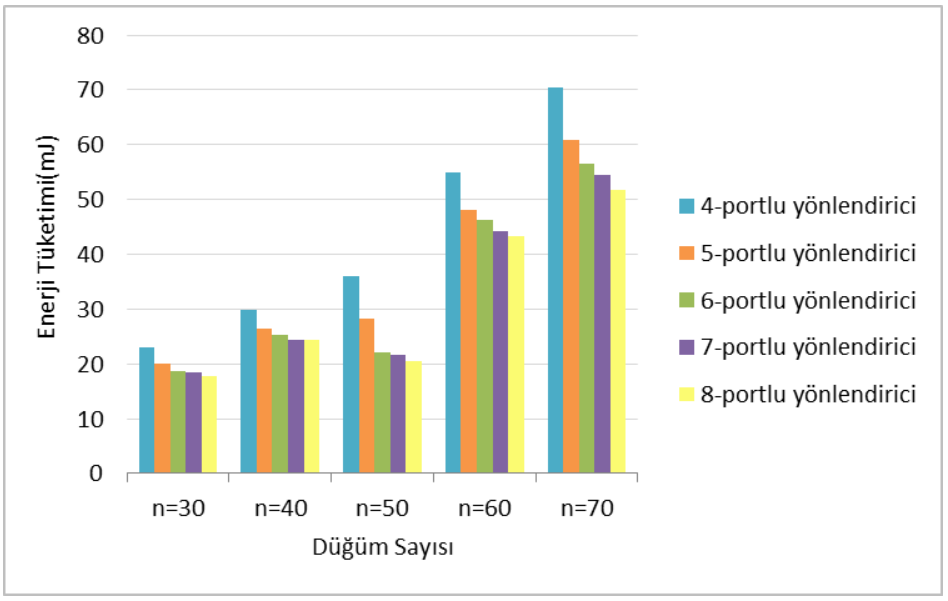
Gecikme süresi karşılaştırması

Bu kısımda yöntemimiz paketlerin gecikmesine dayalı olarak diğer yöntemlerle karşılaştırılmıştır. Gecikme metriği olarak, paketlerin ortalama atlama sayıları (OAS) kullanılmıştır. Her iletişimsel düğüm çifti (yani, uygulama akış çizgesindeki her bir kenar) için atlama sayılarını belirleyip, toplamı grafikteki kenarların sayısına bölerek, uygulamanın tamamının OAS'si hesaplanmıştır.

Çizelge 3.5. Yöntemlerin gecikme süresi karşılaştırması.

	Ortalama atlama sayısı (OAS)				GA-FTCT'nin OAS Gelişimi (%)		
Çizge	GATGA	Halka	FTTG	GA-FTCT	GATGA	Halka	FTTG
G1	0,73	0,64	0,58	0,64	12,50	0,00	-9,72
G2	0,50	0,70	0,75	0,70	-40,00	0,00	6,67
G3	0,73	0,64	0,76	0,64	12,50	0,00	16,27
G4	0,83	0,83	0,95	0,72	13,33	13,33	23,98
G5	3,48	1,70	2,40	0,95	72,66	44,12	60,50
G6	1,26	1,35	1,81	1,10	12,82	19,05	39,40
G7	2,02	1,29	2,69	1,12	44,58	13,21	58,29
G8	3,72	3,33	3,23	2,87	22,76	13,85	11,09
G9	2,84	2,68	4,63	1,58	44,44	41,18	65,88
G10	3,44	3,33	4,76	2,32	32,64	30,37	51,36

Elde edilen sonuçlar Çizelge 3.5.'te verilmiştir. Çizelgedeki ikinci sütun ile beşinci sütun arasında sırasıyla GATGA, halka, FTTG ve GA-FTCT yöntemleri tarafından üretilen topolojiler için ortalama atlama sayıları listelenmiştir. Son üç sütunda ise GA-FTCT'nin sırasıyla GATGA, halka ve FTTG'ye karşı ortalama atlama sayısı oranı gösterilmiştir. Gelişim yüzdeleri yöntemimizin, karşılaştırılan diğer yöntemlere kıyasla çok daha iyi OAS sonuçları elde ettiğini göstermektedir.

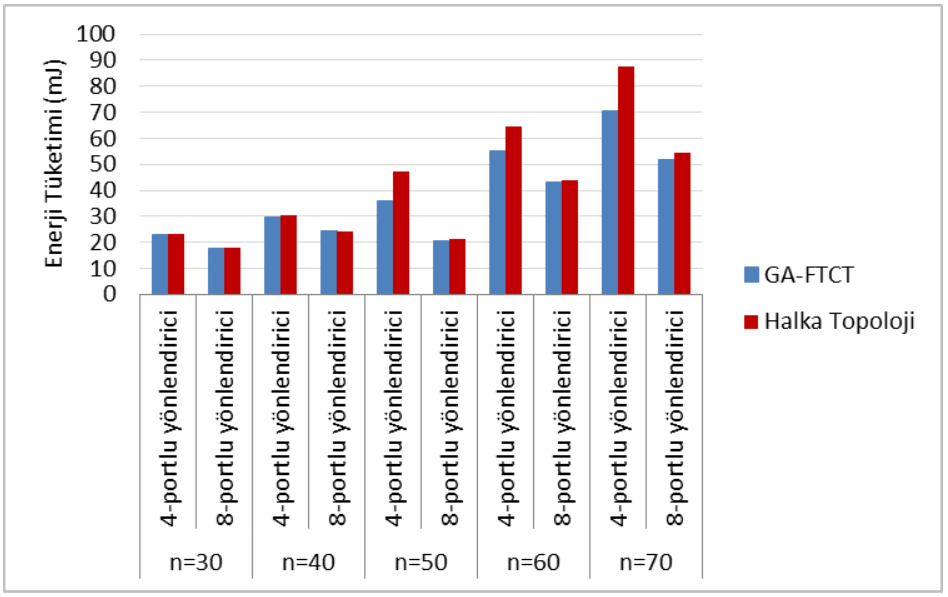


Şekil 3.10. Yönlendiricilerin port sayılarının analizi.

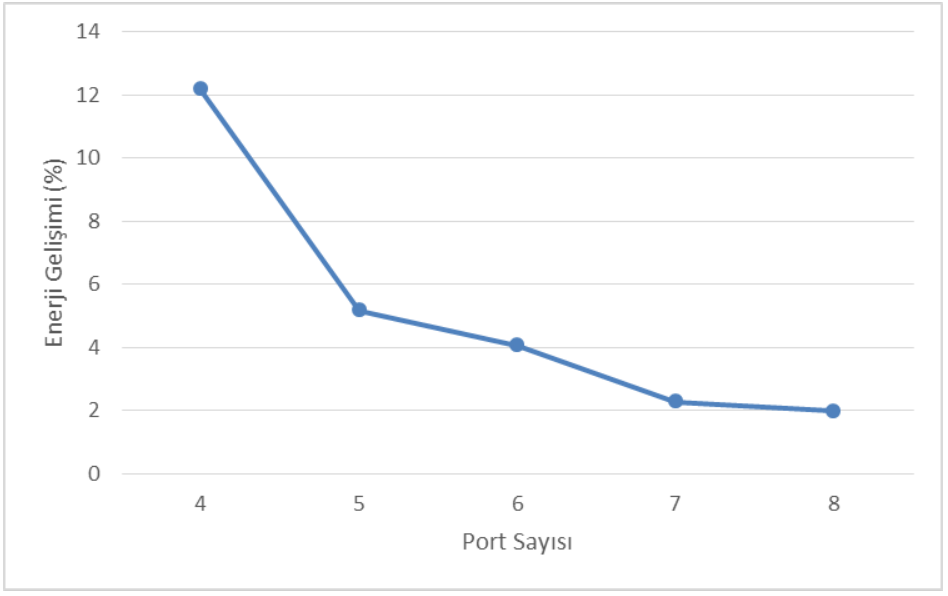
Algoritma parametreleri analizi

Genetik algoritmalar algoritma parametrelerine karşı çok hassas algoritmalardır. Örneğin, çok küçük bir popülasyon boyutu kullanırsak, algoritmanın optimal bir çözüme ulaşmak için çok uzun yürütme sürelerine ihtiyacı olabilir. Öte yandan, geniş popülasyon büyüklüğünün kullanılması genetik operatörün uygulama sürelerini arttırmakta, bu da toplam uygulama süresini arttırmaktadır. Bu nedenle, en iyi algoritma parametrelerini belirlemek için farklı veriler üzerinde deneyler yapmamız gerekebilir. Bu bölümde, GA-FTCT'nin bazı parametrelere duyarlılığını analiz etmek için gerçekleştirdiğimiz deneyleri sunmaktayız. Bu deneyler kapsamında yalnızca Çizelge3.1.'in son beş çizgesi kullanılmıştır.

İlk deneyde, port sayılarının enerji tüketimine etkisi değerlendirilmiştir. Bunun için, dörtten sekize kadar değişen sayıda porta sahip yönlendiriciler ile oluşturulan halka topolojilerin ve GA-FTCT yöntemi ile elde edilen topolojilerin enerji tüketim değerleri karşılaştırılmıştır. Şekil 3.10.'da, her bir uygulama için GA-FTCT yöntemi ile elde edilen sonuçlar gösterilmektedir. Port sayısındaki artışla aynı yönlendiriciye daha fazla uygulama düğümü yerleştirilebilir, böylece sistem tarafından tüketilen enerji de azalır. Ek olarak, port sayısının arttırılması, düğümler arasındaki ortalama atlama sayısını da azaltır.

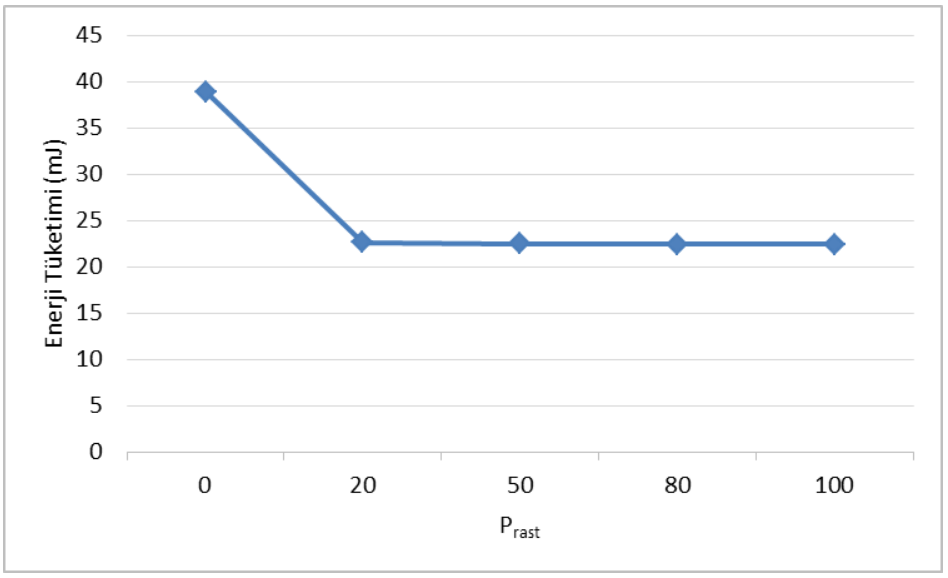


(a)



(b)

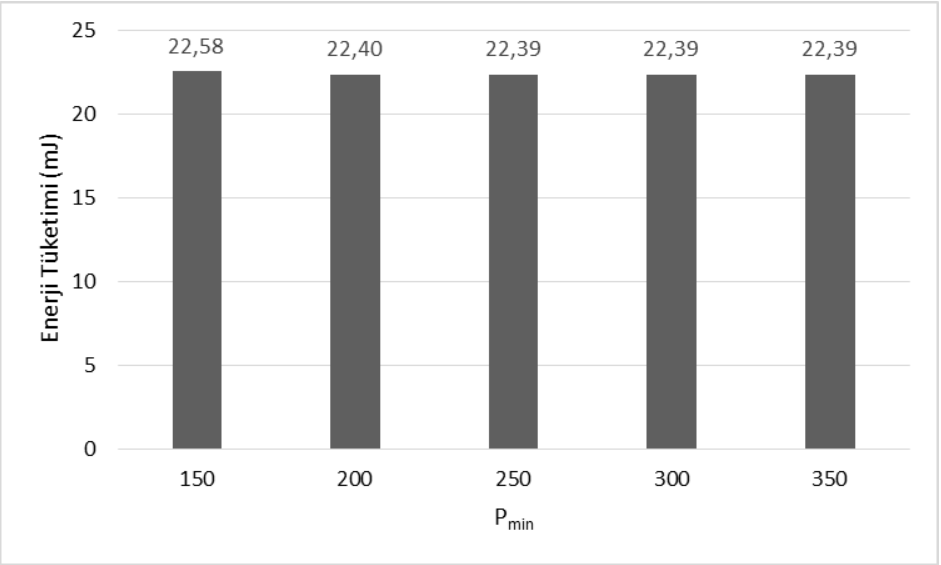
Şekil 3.11. GA-FTCT topolojilerin halka topolojiler ile yönlendirici port sayılarına göre (a) toplam enerji tüketimi ve (b) enerji gelişimi açılarından karşılaştırılması.



Şekil 3.12. P_{rast} parametresinin analizi.

Şekil 3.11.a'da, beş uygulama için dört ve sekiz portlu yönlendiriciler kullanılarak, halka ve GA-FTCT tarafından elde edilen enerji tüketim değerleri gösterilmektedir. GA-FTCT'nin halka topolojisine göre avantajı, topolojideki yönlendiriciler arasına ek bağlantılar eklenmesine izin vermesidir. Böylece yönlendiriciler arasındaki mesafe azalır ve toplam enerji tüketimi de düşer. Bu nedenle, GA-FTCT ile tüm test durumlarında daha iyi sonuçlar elde edilmiştir. Bununla birlikte, port sayısını arttırdığımızda, halka topolojinin de ortalama atlama mesafesi düşmektedir. Bu da Şekil 3.11.b'de görüldüğü gibi enerji kazancının azalmasına neden olmaktadır. Port sayısı artıkça GA-FTCT ile üretilen topolojiler de halka topolojiler gibi görünmeye başlar.

Son deney kapsamında, genetik algoritmaların uygulanacağı birey sayısı (P_{rast}) ile bu bireylerin seçileceği popülasyondaki en iyi bireylerin bulunduğu kümenin büyüklüğü (P_{min}) parametreleri değerlendirilmiştir. Şekil 3.12. ve Şekil 3.13.'de sırasıyla P_{rast} ve P_{min} parametrelerinin değerlerinin etkisi verilmektedir. Deneyler kapsamında G6 kullanımıştır. Sonuçlarda, her iki değerin de toplam enerji tüketimi üzerindeki etkisinin çok küçük olduğu görülmektedir. P_{min} parametresinin değerini 150'den 200'e çıkardığımızda, kazanç çok küçük olmaktadır. Ancak, onu daha fazla arttırmak, enerjiyi azaltmaya yardımcı olmaz. Bu nedenle, bu



Şekil 3.13. P_{min} parametresinin analizi.

değer deneylerde kullanılmak üzere 200 olarak belirlenmiştir. Benzer şekilde, P_{rast} parametresinin 50'nin üstünde olması sonuçları iyileştirmemektedir. Gereksiz sayıda uygulanan genetik operatörler nedeniyle yürütme sürelerini arttırmamak için GA-FTCT'de bu parametrenin değeri 50 olarak seçilmiştir.

3.2. Uygulamaya Özgü Hata Kaldırabilir Topolojiye Hata tespit ve Yeniden Yönlendirme Birimleri Eklenmesi

Belirli bir uygulama için üretilen topolojiler için düzenli bir model yoktur. Bazı düzensiz topolojiler, her yönlendirici en az iki bağlantı ile ağın geri kalanına bağlandığından ağ bileşenlerindeki kalıcı hataları tolere edebilir. Bu nedenle, iki iletişim düğümü arasındaki bağlantılardan birinde bir hata varsa, paketlerin yolu alternatif bir yolla değiştirilebilir. Ancak, bu topolojilerde, sistemin çalışması sırasında hata olması durumunda dinamik olarak paketi alternatif yola yönlendirecek yöntemlere ihtiyaç duyulmaktadır.

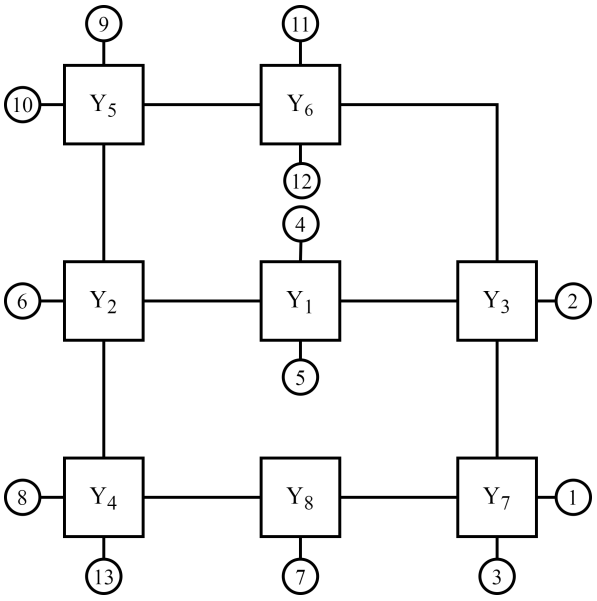
Tezin bu kısmında bu problem ele alınmıştır. Devam eden bölümlerde problemin detayları, tasarlanan sistemin çalışma prensipleri ve gerçekleştirilen deneyler anlatılmaktadır.

