Güvenilirlik Odaklı Tümleşik Sistem Tasarım Yöntemi

Reliability Oriented Embedded System Design Method

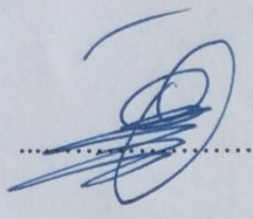
TOHİD TAGHİZAD GOGJEH YARAN

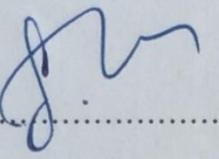
Doç. Dr. SÜLEYMAN TOSUN
Tez Danışmanı

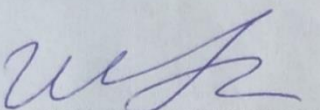
Hacettepe Üniversitesi
Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin
Bilgisayar Mühendisliği Anabilim Dalı için Öngördüğü
YÜKSEK LİSANS TEZİ olarak hazırlanmıştır.

2017

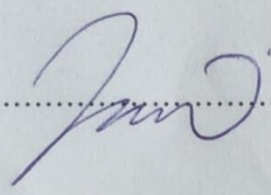
Tohid TAGHİZAD GOGJEH YARAN' in hazırladığı **"Güvenilirlik Odaklı Tümleşik Sistem Tasarım Yöntemi"** adlı bu çalışma aşağıdaki jüri taraflından **BILGİ-SAYAR MÜHENDISLİĞİ ANABILIM DALI'** nda **YÜLSEK LISANS TEZİ** olarak kabul edilmiştir.

Doç. Dr. Özcan ÖZTÜRK
Başkan ..... 

Doç. Dr. Süleyman TOSUN
Danışman ..... 

Prof. Dr. Şahin EMRAH
Üye ..... 

Doç. Dr. Sevil ŞEN
Üye .....

Yrd. Doç. Dr. Adnan ÖZSOY
Üye ..... 

Bu tez Hacettepe Üniversitesi Fen Bilimleri Enstitüsü tarafından YÜKSEK LİSANS TEZİ olarak onaylanmıştır.

Prof. Dr. Menemşe GÜMÜŞDERELİOĞLU

Fen Bilimleri Enstitüsü Müdürü

Aileme ve kız arkadaşıma

YAYINLAMA VE FİKRİ MÜLKİYET HAKLARI BEYANI

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

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

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

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

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

☑ **Serbest Seçenek/Yazarın Seçimi**

08/06/2017

(İmza)

Öğrencinin Adı Soyadı

Tahir Taghizad

ETİK

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

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

beyan ederim.

01/04/2017

Tohid TAGHİZAD GOGJEH YARAN

ÖZET

Güvenilirlik Odaklı Tümleşik Sistem Tasarım Yöntemi

Tohid TAGHİZAD GOGJEH YARAN

Yüksek Lisans, Bilgisayar Mülendisliği Bölümü

Tez Danışmanı: Doç. Dr. Süleyman TOSUN

Mayıs 2017, 59 sayfa

Her bir CMOS teknolojisinin üretim neslinde, kombinasyonel devreler yumuşak hatalara karşı daha hassas duruma gelmektedir. Daha önceki çalışmaların birçoğunda devreleri hatalara karşı sağlamlaştırmak için donanım yedekleme kullanılmıştır. Ancak donanım yedeklemede daha fazla alan ve güç kullanılmaktadır. Ayrıca tasarım kısıtlamaları, yedekleme kaynakların son devrelere eklenmelerine izin vermeyebilir. Bu nedenlerden dolayı, bu tezde kombinasyonel devrelerin güvenirliğini artırmak için genetik algoritma temelli bir yöntem önerilmektedir. Bu yöntemde her bir kaynağın farklı versiyonları kullanılmaktadır ve her bir versiyon farklı alan, gecikme ve güvenirlik değerlerine sahiptir. Bu yöntemin asıl amacı, alan ve gecikme kısıtlamalarının altında, son tasarım güvenirliğini artırmak için en iyi kaynakların kullanılmasını sağlamaktır. Bu tezdeki denemelerin sonuçlarına bakıldığında tezde uygulanan yöntem hiçbir ekstra alan kullanmadan sezgisel yöntemlere göre 19.90% (Ortalama 14.50%) güvenirliği artırmaktadır.

Anahtar kelimeler: CMOS, yumuşak hata, kombinasyonel devre, tasarım kısıtları, genetik algoritma.

ABSTRACT

Reliability Oriented Embedded System Design Method

Tohid TAGHİZAD GOGJEH YARAN

Master of Science; Department of Computer Engineering

Supervisor: Doç. Dr. Süleyman TOSUN

May 2017, 59 pages

Combinational circuits have become more vulnerable to soft errors (SEs) in each CMOS technology generation. Most of the prior studies use hardware redundancy in an attempt to harden the circuits against errors. However, redundancy increases the area and power consumption. Furthermore, the design constraints may not allow adding redundant resources to the final circuit. In this paper, we present a genetic algorithm (GA)-based design method to increase the reliability of combinational circuits. In this method, we use different versions of the same resources, each having different area, latency, and reliability values. The goal of GA-based optimizer is to allocate the best available resources to the application nodes to maximize the reliability of the design under tight area and latency constraints. Our experimental results show that we achieve up to 19.90% (14.50% on average) reliability improvement against a heuristic method with no additional area overhead.

Key words: CMOS, soft errors, combinational circuit, design constraints, genetic algorithm.

Teşekkür

En başta danışmanım Prof. Dr. Süleyman TOSUN'a çalışmalarım sırasındaki desteği, yönlendirmesi ve sabrı için teşekkürü bir borç bilirim. Kendisi çalışmamın her aşamasında desteğiyle ve rehberliğiyle yanımda olmuş çalışmamın tamamlanmasına büyük katkıda bulunmuştur.

Çalışmam süresince bana yeterli zamanı tanıyıp anlayış gösteren Tiga Bilişim ailesine bilhassa Hamit YAŞASIN, Gürkan CANER ve Zehra BOZDOĞAN'a en içten teşekkürlerimi sunarım.

Bu süreç boyunca aynı zamanda destekleriyle yanımda olan aileme, kız arkadaşım Ümran KÖKLÜ'ye ve son düzeltmelerdeki yardımlarından dolayı Yavuz AKSOY'a teşekkürlerimi sunarım.

Bu tez çalışması Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) tarafından desteklenen 116E095 numaralı 1001 projesi kapsamında yapılmıştır.

İÇİNDEKİLER

Sayfa

Çizelgeler Dizini

Sayfa

Şekiller Dizini

Sayfa

Kısaltmalar Dizini

Kısaltmalar

CMOS	Bütünleyici Metal Oksit Yarı İletken (complementary metal oxide semiconductor)
YSP	Yüksek Seviye Programlama
YSS	Yüksek Seviye Sentezleme
VAÇ	Veri Akış Çizelgesi
TDP	Tamsayılı Doğrusal Programlama
PBİ	Paylaşımlı bellek işlem
HBD	Hata Bulma ve Düzeltme
THD	Tek Hata Düzeltme
ÇHB	Çift Hata Bulma
TBHD	Tek Bayt Hata Düzeltme
ÇBHB	Çift Bayt Hata Bulma
ÇHD	Çift bit Hata Düzeltme
ÜHB	Üçlü bit Hata Bulma
TOG	Tek Olaylı Geçiş
TOB	Tek Olaylı Bozulma
HE	Hızlı Ekstrasiyon
ÜMY	Üçlü Modular Yedekleme
SD	Saat Döngüsü

1. GİRİŞ

Günümüzde transistör boyutlarının nano seviyesine ulaşması sonucu bir yonga üstüne milyonlarca transistör yerleştirilebilmektedir. Bu sayede birçok fonksiyon bir yonga üstünde gerçeklenebilmektedir. Yüzlerce bileşenden oluşan bir sistemi alan, performans, enerji verimliliği gibi kıstaslar altında bir yonga üstüne gerçeklemek bir tasarımcı için neredeyse imkansızdır. Bu nedenle otomasyon için kullanılan yazılımlar geliştirilmiştir. Bu yazılımlar genellikle sistem tasarımını mimari seviyesinden değil soyut seviyeyi üst boyutlara taşıyarak mümkün olan en üst seviyeden sentezleme işlemine başlarlar. Bunun nedeni, alt seviyelere inildikçe karmaşıklığın artması, optimize edilecek değişkenlerin istenen sınırlara çekilememe riskinin fazla olmasıdır.

Donanım dilleri ile tanımlanan soyut fonksiyonların ilk kez mimari görünüm kazandırıldığı sentezleme aşaması yüksek seviyede sentezlemedir. Bu seviyede genellikle yonga alanı, sistem performansı, enerji tüketimi gibi kıstas ve/veya optimize edilmesi gereken değişkenler vardır. Teknoloji boyutlarının küçülmesi sonucu, yonga üstündeki geçici hatalar yonga üstündeki ısınma problemleri ve çevresel faktörlere bağlı olarak artmaktadır. Bu nedenle sistemin hatasız, güvenilir bir şekilde çalışması için yüksek seviyede sentezleme işlemi güvenilirlik odaklı yapılmaya başlanmıştır.

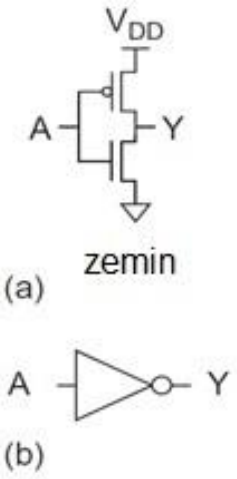
Sistemin güvenilirliği artıracak her bir yeni yöntem, alan, performans ve/veya enerji kıstaslarını olumsuz etkilemektedir. Bu nedenle verilen kıstaslar altında güvenilirliği artırmak bir optimizasyon problemidir. Bu problemin çözümü için daha önce sunulan doğrusal programlama yöntemlerinin çözüm süresi çok fazla iken sezgisel yöntemler her zaman istenen sonucu vermeyebilir. Bu tezde bu problemi çözmek için genetik algoritma tabanlı bir yöntem sunulmuştur.

1.1.CMOS

1970'ten beri bilim insanları donanım cihazlarını daha küçük hale getirmek için çalışmalar yapmaktadır. Donanım cihazlarını küçültme ve performanslarını artırma çabaları bitmez bir şekilde küresel rekabet haline dönüşmüştür. Bu rekabetin başlangıcı sayılabilecek CMOS (Bütünleyici Metal Oksit Yarı İletken) ilk olarak 1963 yılında Frank Walness tarafından icat edilmiş ve günümüze kadar üzerinde çok sayıda çalışmalar yapılmıştır. CMOS ilk başta COS-MOS ismiyle ortaya

çıkmış olup daha sonra Fairchild isimli devrelerde N-tip ve P-tip transistörleri bünyesinde bulundurarak CMOS ismini almıştır. [1]

Şekil 1.1'de CMOS teknolojinin bir NOT devre'de kullanılması gösterilmektedir. Eğer A girişi 0 olursa nMOS transistör kapalı ve pMOS transistör açık durumda olmakta ve Y çıkışı 1 değeri almaktadır, çünkü bu durumda VDD'ye bağlanmış olup ve zeminden bağlantısı kesilmekte, aynı şekilde eğer A girişi 1 değere sahipse nMOS transistör açık ve pMOS transistör kapalı durumda olmakta ve Y çıkışı zemine bağlanarak 0 değeri almaktadır.



Şekil 1.1. Not devre üzerinde kullanılan CMOS teknolojisi.

Sağlam bir CMOS teknoloji elde etmek için hız, devre yoğunluğu , güç tüketimi ve maliyet özelliklerini göz önünde bulundurmak gerekmektedir. CMOS elemanı daha çok entegre devrelerde kullanılmakta olup Statik-RAM , mikrodenetleyici , mikroişlemci ve bir çok dijital mantıksal devrelerde de kullanılabilmektedir. CMOS'un devre açık durumunda işlem yapmadığı sırada neredeyse hiç güç tüketmemesi en önemli özelliklerinden birisi sayılmakta ve bu sayede bu teknolojiyi kullanan elektronik cihazların pil ömrü artmaktadır [2]. Örnek olarak dijital fotoğraf makinelerinde daha az ısınma ve fazla pil ömrünün sağlanması için CMOS daha fazla tercih edilmektedir.

1.2. Yumuşak Hata

Elektronik ortamlarda ve bilgisayar hafıza sisteminde yumuşak hata, program komut değerleri ve verileri bozmaya neden olur. Yumuşak hatalayı düzeltmenin en kolay yolu bilgisayar veya elektronik cihazı kapatıp tekrar çalıştırmak veya işlemi

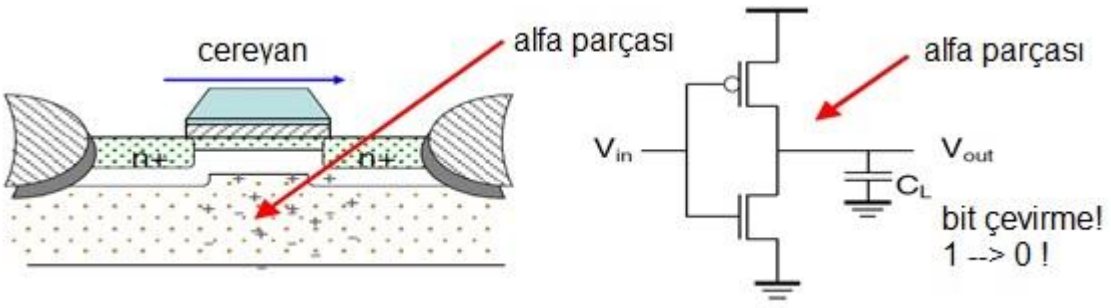
yeniden başlatmaktırdır. Yumuşak hata genelde sistem donanımına hasar vermez, sadece işlem görecek olan verileri değiştirir böylece eğer bu hatalar tespit edilip düzeltilmezse yapılan işlemin sonucu yanlış olur.

Günümüzde yonga-seviye ve sistem-seviye olmak üzere iki tür yumuşak hatalar mevcuttur. Ne zaman radyoaktif atomlardan dolayı alfa parçaları yonga'da yayılmaya başlarsa bu durum yonga-seviyesinde yumuşak hata olduğuna işaret etmektedir. Bu parçalar yonganın donanım yapısına zarar vermeyip sadece hafıza hücrelerinin değerlerine hasar vermektedir [3].

Veri yolunda olan veriler üzerinde işlem yapıldığı sırada eğer ses olaylarından (gürültü) dolayı yumuşak hata gerçekleşirse bu tür yumuşak hatalara, sistem-seviyesinde yumuşak hata ismi verilir. Bu tür hatanın oluşumunda elektronik cihaz ses olayını veri olarak algıladığından dolayı adresleme işleminde veya program kodlarının işlemi sırasında hata oluşur. Hatta bazen bu hatalı veriler hafızada tutulduğunda daha sonraki işlemlerin hatalı olmalarına da sebep olabilir. Bu tür hatalar oluştuktan sonra bazen hatalı verilerin üzerine hatasız veriler tekrar yazılarak düzeltilebilir. Bazı güvenli sistemlerde de hata düzeltme teknikleri kullanılarak yumuşak hatalar, hafıza hücrelerine kaydedilmeden ve üzerinde işlem yapılmadan düzeltilebilmektedir. Ancak bu sistemlerde hatayı bulma ve düzeltme işlemi 100% doğru çalışmayabilir, çünkü çoğu sistemlerde hatalı verileri hatasız verilerden ayırmak çok da kolay olmamaktadır.

Yumuşak hatalar en çok yarı iletken hafızalarda meydana gelmekte bunun dışında iletim hattında, analog devrelerde, manyetik hafızalarda ve sayısal devrelerde de görülmektedir.

Yumuşak hatalara aynı zamanda yarı iletken cihazlardakı aksaklıktan kaynaklanan tek-olaylı-bozukluk ismi de verilir[9]. Bu aksaklık daha önce de anlatıldığı üzere cihaz bozukluğuna neden olmaz. Eğer bir düğümde toplanmış Q enerji, o düğümün kritik yükünden büyük olursa yumuşak hata ortaya çıkar ve o düğümde bit çevirilmesine neden olur [3]. Şekil 1.2'de cihazın ve devrenin görünümünde bit çevrilmesi gösterilmektedir.



Şekil 1.2. Cihaz ve devrelerde bit çevrilme görüntüsü.

Yumuşak hatanın oluşumu genelde tasarlayıcı kontrolunun dışında gerçekleşir. Bir çok sistemde eğer yumuşak hata belli bir ölçünün altında olursa bu durum sistemin işleyişini etkilemediğinden dikkate alınmayabilir. Örneğin bir video uygulamasında oluşan yumuşak hatalar ekrandaki renklerin yanlış gösterilmesine sebep olur fakat bu hatalı renkler görünümü bozmayacak kadar az olursa dikkate alınmayabilinir. Ancak bir cihazın işlevselliğini kontrol etmek için SRAM gibi bir hafıza elemanı kullanılırsa, yumuşak hata etkisi çok daha ciddi olabilir. Çünkü bu durumda verilerin değişimine ek olarak cihazın işlevselliği etkilenir, bu durum da kritik hatalara yol açabilir [10].

1.3.Yumuşak Hata ve Donanım Hata Farkı

Eğer sistemin çalışması sırasında bir hata meydana gelerek işlemin durmasına sebep olursa bu hataya donanım hata ismi verilir. Ama eğer sistemin çalışması sırasında oluşan hata işlemin durmasına sebep olmayıp sadece bit değerlerini değiştiriyorsa buna yumuşak hata denir. Donanım hatası oluştuğunda hatayı gidermek için tek yol donanımın cihazı yeniden başlatılarak işlemin tekrar çalıştırılmasıdır. Yumuşak hata oluştuğunda ise sistemin yeniden başlatılmasına gerek yoktur ancak sistemin yaptığı işlem bittiğinde sonucun ne kadar doğru olup olmadığını anlaşılmaz.

1.4.Yüksek Seviye Sentezleme

Son yıllarda silikon teknolojileri daha karmaşık hale geldi ve bu durumdan dolayı donanım tasarımlarının yüksek soyutlama seviyesine uyarak tasarlanmasını zorunlu hale geldi. Konuyu daha açık hale getirmek için yazılım dünyasındaki gelişmeler örnek verilebilir.

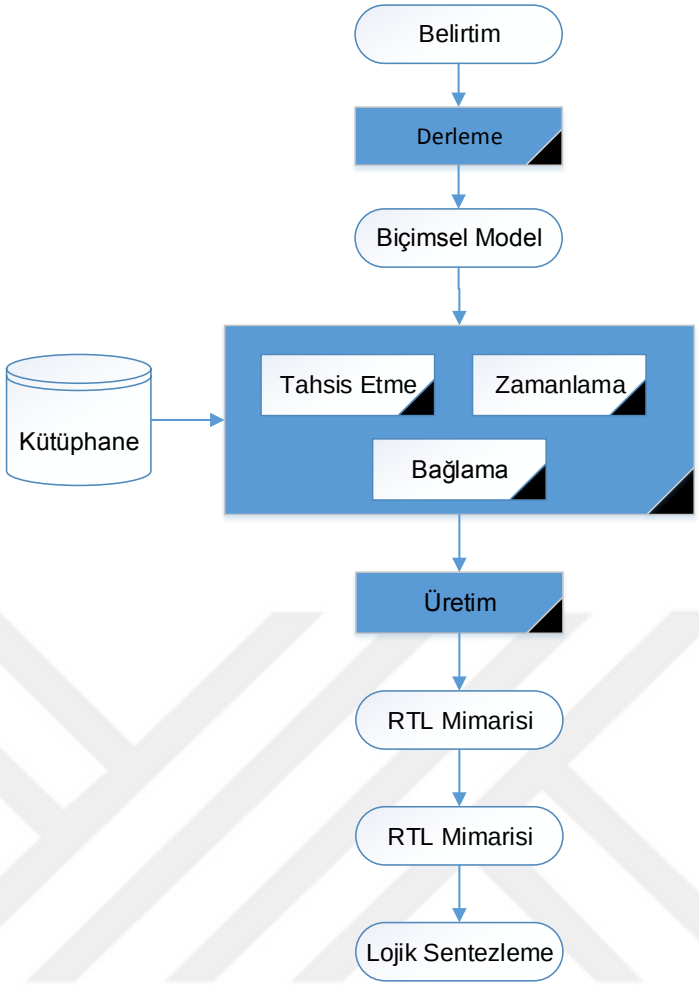
Bilgisayar teknolojisinin başladığı dönemde, bilgisayarların programlanması için 1950'lere kadar sadece makine kodu kullanılıyordu. 1950'li yıllarda assembly dili

icat edildi ve zamanla yüksek seviye programlama dillerinin kullanımı başladı. Böylece eski karmaşık programlama dillerinin yerine insan diline, dil kurallarına ve söz dizim kurallarına uyan diller çıktı. Bu durum YSP (Yüksek Seviye Programlama) dillerinin kullanımının basitleşmesini sağladı. YSP dillerinin basit görünmesinin nedeni; esneklik ve taşınabilirliği için alt planda olan tasarım karmaşıklarını gizli tutmasıdır. Bu basitleşme sayesinde yazılımcıların alt plandaki karmaşık tasarımın nasıl çalıştığını bilmelerine gerek kalmadı.

Donanım dünyası da bu şekilde evrim geçirmiş olup [11], 1960lı yıllarda tümleşik devreler tasarlandı. 1970'li yıllarda geçit (gate) seviyesi simülasiyonu ortaya çıktı ve 1979 yılında devir temelli simülasyonlar elde edildi. Zaman geçtikçe Verlog (1986) ve VHDL (1987) gibi donanım tanımlama dilleri icat edildi ve bunlardan sonra YSS'nin (Yüksek Seviye Sentezleme) ilk ticari nesli 1990'lı yıllarda ortaya çıktı [12]. Aynı yıllarda IP çekirdekli ve platform tabanlı tasarımların kavramı yavaş yavaş ortaya çıkmaya başladı.

YSS deki tasarım adımları kabaca şekil 1.3'de gösterilmektedir:

1- İlk adımda belirtim derlenir
2- Donanım kaynaklarının dağılım işlemleri yapılır.
3- İşletmeler saat döngüsüne göre zamanlanır.
4- İşletmeler işlevsel birimlerine bağlanır.
5- Değişkenler depolama elemanlarına bağlanır.
6- Taşıyıcılar veri yolu ağlarına bağlanır ve
7- En sonunda RTL mimarisi üretilir. [13]



Şekil 1.3. Yüksek seviye sentezleme tasarım adımları.

Karmaşık bir bireşimin yönetimini kolaylıkla ele almak için yukarıdaki adımların sırasıyla yapılması gereklidir. YSS ilk adımda giriş verilerini biçimsel formata çevirmeyle başlar, bu adımda ölü kodları eleme, yanlış veri bağımlılıklarını eleme, döngü dönüşümü gibi işlemler için iyileştirme kodları mevcuttur. İlk adımda veri ve işlemler arasındaki bağımlılıkları gösteren formal bir model elde edilir. Verilerin arasındaki bağımlılıklar Veri Akış Çizelgesi (VAÇ) kullanılarak kolaylıkla gösterilebilmektedir. Bu çizelgede her bir düğüm bir işleme işaret etmekte ve düğümler arasındaki bağlar da giriş, çıkış ve geçici değerleri tutmakla sorumludur. Bu tezde kullanılan yöntemde de işlemleri ve işlemler arasındaki bağları belirlemek için VAÇ kullanılmaktadır [13].

Dağılım işlemi, tasarım kısıtlarını tatmin etmek için donanım kaynaklarının tür ve sayısını belirler. Bileşenler genelde bileşen kütüphanesinde listelenir ve bu

kütüphanede her bir bileşen farklı alan, gecikme ve güç tüketimi değerlerine sahiptir. Her bir işlem için en az bir bileşenin seçilmesi gerekir. Bu bileşenler bazen zamanlama işlemi esnasında işlemler arasında dağılır bazen de zamanlama işlemi başlatılmadan kaynak dağılım işlemi tamamlanmış olur.

Tüm bileşenlerin devirler arasında zamanlanması gerelidir. Örnek olarak a=b+c işlemi için , önce b ve c değişkenlerinin değerleri kaynaklardan okunur, toplama işlemini yapan işlevsel birimin girişine verilir ve işlevsel birimin çıktısı (bu örnek'te a) bu işlemin çıktısı olur. İşlemler genelde bir veya birden fazla devir saatine zamanlanabilmekte, bu işlemlerin sonuçları diğer işlemlerle zincirlenebilmektedir. Yani bir işlemin çıktısı başka bir işlemin girdisine verilebilir. Belirli bir saat döngüsünde eğer beklemede olan kaynak varsa birbirinden bağımsız olan işlemler paralel olarak çalışabilir.

Devirler arasında değerleri taşıyan her bir değişkenin bir hafıza birimine bağlanması gereklidir. Hafıza birimlerinin boş durmasını önlemek ve verimliliğini artırmak için işleme başlama ve bitiş zamanları farklı olan değişkenler aynı hafıza birimini kulllanır ve her bir işlemin bir işlevsel birime bağlanması gerekir. YSS'de harcanan zaman ve alan daha önceden belirlendiği için tasarım daha da optimize hale gelir [13].

1.5.Problem Tanımlama

Son zamanlarda entegre devreler hızlı bir şekilde gelişmekte olup daha az güç tüketerek daha iyi performans göstermeleri için çalışmalar yapılmaktadır. Moore kanununa göre entegre devredeki transistör sayısı her iki yılda bir iki katına çıkmakta böylece entgre devreler daha karmaşık hale gelmektedir. Aynı zamanda transistör sayısı arttıkça yumuşak hataların meydana gelme olaslığı da artmaktadır. Yumuşak hataları gidermek veya oluşmasını engellemek için yazılımsal yöntemler geliştirilmiştir. Bu yazılımsal yöntemlerin daha fazla güç tüketmesi , alan ve gecikme zaman kısıtlarını dikkate almaması vs gibi dezavantajları vardır.

Bu çalışmada diğer yöntemlerden farklı olarak daha hızlı bir şekilde sonuca varmak, aynı zamanda alan ve gecikme kısıtlarını dikkate alarak daha güvenilir kaynakları işlemler arasında dağıtarak sistemin genel güvenirliğinin artırılması için genetik algoritma tabanlı bir seçimsel kaynak atama yöntemi sunulmaktadır.

1.6.Genetik Algoritmanın Kullanma Sebebi

Bu tezde tamsayılı doğrusal programlama (TDP) ve Sezgisel yöntem yerine genetik algoritma yönteminin kullanılması tercih edilmiştir. Daha önceki çalışmalar incelendiğinde YSS üzerinde TDP algoritmasını kullanan yöntemler genelde alan-performans veya alan-performans-güç tüketimi faktörlerini iyileştirmek amaçlı çalışmalar yapılmıştır . TDP algoritmasını kullanan [14][15] gibi çalışmalar mevcuttur, fakat algoritmanın tamamlanma süreci çok fazla zaman aldığı için pek kullanışlı değildir. Kullanılan cihazlarda daha fazla güç tüketilmesine ve kaynak kullanılmasına sebep olmaktadır.

Bu üç algoritma birbiriyle karşılaştırıldığında sezgisel yöntem dayanaklı algoritmalarda, tamsayılı doğrulama programlama yöntemine göre daha hızlı sonuç elde edilir ancak sezgisel yöntemden elde edilen sonuç tahmini sonuç sayılır. Tamsayılı doğrulama programlama yönteminde ne kadar yavaş çalışırsa çalışsın elde edilen sonuç kesin ve oldukça iyi bir sonuçtur. Bu iki yöntemin dezavantajlarına bakıldığında; TDP yöntemi daha yavaş çalışmakta olup işlem sayısı artırıldıkça algoritmanın karmaşıklığı da artmakta, sezgisel yöntemde ise elde edilen sonuç tahmini sonuç olmaktadır. Bu dezavantajları gidermek için bu çalışmada genetik algoritma kulanılmıştır. Bu yöntemde sonuç oldukça hızlı elde edilmiş olup alan ve gecikme kısıtları altında daha iyi bir sonuca varılmıştır.

1.7.Bilim ve Teknolojiye faydası

YSS üzerine yapılmış daha önceki çalışmaların bir çoğunda alan, güç tüketimi ve performansın iyileştirilmesi için çalışmalar yapılmış olup güvenirlik konusu çok dikkate alınmamıştır. Bazı çalışmalarda ise YSS için alan ve performans değerleri düşünülmeden güvenilirliği artırmak odaklı çalışmalar yapılmış ve daha sonra alan ve performansı dikkate alarak güvenilirliği artırmak için bazı yöntemler icat edilmiştir. Ancak sonraki bölümde de anlatıldığı üzere bu yöntemlerin bazı dezavantajları vardır. Bu tezdeki yöntem ise bu dezavantajları gidermek için yapılmıştır. Bu yöntem, üzerinde çalışmalar yapılarak geliştirilip tasarımın güç tüketim değerini azaltmak için de kullanılabilir.

1.8.Tez Planı

2. bölümde bu tezle alakalı olan kaynak özetlerinden bahsedilmiştir. 3. Bölümde tezde kullanılan algoritmalar ve yöntemler tanıtılmış ve açıklanmıştır. 4. Bölümde ise genetik algoritma yönteminin çalışma mekanızması anlatılmıştır. 5. Bölümde tezdeki problemin genetik algoritma yöntemiyle nasıl çözüleceği gösterilmiştir. 6. bölümde farklı VAÇ'leri üzerinde deneyler yapılıp problemin çözümü desteklenmiştir.

2. Kaynak Özetleri

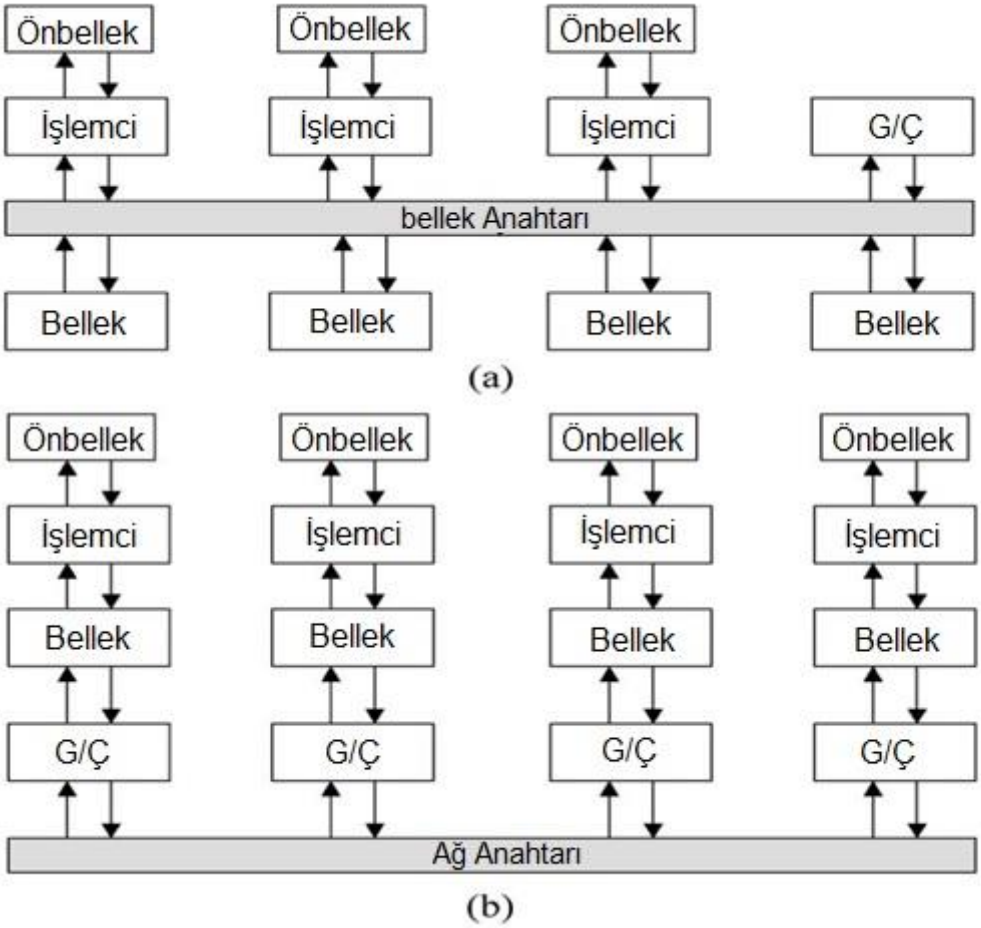
Yumuşak hatalara karşı geçmişte çeşitli çalışmalar yapılmıştır. Önceki çalışmalardan elde edilen sonuçlara göre aynı işlevin uygulaması farklı alan, gecikme ve güvenilirlik değerlerine sahip olabilmektedir [3]. Bu çalışmalardaki yöntemlerden biri bu hatalara karşı daha dirençli olan kaynakları kullanmaktır. Bu durumda tasarımcı tasarımda güvenilirlik değerinin artırması için hatalara karşı daha dirençli olan kaynakları kullanmaktadır. Ancak çoğu zaman bu yöntem işe yaramaz çünkü genelde tasarımda alan ve gecikme zamanı kısıtlanır. Bu durumda eğer hep daha güvenli kaynaklar kullanılırsa alan ve gecikme değerleri artar ve kısıtlanmış alan ve gecikme zamanı karşılanmaz.

Kısıtlanmış alan ve gecikme zamanının altında son tasarımın daha güvenirli olması için hatalara karşı daha dirençli olan kaynakları seçmek üzere bazı çalışmalar yapılmıştır. Bu çalışmaların bazılarında tamsayılı doğrusal programlama algoritması kullanılmıştır [14][15]. Bir önceki bölümde bu çalışmaların dezavantajlarından bahsedilmiş olup bu dezavantajları gidermek için genetik algoritma tabanlı bir yöntem sunulmuştur.

Yumuşak hatalara karşı bazı yöntemler hafıza üzerinde yumuşak hataları bulup düzeltme mekanızmasını kullanmaktadır. [26] makalesinde veriler SRAM, DRAM ve Flash gibi hafıza cihazlarının üzerine yazıldıktan sonra verilerde yumuşak hatanın oluşup oluşmadığı belirlenir. Eğer yumuşak hatalardan dolayı bit dönmesi varsa bunları düzeltmek için hata düzeltme yöntemi kullanılılır. Ancak hafıza üzerinde yumuşak hatalara karşı yapılan çalışmalarda, hata bulma mekanızması yüzde yüz cevap vermeyebilir. Aynı zamanda hata düzeltme kodunun doğru bir şekilde hataları düzelttiğinden emin olmak zordur. Bunlara ek olarak hata bulma ve düzeltme için fazladan kaynak kullanılması gerekir ve bu yüzden bu yöntemler yumuşak hatalara karşı iyi bir yöntem sayılmamaktadır.

Genel olarak hafıza yapıları ve hafızalar üzerinde yumuşak hatalara karşı yapılan çalışmalardaki yöntemlerden bahsedilecek olursa [27] makalesine göre sunucularda ve iş istasyonlarında genel olarak kullanılan iki tür çok işlemci mimarisi Şekil 2.1.'de gösterilmektedir. Paylaşımlı bellek işlem (PBİ) mimarisinde şekilde gösterildiği gibi bellek anahtarı yardımıyla ana hafıza, işlemciler arasında

paylaşılmakta, kümelenmiş mimaride ise her işlemcinin kendisine ait bir ana hafızası var olup veriler ağ anahtarı üzerinden paylaşılmaktalar.



Şekil 2.1. Paylaşımlı bellek ve kümelenmiş mimari.

Hata bulma ve düzeltme yöntemi, genelde büyük bellek dizilerinde yumuşak hatalardan dolayı oluşan bit değişimini bulup düzeltmek için kullanılmaktadır. Bu yöntemde hafıza üzerinde bulunan verilere ilave olarak kelime kontrol bitleri eklenmektedir. Hafızalarda bu tarz hataları bulmak için kullanılan en basit yöntem benzerlik bit yöntemi olarak bilinmektedir. Bu yöntemde verilere benzerlik biti eklenerek verinin tek veya çift olduğu belirlenir ve bu veri üzerinde hata meydana gelip gelmediğini kontrol etmek için verinin tek veya çift olduğu hesaplanır. Eğer bu değer benzerlik bit'inden farklı olursa bit değişimine veya yumuşak hatanın oluşuna işaret etmektedir. Bu yöntemde eğer yumuşak hatalardan dolayı veri üzerinde birden fazla bit değişmiş olursa bu yöntem işe yaramayacaktır. Aynı zamanda değişen bitin hangi bit olduğu bilinmediği için benzerlik bit yöntemi hatayı düzeltmek için kullanılmaz [28].

Hata bulma ve düzeltme(HBD) kodunda kontrol bitin sayısı, korunması gereken veri boyutuna göre belirlenir. Kontrol bit sayısı ne kadar fazla olursa o kadar güçlü bir HBD algoritmasına sahip olunur. Ancak kontrol bit sayısı artıkça hafızada daha fazla yer kullanılır ve bu yüzden korunması gereken kelime boyutuyla dengeli bir şekilde kontrol biti kullanılır.

Hafıza üzerinde hata bulma ve düzeltme tekniklerine bakıldığında; Tek hata düzeltme/çift hata bulma (THD-ÇHB) yöntemi Hamming tarafından icat edilmiştir [29]. Daha sonra sistemin hatalardan korunması için daha kapsamlı ve güçlü olan tek-bayt hata düzeltme ve çift-bayt hata bulma(TBHD-ÇBHB) yöntemi ortaya çıkarılmıştır (Bu yöntemde bayt, 1bayt = 8bit şeklinde olmak zorunda değildir, bayt değeri farklı bit sayısıyla isimlendirilebilir). Hata düzeltmek için diğer bir güçlü yöntem ise çift-bit hata düzeltme/ üçlü-bit hata bulma (ÇHD-ÜHB) yöntemi olarak bilinmektedir ancak bu yöntem diğer yöntemlere göre daha fazla alan ve güç tüketmekte olup aynı zamanda kontrol bitlerini kodlamak ve çözmek için fazladan işlem yaptığından dolayı daha fazla gecikme değerine sahiptir [28].

Tablo 2.1'de her üç yöntem için kullanılan kontrol bit sayısı gösterilmiş ve karşılaştırmaları yapılmıştır. (TBHD-ÇBHB için bayt=4 olarak düşünülmüştür) [28].

	THD-CHB			TBD-CBB (B = 4 bit)			CHD-ÜHB		
Veri bitleri	Kontrol bit	Ek yük	Toplam bit	Kontrol bit	Ek yük	Toplam bit	Kontrol bit	Ek yük	Toplam bit
16	6	%38	22	12	%75	28	11	%69	27
32	7	%22	39	12	%38	44	13	%41	45
64	8	%13	72	14	%22	78	15	%23	79
128	9	%7	137	16	%13	144	17	%13	145

Tablo 2.1. Hata bulma ve düzeltme yöntemlerinde kontrol bit sayısı, ek yük ve toplam bit sayısı.

Kombinasyonel devrelerde yumuşak hatanın oluşumu, Tek-Olaylı-Geçiş (TOG) lere sebep olmaktadır. Diğer taraftan hafıza hücrelerinde oluşan yumuşak hata Tek-Olaylı-Bozulmaya (TOB) neden olur. Her ikisi de (TOG ve TOB) devre işlemlerinde büyük etkiler yaratır.

[30] makalesinde kombinasyonel devrelerde yumuşak hataların oluşma olasılığını azaltmak için simülasyon tabanlı bir yöntemden bahsedilmiştir. Bu yöntemde yumuşak hatanın oluştuğu sırada mantıksal maskeleme işleminin olasılığı artırılmaktadır. Maksimizasyon işlemini yapmak için orjinal çok seviyeli devrelerden, alt devreleri çıkarılmakta olup bu alt devreler üzerinde tekrar sentezleme işlemi uygulanır. Bu işlem yapıldıktan sonra kombinasyonel devreler yumuşak hatalara karşı daha dirençli hale getirilir. Orijinal çok seviyeli devrelerden alt devreler çıkarıldığında, bu alt devrelerin bir tane çıktısı ve M tane girdisi olmasına aynı zamanda diğer devrelere yayılmamasına (çıkışı iki farklı devreye giriş olarak verilmemelidir) dikkat edilmelidir çünkü sentezleme işleminden sonra alt devrelerin yapıları değişmiş olur ve bu değişimin diğer alt devrelerde etki yaratmaması gerekmektedir.