3.2.1. Problem Tanımı

Tasarımcılar, düşük enerji tüketimi ve gecikme süresi açısından emsallerinden daha verimli olmaları nedeni ile düzensiz Yonga-üstü-Ağ (YüA) tasarımlarını tercih etmektedirler. Ancak, düzensiz topolojilerde genellikle iki iletişim düğümü arasında yalnızca tek bir yol bulunur. Kalıcı bir bağlantı arızası, tüm yonganın kullanılamaz hale gelmesine neden olabilir çünkü aktarılan veriler için alternatif yol seçeneği yoktur. Bu tez kapsamında iletişim kuran her bir yönlendirici çifti arasında en az iki alternatif yol bulunan uygulamaya özgü hata kaldırabilir topoloji üreten bir yöntem sunulmuştur.

Amacımız, uygulama akış çizgesi (UAÇ) verilen bir uygulamaya özgü oluşturulan hata-kaldırabilir düzensiz topolojisiye (T), hata tespit ve yol güncelleme mekanizmaları eklemektir. Bir UAÇ, (U(D, B)) düğümler (uygulama çekirdekleri) kümesi (D) ve bu düğümler arasındaki bağlantıların (B) kümesinden oluşur. Her $b_{i,j} \in B$, $d_i \in D$ düğümü ile $d_j \in D$ düğümünün bağımlılığını gösterir. $t_{i,j}$, d_i ve d_j düğümleri arasındaki veri aktarım miktarını belirtir. Bir örnek UAÇ Şekil 3.1.'de verilmiştir. T(Y, I), yönlendiriciler kümesinden (Y) ve bu yönlendiriciler arasındaki bağlantılar kümesinden (I) oluşur. T hata toleranslı bir topolojiyi belirttiği için, burada her iki yönlendirici çifti arasında en az iki farklı yol bulunmaktadır. Şekil 3.14.'te, Şekil 3.1.'de akış çizgesi verilen MP3 Encoder uygulamasının hata kaldırabilir düzensiz topolojiye eşlenmiş bir örneği gösterilmiştir.

Sistem çalışırken, eğer uygulama için oluşturulmuş temel yönlendirme yollarında hata ortaya çıkarsa ve alternatif yol bilgisi mevcut değilse sistem çalışmaya devam edemez. Yönlendiricilerde alternatif yol bilgisinin olması sistemin çalışmaya devam etmesini sağlar. Öncelikle hatanın tespit edilmesi ve daha sonra paketin alternatif yol üzerinden yeniden yönlendirilmesi gerekmektedir. Hatalar kalıcı veya geçici olabilirler. Eğer yönlendiricinin aldığı veride hata varsa, yönlendirici paketin tekrar gönderilmesi için kaynak yönlendiriciye sinyal gönderir. Eğer art arda hata olmaya devam ederse gönderilen yol üzerinde bir kalıcı hata olduğu tespit edilir. Bu durumda yönlendiricilerin yönlendirme tablolarını güncelleyerek yeni bir rota oluşturmaları gerekmektedir.



Şekil 3.14. Şekil 3.1.'te verilen MP3 Encoder uygulamasının hata kaldırabilir düzensiz topoloji üzerine eşlenme örneği.

Paketlerin iletileceği varsayılan yollar ve hata olması durumundaki alternatif yollar belirlenirken amaç yongadaki toplam enerji tüketimini en aza indirgemektedir. Bir bitin kaynak düğümden hedef düğüme iletilmesinden kaynaklı enerji tüketimi Denklem 7'de verilmiştir. Burada $E_{yönlendirici}$ ve $E_{bağlantı}$ sırasıyla bir yönlendiricinin ve bir bağlantının enerji tüketim değerlerini göstermektedir. $y_{sayısı}$, bir bitin kaynak düğümden hedef düğüme iletilirken geçtiği yönlendiricilerin sayısını ifade etmektedir.

$$E_{bit_{(d_i,d_j)}} = y_{sayısı} * E_{yönlendirici} + (y_{sayısı} - 1) * E_{bağlantı} \tag{7}$$

$d_i \in D$ düğümü ile $d_j \in D$ düğümü arasında iletilen verinin miktarının $t_{i,j}$ olduğu düşünüldüğünde bu paketin iki düğüm arasında iletiminden doğan enerji miktarı aşağıdaki gibi hesaplanabilir.

$$E_{(d_i,d_j)} = t_{i,j} * E_{bit_{(d_i,d_j)}}. \tag{8}$$

Sonuç olarak sistemin toplam enerji tüketimi, iletişim kuran bütün düğümler arasındaki paketlerin iletimi ile ortaya çıkan enerji tüketimlerinin toplamı olarak Denklem 9 ile hesaplanmaktadır.

$$E_{toplam} = \sum_{\forall e_{i,j}} E_{(d_i,d_j)}. \tag{9}$$

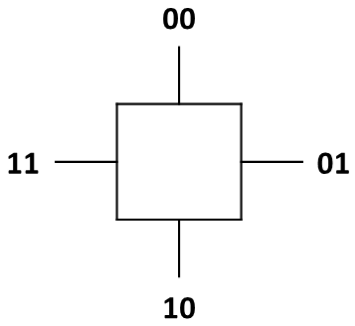
3.2.2. Sunulan Yaklaşım

Bu çalışmada, hataya dayanıklı uygulamaya özel topolojiler için bir hata tespit ve yol güncelleme üniteleri tasarlanmıştır. Amacımız, belirli bir uygulamanın uygulama akış diyagramından oluşturulan hataya dayanıklı düzensiz topolojisine (T(Y, I)) hata tespit ve yol güncelleme mekanizmaları eklemektir. Bu kısımda yönlendiricilerin yönlendirme tablolarını oluşturma işleminden, hata tespit ve yeniden yönlendirme birimlerinden bahsedilmiştir.

Yönlendirme Tabloları Oluşturma

Her bir yönlendiriciye ait bir yönlendirme tablosu (YT) bulunmaktadır. Bir YT, paket kimlik numarasını ve paketin yönlendirileceği hedef portları tutmaktadır. 4 bağlantı noktalı bir yönlendirici için port bilgisi olarak Şekil 3.15.'te gösterilen numaralandırma sistemi kullanılmaktadır. Örneğin hedef port olarak 00 görülüyorsa paket kuzey yöndeki yönlendiriciye gönderilecektir. Bir paket yönlendiriciye girdiğinde, yönlendirme mekanizması paket kimliğini tablodan arar ve paketi ilgili porta gönderir.

Topoloji T(Y, I) bir çizge ve aralarında veri akışı bulunan $d_k \in D$ ve $d_l \in D$ uygulama düğümleri $y_i \in Y$ ve $y_j \in Y$ yönlendiricilerine eşlenmiş olsun. Öncelikle en kısa yol algoritması ile y_i ve y_j yönlendiricileri arasındaki en kısa yol bulunmaktadır. Varsayılan yollar olarak adlandırılan bu yollar, aralarında veri akışı bulunan her bir düğüm çiftinin eşlendiği yönlendiriciler için hesaplanmış ve yönlendiricilerin yönlendirme tablolarına eklenmiştir. T



Şekil 3.15. 4 bağlantı noktalı bir yönlendirici için port numaraları.

çizgesinde başlangıçta her bir bağlantının ağırlığının (a) eşit olduğu varsayılır. Alternatif yolları belirlemeden önce çizgedeki her bir a_j'nin değeri ilgili bağlantıyı kullanan paket sayısı oranında artırılır. Bu işlem, alternatif yollar belirlenirken daha az kullanılan bağlantıların tercih edilmesini sağlar ve böylece yoğun olarak kullanılan bağlantılarda kilitlenme gibi problemlerin ortaya çıkma ihtimali düşer.

Sistemin çalışması sırasında bağlantılarda ortaya çıkabilecek bir hata durumunda, paketlerin hedef yönlendiriciye iletilebilmesi için her bir yönlendiricide alternatif yol bilgisinin bulunması gerekmektedir. Alternatif yolları belirleyebilmek için algoritma rota üzerindeki her bir bağlantıyı tek tek kaldırmakta ve yeni yolları belirlemektedir. Bu yollar ilgili yönlendiricilerin tablolarına kaydedilmektedir. Bu durum sistemin tek bir hata olması durumunda çalışmaya devam etmesini sağlar. Daha sonra alternatif yollar üzerindeki her bir bağlantı da tek tek kaldırılarak belirlenen yollar da tablolara eklenir ve sistemde ortaya çıkabilecek iki hata durumunda sistemin çalışmaya devam etme ihtimali artırılmış olunur.

Örneğin Şekil 3.14.'te birinci düğümden (d_1) ikinci düğüme (d_2) iletilen paket, sistemde hata olmaması durumunda $d_1 \rightarrow y_7 \rightarrow y_3 \rightarrow d_2$ yolunu izlemektedir. Yedinci yönlendirici (y_7) ile üçüncü yönlendirici (y_3) arasındaki bağlantıda ortaya çıkacak bir hata sonucunda bu paket $d_1 \rightarrow y_7 \rightarrow y_8 \rightarrow y_4 \rightarrow y_2 \rightarrow y_5 \rightarrow y_6 \rightarrow y_3 \rightarrow d_2$ yolundan iletilecektir. Bu durumda, y_2 ve y_5 arasındaki bağlantıda ikinci bir hata ortaya çıkarsa bu paket ikinci yönlendiriciden birinci yönlendiriciye iletilecek, ve $y_1 \rightarrow y_3 \rightarrow d_2$ yolu üzerinden hedef düğüme varacaktır. Bu belirlenen yollar sonucunda yönlendiricilerin yönlendirme tablolarının durumu Çizelge 3.6.'da verilmiştir.

Çizelge 3.6. 0 numaralı paket için örnek yönlendirme tablosu oluşturma.

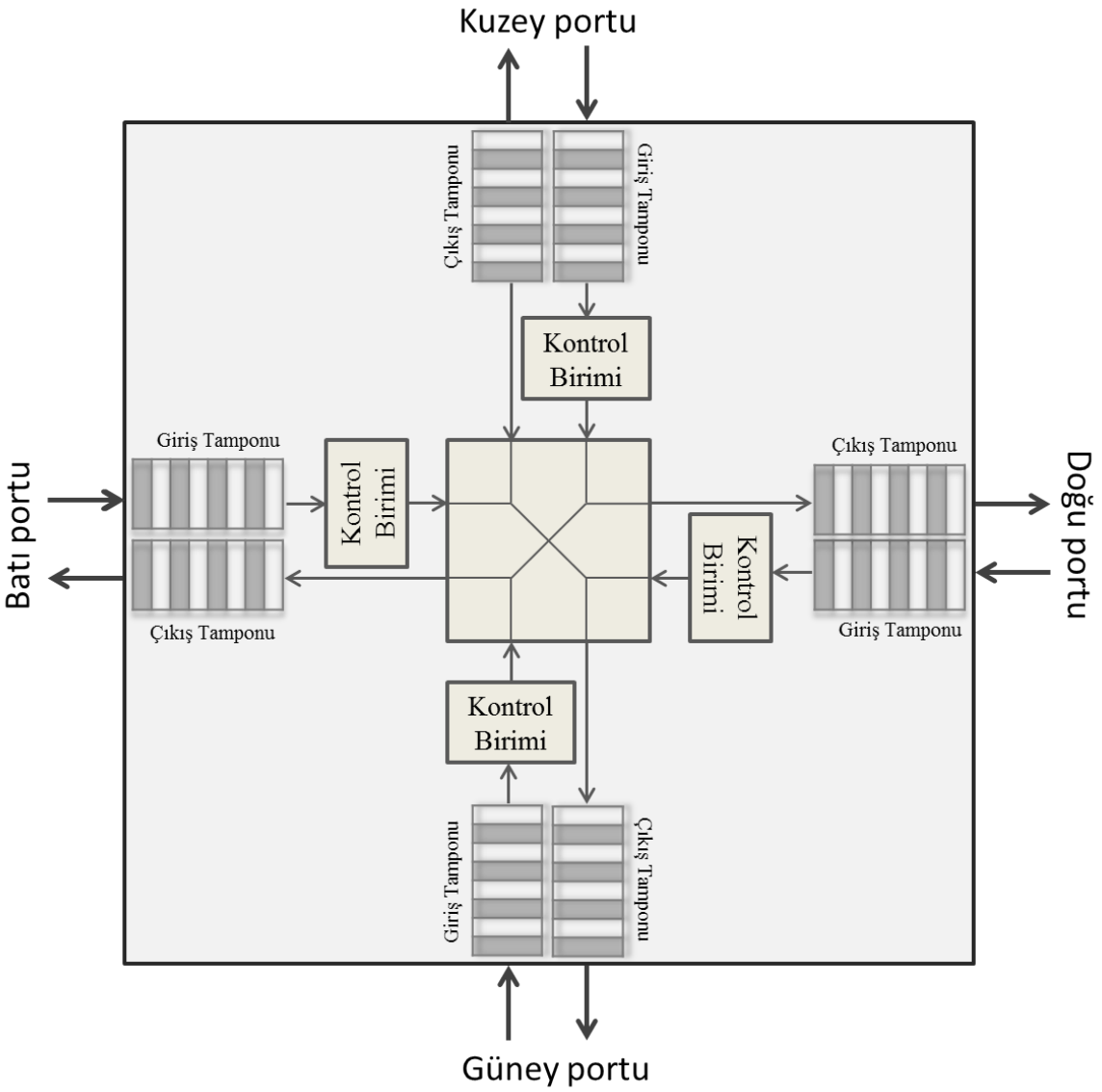
YT_1			YT_2			YT_3			YT_4		
Paket	**Port**	**Port**	**Paket**	**Port**	**Port**	**Paket**	**Port**	**Port**	**Paket**	**Port**	**Port**
0	00	11	0	00	01	0	01		0	00	

YT_5			YT_6			YT_7			YT_8		
Paket	**Port**	**Port**	**Paket**	**Port**	**Port**	**Paket**	**Port**	**Port**	**Paket**	**Port**	**Port**
0	01	10	0	01	11	0	00	11	0	11	

Hata Tespiti ve Yeni yol belirlenmesi

Şekil 3.16.'da görüldüğü gibi tasarlanan sistemde, her bir yönlendiricinin kuzey, doğu, güney ve batı yönlerine karşılık gelen dört giriş ve dört çıkış portu bulunmaktadır. Her yönlendirici bu portlar ile komşu yönlendiricilere veya yerel işlem elemanlarına fiziksel bağlantı kabloları ile bağlıdır. Bu fiziksel bağlantı kablolarından birim zamanda geçebilecek veri miktarı sınırlıdır. İletişim kuran her bir düğüm çifti arasında iletilen verinin miktarı ($t_{i,j}$) birbirinden farklı olduğu için her bir paket parçalara ayrılarak gönderilmelidir.

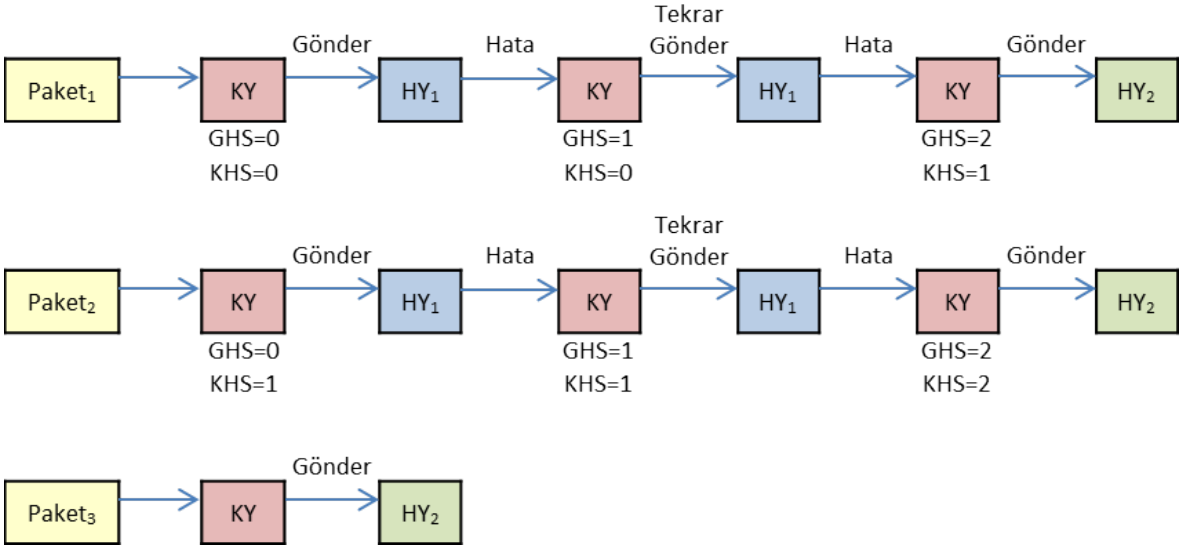
Her bir portta, her birinde en fazla 8 paket parçasının tutulabildiği giriş ve çıkış tamponları bulunmaktadır. Veri iletilirken, kaynak yönlendiriciden iletilen veri ile hedef yönlendiriciye ulaşan verinin içeriklerinin aynı olması gerekir. Veriler iletim hattında bozulursa hedef yönlendiricinin bunu farketmesi beklenmektedir. Her bir giriş tamponuna eklenen kontrol ünitesi döngüsel artıklık denetimi metodu kullanarak gelen paketin içeriğinin doğruluğunu kontrol eder. Döngüsel artıklık metodunda k bit uzunluğundaki veriye sabit n uzunluğunda bir kontrol değeri eklenir [37]. Bu n bitin hesaplanması için hem gönderici hemde alıcı yönlendirici tarafından bilinen $n + 1$ uzunluğunda bir üretici polinoma ihtiyaç duyulmaktadır. Mesaja eklenecek n biti bulmak için k bit uzunluğundaki mesajın kendisi bölünen, $n + 1$ uzunluğundaki üretici polinom da bölen olarak düşünülür. Polinom bölümü sonucunda elde edilen kalan, mesajın sonuna eklenerek veri oluşturur. Bu şekilde iletilen $k + n$ bit uzunluğundaki veriyi hedef yönlendirici kontrol eder. Kontrol işlemi için gelen $k + n$ uzunluğundaki veri üretici polinoma bölünür. Kalan 0 ise pakette hata yoktur. Bu durumda



Şekil 3.16. Yönlendirici konfigürasyonu.

hedef yönlendirici pakette hata olmadığına dair kaynak yönlendiriciyi bilgilendirir ve kaynak yönlendirici ilgili paketi tamponundan siler.

Her yönlendiricinin iki farklı sayacı bulunmaktadır. Bunlardan birincisi (geçici hata sayacı, GHS) bağlantı üzerindeki hatanın geçici bir hata olup olmadığını belirlemek için kullanılırken, ikincisi (kalıcı hata sayacı, KHS) bağlantıdaki hatanın kalıcı olması durumunun kontrolü için tanımlanmıştır. Bir yönlendirici bir paket parçasında bir hata bulursa, kaynak yönlendiriciyi bilgilendirir. Kaynak yönlendirici bu bağlantıya ait GHS değerini artırır



Şekil 3.17. Hatalı link olması durumunda yönlendiricilerin davranış örneği.

ve paketi aynı hedef yönlendiriciye tekrar gönderir. Bu süreç, GHS değeri için belirlenmiş eşik değerine (σ_g) ulaşılana kadar devam eder. Bu değere ulaşıldığında, kaynak yönlendirici KHS'yi artırır ve paketi geçici olarak alternatif bir yoldan iletir. Bir bağlantı için KHS, önceden tanımlanmış değere (σ_k) ulaştığında, bu bağlantıda kalıcı bir hata olduğu anlamına gelir. Kaynak yönlendirici bu yolu yönlendirme tablosundan siler ve paketleri alternatif rotaya yönlendirir.

Örneğin, bir kaynak yönlendirici KY ile hedef yönlendirici HY_1 arasındaki bağlantıda kalıcı hata bulunan bir sistemde, birinci, ikinci ve üçüncü paketlerin bu bağlantıyı kullandığını varsayalım. Bu sistem için kabul edilebilir eşik değerleri σ_g ve σ_k 2 olsun. Bu şartlar altında paketlerin iletimi sırasında yönlendiricilerin davranışı Şekil 3.17.'de gösterildiği gibi olacaktır. KY, $Paket_1$'i HY_1'e gönderir. $Paket_1$ geldiğinde, HY_1 pakette bir hata bulur ve KY'yi bilgilendirir. KY, GHS değerini artırır ve paketi HY_1'e yeniden gönderir. Bu prosedür, GHS, σ_g'ye ulaşana kadar devam eder. KY, $Paket_1$'i alternatif yola yönlendirirken KHS değerini arttırır ve GHS değerini sıfırlar. KHS değeri σ_k değerine ulaştığında, KY bu yolu sonraki paket iletimlerinde kullanılmamak üzere yönlendirme tablosundan siler ve paketleri hedefe alternatif yolla iletir. Bu nedenle, Şekil 3.17.'da $Paket_3$ doğrudan HY_2 üzerinden hedef yönlendiriciye iletilmektedir.

Çizelge 3.7. Deneylerde kullanılan denektaşlarının özellikleri.

Akış Çizgesi	Adı	Düğüm Sayısı	Bağlantı sayısı
MWD [10]	G1	12	12
MP3 Enc. [1]	G2	13	13
H263 Dec. [1]	G3	14	15
VOPD [10]	G4	16	20

Bağlantıdaki geçici hata, verilerde beklenmeyen değişikliklere neden olmaktadır. Hata sayacı artırılmış bir bağlantı üzerinden gönderilen verinin doğru algılanması, her zaman hatanın düzeldiği ve o bağlantının artık güvenli olduğu anlamına gelmez. Verinin içeriğine göre hatalı bağlantının tespit edilememesi durumu ortaya çıkabilir. Örneğin, hata içeren bir bağlantı, ilettiği verilerin üçüncü bitini sıfırlıyor olsun. Eğer üçüncü biti zaten 0 olan bir veri bu bağlantı üzerinden gönderilirse hata algılanamaz. Bu durumda bağlantıyı güvenli olarak belirleyip hata sayaçlarını sıfırlamak doğru bir yaklaşım olmaz. Bu nedenle, sunulan yöntem kapsamında $\sigma_d > 1$ koşulunu sağlayan, σ_d paket peşpeşe bozulmadan iletildiğinde bütün hata sayaçları sıfırlanmaktadır.

3.2.3. Deneysel Sonuçlar

Bu bölümde önerilen yöntemin performansını değerlendirmek için yapılan deneyler ve sonuçları açıklanmaktadır. Deneyler, Multi Window Display (MWD) [10], MP3 Encoder (Mp3 Enc.), 263 Decoder (263 Dec.) [1], ve Video Object Plane Decoder (VOPD) [10] gibi çeşitli denektaşları üzerinde gerçekleştirilmiştir. Bu denektaşlarının ayrıntıları Çizelge 3.7.'de verilmiştir.

Uygulama düğümleri bir topolojiye eşlendikten sonra yonga-üstü-ağ üzerinde veriler uygulama düğümleri arasında iletilir. Deneyler kapsamında, çalışma sırasında yönlendiriciler arasındaki bağlantıların durumuna göre gerçekleşebilecek farklı durumlar değerlendirilmiştir. Bunlar:

- **Durum 1** Bağlantılarda herhangi bir hata oluşmaz ve paketler düzgün bir şekilde hedef yönlendiricilere ulaştırılır.
- **Durum 2** Bağlantılarda geçici hatalar oluşabilir.
  - **2.a** Sistemin çalışma sürecinde, bir zaman biriminde, ağ üzerinde sadece bir bağlantı üzerinde geçici hata olabilir.
  - **2.b** Sistemin çalışma sürecinde, bir zaman biriminde, ağ üzerinde iki bağlantı üzerinde önceden tanımlı bir olasılıkla hata olabilir. Hata sadece bir zaman biriminde geçerlidir. Gerçekleşir ve düzelir.
  - **2.c** Sistemin çalışma sürecinde, bir zaman biriminde, ağ üzerinde iki bağlantı üzerinde belli bir oranda hata olabilir. Hatanın düzelmesi önceden tanımlı bir olasılığa bağlıdır.
- **Durum 3** Bağlantılarda kalıcı hatalar oluşabilir.
  - **3.a** Sistemin çalışma sürecinde, bir zaman biriminde, ağ üzerinde sadece bir bağlantı üzerinde kalıcı hata olabilir.
  - **3.b** Sistemin çalışma sürecinde, bir zaman biriminde, ağ üzerinde iki bağlantı üzerinde kalıcı hata olabilir.
- **Durum 4** Sistemin çalışma sürecinde, bir zaman biriminde, bir bağlantıda geçici bir bağlantıda kalıcı hata olabilir.

Sunulan yöntemin performansı değerlendirilirken, sistem üzerinde bir bağlantıda hata olması ve iki bağlantıda hata olması durumlarının, bağlantı hatası bulunmayan sisteme karşı ortalama enerji tüketimi, paketlerin hedef yönlendiricilere iletilmesi için gerekli ortalama birim zaman ve paketlerin hedef yönlendiricilere iletim oranları açılarından karşılaştırılması yapılmıştır. Gerçekleştirilen sistemin performansını ölçerken kullanılan deneysel kurulum yapılandırmaları aşağıda sıralanmıştır. Değerlendirilen sonuçlar, bütün sistem çalışmalarının ortalamaları göz önünde bulundurularak elde edilmiştir.

- Bağlantılardan birim zamanda en fazla 128 bitlik veri iletilebileceği varsayılmıştır. İletişim kuran uygulama düğümleri arasında iletilen paketler 120 bitlik parçalara ayrılmıştır. Kalan 8 bit hata kontrolünün gerçekleştirilebilmesi için kullanılan döngüsel artıklık metodu için kullanılmaktadır.

- Yönlendiricilerin geçici hata sayaçları için önceden tanımlanmış σ_g değeri 3 olarak belirlenmiştir. Bir bağlantıyı kullanan paket parçalarında 3 kere bozulma meydana geldiğinde paket geçici olarak alternatif yola yönlendirilecektir.

- Yönlendiricilerin kalıcı hata sayaçları için eşik değeri (σ_k) 2 olarak belirlenmiştir. Sayaç 2 değerine ulaştığında yönlendirici bu yolu yönlendirme tablosundan silecek ve bir sonraki paket parçasını alternatif yoldan iletecektir.

- Bir bağlantının güvenli olarak belirlenebilmesi için o bağlantıyı kullanan belli sayıda paket parçasının bozulmadan iletilebilmesi gerekmektedir. Deneyler kapsamında bu sayı (σ_d) 4 olarak kabul edilmiştir.

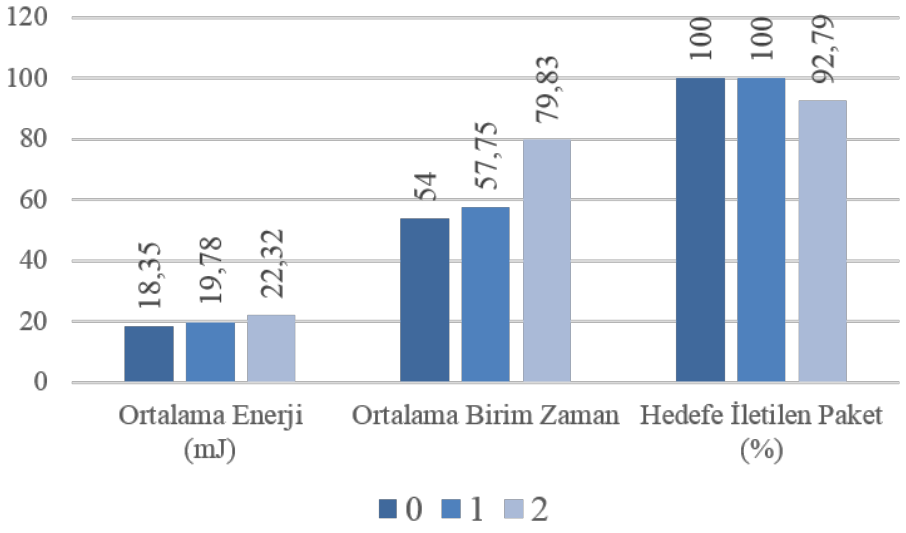
- Geçici hatalı sistemlerde, bağlantılarda hata olması ihtimali %10 ile %50 aralığında değişmektedir.

- Hatalı sistemlerde sonuç elde edilirken, kullanılan her bir bağlantı sırayla değerlendirilmiştir. Her biri hatalı olarak varsayılarak, paketlerin hepsi iletilene kadar sistem çalıştırılmış ve sonuçların ortalaması alınmıştır. Örneğin, n bağlantı içeren bir uygulama için tek bağlantı hatası değerlendirilirken her bir bağlantı tek tek göz önünde bulundurulurken bu n tane sistem çalışmasına denk gelmektedir. İki hatalı sistemlerde ise $C(n, 2)$ çalışma gerekir. Burada $C(n, 2)$, n elemanlı bir kümenin ikili kombinasyonlarının sayısını belirtmektedir.

- Sistemin toplam enerji tüketimini hesaplamak için [38] numaralı çalışmada 100-nm teknolojisi için verilen güç modeli benimsenmiştir. Bu modele göre, yönlendiricilerin giriş ve çıkış portları için enerji tüketimlerinin sırasıyla 328 nJ/Mb ve 65,5 nJ/Mb olduğu varsayılmaktadır. Ayrıca, bir bağlantının enerji tüketimi ise 79,6 nJ/Mb/mm

Çizelge 3.8. Varsayılan, bir bağlantı hatalı ve iki bağlantı hatalı sistemler için elde edilen ortalama enerji tüketimi, paketlerin hedefe ulaşması için gereken ortalama birim zaman, ve hedefe ulaştırılan paketlerin tüm paketlere oranı.

	Ortalama Toplam Enerji Tüketimi (mJ)			**Ortalama Birim Zaman**			**Hedefe ulaştırılan paket (%)**		
	0	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**
G1	13,95	15,93	19,44	14	20,59	34,3	100	100	97,08
G2	12,26	12,53	13,83	108	108,28	121,99	100	100	93,38
G3	12,47	13,09	14,65	74	74,92	108,41	100	100	88,47
G4	34,71	37,55	41,34	20	27,2	54,6	100	100	92,24



Şekil 3.18. Hatalı bağlantı sayısının ortalama enerji tüketimine, paketlerin iletilmesi için gerekli birim zamana ve hedef yönlendiriciye ulaşan paketlerin oranına etkisi.

olarak alınmaktadır. Bu çalışma kapsamında bağlantıların uzunluğunun sabit 3 mm olduğu varsayılmıştır.

- Bir paket parçasının iki yönlendirici arasındaki bağlantıdan geçme süresi 1 birim zaman kabul edilirken, kontrol biriminde paketin değerlendirilip ilgili çıkış tamponuna yönlendirilmesi işlemi 2 birim zaman gerektirmektedir.

Sunulan hata tespit ve yol güncelleme yöntemi sistemin bir hata durumunda dış müdahale gerektirmeden çalışma olanağı getirmiş olsada, eklenen işlemler mimaride enerji tüketimi,

çalışma süresi, gibi değerlerde artışa neden olmaktadır. Tek bağlantı hatalı (Durum2.a, Durum3.a) ve iki bağlantısında hata olan sistemlerin (Durum2.b, Durum2.c, Durum3.b ve Durum4) değişik parametreler ile enerji tüketimleri, çalışma zamanları, ve bu yapılarda paketlerin kaynak yönlendiriciden hedef yönlendiriciye iletilme oranları hesaplanmıştır. Çizelge 3.8.'de sırasıyla bu değerler verilmiştir. Ayrıca Şekil 3.18.'de ortalama sonuçlar grafiksel olarak gösterilmektedir. Çizelge ve grafiklerden de anlaşılacağı üzere sistemde herhangi bir bağlantı hatası olmaması durumda paketler sürekli en kısa yoldan iletildiği için bütün değerler diğer yapılara göre daha düşük çıkmaktadır. Bir ve iki bağlantı hatalı ağlar çalışma süresini sırasıyla %6,93 ve %47,8 artırırken, ortalama %7,78 ve %21,6 daha fazla enerji tüketimine neden olmaktadırlar. Tek hatalı sistemlerde, bu zaman ve enerji artımına rağmen paketlerin hepsinin hedef yönlendiricilere iletilebilmektedir. İki hatalı sistemlerde bu oran dört denektaşı göz önünde bulundurulduğunda ortalama %92,8 oranına kadar düşmektedir. Yonganın normal şartlarda bu maliyetli sistem yerine temel yönlendirmeleri kullanacağı unutulmamalıdır. Bu tarz bir yapı bize bir hata olması durumunda dış müdahale gerektirmeden yapının kullanılmasına ve yonga maliyetinin düşmesine olanak sağlamaktadır.

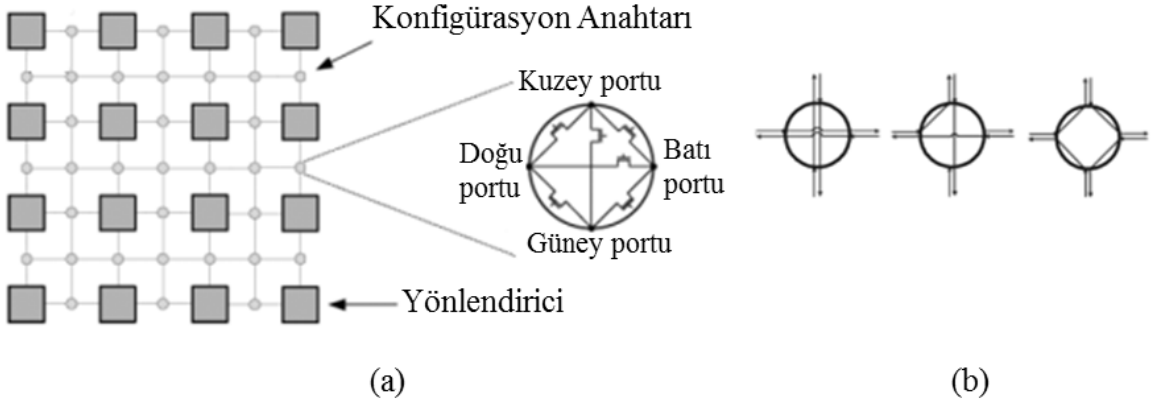
4. YENİDEN YAPILANDIRILABİLİR TOPOLOJİ TASARIMI

Örgü topolojiler, tekrar kullanılabilirlik, hata toleransı, kolaylık ve üretim avantajları nedeniyle en yaygın olarak kullanılan topoloji türüdür. Öte yandan, bazı iyi algoritmalar mevcut olmasına rağmen, bu topolojiler, ağ tıkanıklığına ve fazla enerji tüketimine neden olabilmektedirler. Bu yapılarda çalışmalar, tasarım süresi boyunca tek bir uygulamanın trafik düzenine dayanarak yonga-üstü-ağları optimize eder. Düğümleri eşler ve önceden tanımlanmış yapılandırma ayarlarını kullanarak aralarındaki yönlendirme yollarını belirlerler. Bununla birlikte, geleneksel ağlar çoklu uygulamaları desteklemek için iyi bir aday olmadığından, aynı ağ topolojisinde çalışan çoklu uygulamaları desteklemeye de ihtiyaç vardır. Bu nedenle, geleneksel ağların gecikme ve güç tüketimi bakımından sınırlamalarının üstesinden gelmek için yeniden yapılandırılabilir örgü topoloji yapısı kullanılmaktadır. Aşağıda detayları verilen bu yapıya uygulamaları eşleyen ve eşlenen düğümler arasında paketlerin izleyecekleri güzergahlara karar veren etkili yöntemlere ihtiyaç duyulmaktadır. Tezin bu kısmında bu yapı için sunulan iki farklı yöntemin detaylarına ve deneysel sonuçlarına yer verilmiştir.