Sunulan [30] numaralı makalede alt devrelerde yumuşak hataların maskelenmesi için iki seviyeli sentezleme şeması kullanılmaktadır. İki seviyeli alt devre sentezlemede alan yükünü azaltmak için hızlı ekstrasiyon (HE) algoritması kullanılmaktadır. Alt devreler üzerinde tekrar sentezleme işlemi yapıldıktan sonra kullanılan alanı minimize etmek için ayrıca FX algoritması kullanılır. Bu yöntemin uygulanması tezde sunulan yönteme göre daha zor olmakta ayrıca alan ve gecikme kısıtları dikkate alınmamaktadır.

Devrelerde güvenilirliği artırmak veya yumuşak hataları azaltma için diğer bir yöntem ise [26] makalesinde sunulmuştur. Üçlü-Modular-Yedekleme (ÜMY) yönteminde çok fazla alan (%200 den fazla) kullanılmaktadır. Bu alan yükünü azaltmak aynı zamanda güvenilik, alan , güç tüketimi ve performans arasında bir denge sağlamak için kısmi yedekleme yöntemleri kullanılmaktadır. Geçici, aralıklı ve kalıcı hataları hafifletmek için [26] yönteminde genel bir çerçeve tanımlanmıştır. Bu çerçeve, yüksek olasılıkla oluşan hataları maskeleme için kullanılmakta olup aynı zamanda alan yükünü de azaltmak için çaba gösterilmektedir. Bu makale ÜMY şemasında kullanılmak için olasılıkçı ve evrimsel yöntemleri kullanarak olasılıklı lojik devreler üretmektedir.

Yaklaşık-Logik-devreler orijinal devrelerle ilgili olup farklı bir şekilde uygulanmaktadır. Bu devreler orijinal devrelerle üst üste gelip ve hataları bulup çözmek için kullanılmaktadır. Bu devreler boyut olarak orjinal devrelerden daha

küçük olmaktadır. ÜMY sisteminde orjinal tasarımdan gerçek kopyalama işlemi uygulamak yerine yaklaşık-lojik-devreler kullanılabilmektedir. Bu yöntemde tasarımcı tahmin seviyesini belirleyebilmektedir. Tahmin seviyenin yüksek olması bir o kadar güvenilirliğin ve alanın artması demektır. Bu yöntemde de aynı ÜMY sistemindeki gibi fazladan alan ve güç kullanılmaktadır. Ancak yaklaşık lojik devre mantığında güvenilirlik, alan ve güç tüketimi arasında bir denge bulmaktadır.

[27] numaralı çalışmada yumuşak hataları azaltmak için aşağıdaki katkılar sağlanmaktadır:

1. Sentezleme araçlarında tahmini güvenilirlik ölçme yöntemi sunulmaktadır.
2. Güvenilirlik odaklı sentezlemelerde, optimizasiyon algoritmaların etkisi tanımlanmaktadır.
3. Güvenilirlik odaklı sentezleme tekniği geliştirilmektedir.

Hata oluşma tahmininin doğru bir şekilde olçülmesi için aşağıdaki durumların oluşması gerekmektedir:

1. Geçici dalgaların diğer bir hafıza elemanına erişmesi için mantıksal bir yolun olması gereklidir (mantıksal maskeleme).
2. Geçici dalganın hafıza elemanının durumunu değiştirmesi için yeterli kadar süre ve genişliğe sahip olması gerekmektedir (elektirik maskeleme).
3. Saat darbesinin hafıza elemanını etkileştirdiği sırada geçici dalganın varmış olması gereklidir (mandallı pencere maskeleme).

Bu durumların oluşma tahmininin ölçülmek için aşağıdaki formül tanımlanmıştır:

$$HOİ = P\text{-üretim} * P\text{-yayılma} * P\text{-kilitleme}$$

P-üretim, devre içindeki geçici dalganın oluşma olasılığı, P-yayılma ise geçici dalganın hafıza elemanının durumunun değişmesine sebep olma olsasılığı ve P-kilitleme yanı geçici dalganın hafıza etkileşim sırasında varma anlamına gelmektedir.

Güvenilirlik odaklı sentezleme tekniğini geliştirmek için hata oluşma ihtimali hesaplanır. Daha sonra yumuşak hatalara karşı savunmasız hücreler veya alt devreleri tanımlanır. En sonunda bu alt devreler daha güvenili alt devrelerle yer değiştirilir.

Sonlu durum makinelerinde ise hataları tolere etmek için bir çok çalışma yapılmıştır. Genelde sonlu durum makinelerinde durumları hatalardan korumak için hata oluşma ihtimali yüksek olan durumlara eşdeğer yedekleme durumu eklenir. Bu çalışmaların çoğunda güç ve alan kullanımına dikkat edilmemiştir [28].

[29] makalesinde ise aynı yöntem kullanılmıştır. Bu araştırmada, orjinal tasarım bozulmadan alan ve güç tüketimi optimize edilmeye çalışılmıştır. Bu yöntemde ise hata oluşum ihtimali yüksek olan durumları korumak için ikili kodlarındaki hamming mesafesi kontrol edilir. Korunması gereken bir durumun ikili kodlarının arasındaki hamming mesafesi 1 olursa yedekleme durumu eklenir. Bu teknikte her bir ikili durumun hamming mesafe değeri en az 3 olmak zorundadır. Korunan bir durumun, başka bir durumdan hamming mesafe değeri ise en az 2 olması gerekir.

İki durumun kodları arasındaki hamming mesafe değeri, bu iki durumun kodları arasındaki benzer olmayan bitlerin konum sayısına işaret etmektir. [28] yönteminde beklenen hamming mesafe kısıtı karşılanmaktadır ancak bunun yanında alan ve güç tüketimi optimize edilmemektedir. Bu eksikliği gidermek ve aynı zamanda genel mantığı kullanarak alan ve güç kısıtlarının altında durumları korumak için yeni bir yöntem sunulmuştur [29]. Bu yöntemde önce hatalara karşı daha savunmasız durumlar seçilir. Seçilen durumları korumak için kısıtlanmış alan ve güç değeri altında daha az şifreleme biti kullanılır, böylece sistemin güvenilirliği artırmaya çalışilir.

Yumuşak hataları tolere etmek için [30] makalesinde üçlü yedekleme algoritması ve oylama-geriyazma (vote-writeback) bileşimi kullanarak yeni bir yöntem icat önerilmiştir. Bu şemada oylama işlemi gerçekleştirilmektedir. Yani oylanması gereken veriler kütüklerden (register) okunur. Eğer okunan veriler arasında bir uyuşmazlık keşfedilirse bir hata oluşumuna işaret etmekte olup oylanan veriler kütüklere geri yazılır.

[30][36] da kullanılan Cone bölümler sistemin güvenilirliğini ve kaynak paylaşımını artırmak için kullanılmaktadır, ancak bu başarıyı elde etmek için kaynak atama ve zamanlama işlemlerini optimize edebilecek güçlü bir yüksek seviye sentezleme aracına ihtiyaç vardır.

Cone bölümlerle kombine olmuş ve özelleştirilmiş yüksek seviye sentezleme [37] makalesinde anlatılmıştır. Bu şemada sistemin güvenilirliğini artırmak aynı

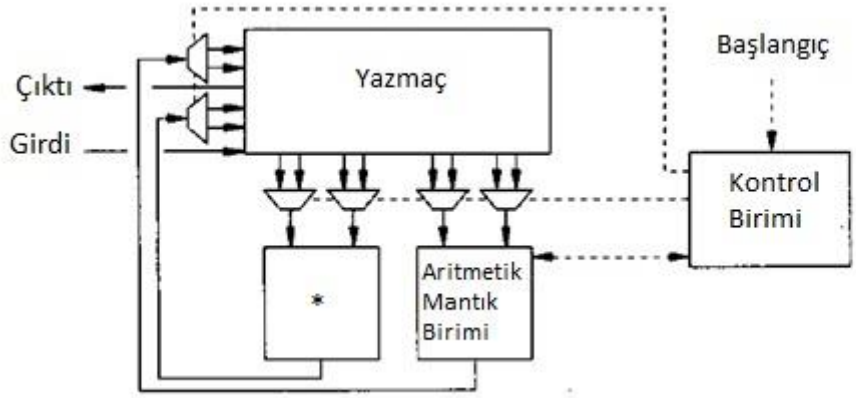
zamanda kaynak atama ve zamanlama işlemini optimize etmek için iki fazlı yeni bir sezgisel yöntemi sunulmuştur. İlk fazda kaynakları cone bölümlerine vermek için üç tane rezerve edilmiş tablo kullanılır. İkinci fazda ise cone'larda kaynak atama ve zamanlama işlemi gerçekleştirilir.

Bu çalışmada önceki çalışmalardan farklı olarak kullanılacak kaynakların birden fazla versiyonu kullanılarak geçici hatalara karşı daha dayanıklı, güvenilirlik değerini artıran bir tasarım yöntemi sunulmuştur. Sunulan genetik algoritma tabanlı yöntem, linear programlama tabanlı yönteme göre daha hızlı bir şekilde sonuçları bulurken, sezgisel yöntemlere göre daha iyi sonuçlar elde etmiştir.

3. KULLANILAN ALGORİTMA ve YÖNTEMLER

3.1. Yüksek Seviye Sentezleme

Yüksek Seviye Sentezleme (YSS), istenilen devre uygulamalarının stiline göre farklı şekillerde gerçekleştirilir. Böylece şekil 3-1'de gösterilen yapıya uygun olan çok sayıda algoritmalar ve araçlar önerilmiştir. YSS'de alan ve performans değerleri; kaynaklar, depolama devreleri ve devre bağlantılarına bağlıdır ancak bu etkenlerin en önemlisi kaynak olarak tanınmakta olup daha önce yapılan çalışmaların çoğunda kaynaklara odaklı yöntemler önerilmiştir [38].



Şekil 3.1. YSS için örnek bir yapı.

Bu tezde alan ve performans kısıtları altında yumuşak hataları önlemek için kaynak odaklı bir yöntem sunulmuştur. Çalışmamızda her bir kaynağın farklı versiyonları mevcut olup alan ve performans kısıtı altında bu kaynaklardan en uygunu seçilerek sistem güvenilirliği artırılmaktadır. Alan hesaplanması için sistemde kullanılan kaynakların toplam alanı sistemin kullandığı alan olarak bilinmekte olup performans hesaplanması için kaynakların işlemlerini bitirmesi için gereken saat döngülerinin sayısı hesaplanmaktadır.

3.2. Uygulama Kısıtlar

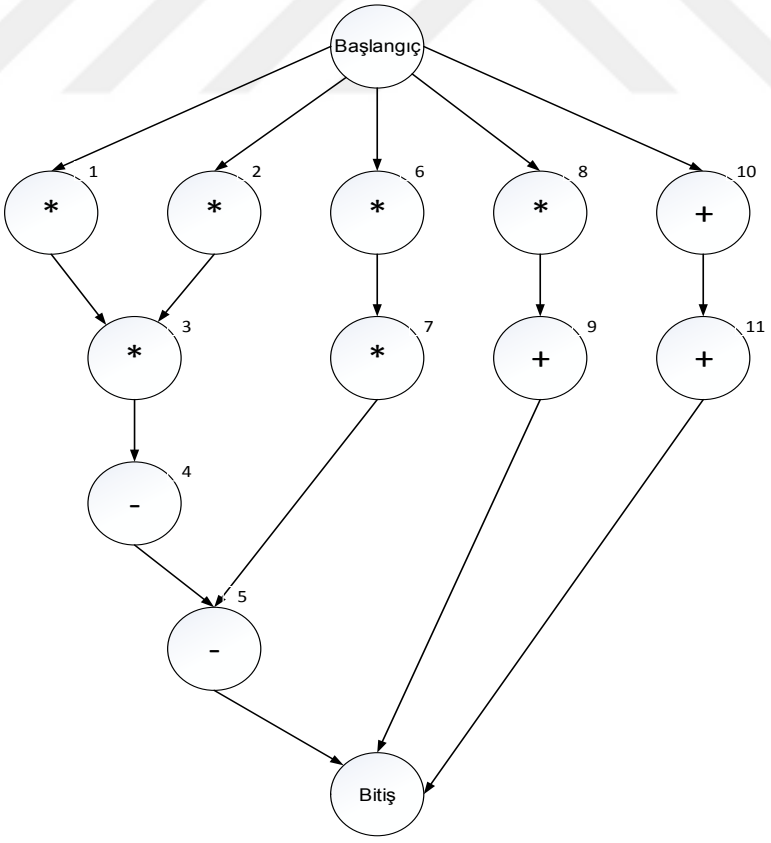
Uygulama kısıtları, tasarımcının alan ve performans gibi özellik beklentilerinin yerine getirilmesi için uygulanmaktadır. Eğer bir sistemde alan kısıtlaması belirlenmişse, işlemler arasında kaynak atamaları sırasında alan kısıtının aşılmamasına dikkat edilmesi gerekmektedir. Bu kısıtları aşmamak için kaynak atama işlemi sırasında farklı yöntemler kullanılmaktadır. Örnek olarak [3] çalışmasında kaynak atama işlemi yapıldıktan sonra beklenen kısıtlar kontrol

edilir, eğer atanan kaynakların özellikleri beklenen kısıtları karşılamıyorsa, kısıtlar sağlanana kadar bu kaynaklar başka kaynaklarla yer değiştirilmektedir.

Bu tezde sadece alan ve performans kısıtlarına dikkat edilmiştir. Sistemdeki alan kısıtının aşılıp aşılmadığını belirlemek için sistemin kullandığı toplam alan hesaplanıp kısıtlanan alan değeriyle karşılaştırılmaktadır. Eğer sistemin kullandığı alan, kısıtlanmış alan değerine eşit veya altında ise alan kısıt aşılmamakta demektir. Performans kısıtının aşılıp aşılmadığını kontrol etmek için uygulamanın kaç saat döngüsünde tamamlandığı hesaplanır, eğer bu değer kısıtlanan performans (gecikme) değeri altında olursa performans kısıtı aşılmamakta demektir.

3.3.Veri Akış Çizergesi

Bu tezde algoritmaya verilen işlemler ve işlemlerin arasındaki bağlantılar veri akış çizer gesi (VAÇ) şeklinde verilmektedir. Şekil 3-2'de gösterildiği gibi her bir işlem bir düğüm olarak tanınmakta olup düğümler arasındaki bağlılıklar oklarla belirlenmektedir.



Şekil 3.2. Örnek veri akış çizelgesi.

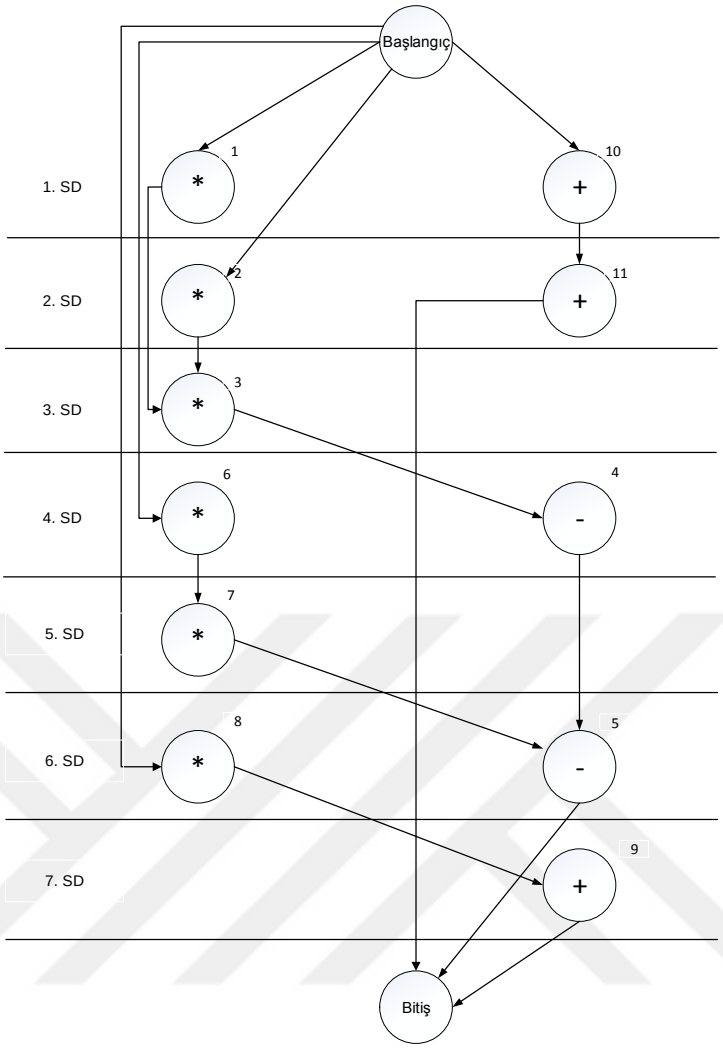
Bu VAÇ'de başlangıç ve bitiş düğümleri işlem olarak algılanmamakta ve bu sırada hiç bir alan, gecikme veya kaynak kullanılmamaktadır. Uygulama başlatıldığında 1,2,6,8 ve 10 düğümlerinden her hangi birisinin işlemi başlatılabilir ancak bunlar dışındaki düğümlerin başlatılması için bir önceki düğümlerin bitmesi gerekmektedir. Örneğin 3 numaralı düğümün başlatılması için 1 ve 2 numaralı düğümlerin işleminin bitmesi, sonuçlarının 3 numaralı düğümün girişine verilmesi gerekir. Uygulamanın bitmesi ise 5, 9 ve 11 numaralı düğümlerin hepsinin sonuçlanmasına bağlıdır.

Bir ugulamanın gecikme zamanı; bitiş düğümünün çalışma zamanının başlangıç düğümünün çalışma zamanından çıkarılmasıyla elde edilir. Eğer 3-2 şeklinde verilen VAÇ'nde gösterilen sistem kaynak ve alan kısıtı olmadan çalıştırıldığındaki düğümlerinin başlangıç zamanları 3-1 tablosunda gösterilmektedir. (Verilen bu örnekte her bir işlemin 1 saat döngüsünde bitmesi öngörülmüştür).

İşlem	Başlangıç Zamanı
D1,D2,D6,D8,D10	1
D3,D7,D9,D11	2
D4	3
D5	4

Tablo 3.1. Şekil 3.2 de gösterilen VAÇ için işlemlerin başlangıç zamanı.

Ama eğer uygulamanın çalışması için kaynak kısıtı getirilirse gecikme zamanı daha da artırılabilir. Şekil 3-2 de gösterilen VAÇ'nin çalıştırılması sırasında bir tane toplayıcı-çıkarıcı kaynak kısıtı ve bir tane de çarpıcı kaynak kısıtı getirilirse şekil 3-3 de gösterildiği gibi gecikme zamanı artırılıbilir.



Şekil 3.3. Kaynak kısıtı uygulanmış zamanlama sonucu

İşlemler tablo 3-2'de gösterilen şekildeki sırayla yapılmaktadır.

İşlemler		Başlama zamanı
Çarpıcı	toplayıcı-çıkarıcı	
D1	D10	1
D2	D11	2
D3	-	3
D6	D4	4
D7	-	5
D8	D5	6
-	D9	7

Tablo 3.2. Şekil 3.3 de gösterilen VAÇ için işlemlerin başlangıç zamanı

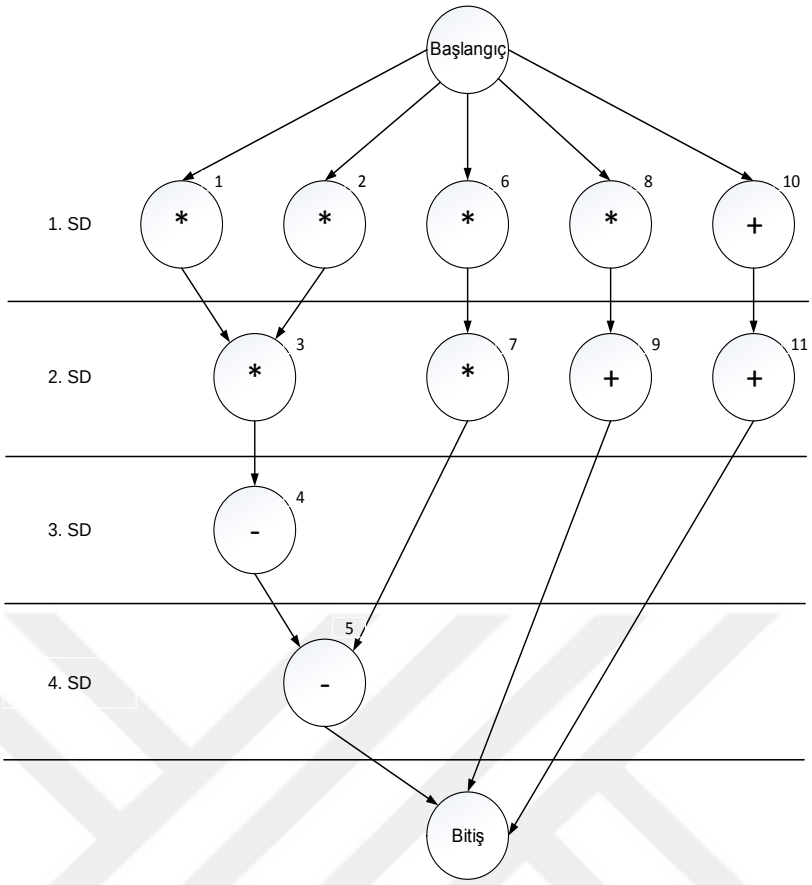
3.4. Zamanlama

Yüksek seviye sentezlemede en önemli adımlardan biri zamanlama olarak bilinmektedir. Sistemin performansı ve kullandığı alan çoğunlukla zamanlama işlemine bağlıdır. Bir önceki bölümde anlatılan VAÇ'de işlemlerin birbiriyle olan bağlantıları gösterilmekte olup zamanlamada ise her bir işlemin başlangıç zamanı belirlenmektedir. Zamanlama algoritması verilen kaynaklara göre paralel olarak da işlemleri başlatabilir ama bazen sistemin tasarımcısının isteklerini karşılamak için kaynak kısıtı getirilebilir. Bu durumda eş zamanlı olarak çalıştırılacak işlemler sayısı kaynak sayısına göre kısıtlanacaktır. Ne kadar kaynak kısıt değeri fazla olursa bir o kadar sistem performansı artırılacaktır, aynı zamanda alan ve güç tüketim değerleri de artırılmış olacaktır.

3.5. Kısıtlanmamış Zamanlama Algoritması (ASAP)

Yüksek seviye sentezlemede kullanılan en basit algoritmalardan birisi ASAP olarak bilinmektedir. Bu algoritma işlemlerin zamanlama hesabını olabildiğince erkene çekmektedir. Bir işlemin başlatılması için bu işlemden önce gelen ve bu işlemle bağlantılı olan işlemlerin bitmesi gerekmektedir, ancak zamanlamanın hesaplanması ilk işlemin başlangıcından önce gerçekleştirilir. Zamanlamaların hesabı; ilk düğümler birinci saat döngüsünde, sonraki bağlantılı düğümler devam eden saat döngülerinde çalışacak şekilde yapılmaktadır.

ASAP algoritması şekil 3.2'de gösterilen VAÇ üzerinde çalıştırıldığında şekil 3.4'de gösterildiği biçimde işlemlerin başlangıçları belirlenmektedir.



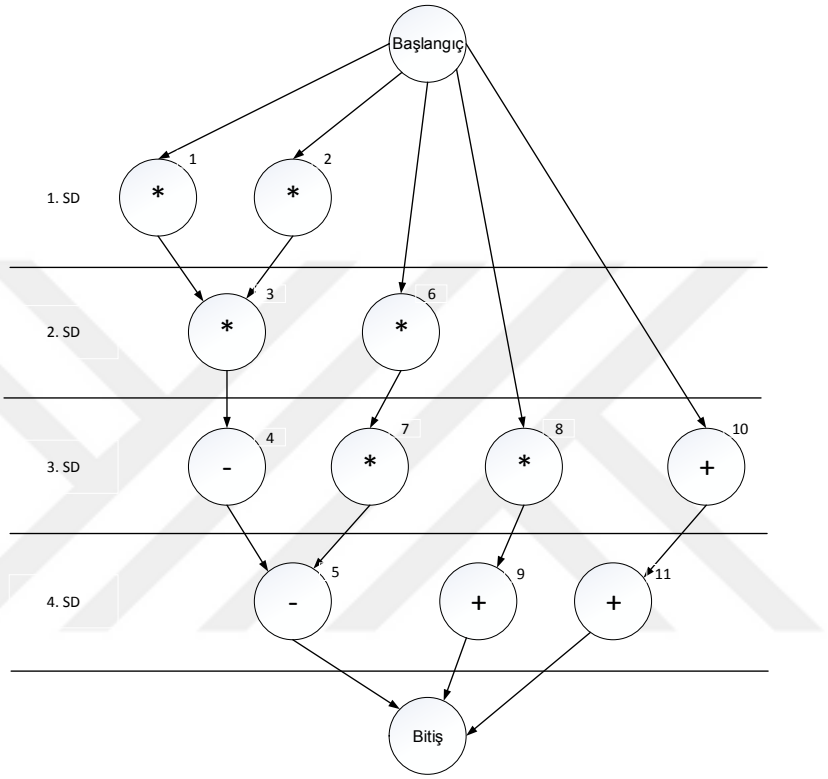
Şekil 3.4. ASAP algoritma çıktısı.

Bu tür zamanlamalarda sistemin genel gecikme değeri daha düşük olmaktadır. Birbirinden bağımsız olan işlemler aynı zamanlarda çalıştırılabilmektedir. Genelde kısıtlanmamış zamanlamalarda ASAP algoritması kullanılır ve bu algoritmayı kullanarak hiç bir kısıt uygulanmaması sağlanmaktadır.

3.6. ALAP Gecikme Kıstlı Zamanlama Algoritması

Yüksek Seviye Sentezlemede ASAP algoritması gecikme değerinin üst sınırını belirlemek için de kullanılmaktadır. Bu üst sınır değeri belirlendikten sonra ALAP algoritmasını kullanarak işlemlerin başlangıç zamanı değiştirilebilir. ASAP algoritmasıyla işlemlerin en erken başlayabileceği saat döngüsü belirlenir, ALAP algoritmasında ise işlemlerin en geç başlayabileceği saat döngüleri belirlenmektedir. ALAP algoritması aynı zamanda kısıt tanımlanmamış bir zamanlama hesabı için de kullanılabilir. Bu durumda ASAP algoritmasıyla gecikme değeri hesaplanıp ALAP algoritmasıyla bu gecikme değerinden başlayarak son düğümden ilk düğüme doğru zamanlama hesabı yapılabilmektedir.

Yukarıdaki şekilde ALAP algoritmasının adımlarında gösterildiği gibi bir işlemin zamanlanma hesabının yapılması için bu işlemden sonra gelen işlemlerin zamanlanma hesabının tamamlanmış olması gerekmektedir. Yani bu algoritma VAÇ'ndeki son düğümden ilk düğüme doğru zamanlama işlemini gerçekleştirmektedir. Örnek olarak şekil 3-4 de gösterilen VAÇ üzerinde ALAP algoritması çalıştırıldığında şekil 3-5'deki VAÇ elde edilecektir.



Şekil 3.5. ASAP algoritma çıktısı.

3.7. Liste Zamanlama Algoritması

Bir önceki bölümde anlatılan ASAP ve ALAP algoritmaları basit zamanlama algoritmaları olarak bilinmektedir. Eğer işlemlerin çalışma süresi bir saat döngüsünden fazla olursa ve tanımlanan VAÇ'sinde farklı türlerde işlemler olduğunda liste zamanlaması gibi güçlü zamanlama algoritmaları kullanılmaktadir. Bu algoritma, kaynak kısıtının altındaki durumlarda düşük gecikme ve gecikme kısıtının altındaki durumlarda az sayıda kaynak kullanım problemleri için uygulanmaktadır.

Liste zamanlama algoritmasında işlemlerden oluşan bir öncelik listesi kullanılmaktadır. Bu öncelik listesinde işlemler sezgisel aciliyet ölçütüne göre sıralanmaktadır. En yaygın olan ve bu makalede kullanılan öncelik listesinin

oluşturulması için son düğümden başlangıç düğümüne doğru işlemler ters bir şekilde sıralanmaktadır (kritik yol üzerinde olan düğümler yüksek öncelik değerine sahiptir). En yüksek öncelik değerine sahip olan düğümler önce zamanlanmaktadır [39].

Örnek olarak şekil 3-2 de gösterilen VAÇ için öncelik listesi şekil 3-6 de gösterildiği şekilde oluşmaktadır. Eğer liste zamanlama algoritması için 3 tane çarpıcı kaynağı ve bir tane toplayıcı-çıkarıcı kaynağı sınırı getirilirse aynı zamanda çarpıcı kaynağının işleminin yapılması 2 saat döngüsü, toplayıcı-çıkarıcı kaynağının işleminin yapılması 1 saat döngüsü içinde gerçekleşir. Şekil 3-2 de verilen VAÇ'nin zamanlama sonucu tablo 3-2 de gösterildği şekilde olacaktır.

Düğüm	1	2	3	4	5	6	7	8	9	10	11
15	4	4	3	2	1	3	2	2	1	2	1

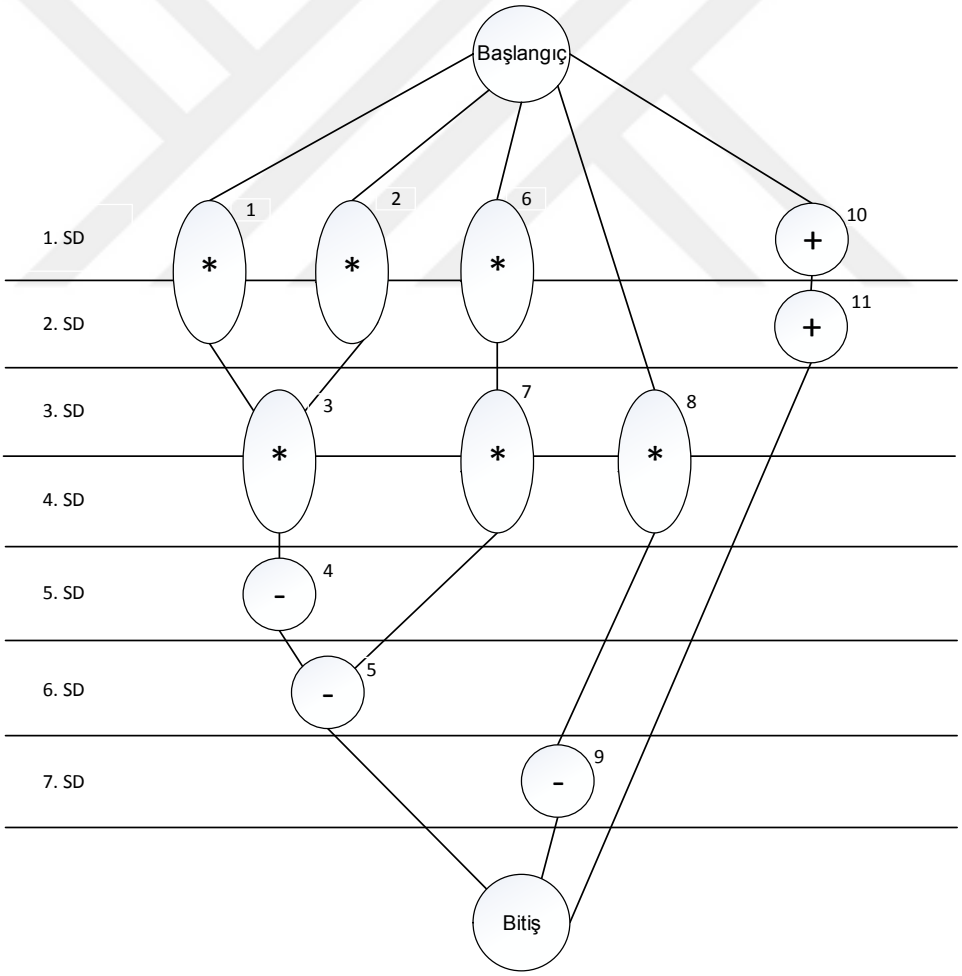
Şekil 3.6. Öncelik listesi.

Bu örnek üzerinde yapılan zamanlama işlem adımlarına bakıldığında, üç çarpıcı işlemi kaynak sınırı olduğu için öncelik listesinden 3 tane en yüksek önceliğe sahip olan çarpıcı işlemi seçilmektedir (d1, d2, d6). Kısıtlanan bir tane toplayıcı-çıkarıcı kaynağı için en yüksek önceliğe sahip olan toplayıcı-çıkarıcı işlemi seçilip (d10) zamanlama hesabı yapılmaktadır. Çarpıcı işleminin iki saat döngüsü sırasında birinci saat döngüsünde d10, ikinci saat döngüsünde d11 toplayıcı işlemleri tamamlanmış olur. d4 numaralı toplayıcı işleminin yapılmamasının sebebi, buna bağlı olan toplayıcı işleminin tamamlanmamış olmasıdır. Bu adımlar tüm düğümlerin zamanlama işleminin tamamlanmasına kadar devam etmektedir.

İşlemler		Başlama zamanı
Çarpıcı	toplayıcı-çıkarıcı	
D1,D2,D6	D10	1
-	D11	2
D3,D7,D8	-	3
-	-	4
-	D4	5
-	D5	6
-	D9	7

Tablo 3.3. liste zamanlama algoritma çıktısı.

Tanımlanan kaynak kısıtlarını uygulayarak yapılan liste zamanlama algoritmasının son durumu şekil 3-7 de gösterilmektedir.



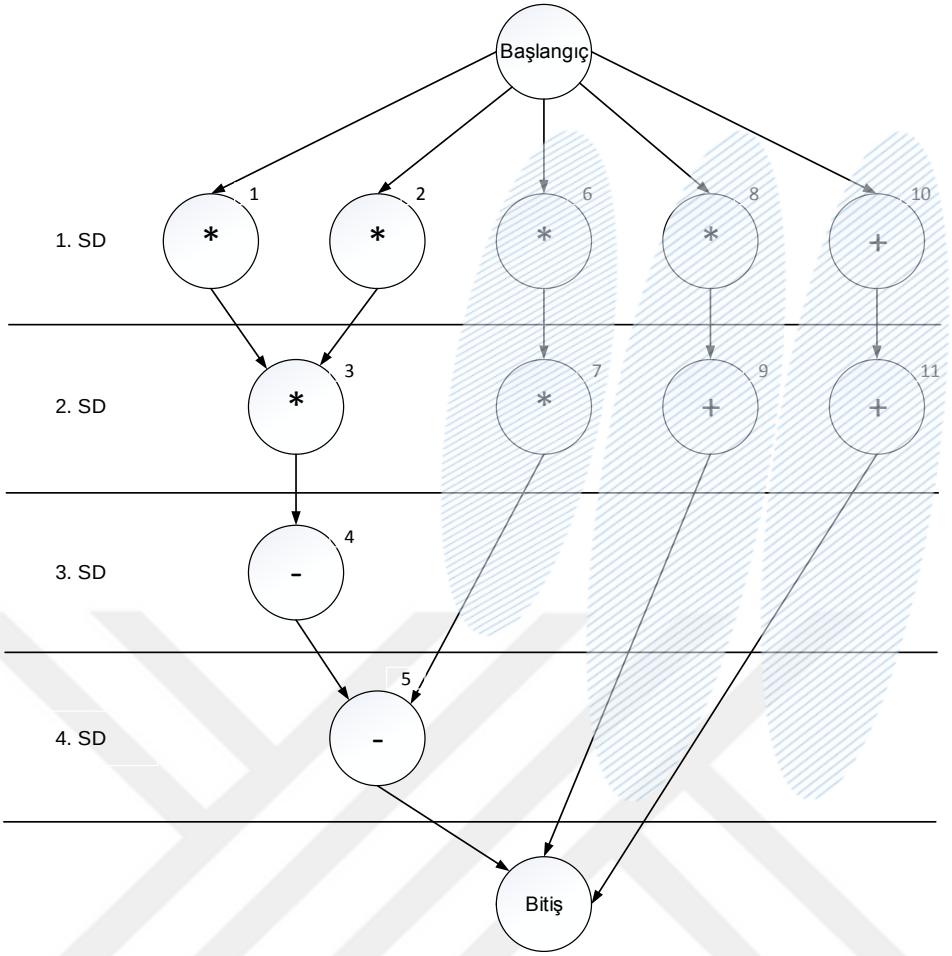
Şekil 3.7. Liste zamanlama algoritma çıktısının son hali.

3.8. Hareketlilik

ASAP algoritması sonucunda çıkan VAÇ'ye bakıldığında işlemlerin başlangıç zamanı olabildiği kadar erken saat döngüsünde zamanlanmaktadır. ALAP algoritması sonucunda çıkan VAÇ'de ise düğümler ASAP algoritmasından alınan toplam gecikme zamanı içerisinde en geç saat döngüsüne zamanlanmaktadır. Bu zamanlama işlemlerinde bazı düğümler kritik yol üzerinde yer almaktadır. Kritik yol üzerinde her işlemin hareketlilik değerinin sıfır olduğu bilinmektedir. Kritik yol üzerinde olan düğümlerin toplam çalışma zamanı ise tasarımın gecikme zamanı olarak değerlendirilir.

Örnek olarak Şekil 3-8 de gösterilen VAÇ'ye bakıldığında düğüm 1,2,3,4 ve 5 kritik yol üzerinde yer almaktadır ve hareketlilik değeri 0'dır. Düğüm 6 ve 7 nin hareketlilik değeri ise 1'dir dolayısıyla 2. ve 3. saat döngüsüne zamanlanabilmektedir. Aynı şekilde 8,9,10 ve 11 düğümlerinin hareketlilik değerleri 2 olarak bilinmektedir. Örneğin 9 numaralı düğüm 2.,3. ve 4. saat döngüsünde zamanlanabilmektedir.

Uygulamalarda tanımlanan gecikme zamanını azaltmak için kritik yol üzerinde yer alan bazı düğümlerin kaynaklarının daha hızlı çalışan kaynaklarla yer değiştirilmesi uygun olur. Fakat şekil 3-8 de gösterilen VAÇ'de kritik yol üzerinde olan tüm düğümlere atanan kaynakların çalışma zamanı en az 1 saat döngüsü olduğu için verilen bu örnekte gecikme zamanını düşürmek mümkün olmamaktadır. Şekil 3-7 de gösterilen VAÇ'de 3 numaralı düğüme çalışma zamanı 1 saat döngüsü olan bir kaynak atanırsa tasarımın gecikme zamanı 7'den 6 saat döngüsüne düşürülecektir. Tasarımda düğümlere atanan kaynakların değiştirilmesi kritik yolun değişmesine sebep olabilmektedir.



Şekil 3.8. Düğümlerin hareketlilik değerleri.

3.9. Bu Tezdeki Fark

Yukarıda anlatılan ASAP, ALAP ve Liste zamanlama algoritmaları bu tezde de kullanılmaktadır. Bu algoritmalarda bir düğümde sadece çarpıcı veya toplayıcı-çıkarıcı işlemi kullanılmıştır. Her bir işlem için işlem türüyle alakalı kaynak atanmaktadır aynı zamanda bu tezde kullanılan şekilde her bir kaynağın kendi içinde farklı versiyonları olabilir. Örnek olarak bir çarpıcı işlemi için tek bir çarpıcı kaynak olurken her bir çarpıcı kaynağının kendi içinde farklı versiyonları da bulunabilmektedir. Kaynakların her bir versiyonunda farklı alan, gecikme ve güvenilirlik değerleri mevcuttur.

Bu tezde kaynaklar ve kaynaklara ait versiyonları kaynak kütüphanesinde tutulmaktadır. Tablo 3-4 de gösterilen örnek kaynak kütüphanesine bakıldığında toplayıcı-çıkarıcı kaynağı için üç farklı versiyon mevcuttur ve her versiyon farklı alan, gecikme ve güvenilirlik değerlerine sahiptir. Bu tezde sistemin genel

güvenilirliği artırılırken, kullanılan alan ve gecikme süre değerlerine dikkat edilerek bu kütüphanedeki kaynaklar kullanılmaktadır.