4.1. Problem Tanımı

Şekil 4.1.'de [2] tarafından önerilen yeniden yapılandırılabilir örgü topoloji yapısı gösterilmektedir. Bu yapıda, geleneksel örgü topolojilerinden farklı olarak yönlendiriciler arasına fazladan konfigürasyon anahtarları yerleştirilir. Bu yapılar aynı anda çip üzerine birden fazla uygulamayı entegre etmeyi sağlar. Uygulamaların tüm düğümleri yapıyla eşleştirildikten sonra, anahtarlarda istenen transistörler etkinleştirilerek, sistemi çalışan uygulamaya göre yeniden yapılandırmak mümkündür. Şekil 4.1.b'de görüldüğü gibi, anahtarlarda istenen transistörleri aktive ederek, paketler herhangi bir yönde iletilebilir. Bu uzak yönlendiriciler arasında doğrudan bağlantı sağlar. Ayrıca, anahtarların enerji tüketimi yönlendiricilerden daha düşük olduğu için, sistemin toplam enerji tüketimi azalmaktadır. Şekil 4.1.b, bir anahtarın

alabileceği üç farklı yapılandırmayı göstermektedir. Örneğin, Şekil 4.1.a'daki düğümler arasındaki yapılandırma anahtarları Şekil 4.1.b'deki ilk konfigürasyondaki gibi yapılandırılırsa, sonuç topolojisi örgü topolojisi olacaktır. Kullanılan yapı aracılığıyla uygulamanın yapısına göre bir topoloji yaratmak mümkündür. Bu, performans ve enerji gibi kriterleri daha iyi optimize etmeyi sağlar. Yeniden yapılandırılabilir ağların, geleneksel olanlara göre dezavantajı, [2] numaralı çalışmada da ifade edildiği gibi kapladıkları alandır. 128 bitlik bir yeniden yapılandırılabilir YüA alanı, neredeyse 158 bitlik geleneksel bir yonga-üstü-ağ alanıyla aynıdır.

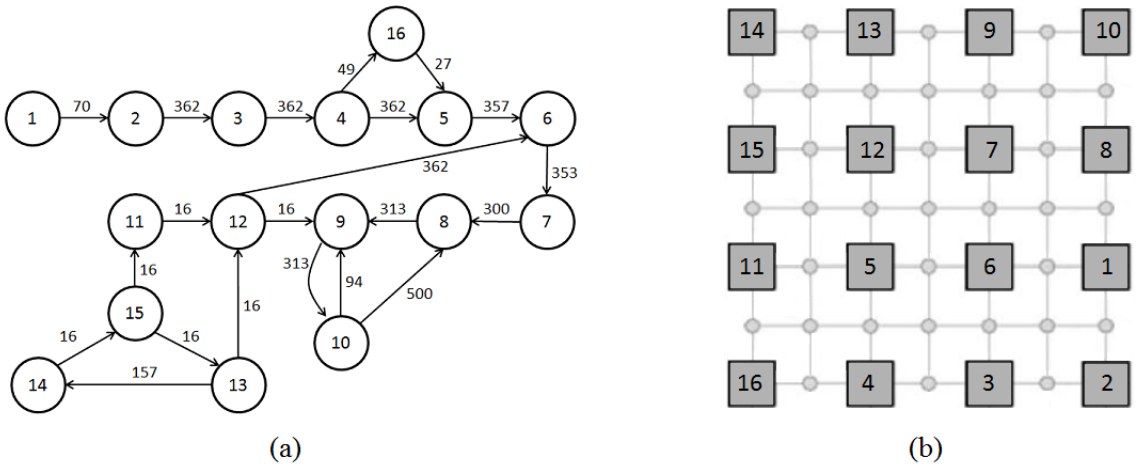


Şekil 4.1. Yeniden yapılandırılabilir örgü mimarisi ve (b) Anahtarların konfigürasyon örnekleri [2].

Şekil 4.2.b, Şekil 4.2.a'da verilen VOPD uygulaması için oluşturulmuş örnek bir yeniden yapılandırılabilir örgü yapısını göstermektedir. Bir uygulamanın akış grafiği U(D, B) ile temsil edilir. D uygulama düğümlerini ve B iletişim kuran uygulama düğümleri arasındaki bağlantı kümesini temsil eder. $d_i \in D$ ve $d_j \in D$ düğümleri arasındaki $b_{i,j}$ bağlantısındaki iletişim tutarı $t_{i,j}$ ile gösterilir. Tüm tanımları göz önünde bulundurarak, topolojinin toplam enerji tüketimini hesaplamak için Denklem 10'da verilen enerji modeli kullanılmıştır.

$$E_{\text{Toplam}} = \sum_{\forall b_{i,j} \in B} t_{i,j} * (\alpha_{i,j} E_{\text{anahtar}} + \beta_{i,j} E_{\text{yönlendirici}}) \tag{10}$$

Yukarıdaki denklemde, E_{anahtar} ve $E_{\text{yönlendirici}}$, sırasıyla bir anahtarın ve yönlendiricinin enerji tüketimini temsil eder ve α ve β, bitin geçtiği anahtar ve yönlendiricilerin sayısıdır. Bir



Şekil 4.2. (a) VOPD uygulamasının akış grafiği ve (b) yeniden yapılandırılabilir örgü yapısı üzerine eşleme örneği.

yapılandırma anahtarının güç tüketimi bir yönlendiriciden daha azdır. Bu çalışma kapsamında [2]'de önerildiği gibi bir yapılandırma anahtarının enerji tüketiminin 1 ve bir yönlendiricinin enerji tüketiminin 5 olduğu varsayılmıştır.

Bir topoloji üretildikten sonra, toplam enerji tüketimini hesaplayabilmek için yönlendiriciler arasındaki anahtarlar yapılandırılır. İlk olarak, topoloji üzerinde bir atlama uzaklığındaki yönlendiricilere eşlenmiş olan d_i ve d_j düğümleri arasındaki anahtarların konfigürasyonları belirlenir. Ardından kalan kenarlar, iletişim miktarına göre azalan sıraya göre sıralanır. En yüksek ağırlıklı kenardan başlayarak, sırasıyla bütün kenarlar için Denklem 10'da verilen toplam enerji tüketimini en aza indiren en kısa yol belirlenir. İletişim kuran herhangi iki düğüm çifti arasında yol bulunamamış ise bu yolun maliyeti sonsuz olarak belirlenir. Bu nedenle, ilgili topolojinin toplam enerji tüketimi de sonsuz olacak ve değerlendirmeye alınmayacaktır.

4.2. Uygulamaların Yeniden Yapılandırılabilir Örgü Topoloji Üzerine Eşlenmesi

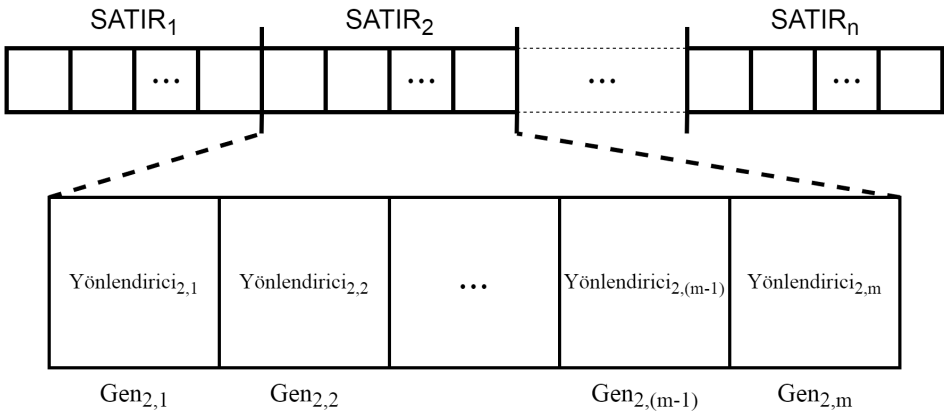
Bu çalışmada, uygulama düğümlerini yeniden yapılandırılabilir ağ yapısına eşleyen ve yönlendirme yollarını ayarlamak için anahtarları yapılandıran genetik algoritma tabanlı bir yöntem önerilmiştir. Sunulan algoritmanın amacı ağ üzerinde iletilen paketlerden kaynaklı Denklem 10 ile hesaplanan sistemin toplam enerji tüketimini en aza indirgemektir.

4.2.1. Sunulan Yaklaşım

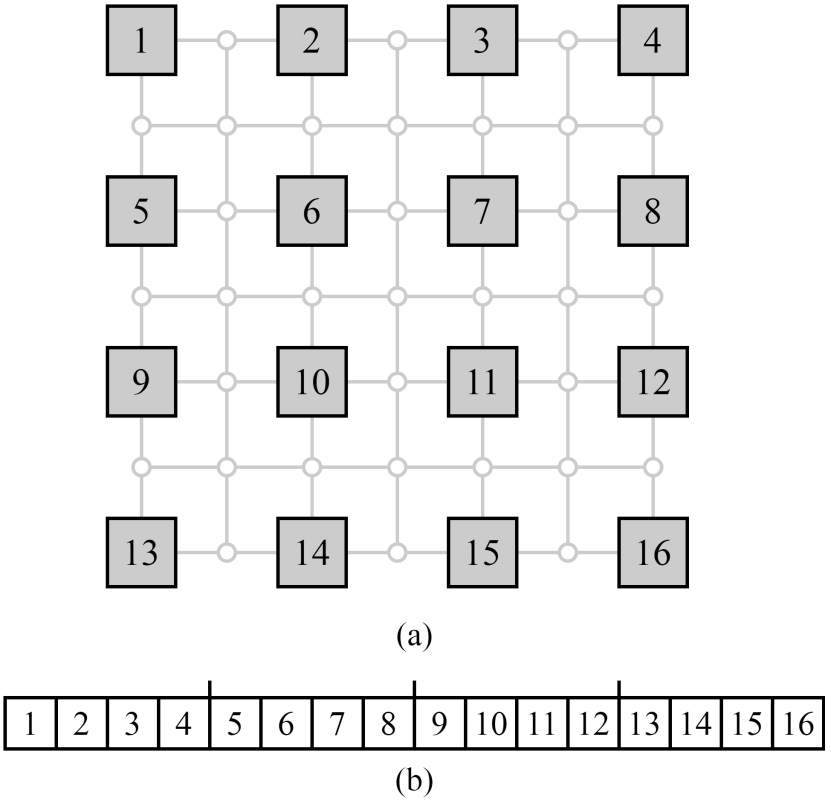
Sunulan algoritma genetik algoritmaya dayanmaktadır. Genetik algoritma bilgisayar ortamında evrimsel süreci simüle eder. Algoritma, yerel minimumdan kaçmak için arama alanında rastgele hareketler yapar ve en uygun çözümü bulmaya çalışır. Haritalama prosedürü çok zaman aldığından, genetik algoritma, problemin makul bir zamanda çözülmesi için seçilmiştir [39].

Genetik algoritmaya dayalı yöntemler üç ana aşamadan oluşur: ilk popülasyonun oluşturulması, genetik operatörlerin uygulanması ve yeni popülasyonun belirlenmesi. İlk aşamada, MARM-GA olarak adlandırılan yöntem, uygulama düğümlerini yeniden yapılandırılabilir örgü yapısı üzerine rastgele eşlemektedir. Daha sonra, her bir yinelemede daha iyi çözümler elde etmek ve nüfus çeşitliliğini arttırmak için genetik operatörler (çaprazlama ve mutasyon) uygulanmaktadır. Son aşamada, eski ve yeni oluşan nesiller birleştirilmiş ve en düşük enerji değerlerine sahip bireyler yeni bir nüfus olarak seçilmiştir.

Uygulama düğümlerinin eşlendiği her yeniden yapılandırılabilir ağ yapısı popülasyondaki bir bireyi gösterir. Bireyler kromozomlar ile temsil edilir. Kromozomlar genlerden oluşur. Şekil 4.3.'te, $n \times m$ boyutunda yeniden yapılandırılabilir bir örgü yapısı için kromozom temsilinin genel biçimi gösterilmektedir. Her bir gen, yapıdaki yönlendiricilere eşlenen uygulama düğümlerini belirtmektedir. İlk sıradaki yönlendiricileri ikinci sıradaki yönlendiriciler izler. Bir kromozom, $n \times m$ büyüklüğünde bir yapı için tam olarak $n \times m$ gen içerir.



Şekil 4.3. Kromozom gösterimi.



Şekil 4.4. (a) 16 tane eşlenmiş uygulama düğümü içeren örnek bir 4×4'lük yeniden yapılandırılabilir örgü topoloji ve (b) bu topolojinin kromozom temsili.

Boş yönlendiricilerin genleri -1 ile işaretlenmiştir. Şekil 4.4.a'daki 4×4 boyutundaki örnek yeniden yapılandırılabilir örgü yapısının kromozom temsili Şekil 4.4.b'de gösterilmektedir.

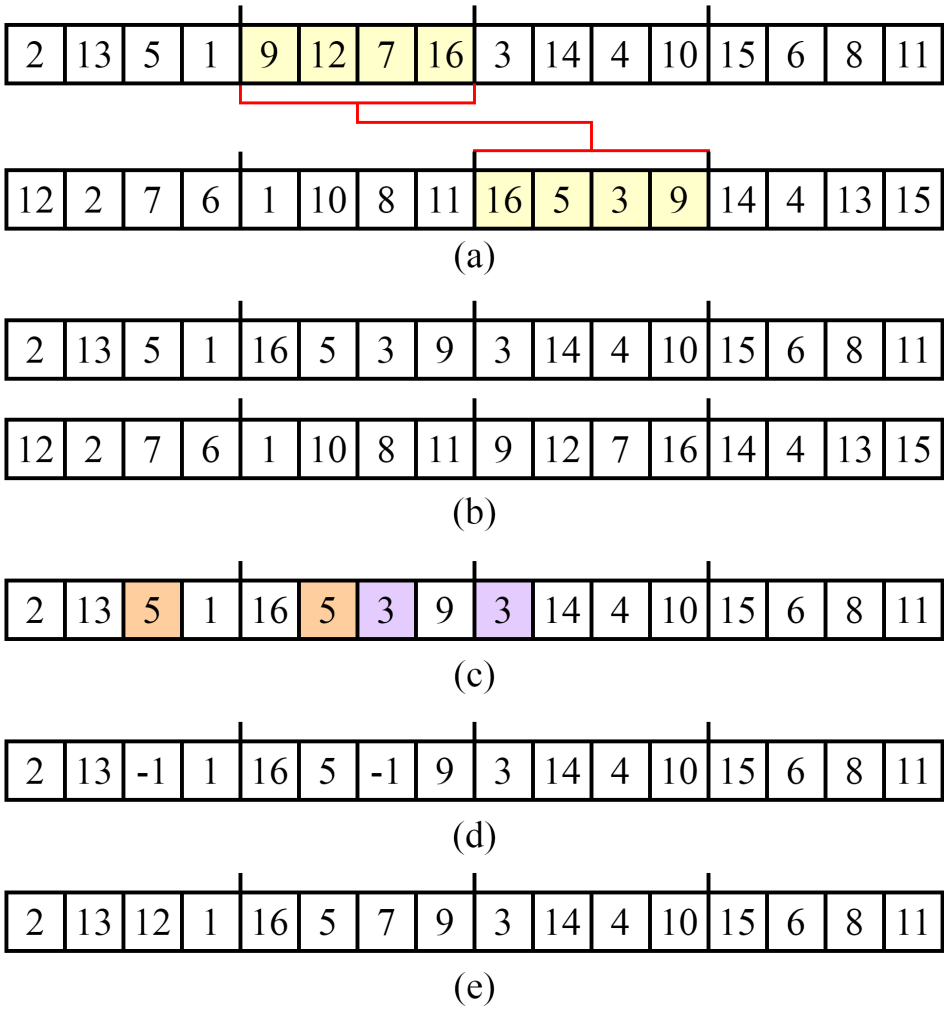
İlk nesil üretimi

MARM-GA'nın ilk adımı ilk neslin üretilmesidir. Popülasyondaki her bir kromozom, haritalanmış uygulama düğümlerine sahip yeniden yapılandırılabilir bir örgü yapısını temsil eder. Her kromozom, örgü yapısıyla aynı boyuttadır. İlk nesil, popülasyonun büyüklüğü P_{boyut} olana kadar her bir bireyin genlerine rastgele değerler atanarak oluşturulur.