	Kaynak türü	Alan	Gecikme	Güvenilirlik
1	Toplayıcı-çıkarıcı 1	1	2	0.999
2	Toplayıcı-çıkarıcı 2	2	1	0.969
3	Toplayıcı-çıkarıcı 3	4	1	0.987
4	Çarpıcı 1	2	2	0.999
5	Çarpıcı 2	4	1	0.969

Tablo 3.4. Kaynak kütüphanesi.

3.10. Kaynak Payşalma ve Atama

Şekil 3-2 de gösterilen VAÇ'de 1 ve 2 numaralı düğümler birbirinden bağımsız oldukları için işlemleri aynı zamanda başlatılabilmektedir. Bunun yanında 1 numaralı düğüm ve 3 numaralı düğüm birbirine bağlı olup 3 numaralı düğümün başlatılması için 1 numaralı düğümün bitmesi gerekmektedir. Bu durumda eğer birden fazla çarpıcı işlemi kaynağı mevcut olursa düğüm 1 ve 2 aynı zamanda başlatılbilmektedir fakat düğüm 1 ve 3 hiç bir şekilde aynı zamanda başlatılmaz.

Eğer bir kaynak birden fazla işlem arasında paylaştırılırsa buna kaynak paylaşma denir. Kaynak paylaşımın en önemli amacı, kaynakları eşzamanlı olmayan işlemler arasında dağıtarak kullanılan alan değerini düşürmektir. Kaynak atama ise kaynak paylaşımının bu özelliğini kullanarak kaynakları işlemler arasında uygun şekilde atamaktadır. Kaynak atama işlemi bir kaynağın birden fazla eş zamanlı işleme atanmasına izin vermemektedir. Kaynak atama, sistemin gecikme değerini etkileyebilmektedir. Bunun sebebi aynı kaynağı kullanan işlemlerin eş zamanlı olarak çalışamamasıdır. Bir işlemin çalışmasının başlatılması için paylaşılan kaynağın boş durumda olması gerekmektedir.

3.11. Problem Tanımlama

Bu tezde kaynakları işlemler arasında doğru bir şekilde atayarak sistemin genel güvenilirliği artırılmaktadır. Tablo 3-4 de listelenen kaynaklarda, her bir kaynağın farklı alan, gecikme ve güvenilirlik değerleri mevcuttur. Eğer sistem tasarımı için hiç bir kısıt belirlenmemişse, kaynak kütüphanesinden en yüksek güvenilirlik değerine sahip olan kaynakları düğümlere atayarak sistem yumuşak hatalara karşı daha dirençli hale getirilir. Bu durumda hep yüksek güvenilirlik değerine sahip

kaynaklar atandığında sistemin kullandığı alan yüksek, performansı ise düşük olacaktır.

Tezin devamında kombinasyonel devrelerin yumuşak hatalara karşı daha dirençli olması için bir yöntem sunulmuştur. Bu yöntemin uygulaması için belirlenen VAÇ yöntemin girdisi olarak verilmektedir. Bir önceki bölümde anlatılan ve örnek olarak Şekil 3-2 de gösterilen VAÇ'de düğümler işlemleri işaret etmekte olup düğümler arasındaki bağlar ise düğümlerin birbirine olan bağlantılarını göstermektedir. Yöntemde kullanılan VAÇ'de başlangıç ve bitiş düğümleri uygulamanın bir parçası olarak görünmemektedir ve hiç bir alan ve kaynak kullanmamaktadır. Bu başlangıç ve bitiş düğümleri uygulamanın ilk başladığı ve bittiği noktaları göstermek için kullanılmaktadır.

Bu tezdeki yöntemin uygulamasına verilen diğer girdi değeri kaynak kütüphanesi olarak belirlenmektedir. Bu kaynak kütüphanesinde bir kaynak türü için birden fazla versiyon mevcut olabilir ve her versiyon farklı alan, gecikme ve güvenilirlik değerine sahip olabilmektedir. Kaynak kütüphanesinde belirlenen güvenilirlik değeri, ilgili kaynağın yumuşak hatalara karşı ne kadar dirençli olabileceğini göstermektedir. Güvenilirlik değeri ne kadar büyük olursa kaynak o kadar hatalara karşı dirençli olabilir. Tablo 3-4 de gösterilen kaynak kütüphanesi [3] makalesinden alınmıştır (bu tezdeki yöntemi [3]'deki yöntemle karşılaştırmak için alınmıştır). Tezde anlatılan yöntem için çeşitli kaynaklar ve kaynak versiyonları kullanılabilmektedir.

Yöntemin uygulamasına verilen son girdi değeri, kısıtlanmış alan ve gecikme değeri olarak tanımlanmaktadır. Eğer her bir düğüm için bir tane kaynak atanmış olursa tasarımın kullandığı alan fazla olur, dolayısıyla bu yöntemin çıktısı masraflı bir çözüm oluşturacaktır. Bu durumu engelleyebilmek için son tasarıma alan kısıtı getirilir. Belirlenen alan kısıtını karşılamak için kaynak paylaşım yönteminin kullanılması gerekmektedir [40]. Eğer her bir işlem türü için bir tane kaynak kullanılmış olursa tasarımın toplam alan değeri düşük ancak gecikme değeri artımış olacaktır. Böylece her bir işlem için dengeli bir şekilde kaynak kullanılması gerekmektedir. Bir önceki çalışmalara bakıldığında genelde gecikme kısıtı altında alan değerini düşürmek ve ya alan kısıtı altında gecikme değerini azaltmak için yöntemler sunulmuştur. Bu tezde ise alan ve gecikme kısıtını aşmadan tasarımın

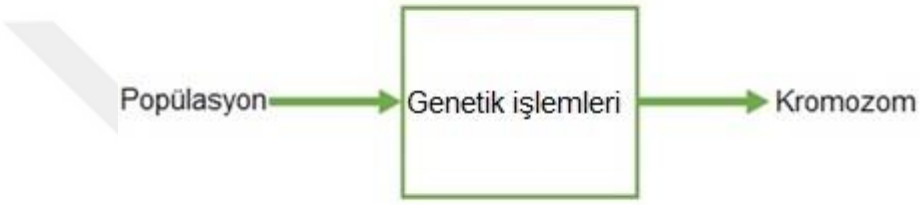
son güvenilirliğini artırmak için bir yöntem anlatılmıştır. Sistemin genel güvenilirlik değerini hesaplamak için aşağıdaki formül kullanılmaktadır.

$$R_S = \prod_{i=0}^{n} R_{i \to r} \qquad (1)$$

Bu formülde ***i*** düğümü ve ***r*** kaynağı ifade eder. ***Ri --> r*** ise ***i*** düğümüne ***r*** kaynağı atandığında bu düğümün güvenilirlik değerine işaret etmektedir. Aslında bu tezde sunulan yöntem, zamanlama ve kaynak atama için kullanılan bir yüksek seviye sentezleme aracı olarak da ifade edilebilir. Kaynaklar düğümlere atanarak her bir düğümün işleminin başlama zamanı belirlenir.

4. GENETİK ALGORİTMA

Genetik algoritma arama tabanlı bir optimizasyon tekniği olarak tanınmakta olup bilgisayar biliminde NP-Zor problemlerini çözmek için iyi bir yöntem olarak bilinmektedir. NP-Zor problemlerinde sonuca varmak için en güçlü bilgisayarların günlerce hatta aylarca çalışılmaları gerekmektedir. Bu algoritmada en iyi veya iyiye yakın çözümler, olabilecek en kısa zamanda bulunmaktadır. Şekil 4-1 de gösterildiği gibi bu teknikte veriler algoritmaya girdi olarak verilip bu veriler içinden en iyi çözüm veya çözümler algoritmanın çıktısı olarak belirlenmektedir.



Şekil 4.1. Genetik algoritma genel işleyişi

Genetik algoritma işlemi başlatılmadan önce ilgili problem için olası çözümlerden bir havuz oluşturulmaktadır. Daha sonra algoritmanın işleyişi sırasında bu olası çözümler üzerinde mutasyon veya çaprazlama yapılarak yeni çözümler üretilmektedir. En iyi çözüm veya çözümler bulunana kadar bu işlem tekrarlanmaktadır.

4.1. Genetik Algoritmasyıla Alakalı Kullanılan Terimler

Genetik algoritmanın genel işleyişi anlatılmadan önce, genetik algoritmada ve bu tezde kullanılan bazı terimler aşağıda listelenmiştir:

4.1.1. Popülasyon

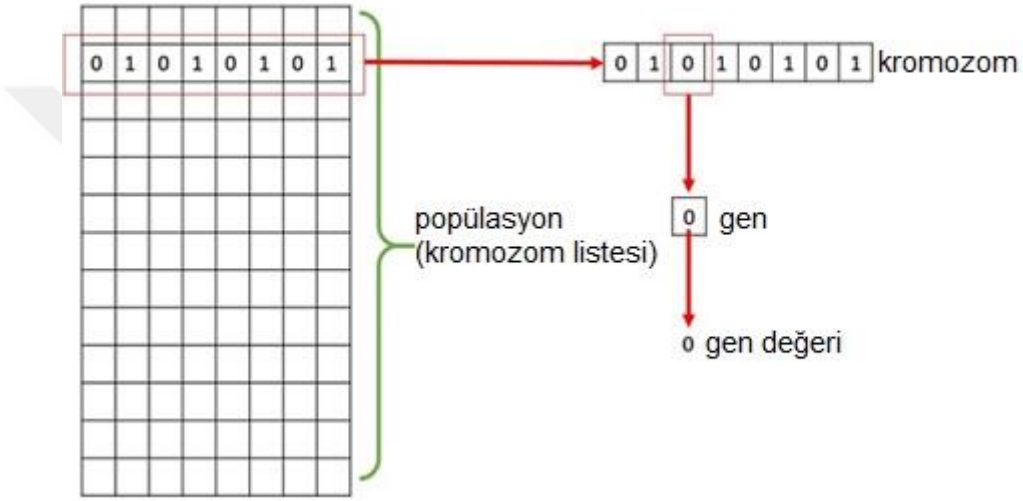
Olası çözümlerin bir araya gelmesiyle oluşan listeye popülasyon adı verilmektedir. Popülasyonu oluşturan olası çözümlerin sayısının doğru bir şekilde belirlenmesi önem taşımaktadır. Genelde bu sayı, problem yapılarına göre tasarımcı tarafından belirlenmektedir. Bu tezde popülasyondaki olası çözümler sayısı, işlem sayısı * 3 formülünden elde edilmektedir. Örnek olarak Şekil 3-2 de gösterilen VAÇ'de işlem sayısı 11 tane olup popülasyonu oluşturan çözümlerin sayısı 33 olmaktadır.

4.1.2. Kromozom

Popülasyonun içerdiği olası çözümlerden birine kromozom adı verilmektedir.

4.1.3. Gen yapısı

Probleme ait en küçük bilgiyi taşıyan değere gen adı verilmektedir. Probleme ait bu genlerin bir araya gelmesiyle kromozom oluşmaktadır. Gen, problem yapısına göre ikili ve sembol gibi farklı değerlerle tanımlanabilmektedir. Bu tezde her bir gen kaynak numaralarına işaret etmektedir.



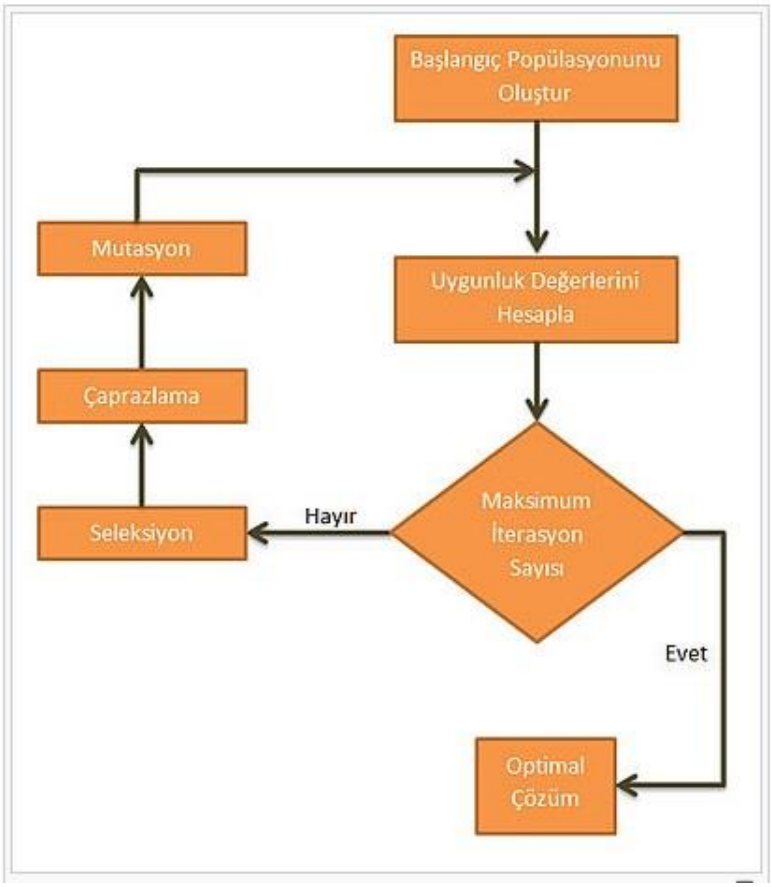
Şekil 4.2. Genetik algoritma yapısı

Şekil 4-2 de gösterildiği gibi popülasyon olası çözümlerden oluşmakta olup her bir olası çözüme kromozom adı verilmekedir. Her bir kromozom problem için iyi bir çözüm olabilmektedir ve bu kromozomlar üzerinde ileride anlatılacak çaprazlama ve mutasyon işlemleri ile popülasyon için yeni kromozomlar üretilebilmektedir. Her bir kromozom farklı genlerden oluşmakta olup bu tezdeki yapıya göre her bir gen bir işleme atanan kaynak numarasına işaret etmektedir [41].

4.2. Genetik Algoritmanın Genel İşleyişi

Genetik algoritma adımları şekil 4-3 de gösterildiği gibi çalışmaktadır. Bu algoritma işleme başlatıldığında ilk adımı popülasyon oluşturmaktır. Popülasyonun oluşturulması için genel olarak Rasgele veya Sezgisel yöntemleri kullanılmaktadır. Rasgele yönteminde popülasyon içindeki kromozomlar, çözülmesi gereken probleme göre rasgele değer almakta olup Sezgisel adlı yöntemde, problem

çözümüne yakın olan bir kaç kromozom oluşturulmakta ve diğer kromozomlar ise bir önceki yöntemdeki gibi rasgele oluşturulmaktadır [41].



Şekil 4.3. Genetik algoritma çalışma adımları

Popülasyon içindeki her bir kromozomun problem için iyi çözüm olup olmadığına uygunluk fonksiyonu karar vermektedir. Bu fonksiyonun girdisi olarak kromozom gönderilip bu kromozomun ne kadar iyi olup olmadığı hesaplanır ve fonksiyonun çıktısı uygunluk olarak belirlenir. Fonksiyondan çıkan, kromozumun uygunluk değeri düşük olursa kromozom üzerinde çaprazlama veya mutasyon yöntemleri kullanılarak yeni kromozom oluşturulur ve bu kromozomla yer değiştirilir.

Çaprazlama işleminde uygunluk fonksiyonundan çıkan değere göre kromozomun bazı genleri, uygunluk değeri yüksek olan kromozomla yer değiştirilir ve böylece yeni bir kromozom elde edilir. Mutasyon işlemi ise kromozomun içindeki genlerin birbiriyle yer değiştirmesi şeklinde gerçekleşmektedir.

İşlemler maksimum iterasyon sayısı kadar tekrarlanmaktadır. Bazı problemlerde bu işlemler maksimum iterasyon sayısına bağlı olmadan tekrarlanır ve bir kromozomun uygunluk değeri, optimum değere yakın veya aynı olduğu zaman tekrarlama işlemi sonlanır daha sonra bu kromozom algoritma çıktısı olarak tanımlanır. Bazı problemlerde ise bu işlemler maksimum iterasyon sayısı kadar devam eder ve döngü işlemi tamamlandıktan sonra birden fazla uygunluk değeri yüksek olan kromozom algoritma çıktısı olarak belirlenir [42].

5. GENETİK ALGORİTMA TABANLI YÖNTEM

Bu bölümde Yüksek Seviye Sentezleme problemini çözmek için uygulanan genetik algoritma yöntemi anlatılmıştır. Bir önceki bölümde anlatıldığı gibi bu tezde ele alınan problem bir NP-sert problemi olarak bilinmektedir. Bu problemi çözmek için daha önce yapılan tamsayılı doğrusal programlama tabanlı yöntem ve sezgisel yöntemin dezavantajlarını gidermek için genetik algoritma tabanlı yöntem sunulmuştur. Bir önceki bölümde anlatılan genetik algoritma adımları tezdeki algoritmada kullanılmakta olup bu adımların problem tanımına uygulanışı aşağıda anlatılmıştır.

5.1. Popülasyon Üretimi

Genetik algoritmayla çözülen diğer problemler gibi bu tezde anlatılan yöntemde de ilk önce popülasyon üretilmektedir. Bir önceki bölümde anlatıldığı gibi popülasyon kromozomlardan oluşur ve her bir kromozom kaynak dağıtım problemi için bir çözüm olabilir. Bu yöntemde algoritma girişine verilen VAÇ'nin toplam n düğümden oluştuğu düşünüldüğünde popülasyondaki kromozom sayısı n*3 olarak belirlenir. Örnek olarak Şekil 3.2 de gösterilen VAÇ için kromozom sayısı toplam 39 (13*3) olarak belirlenmektedir. Belirlenen bu sayı sezgisel olarak tanımlanmakta olup algoritmanın yönetimini kolaylaştırır. Bu yöntemde kullanılan ve Şekil 3.2 de gösterilen VAÇ'ye ait kromozom yapısı Şekil 5.1 de gösterilmiştir.

Düğüm numaraları	1	2	3	4	5	6	7	8	9	10	11
Kaynaklar	Ç2	Ç2	Ç1	T1	T2	Ç2	Ç1	Ç2	T1	T3	T3

Şekil 5.1. Rasgele oluşan Şekil 3.2'de gösterilen VAÇ için örnek bir kromozom.

Yukarıdaki şekilde gösterildiği gibi her kromozom genlerden oluşmakta olup kromozomdaki her gen VAÇ'deki bir düğüme işaret etmektedir. Her genin tuttuğu değer ise kaynak kütüphanesindeki kaynaklara işaret etmektedir. Örnek olarak bir numaralı gen, Tablo 3.4 de gösterilen kaynak kütüphanesindeki ikinci çarpıcı kaynağına işaret etmektedir. İkinci bir örnek olarak 4 numaralı gen kaynak kütüphanesindeki birinci toplayıcı-çıkarıcı kaynağına işaret etmektedir. Bu tezde sunulan yöntemde gen değerlerine rasgele kaynak numarası atanmaktadır.

Kromozomdaki genlere kaynaklar rasgele atandıktan sonra popülasyon oluşturulur (Tablo 5.1). Oluşan popülasyondan iki kromozom rasgele seçilip bu iki kromozom üzerinde yüksek seviye sentezleme işlemi yapılır ve uygunluk değeri hesaplanır. Yüksek seviye sentezleme işleminin ilk adımında işlemlerin başlangıç zamanı hesaplanır. İşlemlerin başlangıç zamanını hesaplamak için liste zamanlama algoritması kullanılmaktadır [42].