Genetik operatörin (çaprazlama-mutasyon) uygulanması

İlk popülasyon oluşturulduktan sonra, algoritma, çeşitliliği arttırmak için popülasyondaki bireyler üzerinde genetik algoritma operatörlerini uygular. İlk olarak, nüfus içindeki bireyler Denklem 10 ile hesaplanan enerji değerlerine göre artan sırada sıralanmıştır. Daha sonra en düşük enerji değerine sahip olan P_{min} bireyler seçilir ve bu bireylere genetik operatörler uygulanır.

Çaprazlama işlemi için, P_{rast} çift birey P_{min} kümesinden rastgele seçilir. Daha sonra algoritma, bu bireylerin rastgele seçilen sıralarını alır ve bunları değiştirerek iki yeni birey oluşturur. Şekil 4.5.'te örnek çaprazlama işlemi verilmiştir. Öncelikle, MARM-GA rastgele seçtiği birinci ve ikinci bireyin, sırasıyla ikinci ve üçüncü sıralarını değiştirilmek üzere belirlemiştir (Şekil 4.5.a). Ardından, bu satırlardaki değerler değiştirilmiştir (Şekil 4.5.b). Geçerli bir bireyde uygulama düğümleri sadece bir gende bulunabilir. Ancak çaprazlama işleminden sonra, Şekil 4.5.b'de görüldüğü gibi yeni bireylerde geçersiz genler oluşmuştur. Bu tür genlerin onarılması gerekmektedir. Geçersiz genleri düzeltmek için iki adımlı bir prosedür izlenmektedir. 1) Aynı düğümleri içeren rastgele seçilmiş genlerden birisine -1 atama ve 2) Kromozomda bulunmayan uygulama düğümlerini rastgele seçilen boş genlere atama. Örneğin, Şekil 4.5.'teki ilk bireyde 3. ve 6. genlere beşinci düğüm (d_5) atanmıştır. Ayrıca, d_3 aynı anda hem 7. hemde 9. gende bulunmaktadır (Şekil 4.5.c). Bunu düzeltmek için: 1) Rastgele seçilen 3. ve 7. genlerin değerleri -1 olarak değiştirilmiş (Şekil 4.5.d) ve 2) d_7 ve d_{12}, sırasıyla 7. ve 3. genlere atanmıştır (Şekil 4.5.e).

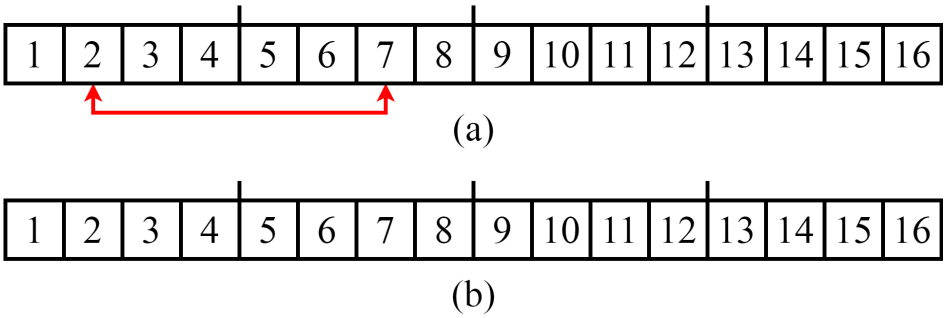


Şekil 4.5. Çaprazlama işlemi aşamaları. (a) Ebeveyn bireyler. (b) Çaprazlama işleminden sonra yeni oluşan bireyler. (c)-(e)Geçersiz genlerin düzeltilmesi.

Çaprazlama işleminden sonra P_{min} kümesinden rastgele seçilen P_{rast} birey üzerinde mutasyon işlemi gerçekleştirilir. Bunun için, MARM-GA, seçilen her bir bireyden rastgele seçilen iki değeri değiştirmektedir. Şekil 4.6.'da mutasyon işlemi örneği verilmiştir. Şekilde görüldüğü üzere, d_7 ve d_2 yer değiştirilmiş ve yeni bir birey oluşturulmuştur.

Yeni Nüfus Seçimi

Çaprazlama ve mutasyon işlemleri gerçekleştirildikten sonra, eski nesil ve yeni oluşmuş bireyler yeni bir listede birleştirilmektedir. Bunlar Denklem 10 ile hesaplanan enerji değerlerine göre sıralanmış ve listeden en düşük enerji değerine sahip P_{min} birey belirlenmiştir.



Şekil 4.6. Mutasyon işlemi örneği. (a) Eski nesil birey ve (b) mutasyon işleminden sonra oluşan yeni birey.

Seçilen en iyi bireyler, bir sonraki nesli üretmekte daha iyi çözümler elde etmeye yardımcı olmaktadır.

Önceden tanımlanmış yineleme sayısına ulaşana kadar popülasyon seçimi ve genetik operatör uygulamaları işlemleri tekrarlanır. Bu işlemler sonucunda en düşük enerji tüketimine sahip birey çözüm olarak değerlendirilmektedir.

4.2.2. Deneysel Sonuçlar

Bu bölümde, önerilen yöntemin performansını değerlendirmek için gerçekleştirilen deneyler ve sonuçları açıklanmıştır. Deneyler, Multi Window Display (MWD) [10], Video Object Plane Decoder (VOPD) [10], Depth Map Computation (DMC) [40] and multi-media system (MMS) [41] gibi çeşitli multimedya denektaşları üzerinde gerçekleştirilmiştir. Elde edilen sonuçlar toplam enerji tüketimi açısından, çalışmalarını bizim kullandığımız yeniden yapılandırılabilir örgü yapısı üzerinde gerçekleştiren [2] ve [3] numaralı çalışmalarda sunulan yöntemler ile karşılaştırılmıştır. Bu yöntemler ile doğru bir karşılaştırma yapmak için enerji hesaplamalarında [2] tarafından sunulan enerji modeli benimsenmiştir. Bu model göz önünde bulundurularak, bir yapılandırma anahtarı ve bir yönlendiricinin maliyetinin sırasıyla bir ve beş Joule birimi olduğu varsayılmıştır.

Hem [2] hem de [3] ilk önce uygulama düğümerini yeniden yapılandırılabilir örgü yapısı üzerine eşleme işlemlerini gerçekleştirmektedirler. [2] numaralı çalışmada sunulan eşleme

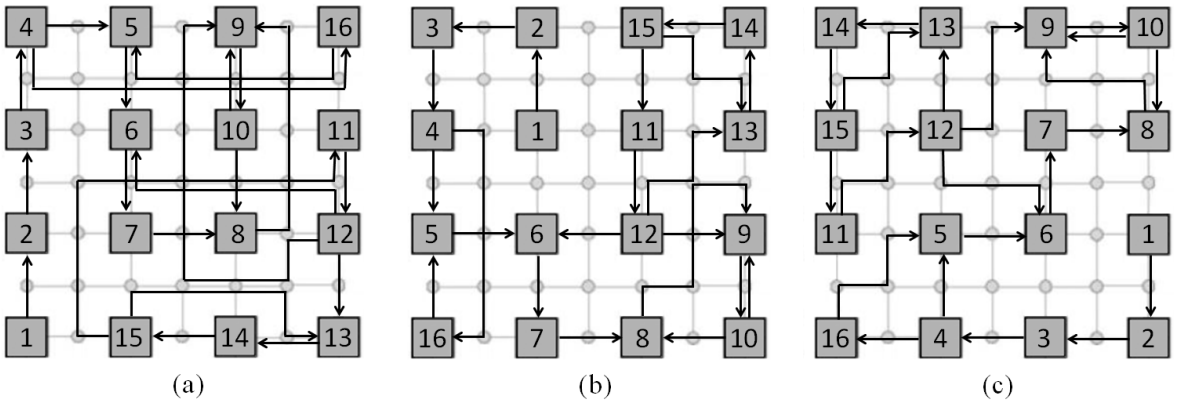
algoritması, diğer düğümlerle en fazla iletişim kuran düğümü seçmekte ve bunu yapı üzerinde en fazla komşuya sahip olan yönlendiricilerden birine yerleştirmektedir. Bu yinelemeli prosedür, tüm düğümler yapı üzerine yerleştirilene kadar devam eder. [3] numaralı çalışmada ise yazarlar, parçacık filtrelemesi tabanlı yeniden örnekleme algoritması kullanan bir eşleme yöntemi sunmuşlardır. İlk olarak, ilk partiküller olarak rastgele topolojiler oluşturmuşlar ve iyi partiküllerin sayısının artması için yeniden örneklemeye devam etmişlerdir. Bu iki yöntem, yeniden yapılandırılabilir anahtarları eşleştirme süreci bittikten sonra en kısa yol algoritmasına göre yapılandırmaktadırlar.

Genetik algoritmalar parametrelere karşı çok hassas algoritmalardır. Bu yüzden önerilen yöntem, en iyi algoritma parametrelerini belirlemek için popülasyon boyutu, yineleme sayısı, çapraz geçiş ve mutasyon oranlarının farklı değerleri üzerinde çalıştırılmıştır. Ardından, uygulamaların enerji tüketimini en aza indiren parametreler seçilmiştir. Deneyler kapsamında, ilk popülasyon için rastgele 1000 birey üretilmiştir ($P_{boyut} = 1000$). Ardından en düşük enerji değerine sahip ilk 200 bireye ($P_{min} = 200$) çaprazlama ve mutasyon işlemleri uygulanmıştır. MARM-GA, P_{min} kümesinden 50 çift bireyi rastgele seçmekte ($P_{rast} = 50$) ve çaprazlama işlemi uyglamaktadır. Aynı kümeden rastgele seçilen 50 bireye de mutasyon işlemi gerçekleştirilmiştir. Yineleme sayısı 1000'e ulaşana kadar tüm işlemler P_{min} birey üzerinde tekrarlanmıştır.

Çizelge 4.1. Yöntemimizin toplam enerji tüketimi açısından diğer yöntemler ile karşılaştırması.

Uygulama	Düğüm sayısı	Toplam Enerji Tüketimi (J)			Gelişim (%)	
		[2]	[3]	MARM-GA	[2]	[3]
MWD	12	1632	1504	1376	15,69 %	8,51 %
VOPD	16	5753	5243	4539	21,10 %	13,43 %
DMC	25	114858	59666	31305	72,74 %	47,53 %
MMS	12	1561372	944741	943007	39,60 %	0,18 %
				Ortalama:	**37,28 %**	**17,41 %**

Çizelge 4.1.'de MWD, VOPD, DMC ve MMS denektaşları için elde edilen enerji değerleri listelenmiştir. İlk iki sütunda, bu denektaşlarının isimleri ve düğüm sayıları verilmiştir. Sonraki üç sütunda, sırasıyla [2], [3] ve MARM-GA tarafından oluşturulan topolojilerin toplam enerji tüketimleri gösterilmiştir. Son iki sütun, sunulan yöntemin [2] ve [3]'e karşı iyileştirme oranlarını göstermektedir. Ortalama olarak, MARM-GA, enerji tüketimini



Şekil 4.7. (a) [2], (b) [3], ve (c) MARM-GA ile Şekil 4.2.'de verilen VOPD uygulaması için oluşturulan sonuç eşleme ve yönlendirme.

[2] numaralı çalışmaya karşı %37,28 oranında, [3] numaralı çalışmaya karşı ise %17,41 oranında azaltmıştır. MWD ve VOPD, daha az sayıda uygulama düğümüne sahip nispeten daha basit denektaşlarıdır. Çizelgede görüldüğü gibi, yöntemimizin enerji iyileştirmesi, düğüm sayısının DMC gibi büyük olduğu uygulamalarla artmaktadır.

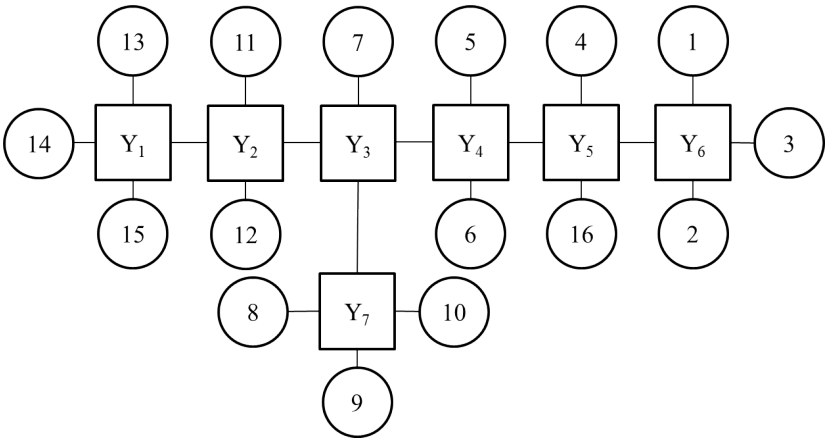
MMS denektaşı, H.263 Decoder (H.263 Dec.), H.263 Encoder (H.263 Enc.), MP3 Decoder (MP3 Dec.) ve MP3 Encoder (MP3 Enc.) olmak üzere dört uygulamadan oluşmaktadır. Bu dört uygulama aynı eşleme yapısını kullanır, ancak mimarinin yeniden yapılandırılabilirlik özelliğini kullanarak her uygulama için düğümler arasındaki yollar farklı şekilde konfigüre edilmektedir. Bu deney setinde, bireylerin enerji tüketimlerini hesaplamak için dört uygulamanın akış grafikleri kullanılmıştır. İlk olarak, her uygulama için anahtarlar ayrı ayrı yapılandırılmış ve her bir uygulama için bireyin toplam enerji tüketimi hesaplanmıştır. Daha sonra dört uygulamanın enerji değerlerinin toplamı, bireyin enerji tüketimi olarak belirlenmiştir. Çizelge 4.1.'de dört uygulamanın toplam enerji tüketimi verilmiştir. Metodumuzun enerji kazancı [3]'e karşı çok küçüktür (%0,18). Bununla birlikte, MARM-GA, toplam enerji tüketimini [2] numaralı çalışmaya karşı ortalama olarak %39,60 oranında düşürmektedir. MARM-GA, tüm denektaşları için bu yönteme kıyasla daha iyi enerji değerleri elde etmektedir.

Şekil 4.7.'de, VOPD uygulaması için oluşturulan topolojiler için [2], [3] ve MARM-GA tarafından önerilen yöntemlerle eşleme ve yönlendirme sonuçları gösterilmiştir. [2] ve [3]

numaralı çalışmalarda sunulan yöntemler, önce ağ yapısındaki uygulama düğümlerinin eşlenmesine karar verir. Optimum eşleşmeyi elde ettiklerinde anahtarları yapılandırırlar. Haritalama algoritmasındaki küçük bir gelişme, sistemin toplam enerji tüketiminde daha fazla kazanç sağlar. Bu yüzden [3] numaralı yöntem ile [2] numaralı yönteme göre daha düşük enerji değerleri elde edilmektedir. Diğer çalışmaların aksine, MARM-GA, eşleştirme ve yönlendirmeyi aynı anda belirler ve toplam enerji tüketimini optimize eder. Sunulan yöntemin diğer iki yöntemden daha iyi performans göstermesinin ana nedeni budur.

4.3. Uygulamaya Özgü Oluşturulan Topolojilerin Yeniden Yapılandırılabilir Örgü Topoloji Üzerine Eşlenmesi

Genetik algoritma tabanlı bir yöntem olan GATGA [9], uygulamaya özgü topolojilerin oluşturulması için kullanılır. Bu yöntemde her olası topoloji bir birey olarak kabul edilir ve her birey yönlendirici portlarından oluşur. Her adımda, algoritma mevcut popülasyondan bireyleri rastgele seçer ve optimal çözümü elde etmek için genetik operatörleri uygular. Şekil 4.8., Şekil 4.2.a'da akış çizgesi verilen VOPD uygulaması için bu yöntemle üretilen topoloji örneğini göstermektedir. Bu yöntem, bir uygulamanın belirli bir uygulama akış çizgesi için bir topoloji tasarlarken, minimum sayıda bileşen kullanarak enerji tüketimini en aza indirmeye odaklanmıştır. Bununla birlikte, hata toleransını dikkate almaz. Bir topolojinin hata kaldırabilir olması için iletişim kuran her uygulama düğümü arasında en az iki yol olması gerekmektedir. Böylece ana yol üzerinde bir hata olması durumunda paket alternatif yoldan iletilebilir. GATGA gibi sadece alan, maliyet performans gibi kriterler göz önünde bulundurularak oluşturulan topolojilerde sistem herhangi bir yol üzerinde bir hata olması durumunda çalışmaya devam edemez. Tezin bu kısmında hedeflenen geleneksel örgü yapılarının ölçeklenebilirlik ve yeniden kullanılabilirlik özellikleri ile uygulamaya özgü topolojilerin enerji ve maliyet optimizasyonu özelliklerini birleştirecek bir algoritma geliştirmektir. Bu amaçla iki aşamalı bir algoritma sunulmuştur: (1) uygulamaya özgü düzensiz topolojiyi yeniden yapılandırılabilir örgü topolojisine eşlemek ve (2) yönlendiriciler arasındaki anahtarların konfigürasyonunu gerçekleştirmek.



Şekil 4.8. VOPD uygulaması için GATGA yöntemi ile oluşturmuş düzensiz topoloji örneği.

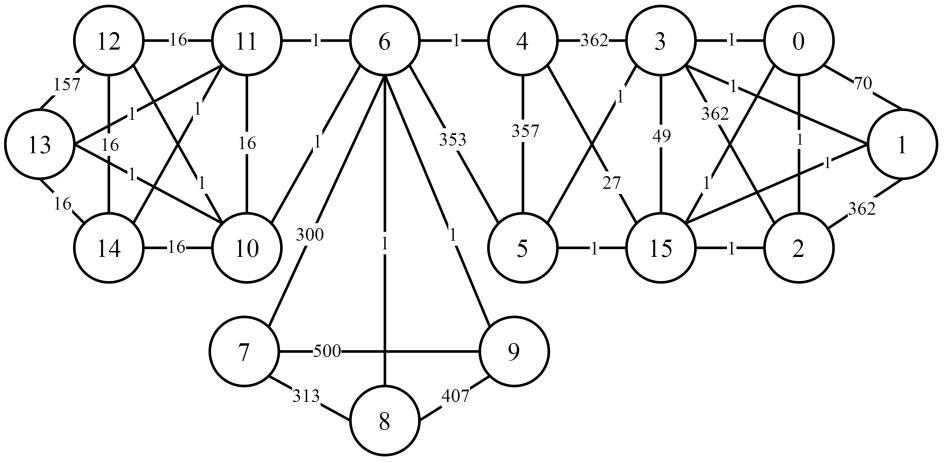
4.3.1. Sunulan Yaklaşım

Bu kısımda algoritmanın detayları GATGA yöntemi ile oluşturulmuş Şekil 4.8.'te verilen topoloji üzerinden anlatılmıştır.

Eşleme Algoritması

Şekil 4.8.'te verilen düzensiz topolojide, yüksek iletişimde olan düğümler birbirine yakın veya aynı yönlendiricilere eşlenmiştir. Enerji tüketimini azaltmak için benzer şekilde bu düğümlerin yeniden yapılandırılabilir örgü topoloji üzerinde de yakın yönlendiricilere eşlenmesi beklenmektedir. Düzensiz topolojide bir yönlendirici birden fazla düğüme sahip olabileceğinden, bunları doğrudan yeniden yapılandırılabilir ağa eşlemek kolay değildir. Bu nedenle, eşleme işlemi üç aşamaya bölünmüştür: 1) düzensiz topolojinin bağlantı yapısını yansıtan yeni bir grafik oluşturulması, 2) üretilen grafiğin maksimum yayılma ağacının belirlenmesi ve 3) ağacın yeniden yapılandırılabilir topolojiye eşlenmesi.

İlk adımda, uygulamaya özgü topoloji, düzensiz topolojideki düğümlerin aralarındaki mesafeleri yansıtan Düzensiz Bağlantı Çizgesi (DBÇ(D, K)) adı verilen yeni bir çizgeye çevrilmiştir. Düzensiz topolojide iki düğüm aynı yönlendiriciye veya komşu yönlendiricilere (yani bir atlama mesafesine sahip yönlendiriciler) atanmışlarsa, yeniden yapılandırılabilir ağdaki herhangi iki yönlendirici çifti arasındaki en kısa atlama mesafesi bir olduğu için DGB'de



Şekil 4.9. Şekil 4.8. için oluşturulmuş Düzensiz Bağlantı Çizgesi (DBÇ).

bu iki düğüm arasına bir kenar eklenmiştir. Hem uygulama akış çizgesinde hem de DBÇ'de iki düğüm arasında bir bağlantı bulunuyorsa, DBÇ'deki kenarın ağırlığı UAÇ'deki değer olarak belirlenir. Aksi takdirde, DBÇ'deki kenarın ağırlığı birdir. Örneğin, Şekil 4.9.'da verilen DBÇ'de, d_{12} ve d_{13} düğümleri arasında bir bağlantı oluşmuştur. Şekil 1.2.a'daki akış çizgesinde bu değer 157'dir. Dolayısı ile yeni çizgede de bu değer 157 olarak belirlenecektir. Yeni oluşan çizgedeki d_6 ve d_7 düğümleri arasında oluşan bağlantı, uygulama akış çizgesinde bulunmadığından yeni çizgede bu değer 1 olarak atanacaktır. DBÇ'de $k_{i,j}$ kenarının ağırlığı $a_{i,j}$ aşağıdaki formül ile belirlenmiştir:

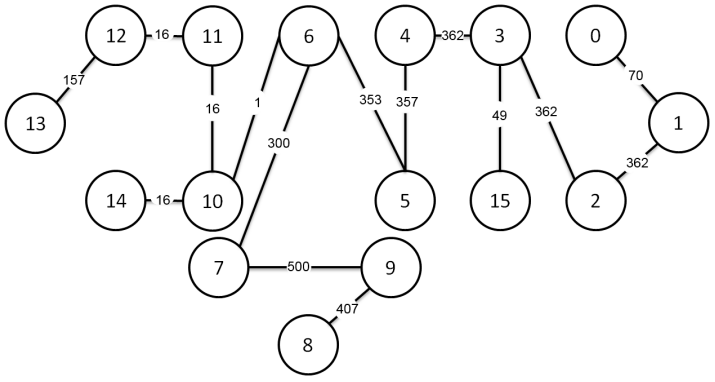
$$a_{i,j} = \begin{cases} 1 & \text{if } t_{i,j} = 0 \\ t_{i,j} & \text{if } t_{i,j} > 0 \end{cases} \tag{11}$$

Yeni oluşturulan çizge çok sayıda bağlantı içermektedir. Bu tarz bir yapıyı örgü yapısı üzerine eşlemek çok zordur. Bu yüzden bu çizge mümkün olduğunca sadeleştirilmelidir. Sadeleşme işlemi sonucunda her bir düğüm, çizgenin geri kalanına en az bir kenar ile bağlı olmalıdır. Yani bağlı çizge, sadeleşme işlemi sonucunda da bu özelliğini korumalıdır. Mümkün olduğunca yüksek ağırlıklı kenarların tutulması, yüksek iletişimli düğümler arasındaki bilginin korunmasını sağlar. Bu tarz bir çizgeden, her düğümü mutlaka içerecek, maksimum ağırlıklı bir ağaç çıkarmak için Kruskal algoritması modifiye edilmiştir. Bir çizgedeki en az

maliyetli örten ağacı bulan Kruskal algoritması, n sayıda düğüm içeren bir çizgede herhangi bir düğümden başlayarak ve her seferinde en kısa kenarı yapıya ekleyerek ağacı oluşturur. Maksimum örten ağacı bulmak için modifiye edilmiş Kruskal algoritması Algoritma 1'de görülmektedir. Şekil 4.9.'daki çizgenin bu algoritma ile üretilen maksimum ağırlıklı örten ağacı, $T(D, L)$, Şekil 4.10.'da verilmiştir.

Algoritma 1 Maksimum örten ağaç

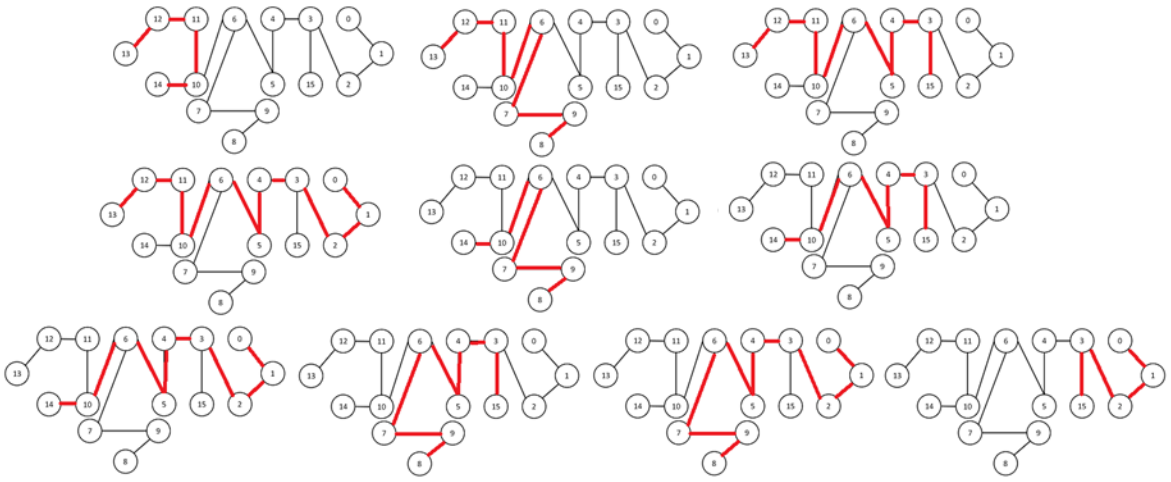
Girdi: N (Düğüm sayısı), K (Kenar kümesi)
Çıktı: T: Maksimum örten ağaç
  K kümesindeki kenarları ağırlıklarına göre azalan sırada sırala.
  while $|T| < |N| - 1$ **do**
    $(u, v) \leftarrow$ K kümesindeki sıradaki kenar
    if (u, v) döngüye neden olmuyorsa **then**
      $T = T \cup \{(u, v)\}$
    end if
  end while



Şekil 4.10. Şekil 4.9.'un Algoritma 1 ile üretilen maksimum ağırlıklı örten ağacı.

Ağacı elde ettikten sonra, algoritmanın son aşaması olarak, ağaç düğümleri yeniden yapılandırılabilir ağ yapısına eşlenmiştir. Aralarındaki iletişim miktarı fazla olan düğümleri yeniden yapılandırılabilir örgü yapısı üzerinde yakın konumlara eşlemek için bu düğümleri içeren ağaç dalları belli bir öncelikte eşlenmelidir. Bunun için, öncelikle T(D, L) kümesinin bütün uç düğümleri arasındaki tüm yolların belirlenmesi gerekir. Şekil 4.11.'de verilen bütün yolların toplam ağırlıkları hesaplandıktan sonra, yollar azalan sıra ile sıralanmıştır. Şekil 4.10.'da verilen T için, bu yollar Çizelge 4.2.'de iletişim maliyetlerine göre azalan sırayla listelenmiştir. Her yol $G_p(D_p, L_p)$ ile gösterilir. Burada $D_p \in D$, bu yoldaki düğümler

kümesidir ve $L_p \in L$, G_p yolundaki kenar kümesini temsil eder. $l_{i,j} \in L$ ise $d_i \in D$ ve $d_{i+1} \in D$ düğümleri arasındaki kenarı gösterir.



Şekil 4.11. Şekil 4.10. için uç düğümler arasındaki yollar.

Çizelge 4.2. Şekil 4.10.'da verilen maksimum ağırlıklı örten ağaçtaki uç düğümler arasındaki yollar ve ağırlıkları.

Yoldaki düğümler	Yolun toplam ağırlığı
0-1-2-3-4-5-6-7-9-8	3073
15-3-4-5-6-7-9-8	2328
0-1-2-3-4-5-6-10-11-12-13	2056
0-1-2-3-4-5-6-10-14	1883
13-12-11-10-6-7-9-8	1397
15-3-4-5-6-10-11-12-13	1311
14-10-6-7-9-8	1224
15-3-4-5-6-10-14	1138
0-1-2-3-15	843
13-12-11-10-14	205

Belirlenen yolları yeniden yapılandırılabilir örgü yapısına eşlerken ana mantık ağırlığı diğerlerinden yüksek olan $G_{max} \in G$ ile başlamak ve daha sonra kalan yolları yüksek iletişim kuran düğümleri birbirine yakın konumlanacak şekilde yerleştirmektir. Bir yol seçildikten ve yapıya eşlendikten sonra kalan yollar, bu yerleşen yolla kesişen düğümlere göre güncellenir. Seçili G_p yolundaki d_i ve d_j düğümleri yapıya eşlendiğinde, güncelleme işlemi $l_{i,j}$ diğer yollardan kaldırılarak gerçekleştirilmektedir. Örneğin, eşleme için n_0 ile başlayan ve d_8 ile biten ilk yolun seçildiğini varsayalım. d_0 ve d_1 eşlendikten sonra, bunlar diğer yollardan kaldırılmış ve iletişim maliyetleri güncellenmiştir. Çizelge 4.2.'deki yolların bu işlem

Çizelge 4.3. İlk güncelleme işleminden sonra yollar ve ağırlıkları.

Yoldaki düğümler	Yolun toplam ağırlığı
0-1-2-3-4-5-6-7-9-8	3073
15-3-4-5-6-7-9-8	2328
1-2-3-4-5-6-10-11-12-13	1986
1-2-3-4-5-6-10-14	1813
13-12-11-10-6-7-9-8	1397
15-3-4-5-6-10-11-12-13	1311
14-10-6-7-9-8	1224
15-3-4-5-6-10-14	1138
1-2-3-15	773
13-12-11-10-14	205

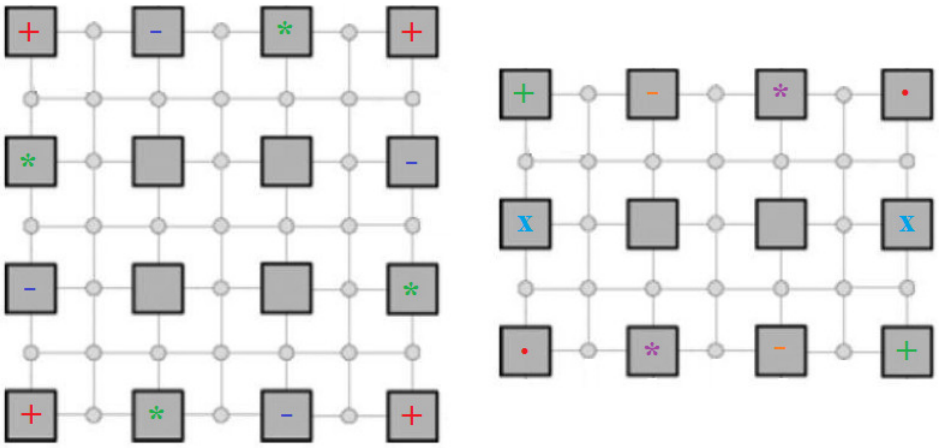
Çizelge 4.4. İlk yol yapıya eşlendikten sonra güncellenmiş yollar.

Yoldaki düğümler	Yolun toplam ağırlığı
6-10-11-12-13	190
3-15	49
6-10-14	17

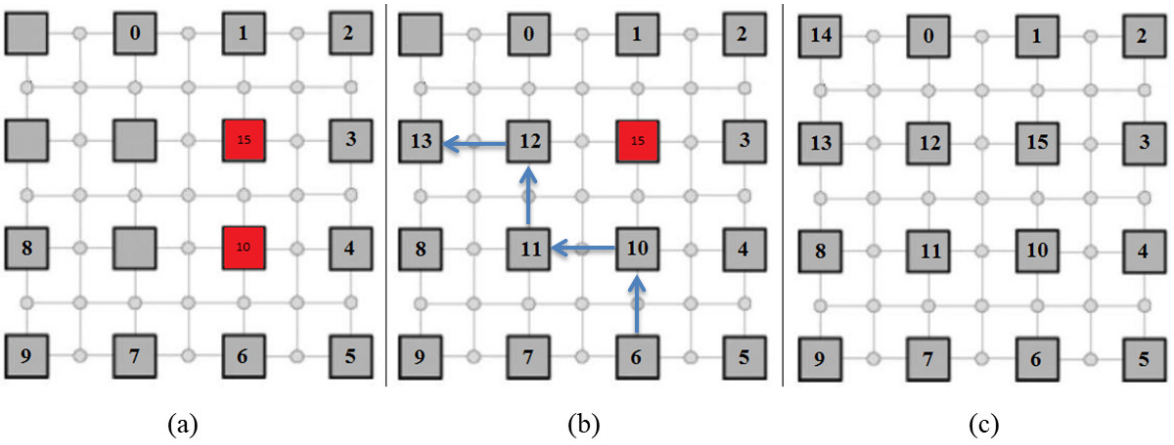
sonucunda güncellenmiş halleri Çizelge 4.3.'te verilmiştir. İlk yolun tüm düğümlerini yerleştirdikten sonra diğer yolların son durumu ise Çizelge 4.4.'te gösterilmiştir.

Eşleme prosedürü sırasında, bir diğer önemli karar ise ilk düğümün yeniden yapılandırılabilir yapı üzerine eşlenmesi için başlangıç yönlendiricinin belirlenmesidir. Bu işlem geri kalan düğümler için eşleme kararını ve sistemin toplam enerji tüketimin etkiler. Trafik yoğunluğu, genellikle ağ topolojilerinin merkezinde çok yüksektir. Trafiği ağ ağının dış yönlendiricilerine dağıtmak ve geri kalan düğümler için daha fazla alternatif yol sunabilmek için, dış şeritteki yönlendiricilerden başlamak tercih edilmiştir. Ancak, dış yönlendiriciler arasında başlangıç noktası olarak da farklı alternatifler bulunmaktadır. $a \times a$'lık bir örgü yapısı için, Şekil 4.12.a'daki yapıda görüldüğü gibi $a - 1$ farklı başlangıç noktası alternatifi bulunmaktadır. Bu şekilde simetriden dolayı aynı sonucu üretecek noktalar benzer işaretçiler ile eşlenmişlerdir.