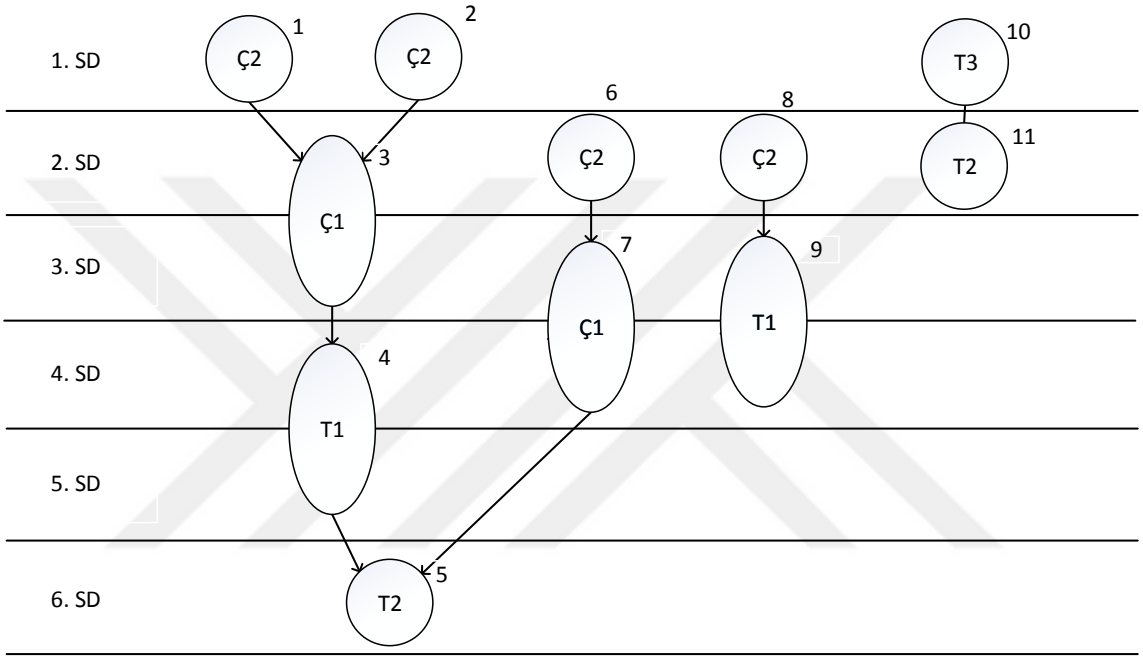
	1	2	3	4	5	6	7	8	9	10	11
1	5	4	4	2	2	5	5	4	2	1	3
2	5	4	4	2	1	5	5	4	2	1	3
3	4	4	4	3	2	4	5	5	3	2	1
4	5	5	4	3	3	5	4	4	2	3	2
5	5	5	5	2	2	5	5	5	2	1	3
6	5	4	5	3	2	5	4	5	2	2	2
7	4	5	4	3	3	4	4	4	3	3	2
...	...	...	...	...	...	...	...	...	...	...	...
32	5	5	5	3	3	4	5	5	2	3	2
33	4	4	5	2	3	4	4	4	3	1	2

Tablo 5.1. Kromozomlardan oluşan örnek bir popülasyon (sutün başlığı işlem numarasına ve satır başlığı kromozom numarasına işaret etmektedir).

Liste zamanlama algoritmasında ilk önce ASAP ve ALAP zamanlama algoritmaları kullanılarak düğümlerin hareketlilik değerleri hesaplanır. Daha sonra tasarımın kullandığı alan değerini minimize etmek için liste zamanlama algoritması uygulanır. Bu adımda dikkat edilmesi gereken husus; eğer bir kromozom da x kaynak r kere atanmışsa, tasarımın genel alanı için atanan x kaynağının alanı r kere toplanmamaktadır, çünkü bu tezde sunulan yöntemde kaynak paylaşım tekniği kullanılmıştır. Kaynak paylaşım tekniğinde bir kaynak boş olsa da diğer işlemler tarafından kullanılabilir. Bu yüzden tasarımın kullandığı alanı minimize etmek için işlemler üzerinde zamanlama uygulanmaktadır.

Zamanlama işlemi bitikten sonra alan, gecikme ve tasarımın genel güvenilirlik değeri hesaplanır. Güvenilirlik değeri sistemin uygunluk değeri olarak bilinmekte olup (1) numaralı formülü kullanılarak hesaplanır. Örnek olarak şekil 5.1 de

gösterilen kromozom üzerinde zamanlama işlemi yapıldıktan sonra Şekil 5.2 de gösteilen VAÇ elde edilir. Diğer bir örnek olarak, eğer tasarımda alan kısıtı 11 olarak, gecikme kısıtı 5 saat döngüsü olarak tanımlanmışsa, elde edilen örnek kromozomun VAÇ'sine bakıldığında bu kısıtların sağlanmadığı gösterilebilir. Belirlenen bu kısıtları sağlamak için asağıda anlatılan genetik algoritma adımları uygulanabilir.



Şekil 5.2. Şekil 5.1'de gösterilen kromozom üzerinde zamanlama işlemi yapıldıktan sonra elde edilen VAÇ.

5.2. Genetik işlewleri

Popülasyon üretildikten sonra bir çift kromozom seçilir. Seçilen kromozomlar uygunluk fonksiyonuna girdi olarak verilip her bir kromozomun uygunluk değeri hesaplanmaktadır. Bu iki koromozomdan uygunluk değeri yüksek olan kromozoma kazanan kromozom ve uygunluk değeri düşük olan kromozoma kaybeden kromozom adı verilir. Kazanan kromozomun gen değerleri değişmez, kaybeden kromozom üzerinde ise mutasyon ve çaprazlama işlemi yapılarak yeni bir kromozom üretilir. Bu kromozomun yerine kaydedilir. Bu işlem tanımlanan iterasyon sayısı kadar tekrarlanmakta, bu döngüden çıktıktan sonrada popülasyondaki her bir kromozomun uygunluk değeri kısıtlanan alan ve gecikme

değeri altında en iyi veya iyiye yakın olmaktadır. En sonunda popülasyondaki en yüksek uygunluk değerine sahip olan kromozom algoritma çıktısı olarak belirlenir. Bu yöntem ile birden fazla zamanlama tasarımı elde edilebilir.

5.2.1. Seçmeli Mutasyon

Genetik algoritmada bir kromozomdan daha iyi bir kromozom elde etmek için mutasyon işlemi gerçekleştirilmektedir. Mutasyon işleminde kromozoma ait bir veya birden fazla gen yer değiştirilmekte ve yeni bir kromozom elde edilmektedir. Bu tezde uygulanan mutasyon işlemi seçilen çift kromozomdan kaybeden kromozom üzerinde yapılmakta olup üretilen yeni kromozom bu kromozom yerine kaydedilmektedir. Bazı durumlarda ise kaybeden kromozomun bazı genlerindeki değerleri (kaynak numaraları) değiştirilerek yeni bir kromozom elde edilebilir. Bu tezde yapılan mutasyon işlemleri kısmen rasgele olarak yapılmaktadır. Örnek olarak zamanlama işlemi yapıldıktan sonraki yani Şekil 5.2 de gösterilen VAÇ'ye bakıldığında alan ve gecikme kısıtları sağlanmadığı görülür. Gecikme kısıtının sağlanması için VAÇ'de kritik yol üzerinde olan (hareketlilik değeri 0 olan) bazı düğümlerin kaynakları daha hızlı kaynaklarla yer değiştirilir. Eğer kısıtlanan alan değeri hala karşılanmadıysa seçmeli mutasyon işlemi uygulanarak işlemlerin kaynakları değiştirilmektedir. Tasarımda kullanılan alan değerini düşürmek için kromozomdaki düğümler tarafından en az kullanılan kaynak kaldırılıp yerine daha çok kullanılan kaynak atanarak tasarımda kullanılan kaynak sayısı düşürülür, bu şekilde tasarımın genel alan değerinin düşük olması sağlanır.

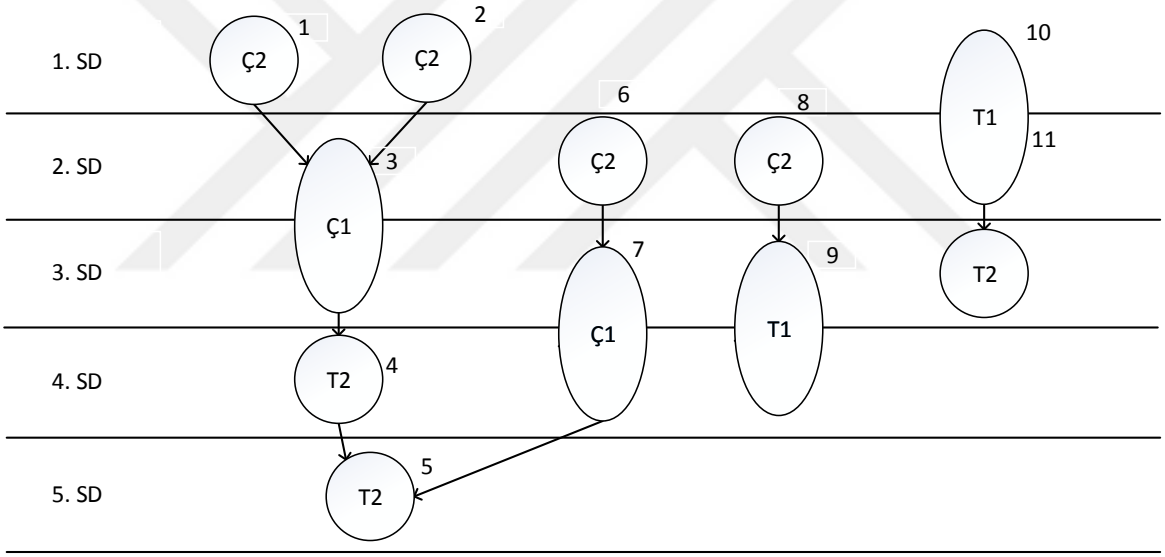
Örnek olarak Şekil 5.2 de gösterilen VAÇ'ye ait kromozomun yapısına bakıldığında 1,2,3,4 ve 5 düğümlerinin kritik yol üzerinde ve 0 hareketlilik değerinde olduğu görülür. Bu kromozomda kullanılan gecikme değerini düşürmek için düğüm 3 ve 4'ün kaynakları daha hızlı kaynaklarla yer değiştirilir. Tezde kullanılan yöntemde düğüm 4'ün rasgele seçildiği düşünülürse bu düğüme atanan A2 kaynağı yerine A1 kaynağının atanması gecikme zamanını düşürür.

Aynı şekilde kullanılan toplam alan değerini düşürmek için kromozomda en az kullanılan kaynak yani A3 kaynağı seçilir. Bu kaynağı kullanan düğüm veya düğümlere en çok kullanılan kaynak yani A1 kaynağı atanırsa kullanılan alan değeri düşürülür. Şekil 5.1 de gösterilen kromozom üzerinde bu değişiklikler yapılarak Şekil 5.3 de gösterilen kromozom elde edilir.

Düğüm numaraları	1	2	3	4	5	6	7	8	9	10	11
kaynaklar	Ç2	Ç2	Ç1	T1	T2	Ç2	Ç1	Ç2	T1	T3	T3

Şekil 5.3. Mutasyon işlemi uygulandıktan sonra elde edilen kromozom.

Seçmeli mutasyon işlemi yapıldıktan sonra elde edilen kromozomun VAÇ'si Şekil 5.4 de gösterilmektedir. Bu VAÇ'ne bakıldığında 10 numaralı düğüme A3 yerine A1 kaynağı atanarak kullanılan toplam alan değeri 17'den 13'e düşürülmüştür. Aynı şekilde 4 numaralı düğüme A1 yerine A2 kaynağı atanarak tasarımın gecikme zamanı 6'dan 5'e düşürülmüştür. Bu işlemler sonunda üretilen yeni kromozom eskisi yerine kaydedilir.



Şekil 5.4. mutasyon işlemi uygulandıktan sonra elde edilen VAÇ.

5.2.2. Çaprazlama İşlemi

Bir kromozomdan yeni bir kromozom üretmek için kullanılan genetik algoritmanın bir diğer adımı çaprazlama olarak bilinmektedir. Genel olarak tek nokta ve çift nokta olarak adlandırılan iki tür çaprazlama işlemi kullanılmaktadır. Genetik algoritmanın çaprazlama işleminde iki kromozomun belirlenen yerleri kesilir ve

birbiriyle değiştirilir. Bu tezde kullanılan yöntemde ise kazanan kromozom değiştirilmemektedir. Bu kromozomdan bazı genler seçilir ve kaybeden kromozoma aktarılır.

Bu tezde uygulanan çaprazlama işlemlerinin bir örneği Şekil 5.5 de gösterilmiştir. Bu örnekte 1 numaralı kromozom kazanan kromozom ve 2 numaralı ise kaybeden kromozom olarak düşünüldüğünde, kazanan kromozomdan bazı genler rasgele olarak seçilip kaybeden kromozoma aktarılır.

Düğümler	1	2	3	4	5	6	7	8	9	10	11	A	P	G
Kazanan	Ç2	Ç2	Ç1	T2	T2	Ç2	Ç1	Ç2	T1	T1	T2	12	5	0,849
Kaybeden	Ç1	Ç1	Ç2	T2	T2	Ç1	Ç1	Ç1	T2	T2	T1	10	7	0,931

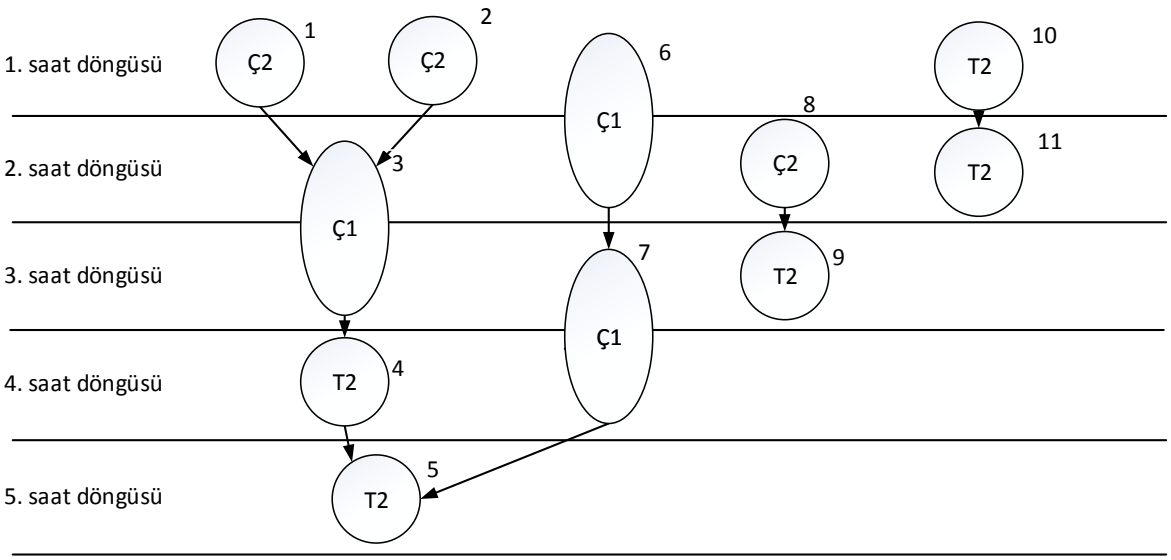
Şekil 5.5. Kazanan ve kaybeden kromozom.

Şekil 5.6 de gösterildiği gibi kazanan kromozomun gen değerleri değişmemekte ve yeni elde edilen kromozom kaybeden kromozom yerine kaydedilmektedir. Üretilen yeni kromozoma ait VAÇ ise Şekil 5.7 de gösterilmektedir.

Düğümler	1	2	3	4	5	6	7	8	9	10	11	A	P	G
Kazanan	Ç2	Ç2	Ç1	T2	T2	Ç2	Ç1	Ç2	T1	T1	T2	12	5	0,849
Kaybeden	Ç2	Ç2	Ç1	T2	T2	Ç1	Ç1	Ç2	T2	T2	T2	10	7	0,931

Şekil 5.6. çaprazlama işlemi yapıldıktan sonra kaybeden kromozomdaki değişim

Çaprazlama işlemi bittikten sonra birinci döngü tamamlanmış sayılır. Yine rasgele olarak iki kromozom popülasyondan seçilir ve yukarıda anlatılan işlemler tekrarlanır. Bu döngü tasarımcının belirlediği sayı kadar devam etmekte olup döngünün sonuna gelindiğinde populasyondaki kromozomların bir çoğunun uygunluk değeri alan ve gecikme kısıtı altında, yüksek ve birbirine yakın olacaktır. Algoritma çıktısında popülasyondan en yüksek uygunluk değerine sahip olan aynı zamanda belirlenen alan ve gecikme kısıtını sağlayan kromozom seçilir ve algoritma çıkışı olarak belirlenir.



Şekil 5.7. Çaprazlama işlemi uygulandıktan sonra elde edilen VAÇ.

Bu tezde kullanılan adımları özetlemek için Şekil 5-8 de algoritma adımları gösterilmiştir. Daha önce de anlatıldığı gibi algoritmanın ilk adımında popülasyon üretilir daha sonra bu populasyon tasarımcı tarafından sayısı belirlenen döngü içine girip iki kromozom a ve b rasgele olarak seçilir(seçilen kromozomlar birbirinden farklı olmak zorundadır). Seçilen bu kromozomların uygunluk değerleri uygunluk fonksiyonu tarafından hesaplanır. Elde edilen uygunluk değerlerine göre en yüksek değere sahip olan kromozom kazanan kromozom diğeri ise kaybeden kromozom olarak belirlenir. En sonunda kazanan ve kaybeden kromozomlar, üzerlerinde algoritma adımlarının(mutasyon , çaprazlama vs) uygulanması için algoritma fonksiyonuna gönderilir. Bu döngü belirlenen sayı kadar tekarlanmakta olup nihayetinde alan ve gecikme kısıtları altında en yüksek uygunluk (güvenilirlik) değerine sahip olan kromozom algoritma çıktısı olarak belirlenir.

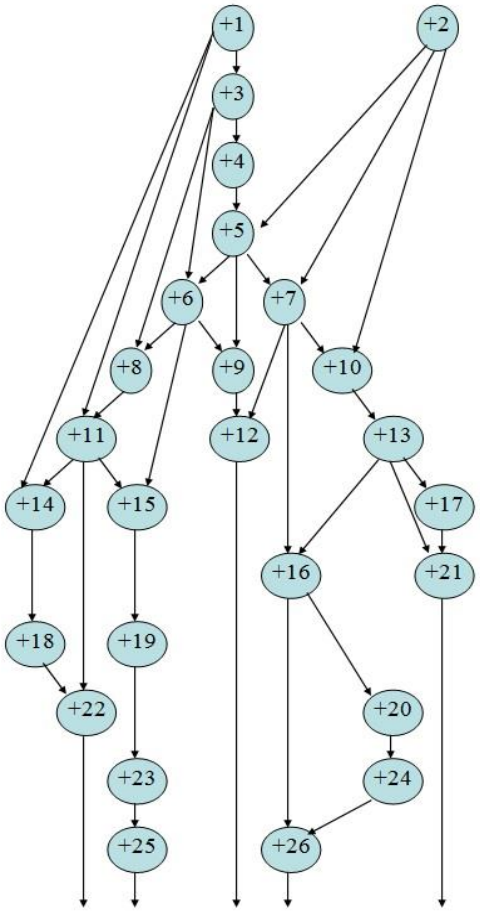
```
Popülasyon üretilir
for (int tornoment = 0; tournament < 1000; tournament ++)
        {
           a = Popülasyondan rastgele bir kromozom seçilir
           b = Popülasyondan rastgele bir kromozom seçilir
           if (a != b)
           {
             uygunlukA = UygunlukFonksiyon(a);
             uygunlukB = UygunlukFonksiyon (b);
              if (uygunlukB < uygunlukA)
              {
                 kazanan = a;
                 kaybeden = b;
              }
              else
              {
                 kazanan = b;
                 kaybeden = a;
              }
              GenetikFonksiyon(kazanan, kaybeden);
           }
        }
 UygunlukFonksiyon(Kromozom a){
          ASAP uygulanır
          ALAP uygulanır
          Liste zamanlama uygulanır
          Güvenilirlik değeri hesaplanıp geri gönderilir
 }
  GenetikFonksiyon(kazanan, kaybeden){
             Mutasyon uygulanır
             Çarprazlama uygulanır
             Elde edilen yeni kromozom kaybeden yerine yazılır
 }
```

Şekil 5.8. Tezde kullanılan algoritma adımları

6.DENEYLER VE SONUÇLAR

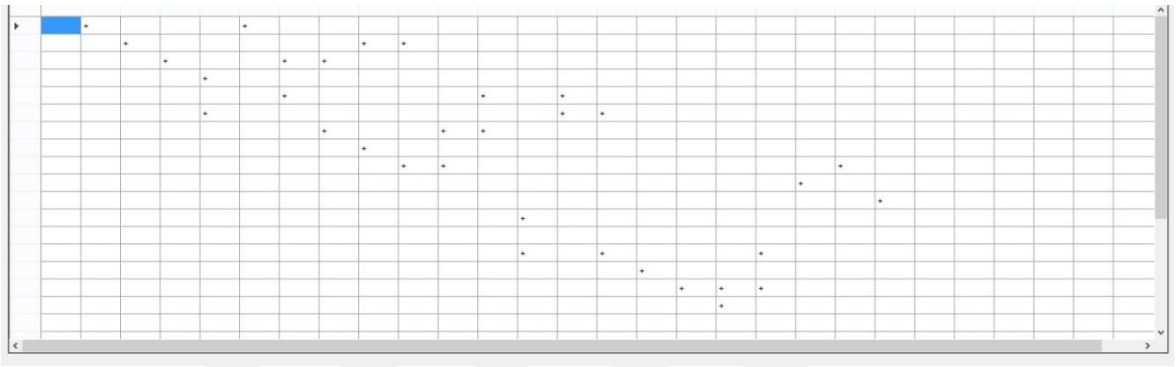
Yukarıda anlatılan yöntemi denemek için bu tezde C# yazılım dilinde bir uygulama geliştirilmiştir. Birinci adımda uygulama girdisi olarak kaynak kütüphanesi girilmektedir. Daha sonra zamanlanması istenilen problem iki boyutlu dizi şeklinde uygulama girdisine verilir ve tasarım için belirlenen alan ve gecikme kısıtı değerleri uygulamaya tanımlanır. Uygulama çalıştırılıp işlemler bittikten sonra popülasyonda tanımlanan alan ve gecikme kısıtı altında en yüksek güvenilirlik değerine sahip olan kromozom çıktı olarak gönderilir.

Örnek olarak Şekil 6-1 de gösterilen EW VAÇ Tablo 3-4 de gösterilen kaynaklarla zamanlanması istenilirse önceden belirlenen kaynak kütüphanesi ve Şekil 6-2'de gösterilen iki boyutlu dizi uygulama girdisi olarak verilir daha sonra Şekil 6-3 de gösterilen iki boyutlu dizi çıktı olarak alınır.

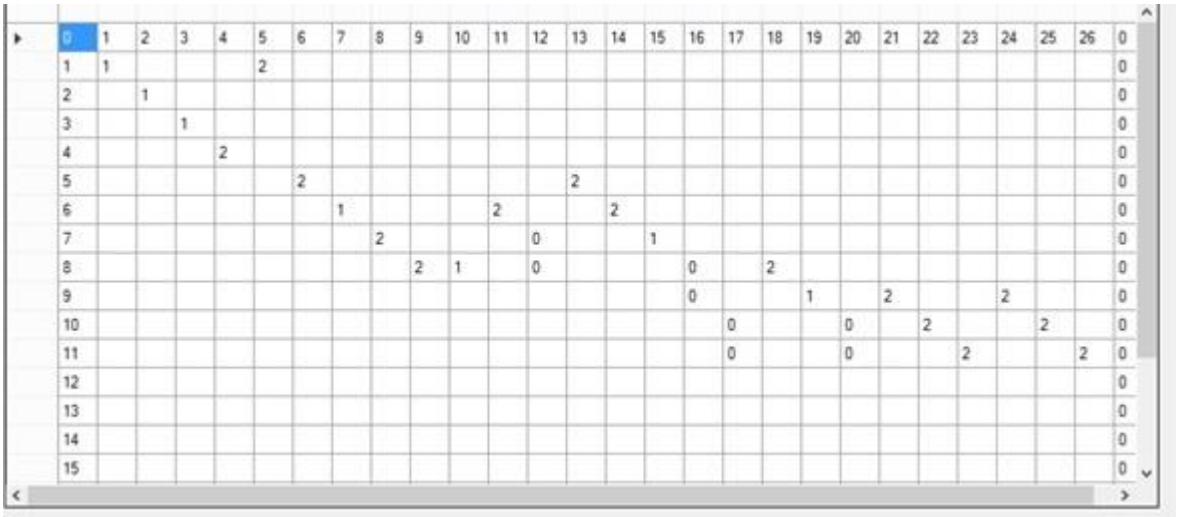


Şekil 6.1. EW Veri Akış Çizelgesi.

Zamanlama için alan kısıtı 12 gecikme kısıtı 11 olarak belirlenmişse, uygulamanın işlemleri sonucunda Şekil 6-3 de gösterilen iki boyutlu dizi uygulama çıktısı olarak belirlenir. Bu dizide satır numaraları saat döngüsüne, sütün numaraları ise düğümlere işaret etmektedir. Dizi içindeki değerler o düğüm için zamanlanan kaynak numarasını göstermektedir. Eğer bu dizi tekrar VAÇ'ye çevirilirse Şekil 5-12 de gösterilen VAÇ elde edilir. Bu uygulamada farklı kaynak kütüphaneleriyle her türlü VAÇ için zamanlama işlemi yapabilir.



Şekil 6.2. Uygulama girişine verilen örnek iki boyutlu dizi.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	0
0																												
1	1					2																						0
2		1																										0
3			1																									0
4				2																								0
5						2								2														0
6							1					2			2													0
7								2					0			1												0
8									2	1			0				0		2									0
9																	0			1		2			2			0
10																		0			0		2			2		0
11																		0			0			2			2	0
12																												0
13																												0
14																												0
15																												0

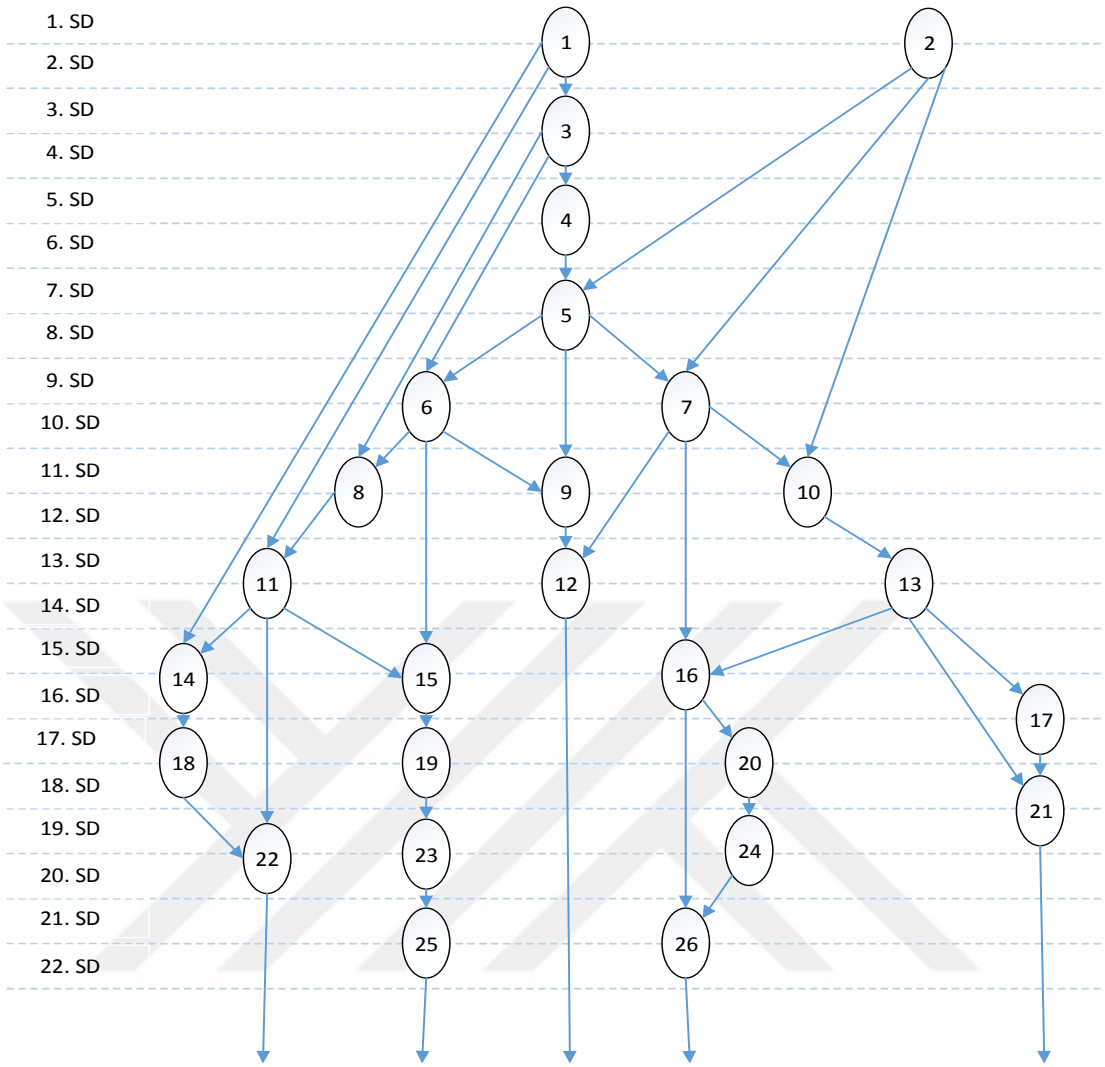
Şekil 6.3. Alan 12 ve gecikme 11 kısıtlar altında uygulama çıktısı.

6.1. Sonuçlar

Bu tezde sunulan yöntem DES (11 düğüm) , EWF (26 düğüm), FIR (23 düğüm), AR (28 düğüm) ve FDT (42 düğüm) veri akış çizelgeleri üzerinde test edilmiştir. Test sırasında her VAÇ için farklı alan ve gecikme kısıtı değerleri verilmiştir. Daha önce anlatıldığı gibi bu tezde her bir kaynağın farklı versiyonları kullanılmaktadır. Eğer bir VAÇ'nin zamanlanması için belirlenen kaynak versiyonlarından sadece bir tanesi kullanılırsa, algoritma çıktısında gecikme değeri yüksek veya güvenilirlik değer düşük olabilmektedir. Bu sebepten dolayı tanımlanan alan ve gecikme kısıtlarını karşılayıp en yüksek güvenilirlik değrini elde etmek için zamanlama işleminde farklı versiyonlar kullanılır. Örnek olarak EW üzerinde agoritma girdisine ayrı ayrı A0, A1 ve A2 kaynak versiyonları verilirse Tablo 6.1 da gösterilen sonuçlar elde edilir. Bu sonuç tablosuna bakıldığında eğer zamanlama için sadece A0 kaynak versiyonu kullanılırsa güvenilirlik değeri en yüksek fakat aynı zamanda gecikme değeri ise fazla olmaktadır. Aynı şekilde eğer zamanlama işlemi için sadece A1 kaynağı atanırsa alan ve gecikme değeri düşük ve güvenilirlik değeri ise düşük olmaktadır. Bu sebepten dolayı zamanlama işlemi uygulandığı sırada alan, gecikme ve güvenilirlik değerlerin arasında bir denge sağlanması için kaynakların farklı versiyonları kullanılır. A0 kaynak versiyonu kullanıldğında algoritma çıktısı şekil 6.4 te gösterilmektedir.

Kullanılan kaynak	Alan	Gecikme	Güvenilirlik değer	Zaman / saniye
A0	3	22	0.974	0,061
A1	6	12	0,440	0,057
A2	12	12	0,711	0,059

Tablo 6.1. EW üzerinde toplayıcı-çıkarıcı kaynak versiyonlarının ayrı ayrı kullanımı.



Şekil 6.4. EW üzerinde A0 kaynak versiyonu kullanıldığında algoritma çıktısı.

Test sonuçları [3] makalesinde önerilen sezgisel yöntemle karşılaştırılıp sonuçlar Tablo 6.2 de gösterilmiştir.

VAÇ	Alan	Gecikme	Kullanılan Alan	Kullanılan Gecikme	Güvenilirlik		
					Sezgisel	Tezdeki yöntem	İyileştirme
DES 11 Düğüm	11	5	11	5	0.775	0.872	%12,52
	13	5	13	5	0.824	0.904	%9,70
	15	5	13	5	0.828	0.904	%9,17
	11	6	11	6	0.824	0.985	%19,53
	13	6	11	6	0.824	0.985	%19,53
	15	6	11	6	0.828	0.985	%18,96
	7	7	7	7	0.903	0.959	%6.20
	9	7	9	7	0.824	0.977	%18.56

	11	7	11	7	0.828	0.987	%19.20
	DES üzerinde ortalama iyileştirme oranı						%14.82
EWF 26 Düğüm	7	13	7	13	0.703	0.687	%-2,32
	9	13	9	13	0.785	0.793	%1,02
	11	13	10	13	0.785	0.813	%3,57
	7	14	7	14	0.711	0.853	%19.97
	9	14	7	14	0.794	0.853	%7,43
	11	14	7	14	0.794	0.853	%7,43
	5	15	5	15	0.694	0.694	0
	7	15	7	15	0.804	0.873	%8,58
	9	15	9	15	0.804	0.883	%9.83
	EWF üzerinde ortalama iyileştirme oranı						%6,17
FIR 23 Düğüm	9	10	9	10	0,599	0,657	%9,68
	11	10	11	10	0,695	0,756	%8,78
	13	10	12	10	0,695	0,84	%20,86
	9	11	9	11	0,789	0,855	%8,36
	11	11	10	11	0,897	0,897	0
	13	11	10	11	0,897	0,897	0
	9	12	9	12	0,813	0,897	%10,33
	11	12	11	12	0,908	0,908	0
	13	12	11	12	0,908	0,908	0
	FIR üzerinde ortalama iyileştirme oranı						%6,45
AR 28 Düğüm	14	11	14	11	0,716	0,741	%3,49
	16	11	16	11	0,771	0,841	%9,08
	18	11	17	11	0,861	0,861	0
	20	11	17	11	0,861	0,861	0
	12	12	12	11	0,674	0,714	%5,93
	14	12	14	12	0,761	0,761	0
	16	12	16	11	0,820	0,841	%2,56
	20	12	18	12	0,882	0,882	0
	AR üzerinde ortalama iyileştirme oranı						%2,63
FDT 42 Düğüm	8	14	8	14	-	0433	-
	8	16	8	16	-	0,958	-
	9	14	9	14	-	0,958	-
	12	16	12	16	-	0,958	-
	12	13	11	13	-	0,902	-
	13	12	13	12	-	0,924	-
	14	12	13	12	-	0,924	-
	FDT üzerinde ortalama iyileştirme oranı						

Tablo 6.2. DES, EWF, FIR, AR ve FDT üzerinde uygulamadan çıkan sonuçların [3] yöntemiyle karşılaştırılması.

Tablo 6.2 de sonuç tablosuna bakıldığında; tezde sunulan yöntemin sezgisel yönteme göre en iyi durumunda DES ‘de %19,53 , EW’de %19,97, FIR’de %20,86 ve AR’de %9,08 daha yüksek güvenilir değerlerine sahip olduğu ve ortalama olarak da DES %14.82, EWF %6,17 ,FIR %6,45 ve AR %2,63 daha iyi güvenilirlik değerlerinin elde edildiği görülmektedir (Tablo 6.3).

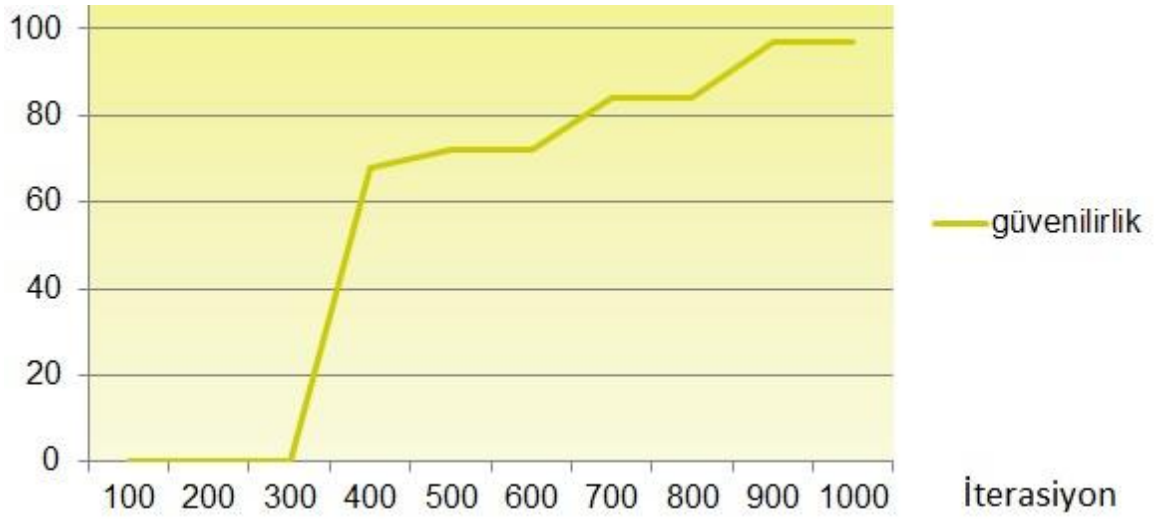
VAÇ	En iyi durumda iyileştirme	Ortalama iyileştirme
DES	%19,53	%14,82
EW	%19,97	%6,17
FIR	%20,86	%6,45
AR	%9,08	%2,63

Tablo 6.3. DES, EW, FIR, AR ve FDT VAÇ’leri için elde edilen sonuçların [3] yöntemiyle karşılaştırılıp ve iyileştirme oranları.

6.2. Sonuç Değerlendirmesi

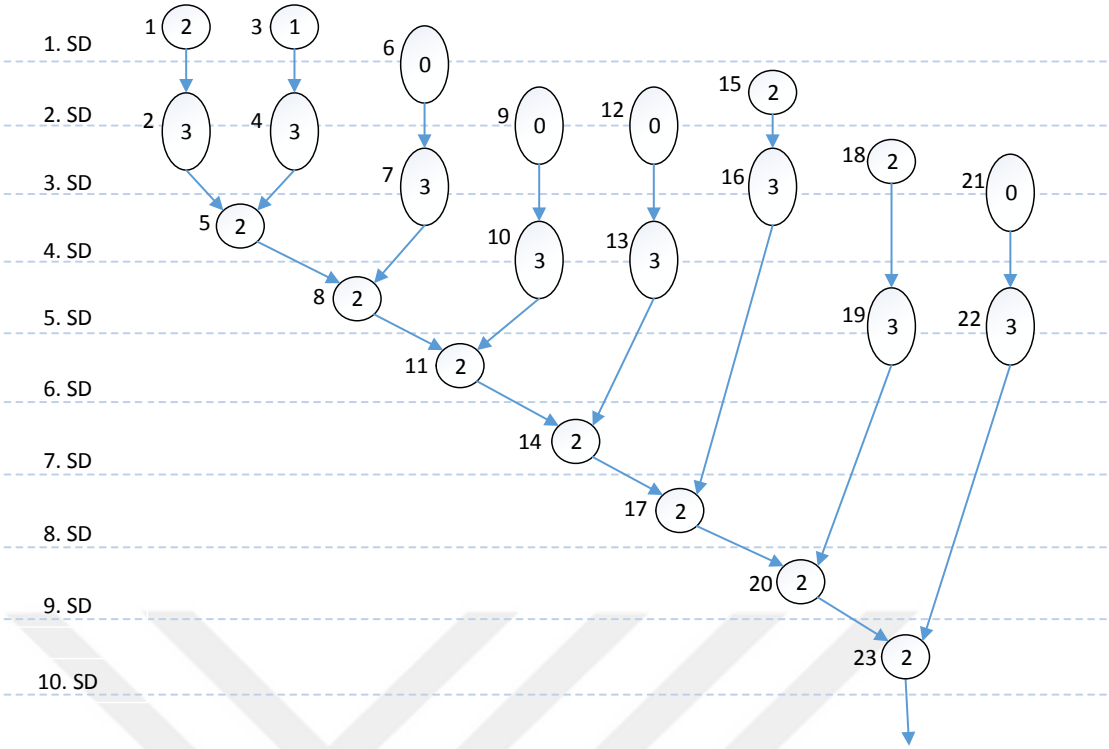
Bu tezde sunulan yöntemin çalışma süresi daha düşüktür ve sonuçlar sezgisel yönteme göre daha hızlı elde edilir. Örneğin uygulama DES’in çözümünü 5 saniye sonunda tamamlarken aynı şekilde EWF için sonucu ortalama 9 saniyede hesaplamaktadır.