$a \times b$'lik bir yapı için alternatif sayısı $(a-1)+(b-1)$'e eşittir. Şekil 4.12.b'de 3×4'lük bir yapı için başlangıç noktası örnekleri verilmiştir. Algoritma var olan bütün alternatif başlangıç noktaları ile eşleme işlemini gerçekleştirmekte daha sonra minimum enerji tüketimini sağlayan



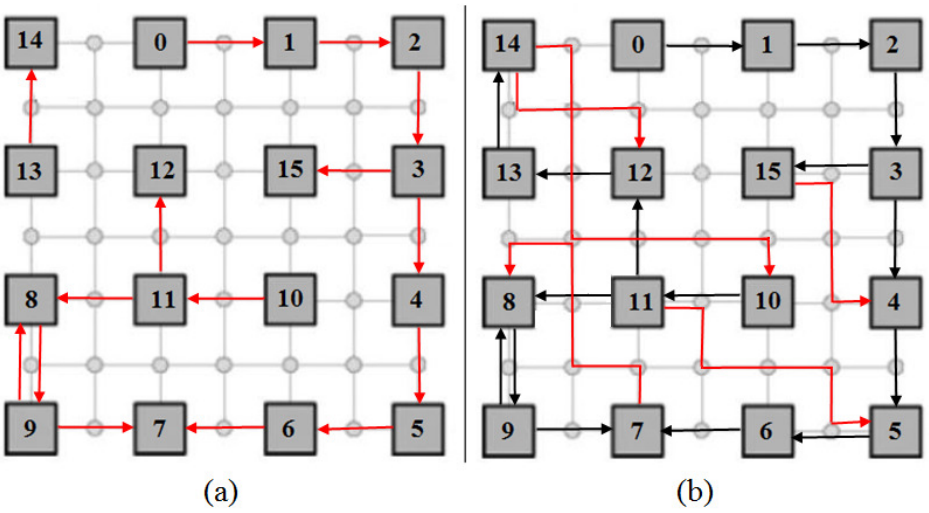
Şekil 4.12. (a) 4×4'lük, (b) 3×4'lük yeniden yapılandırılabilir örgü yapıları için başlangıç noktası alternatifleri.



Şekil 4.13. Örnek eşleme prosedürü. (a) En yüksek ağırlıklı yolun örgü topolojinin en dış şeridine yerleştirilmesi ve kalan yollar için bazı yönlendiricilerin işaretlenmesi. (b) Güncellenmiş yol çizelgesindeki kalan yolların eşlenmesi. (c) Sonuç eşleme.

yerleşimi seçmektedir.

Başlangıç noktasını seçtikten sonra, düğümler, bu noktadan başlayarak dış hattaki yönlendiricilere saat yönünde yerleştirilmiştir. Bu, düğüm sayısı, ağın dış kısmındaki yönlendirici sayısından az veya ona eşit olduğu sürece tüm komşu düğümlerin bir atlama mesafesinde eşlenmesini sağlar. Aksi takdirde, düğümler mümkün olduğunca yakın konuma yerleştirilir.



Şekil 4.14. (a) Bir atlama mesafesindeki iletişim kuran düğümler arasındaki anahtarların yapılandırılması. (b) Bütün anahtarların yapılandırılması ile oluşan sonuç yönlendirme.

İlk yol en dış şeride yerleştirilirken kalan her yol için en iyi başlangıç noktası işaretlenmektedir. Şekil 4.13.a'da, ilk yolun eşlenmesi işlemi ve kalan yollar için işaretlenmiş yönlendiricilerin durumu gösterilmiştir. Burada, diğer yollardaki d_{10} ve d_{15} düğümleri için, y_6 ve y_3 yönlendiricilerine en yakın boş yönlendiriciler işaretlenmiştir. İşaretli bu yönlendiriciler Çizelge 4.4. içindeki kalan yollar (yani, 190, 49 ve 17 ağırlıkları olan yollar) için en iyi adaylardır.

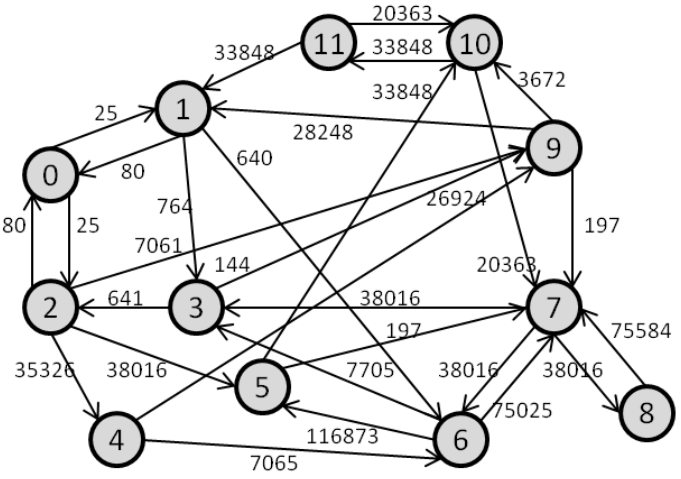
Daha sonra, bir sonraki yol, iletişim maliyeti göz önünde bulundurularak belirlenen en iyi yönlendirme noktasından başlayarak eşlenmiştir. Örneğimizde, bir sonraki yol Çizelge 4.4. içindeki ilk yoldur. Şekil 4.13.b'de bu yolun eşlenme işlemi gösterilmiştir. d_{12} düğümünü ilgili yönlendiriciye yerleştirdikten sonra, d_{13} düğümünü, d_{12} düğümünün doğusuna veya batısına eşleyebiliriz. Ancak, d_{12} düğümünün doğu yönlendiricisi daha önce işaretlenmiş olduğundan d_{13} düğümü için batı yönlendirici seçilmiştir. Bu erken işaretleme kararı ile, sonraki yolların da en iyi konumlarıyla eşleştirilebilmesi sağlanmaktadır. Tüm uygulama düğümleri eşleştirilene kadar haritalama prosedürü bu şekilde devam eder. Örnek uygulamamız için ortaya çıkan eşleme Şekil 4.13.c'de gösterilmiştir.

Anahtarların Yapılandırılması

Uygulama düğümleri ağ topolojisine eşlendikten sonra, düğümler arasındaki anahtarların yapılandırılması gerekir. Bu yapılandırma için iki adımlı bir prosedür izlenmiştir: 1) bir atlama mesafeli düğümler arasındaki anahtarları yapılandırma ve 2) kalan anahtarları yapılandırma. İlk adım olarak, bir atlama mesafede yani komşu yönlendiricilere eşlenmiş ve aralarında veri aktarımı bulunan düğümler belirlenmiş ve anahtarlar, doğrudan iletişime izin verecek şekilde yapılandırılmıştır. Bu düğümler veri paylaştıklarında yalnızca bir anahtar kullanırlar. Bu adımın sonucunu Şekil 4.14.a'da gösterilmiştir. Kalan iletişim düğümleri öncelikle iletişim ağırlıklarına göre sıralanır. Daha sonra, anahtarların maliyetinin yönlendiricilerden çok daha az olduğu göz önünde bulundurularak bu düğümler için sırasıyla en düşük maliyetli yollar belirlenmiştir. Deneylerde, bir anahtarın ve bir yönlendiricinin maliyetinin [2] tarafından önerildiği gibi sırasıyla bir ve beş olduğunu varsayılmıştır. Anahtar yapılandırmalarından sonraki yönlendirme sonucu Şekil 4.14.b'de verilmiştir.

4.3.2. Deneysel Sonuçlar

Tezin bu bölümünde, önerilen yöntemin performansını değerlendirmek için yapılan deneyler ve sonuçları açıklanmıştır. Deneyler, Multi Window Display (MWD) [10], Multi-Media System (MMS) [41], Video Object Plane Decoder (VOPD) [10] ve Depth Map Computation (DMC) [40] gibi bazı uygulama akış çizgeleri üzerinde gerçekleştirilmiştir. Sonuçlar, [2, 3] numaralı çalışmalardaki yöntemler ile karşılaştırılmıştır. Bu yöntemler YüA tasarımı için bizim de kullandığımız aynı yeniden yapılandırılabilir örgü topoloji mimarisini kullanmaktadırlar. Eşleme ve yönlendirme algoritmalarının başarısı sistemin toplam iletişim maliyetine ve ortalama atlama sayısına göre değerlendirilmiştir. Enerji hesabı için, [2] yöntemindeki enerji parametreleri kullanılmıştır. Buna göre bir bitin bir yapılandırma anahtarı üzerinden iletilmesinin maliyeti bir Joule iken bir yönlendirici üzerinden geçmesinin maliyeti beş Joule birimidir.



Şekil 4.15. Ortalama akış çizgesi [2].

Enerji Tüketimi Değerlendirilmesi

Çizelge 4.5. Enerji tüketimi açısından yöntemimiz ile [2] ve [3] yöntemlerinin karşılaştırması.

Uygulama	**Düğüm Sayısı**	**Toplam Enerji Tüketimi (J)**			**Enerji gelişimi(%)**	
		[2]	[3]	Sunulan Yöntem	[2]	[3]
MWD	12	1632	1504	1568	3,92	-4,26
MMS	12	1561372	944741	943141	44,79	0,17
VOPD	16	5753	5243	5229	9,11	0,27
DMC	25	114858	59666	50053	56,42	16,11

Çizelge 4.5.'te, MWD, MMS, VOPD ve DMC uygulamalarında yöntemimiz ile elde edilen enerji değerlerinin [2] ve [3] yöntemleri ile kıyaslaması listelenmiştir. Bu uygulamalarda düğüm sayıları sırasıyla 12, 12, 16 ve 25'tir. MMS, dört uygulamadan oluşur: H.263 Decoder (H.263 Dec.), H.263 Encoder (H.263 Enc.), MP3 Decoder (MP3 Dec.) ve MP3 Encoder (MP3 Enc.). Bu denektaşı için, dört uygulamanın düğümleri ağ yapısına bir kez eşleştirilir. Ardından, mimarinin yeniden yapılandırılabilirlik özelliği kullanılarak her uygulama için farklı yollar belirlenmiştir. Eşleştirme algoritması için, Şekil 4.15.'te verilen ve [2] tarafından önerilen birleşik çizge kullanılmıştır. Bu grafik, MMS karşılaştırması altında dört uygulamanın ortalama akış grafiği olarak oluşturulur. Bu yeni grafiğin her kenarı, uygulama düğümleri arasındaki toplam akışın ağırlıklı ortalaması alınarak hesaplanır. Bir uygulamada iki düğüm arasında akış yoksa, bu değerin sıfır olduğu varsayılır.

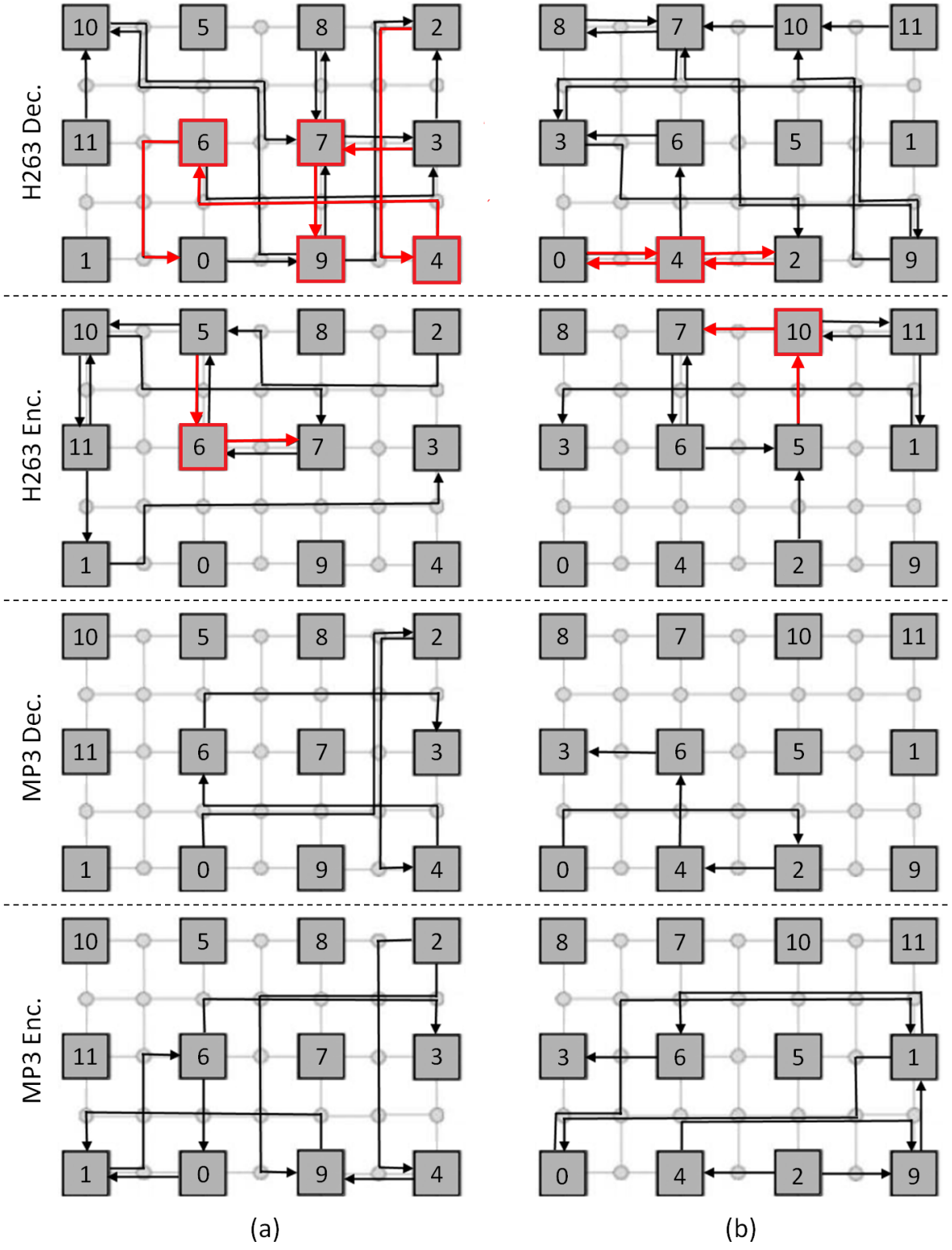
Çizelge 4.5. içinde son iki sütun yöntemin [2] ve [3]'e karşı gelişimini göstermektedir. Önerilen yöntem, enerji tüketimini [2] numaralı çalışmaya karşı %56,42'ye kadar ve [3] numaralı çalışmaya karşı %16,11 oranında kadar azaltır. Her bir uygulama için [3]'da sunulan yönteme karşı enerji gelişimine bakıldığında, uygulama düğümü sayısı yüksek olduğunda iyileşmenin de arttığı gözlemlenmektedir. Bu nedenle, sunulan yöntem çok sayıda uygulama düğümü olan uygulamalar için daha iyi sonuçlar vermektedir.

[2] içindeki yöntem bir kaba kuvvet algoritması olarak tasarlanmıştır ve zaman karmaşıklığı $\Omega(n^3)$ şeklindedir. Kullandığımız düzensiz topoloji oluşturma yönteminin (GATGA) ve [3] içindeki yöntemin eşleme algoritması zaman karmaşıklığı açısından eşittir. Her ikisi de popülasyon/parçacık büyüklüğüne ve yineleme sayısına bağlıdır. Sunulan algoritma uç düğümler arasındaki yolları hesaplamak için fazladan $O(n^2)$ zaman karmaşıklığı getirir; burada n uygulamadaki düğüm sayısıdır.

MMS denektaşındaki her bir uygulama için, Şekil 4.16.a ve Şekil 4.16.b'de sırasıyla [2] ve sunulan yöntem tarafından elde edilen sonuç eşleme ve yönlendirme sonuçları gösterilmiştir. Her iki yöntemde de, paketleri yönlendirmek için hem anahtarlar hem de yönlendiriciler kullanılır. Kırmızı çizgili yollar, bazı yönlendiricilerin paketlerin iletimi sırasında kullanıldığını gösterir. Bu şekilde görüldüğü gibi, [2] içindeki yöntem toplamda altı yönlendirici kullanırken, sunulan yöntem ile yönlendirme için sadece üç yönlendirici kullanılmıştır. Yöntemimiz, yönlendirme için kullanılan yönlendirici sayısını [2] ile karşılaştırıldığında %50 oranında azaltır. Bu yönlendirici kullanımının azaltılması, yönlendiricilerin enerji tüketimi anahtarlardan daha yüksek olduğu için toplam enerji tüketimini de azaltır.

Çizelge 4.6. Ortalama atlama sayısı açısından yöntemimiz ile [2] yönteminin MMS denektaşında karşılaştırılması.

MMS Uygulamaları	OAS Sunulan Yöntem	OAS [2]	OAS Gelişim (%)
H.263 Dec.	1,93	2,14	9,81
H.263 Enc.	1,27	1,64	22,56
MP3 Dec.	1,25	2,75	54,55
MP3 Enc.	2,00	1,75	-14,29
Ortalama	**1,61**	**2,07**	**18,16**



Şekil 4.16. (a) [2] ve (b) sunulan yöntem ile MMS denektaşı için elde edilen eşleme ve yönlendirme sonuçları.

Bir sonraki deneylerde, haritalama algoritması ortalama atlama sayısı (OAS) açısından değerlendirilmiştir. MMS denektaşı için sonuçlar [3] ile oldukça yakın olduğundan karşılaştırma işlemi sadece [2] numaralı çalışma ile gerçekleştirilmiştir. OAS hesaplamaları için, topolojinin yeniden yapılandırılabilir bir ağdan ziyade geleneksel bir ağ olduğu varsayılmıştır. OAS'yi hesaplamak için, önce her bir iletişim kuran düğüm için atlama sayıları toplanmış, ardından, bu toplam uygulamanın akış çizgesindeki toplam kenar sayısına bölünmüştür.

Sunulan eşleme yöntemi ve [2] tarafından önerilen yöntem ile elde edilen OAS sonuçları Çizelge 4.6.'da verilmiştir. MMS denektaşında sunulan yöntem ile %18,16 daha iyi OAS sonuçları elde edilmiştir. Çizelgede gösterildiği gibi, eşleme algoritmamız, negatif değerle gösterilen bir durum hariç bütün uygulamar için [2] numaralı çalışmadan daha iyi sonuçlar vermektedir. Dört uygulama da dikkate alındığında, en düşük iletişim miktarı 0 ve 1 düğümleri arasındadır. [2] numaralı çalışmada elde edilen sonuçta bu düğümler komşu yönlendiricilere yerleştirmektedir. MP3 Encoder uygulaması için, d_0 ve d_1 arasında bir iletişim bulunmaktadır. Yöntemimizin temel amacı enerji tüketimini azaltmak olduğundan, aralarındaki iletişim miktarı yüksek olan düğüm çiftleri daha yakın yönlendiricilere yerleştirilmiştir. Bu nedenle, d_0 ve d_1, yöntemimizle [2]'ye göre daha uzak yönlendiricilere eşlendiğinden MP3 Encoder denektaşı için OAS ölçütümüz [2] numaralı çalışmadan daha yüksektir.

5. TARTIŞMA

Yonga-üstü-Ağ mimarisi tasarım denemelerinin çoğunda bant genişliği, performans, enerji tüketimi, maliyet, yeniden kullanılabilirlik ve hataya dayanıklılık gibi etkileşimli kriterleri aynı anda karşılamak çok zordur. Bu parametrelerin bazıları, yeniden kullanılabilirlik ve hata tolerans özellikleri nedeniyle düzenli topolojiler tarafından kolayca optimize edilebilir ve karşılanabilir. Öte yandan, enerji tüketimi, performans ve yonga alanı gibi diğer parametreler düzensiz topolojilerde daha iyi optimize edilmektedir.

Tez çalışması kapsamında ilk olarak, Yonga-üstü-Ağ mimarileri için genetik algoritmaya dayalı bir uygulamaya özgü hataya dayanıklı topoloji üretme yöntemi sunulmuştur. Yöntem, iletilen paketlerin ortalama atlama sayısını ve sistemin toplam iletişim maliyetini azaltmak için hataya dayanıklı halka topolojilere minimum sayıda bağlantı ve yönlendirici eklemektedir. Topolojilerin çeşitliliğini artırmak için genetik algoritma operatörleri uygulanmıştır. Diğer uygulamaya özgü topoloji oluşturma yöntemlerinden farklı olarak, topolojideki her yönlendiricinin diğer yönlendiricilere en az iki bağlantısı olması garanti altına alınmıştır. Bu, topolojinin hataya dayanıklı olmasını sağlar. Önerilen yaklaşım, hataya dayanıklı topoloji üreten alternatif bir yöntem, hata kaldırma özelliği olmayan uygulamaya özgü topoloji üretme yöntemi ve halka topolojisi ile karşılaştırılmıştır. Sonuçlar, yöntemimizin çok sayıda düğümü olan uygulamalar için toplam enerji tüketimi, çalışma zamanı ve gecikme süresi bakımından diğer yöntemlerden daha iyi performans gösterdiğini göstermektedir.

Uygulamalar, uygulamaya özgü oluşturulan hataya dayanıklı topolojilere eşlenmekte, paketlerin yönlendirme bilgileri belirlenmekte ve sistem bu şekilde çalışmasını sürdürmektedir. Yonganın çalışma sürecinde paketlerin iletildiği yollardaki bir bağlantı üzerinde hata olması durumunda, paketler hedef yönlendiricilere doğru içerikle gönderilemeyecektir. Bu nedenle, sistemler paketin yönleneceği yolun hatalı olup olmadığını göz önüne alarak dinamik olarak paketi uygun çıkışa yönlendirecek bir birime ihtiyaç duymaktadırlar. Bu tez çalışması kapsamında tasarlanan kontrol birimi, paketlerin içeriğini kontrol etmekte ve hatanın geçici mi yoksa kalıcı mı olduğuna karar vermektedir. Yönlendirici, aldığı veride hata varsa, veriyi

gönderen yönlendiriciye tekrar göndermesi için sinyal verir. Eğer art arda hata olmaya devam ederse gönderilen yol üzerinde bir hata olduğu tespit edilir. Bu durumda yönlendiriciler yönlendirme tablolarını güncelleyerek yeni bir rota oluşturmaktadırlar. Hata tespit ünitesi ve paket yönlendirme ünitesi donanım olarak yönlendiriciye eklenirken yonganın alanını, enerji tüketimini, ve belli bağlantıların daha fazla kullanılma ihtimalini en az şekilde etkileyecek şekilde tasarlanmıştır.

Uygulamaya özgü oluşturulan düzensiz topolojilerin dezavantajı tek bir uygulama için optimize edildikleri için başka uygulamalarda kullanılamamasıdır. Düzenli topolojilerde farklı uygulamalar ağ bileşenlerine eşlenerek ağ tekrar tekrar kullanılabilir. Fakat her bir uygulamanın veri akış yapısı farklı olacağından her uygulama için yeni bir eşleme yapılmalıdır. Uygulamaların farklı yapıları örgü topoloji için uygun olmayabilir. Yani uygulamaların veri akışı ve bileşenlerinin birbirleriyle etkileşimi nedeniyle farklı topolojilerde daha olumlu sonuçlar alınmasına neden olabilir. Bu nedenle örgü topolojiye yapılan eşlemelerde performans gibi sağlanması gereken kısıtlar sağlanamayabilir veya enerji tüketimi gibi optimize edilmek istenen değerler istenen ölçüde optimize edilemeyebilir. Yeniden yapılandırılabilir örgü ağlarda, geleneksel örgü topolojierinden farklı olarak yönlendiriciler arasına anahtarlar yerleştirilmiştir. Bu anahtarların konfigürasyonu değiştirilerek, ağ çalışan uygulamaya göre yeniden yapılandırılabilir.

Tez kapsamında, bu yapı üzerinde gerçekleştirilen ilk çalışmada, uygulama düğümlerinin bu yapılara eşlenmesi ve paketlerin yönlendirilmesi problemlerine çözüm getiren genetik algoritma tabanlı bir yöntem önerilmiştir. Yöntemin amacı, ağ üzerinde, paketlerin anahtarlar ve yönlendiriciler aracılığı ile hedef yönlendiricilere iletilmesi sonucunda sistemin tükettiği toplam enerjiyi en aza indirgemektir. Önceki çalışmalardan farklı olarak, yöntem uygulama düğümlerinin yapıya eşlenmesi ve yönlendiriciler arasındaki anahtarların yapılandırılması işlemlerini aynı anda yapmaktadır. Önerilen yöntem, çeşitli denektaşlarında, aynı yeniden yapılandırılabilir yapıyı kullanan iki mevcut yöntemle karşılaştırılmıştır. Sonuçlar, yöntemin toplam enerji tüketimi açısından muadillerinden daha iyi performans gösterdiğini göstermektedir.

Son olarak tez çalışması kapsamında, örgü topolojisinin yeniden kullanılabilirlik ve ölçeklenebilirlik özelliklerini uygulamaya özgü düzensiz topolojinin enerji ve maliyet optimizasyon özellikleri ile birleştiren yeni bir yaklaşım önerilmiştir. Araştırmalarımız, bu yöntemin bu problem için ilk girişim olduğunu göstermektedir. Sunulan yöntem, uygulamaya özgü oluşturulan topolojilerdeki bilgiyi kullanarak uygulamaları yeniden yapılandırılabilir örgü topoloji üzerine eşler. Uygulamaya özgü topolojiler oluşturulurken enerji ve alan optimizasyonu en önemli kriterdir. Bu şekilde oluşturulan topolojilerde yüksek etkileşimli düğümler mümkünse aynı, değilse komşu yönlendiricilere konumlanırlar. Bu bilgi kullanılarak oluşturulan, uygulama düğümlerinden oluşan ağaç yeniden yapılandırılabilir örgü topoloji üzerine eşlendiğinde enerji minimizasyonu kriteri büyük ölçüde sağlanmaktadır. Sunulan yöntem ilk aşamada bu eşleme işlemini gerçekleştirmektedir. İkinci aşamada yapılandırma anahtarlarının durumlarına karar vermekte, yani; paketler için yönlendirmeleri belirlemektedir. Oluşturulan topolojilerin toplam enerji tüketimlerini, iletişim maliyetlerini ve paketlerin ortalama atlama sayılarını karşılaştırmak için gerçekleştirilen deneyler, yöntemin mevcut algoritmalara karşı üstünlüğünü göstermektedir.

Sunulan yaklaşımlarda düğümler arasındaki bağlantıların uzunluğu eşit olarak varsayılmıştır. Yonga üstüne donanımsal olarak eşlemelerde bu varsayım sağlanamayabilir. Bu nedenle tez kapsamında yapılabilecek iyileştirmelerden en önemlisi tasarımların donanımsal olarak gerçekleştirilmesi olmalıdır. Bu şekilde yöntemlerin karşılaştırılmasını sağlayacak daha gerçekçi bir enerji değerlendirmesi yapılması mümkün olabilir. Bunun yanında hata tespit ve yeniden yönlendirme biriminin eklenmesinin gerektirdiği donanımsal artışın maliyeti ile yeni bir yonga edinmenin maliyeti daha sayısal verilere dayandırılarak karşılaştırılabilir.

KAYNAKLAR

[1] K. Srinivasan, K. S. Chatha, G. Konjevod, Linear-programming-based techniques for synthesis of network-on-chip architectures, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 14 (4) (2006) 407–420.

[2] M. Modarressi, A. Tavakkol, H. Sarbazi-Azad, Application-aware topology reconfiguration for on-chip networks, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 19 (11) (2011) 2010–2022.

[3] S. Bayar, A. Yurdakul, An efficient mapping algorithm on 2-d mesh network-on-chip with reconfigurable switches, in: Design and Technology of Integrated Systems in Nanoscale Era (DTIS), 2016 International Conference on, IEEE, 2016, pp. 1–4.

[4] W. J. Dally, B. P. Towles, Principles and practices of interconnection networks, Elsevier, 2004.

[5] A. Jantsch, H. Tenhunen, et al., Networks on chip, Vol. 396, Springer, 2003.

[6] K. Srinivasan, K. S. Chatha, A technique for low energy mapping and routing in network-on-chip architectures, in: Proceedings of the 2005 international symposium on Low power electronics and design, ACM, 2005, pp. 387–392.

[7] J. Hu, R. Marculescu, Exploiting the routing flexibility for energy/performance aware mapping of regular noc architectures, in: Design, Automation and Test in Europe Conference and Exhibition, 2003, IEEE, 2003, pp. 688–693.

[8] S. Murali, G. De Micheli, Bandwidth-constrained mapping of cores onto noc architectures, in: Proceedings of the conference on Design, automation and test in Europe-Volume 2, IEEE Computer Society, 2004, p. 20896.

[9] S. Tosun, Y. Ar, S. Ozdemir, Application-specific topology generation algorithms for network-on-chip design, IET computers & digital techniques 6 (5) (2012) 318–333.

[10] K.-C. Chang, T.-F. Chen, Low-power algorithm for automatic topology generation for application-specific networks on chips, IET Computers & Digital Techniques 2 (3) (2008) 239–249.

[11] S. Tosun, V. B. Ajabshir, O. Mercanoglu, O. Ozturk, Fault-tolerant topology generation method for application-specific network-on-chips, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 34 (9) (2015) 1495–1508.

[12] M. Becker, W. Sarasureeporn, H. Szczerbicka, Comparison of bio-inspired and graph-theoretic algorithms for design of fault-tolerant networks, in: ICAS 2012, The Eighth International Conference on Autonomic and Autonomous Systems, 2012, pp. 1–7.

[13] M. Becker, M. Krömker, H. Szczerbicka, Evaluating heuristic optimization, bio-inspired and graph-theoretic algorithms for the generation of fault-tolerant graphs with minimal costs, in: Information Science and Applications, Springer, 2015, pp. 1033–1041.

[14] P. Shah, A. Kanniganti, J. Soumya, Fault-tolerant application specific network-on-chip design, in: Embedded Computing and System Design (ISED), 2017 7th International Symposium on, IEEE, 2017, pp. 1–5.

[15] Z. Li, J. Huang, Q. Xu, S. Chen, Integer linear programming based fault-tolerant topology synthesis for application-specific noc, in: ASIC (ASICON), 2017 IEEE 12th International Conference on, IEEE, 2017, pp. 96–99.

[16] J. Soumya, K. N. Babu, S. Chattopadhyay, Multi-application mapping onto a switch-based reconfigurable network-on-chip architecture, Journal of Circuits, Systems and Computers 26 (11) (2017) 1750174.

[17] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, D. Blaauw, A highly resilient routing algorithm for fault-tolerant nocs, in: Proceedings of the Conference

on Design, Automation and Test in Europe, European Design and Automation Association, 2009, pp. 21–26.

[18] M. E. Gomez, J. Duato, J. Flich, P. Lopez, A. Robles, N. A. Nordbotten, O. Lysne, T. Skeie, An efficient fault-tolerant routing methodology for meshes and tori, IEEE Computer Architecture Letters 3 (1) (2004) 3–3.

[19] C.-T. Ho, L. Stockmeyer, A new approach to fault-tolerant wormhole routing for mesh-connected parallel computers, IEEE Transactions on Computers 53 (4) (2004) 427–438.

[20] S. Rodrigo, J. Flich, J. Duato, M. Hummel, Efficient unicast and multicast support for cmps, in: Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture, IEEE Computer Society, 2008, pp. 364–375.

[21] J. Wu, A fault-tolerant and deadlock-free routing protocol in 2d meshes based on odd-even turn model, IEEE Transactions on Computers 52 (9) (2003) 1154–1169.

[22] K. Constantinides, S. Plaza, J. Blome, B. Zhang, V. Bertacco, S. Mahlke, T. Austin, M. Orshansky, Bulletproof: A defect-tolerant cmp switch architecture, in: High-Performance Computer Architecture, 2006. The Twelfth International Symposium on, IEEE, 2006, pp. 5–16.

[23] S.-J. Pan, K.-T. Cheng, A framework for system reliability analysis considering both system error tolerance and component test quality, in: Proceedings of the conference on Design, automation and test in Europe, EDA Consortium, 2007, pp. 1581–1586.

[24] V. B. Ajabshir, S. Tosun, Fault-tolerant routing for irregular-topology-based network-on-chips, in: 2014 Second International Symposium on Computing and Networking, IEEE, 2014, pp. 123–129.

[25] W. J. Dally, B. Towles, Route packets, not wires: On-chip interconnection networks, in: Design Automation Conference, 2001. Proceedings, IEEE, 2001, pp. 684–689.

[26] L. Benini, G. De Micheli, Networks on chips: A new soc paradigm, computer 35 (1) (2002) 70–78.

[27] M. Janidarmian, A. Khademzadeh, M. Tavanpour, Onyx: A new heuristic bandwidth-constrained mapping of cores onto tile-based network on chip, IEICE Electronics Express 6 (1) (2009) 1–7.

[28] P. K. Sahu, S. Chattopadhyay, A survey on application mapping strategies for network-on-chip design, Journal of Systems Architecture 59 (1) (2013) 60–76.

[29] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, D. Sylvester, Vicis: A reliable network for unreliable silicon, in: Proceedings of the 46th Annual Design Automation Conference, ACM, 2009, pp. 812–817.

[30] A. Vitkovskiy, V. Soteriou, C. Nicopoulos, A dynamically adjusting gracefully degrading link-level fault-tolerant mechanism for nocs, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 31 (8) (2012) 1235–1248.

[31] K. Aisopos, A. DeOrio, L.-S. Peh, V. Bertacco, Ariadne: Agnostic reconfiguration in a disconnected network environment, in: Parallel Architectures and Compilation Techniques (PACT), 2011 International Conference on, IEEE, 2011, pp. 298–309.

[32] R. Parikh, V. Bertacco, udirec: unified diagnosis and reconfiguration for frugal bypass of noc faults, in: Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture, ACM, 2013, pp. 148–159.

[33] D. DiTomaso, A. Kodi, A. Louri, Qore: A fault tolerant network-on-chip architecture with power-efficient quad-function channel (qfc) buffers, in: High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on, IEEE, 2014, pp. 320–331.

[34] S. P. Azad, B. Niazmand, J. Raik, G. Jervan, T. Hollstein, Holistic approach for fault-tolerant network-on-chip based many-core systems, arXiv preprint arXiv:1601.07089 (2016).

[35] A. Pullini, F. Angiolini, P. Meloni, D. Atienza, S. Murali, L. Raffo, G. De Micheli, L. Benini, Noc design and implementation in 65nm technology, in: Networks-on-Chip, 2007. NOCS 2007. First International Symposium on, IEEE, 2007, pp. 273–282.

[36] H. Kim, P. Ghoshal, B. Grot, P. V. Gratz, D. A. Jiménez, Reducing network-on-chip energy consumption through spatial locality speculation, in: Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip, ACM, 2011, pp. 233–240.

[37] W. W. Peterson, D. T. Brown, Cyclic codes for error detection, Proceedings of the IRE 49 (1) (1961) 228–235.

[38] G. Leary, K. Srinivasan, K. Mehta, K. S. Chatha, Design of network-on-chip architectures with a genetic algorithm-based technique, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 17 (5) (2009) 674–687.

[39] P. Kullu, S. Tosun, Energy-aware and fault-tolerant custom topology design method for network-on-chips, Nano Communication Networks 19 (2019) 54–66.

[40] T. Schönwald, A. Viehl, O. Bringmann, W. Rosenstiel, Distance-constrained force-directed process mapping for mpsoc architectures, in: Digital System Design (DSD), 2012 15th Euromicro Conference on, IEEE, 2012, pp. 592–599.

[41] J. Hu, R. Marculescu, Energy-and performance-aware mapping for regular noc architectures, IEEE Transactions on computer-aided design of integrated circuits and systems 24 (4) (2005) 551–562.