Uygulamada döngü içerisinde genetik algoritma adımları uygulandıkça popülasyondaki kromozomların güvenilirlik değeri artış göstermektedir. Örnek olarak eğer DES üzerinde yapılan deneme için alan kısıtı 11 ve gecikme kısıtı 7 olarak belirlenmişse uygulama sonucunda en yüksek güvenilirlik değerine sahip olan bir kromozomun güvenilirlik değeri şekil 5-13’de gösterildiği gibi artırılmaktadır. Uygulamanın ilk başladığı sırada kromozomlar rasgele oluştukları için ilk başta bu kromozomların güvenilirlik değerleri düşüktür veya bu kromozomlar tasarım için tanımlanan kısıtları sağlamamaktadır. İterasyon sayısı arttıkça kısıtlar sağlanmaya başlamakta aynı zamanda güvenilirlik değerleri yükselmektedir.



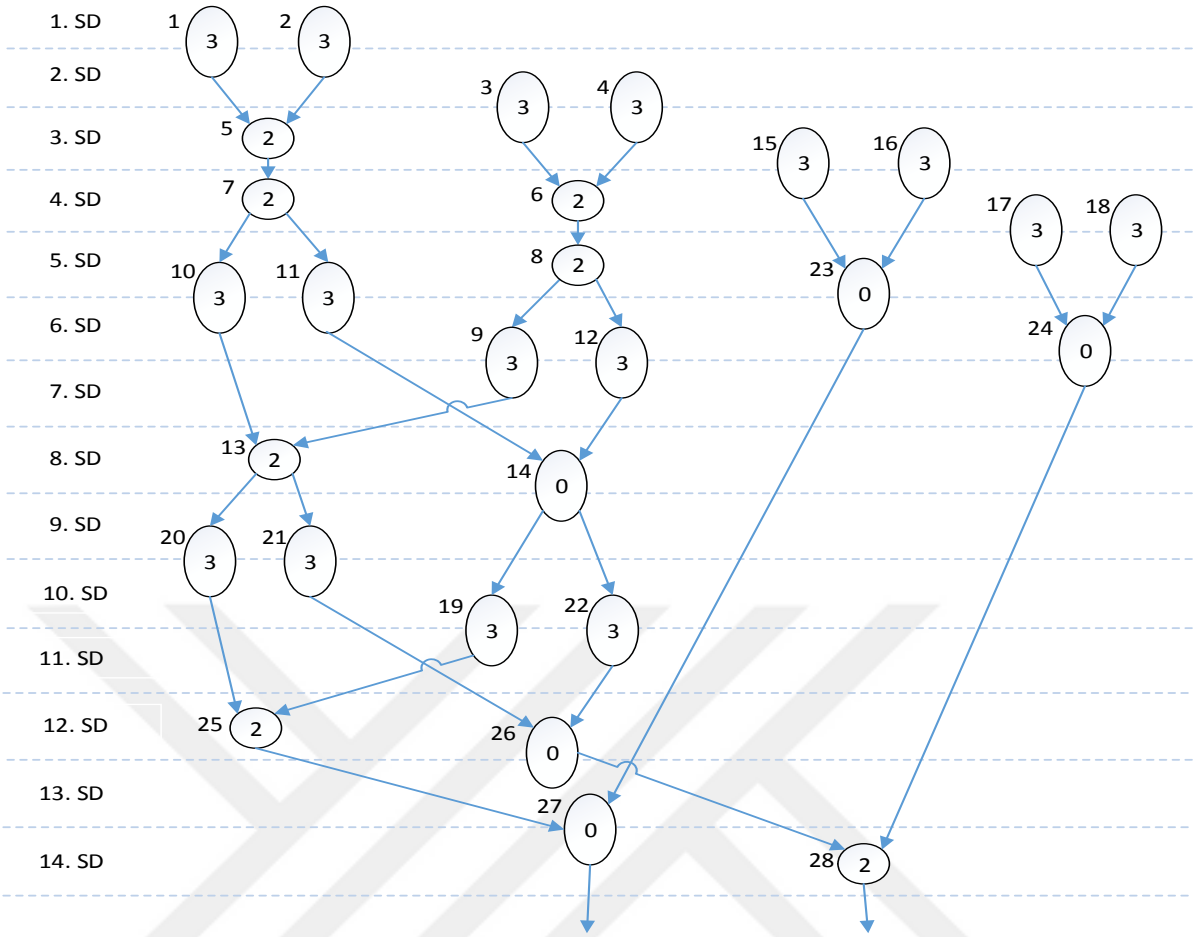
Şekil 6.5. DES üzerinde alan 11 ve gecikme 7 kısıtları altında bir kromozoma ait değişim şekli (0 güvenilirlik kısıtların sağlanmadığı anlamına gelmektedir).

Uygulama işlemi bittikten sonra en yüksek güvenilirlik değerine sahip olan birden fazla zamanlama sonucu elde edilebilir. Örnek olarak FIR üzerinde alan 13 ve gecikme 10 kısıtları altında uygulama çıktısı şekil 6.6 de gösterildiği gibi olabilmektedir. Eğer 1 numaralı düğüme 1 numaralı kaynak ve 2 numaralı düğüme ise 2 numaralı kaynak atanırsa kullanılan alan, gecikme ve güvenilirlik değerleri değişmeden farklı bir zamanlama elde edilebilir. Aynı şekilde düğümlerin başlangıç zamanı değişebilmektedir örneğin 22 numaralı düğüm 5.SD yerine 8.SD'de ise başlatılabilmektedir.



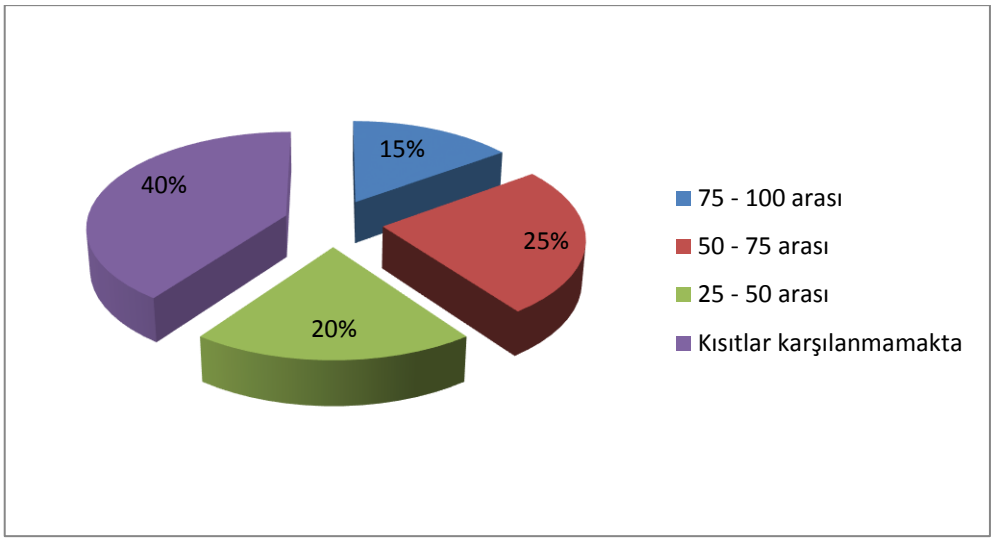
Şekil 6.6. FIR üzerinde alan 13 ve gecikme 10 kısıtları altında algoritma çıktısı.

Bu tezde denemeler için kullanılan veri akış çizergelerden bir diğer örnek ise AR olarak tanınmaktadır. Bu VAÇ denemesinde alan 13 ve gecikme 14 olarak belirlenirse şekil 6.7 de gösterilen zamanlama, uygulama çıktısından alınan örnek bir sonuçtur.



Şekil 6.7. AR üzerinde alan 13 ve gecikme 14 kısıtları altında algoritma çıktısı.

Zamanlama işlemi bittikten sonra popülasyondaki kromozomların güvenilirlik değeri farklı olmaktadır. DES üzerinde yapılan bir denemedeki popülasyonun kromozomlarının son durumu şekil 6.8'deki çizelgede gösterilmiştir. Çizelgeden anlaşıldığı gibi deneme sonucunda kromozomların %15'i en iyi veya iyiye yakın güvenilirlik değerlerine sahiptir. Bunların içinde 3 tane kromozomun güvenilirlik değeri 0.987 olarak hesaplanmıştır yani DES üzerinde yapılan bu denemede 3 tane farklı zamanlama elde edilmiştir.



Şekil 6.8. DES üzerinde alan 11 ve gecikme 7 kısıtları altında uygulama sonucunda kromozomların güvenilirlik değeri.

7.SONUÇ

Bu tezde kombinasyonel devrelerin yumuşak hatalara karşı daha dayanıklı olması için genetik algoritma tabanlı bir yöntem sunulmuştur. Bu yöntemde işlemler VAÇ şeklinde girdi olarak alınır. Bir kaynak kütüphanesinde kaynakların farklı versiyonları tutulur. Tanımlanan alan ve gecikme kısıtları altında kaynak kütüphanesindeki kaynaklar işlemler arasında dağıltılır. Sunulan yöntemin alan ve gecikme kısıtları altında güvenilirlik değerini maksimize edilmesi için kullanılması amaclanmıştır. Tezde anlatılan yöntem daha önce Tamsayılı Doğrusal Programlama ve Sezgisel yöntemlerle sunulmuştur. Tezdeki yöntem daha önce sunulmuş olan sezgisel yöntemle farklı VAÇ'ler üzerinde test edilmiş ve sonuçları karşılaştırılmıştır. Tezde anlatılan yöntem Tamsayılı Doğrusal Programlamaya göre daha hızlı bir şekilde sonuçlanmış ve sezgisel yönteme göre daha yüksek güvenilirlik değeri elde etmiştir. Tezde anlatılan yöntemin üzerinde tekrar çalışma yapılıp ve bu yöntem güç tüketim değerini minimize etmek için de kullanılabilir.

KAYNAKLAR

[1] Coussy, P.; Gajski, D. D.; Meredith, M.; Takach. "An Introduction to High-Level Synthesis". IEEE Design & Test of Computers 26 (4): 8–17, 2009.

[2] Alice C. Parker, Yosef Tirat-Gefen, Suhrid A. Wadekar . "System-Level Design". In Wai-Kai Chen. The VLSI handbook (2nd ed.). CRC Press. chapter 76,2007.

[3] Tosun, Suleyman, Nazanin Mansouri, Ercument Arvas, Mahmut Kandemir, and Yuan Xie. "Reliability-centric high-level synthesis." In Design, Automation and Test in Europe, pp. 1258-1263. IEEE, 2005.

[4] Glaß, Michael, et al. "Reliability-aware system synthesis." *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*. IEEE, 2007.

[5] Streichert, Thilo, et al. "Design space exploration of reliable networked embedded systems." *Journal of Systems Architecture* 53.10 (2007): 751-763.

[6] Tanaka, Sho, et al. "A fault-secure high-level synthesis algorithm for RDR architectures." *IPSJ Transactions on System LSI Design Methodology* 4 (2011): 150-165.

[7] Mitra, Subhasish, et al. "Robust system design with built-in soft-error resilience." *Computer* 38.2 (2005): 43-52.

[8] Inoue, Tomoo, et al. "High-level synthesis for multi-cycle transient fault tolerant datapaths." *On-Line Testing Symposium (IOLTS), 2011 IEEE 17th International*. IEEE, 2011.

[9] J. F. Ziegler et.al. “IBM experiments in soft fails in computer electronics (1978-1994)”, IBM Journal of Research and Development, 1996.

[10] Liang Chen, Mojtaba Ebrahimi, and Mehdi B. Tahoori. 2016. Reliability-Aware Resource Allocation and Binding in High-Level Synthesis. ACM Trans. Des. Autom. Electron. Syst. 21, 2, Article 30 (January 2016), 27 pages.

[11] D. MacMillen et al., ‘‘An Industrial View of Electronic Design Automation,’’ IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 19, no. 12, 2000, pp. 1428-1448.

[12] 4. J.P. Elliot, Understanding Behavioral Synthesis: A Practical Guide to High-Level Design, Kluwer Academic Publishers, 1999.

[13] Philippe Coussy., Michael Meredith, Daniel D. Gajski, Andres Takach,”An Introduction to High-Level Synthesis”.

[14] S. Tosun, O. Ozturk, N. Mansouri, E. Arvas, M. Kandemir, Y. Xie, and W-L. Hung. 2005. An ILP Formulation for Reliability-Oriented High-Level Synthesis. In Proceedings of the 6th International Symposium on Quality of Electronic Design (ISQED '05). IEEE Computer Society, Washington, DC, USA, 364-369.

[15] Paulin, Pierre G., and John P. Knight. "Scheduling and binding algorithms for high-level synthesis." Design Automation, 1989. 26th Conference on. IEEE, 1989.

[16] Slayman, Charles. "Soft error trends and mitigation techniques in memory devices." Reliability and Maintainability Symposium (RAMS), 2011 Proceedings-Annual. IEEE, 2011.

[17] A. Charlesworth, “STARFIRE: Extending the SMP envelope,” IEEE Micro, vol. 18, no. 1, pp. 39–49, Jan./Feb. 1998.

[18] C. Slayman, “Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations”, IEEE Trans. Device and MaterialsReliability, Vol. 5 , No. 3, 2005, pp. 397-404.

[19] R. W. Hamming, “Error detecting and error correcting codes,” Bell Syst. Tech. J., vol. 29, no. 2, pp. 147–160, Apr. 1950.

[20] A. H. El-Maleh and K. A. K. Daud, “Simulation-based method for synthesizing soft error tolerant combinational circuits,” IEEE Trans. Rel., vol. 64, no. 3, pp. 935–948, Sep. 2015.

[21] Xie, Yuan, et al. "Reliability-aware co-synthesis for embedded systems." *The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology* 49.1 (2007): 87-99.

[22] Hung, W-L., Xiaoxia Wu, and Yuan Xie. "Guaranteeing performance yield in high-level synthesis." *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*. ACM, 2006.

[23] Cong, Jason, Albert Liu, and Bin Liu. "A variation-tolerant scheduler for better than worst-case behavioral synthesis." *Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis*. ACM, 2009.

[24] Mittal, Kartikey, Arpit Joshi, and Madhu Mutyam. "Timing variation-aware scheduling and resource binding in high-level synthesis." *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 16.4 (2011): 40.

[25] Del Barrio, Alberto A., Jason Cong, and Román Hermida. "A distributed clustered architecture to tackle delay variations in datapath synthesis." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35.3 (2016): 419-432.

[26] Sanchez-Clemente, Antonio J., Luis Entrena, Radek Hrbacek, and Lukas Sekanina. "Error Mitigation using Approximate Logic Circuits: A Comparison of Probabilistic and Evolutionary Approaches." IEEE Transactions on Reliability 65, no. 4 (2016): 1871-1883.

[27] Limbrick, Daniel B. "Impact of logic synthesis on soft error rate of digital integrated circuits." VLSI (ISVLSI), 2012 IEEE Computer Society Annual Symposium on. IEEE, 2012.

[28] A. H. El-Maleh, A. S. Al-Qahtani, “A Finite State Machine Based Fault Tolerance Technique for Sequential Circuits, Microelectronics Reliability, vol. 54, pp. 491-662, 2014.

[29] El-Maleh, Aiman H. "A sequential circuit fault tolerance technique with enhanced area and power." Signal Processing and Information Technology (ISSPIT), 2015 IEEE International Symposium on. IEEE, 2015.

[30] Mineo Kaneko, Kazuaki Oshio, “Fault Tolerant Datapath Based on Algorithm Redundancy and Vote-Writeback Mechanism”, Proc. IEEE International Symposium on Circuits and Systems, Vol. V, pp.645-648, 2003.

[31] Hara-Azumi, Yuko, and Hiroyuki Tomiyama. "Cost-efficient scheduling in high-level synthesis for Soft-Error Vulnerability Mitigation." *Quality Electronic Design (ISQED), 2013 14th International Symposium on*. IEEE, 2013.

[32] Chen, Liang, Mojtaba Ebrahimi, and Mehdi B. Tahoori. "Reliability-aware operation chaining in high level synthesis." *Test Symposium (ETS), 2015 20th IEEE European*. IEEE, 2015.

[33] Cong, Jason, et al. "High-level synthesis for FPGAs: From prototyping to deployment." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30.4 (2011): 473-491.

[34] Nicolaidis, Michael. "Circuit-Level Soft-Error Mitigation." *Soft errors in modern electronic systems*. Springer US, 2011. 203-252.

[35] Canis, Andrew, et al. "LegUp: high-level synthesis for FPGA-based processor/accelerator systems." *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*. ACM, 2011.

[36] Anna Antola, Vincenzo Piuri, and Mariagiovanna Sami, “High Level Synthesis of Data Paths with Concurrent Error Detection”, Proc. IEEE Symp. DFT in VLSI Systems, pp.292-299, 1998.

[37] Kaneko, Mineo, and Yutaka Tsuboishi. "Constrained binding and scheduling of triplicated algorithm for fault tolerant datapath synthesis." Circuits and Systems (ISCAS), 2014 IEEE International Symposium on. IEEE, 2014.

[38] Micheli, Giovanni De. Synthesis and optimization of digital circuits McGraw-Hill Highter Education, 1994.

[39] I. Nestor and D. Thomas. "Behavioral Synthesis with Interfaces," ICCAD, Proceedings of the international Conference on Computer Aided Design, pp. 112-1 15, 1986.

[40] Hwang, C-T., J-H. Lee, and Y-C. Hsu. "A formal approach to the scheduling problem in high level synthesis." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 10.4 (1991): 464-475.

[41] BEASLEY, D., BULL, D.R., and MARTIN, R.R., 1993a. An Overview of Genetic Algorithms.

[42] Practical Genetic Algorithms by Randy L. Haupt and Sue Ellen Haupt.

[43] Kavi, Krishna M., Bill P. Buckles, and U. Narayan Bhat. "A formal definition of data flow graph models." IEEE Transactions on computers 35.11 (1986): 940-948.

[44] Cong, Jason, et al. "High-level synthesis for FPGAs: From prototyping to deployment." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 30.4 (2011): 473-491.

[45] Ku, David C., and Giovanni DeMicheli. High level synthesis of ASICs under timing and synchronization constraints. Vol. 177. Springer Science & Business Media, 2013.

[46] Mohanram, Kartik, and Nur A. Touba. "Partial error masking to reduce soft error failure rate in logic circuits." *Defect and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on*. IEEE, 2003.

[47] Inoue, Tomoo, et al. "High-level synthesis for multi-cycle transient fault tolerant datapaths." *On-Line Testing Symposium (IOLTS), 2011 IEEE 17th International*. IEEE, 2011.

[48] Chen, Deming, Jason Cong, and Yiping Fan. "Low-power high-level synthesis for FPGA architectures." *Low Power Electronics and Design, 2003. ISLPED'03. Proceedings of the 2003 International Symposium on*. IEEE, 2003.

ÖZGEÇMİŞ

Kimlik Bilgileri

Adı Soyadı : Tohid Taghizad Gogjeh Yaran

Doğum Yeri : Orumiyeh - İran

Medeni Hali : Bekar

E-posta : tohid.taghizadeh66@gmail.com

Adres : Ertuğrul Gazi Mah. Şehit İsmail Kılıç SK. 16/7 Çankaya/ANKARA

Eğitim

Lise : TabaTabaie Lisesi Orumiyeh İran

Lisans : Zanjan Üniversitesi

Yabancı Dil ve Düzeyi

İngilizce (İyi), Azerice (Çok İyi), Farsça (Çok İyi)

İş Deneyimi

2015'den beri Tiga yazılım şirketinde çalışmaktadır.

Deneyim Alanı

Web, Application yazılım uzmanı

Tezden Üretilmiş Projeler ve Bütçesi

Tezden Üretilmiş Yayınlar

Tezden Üretilmiş Tebliğ ve/veya Poster Sunumu ile Katıldığı Toplantılar

Tohid Taghizad Gogjeh Yaran, Süleyman Tosun ." Improving Combinational Circuit Resilience against Soft Errors via Selective Resource Allocation " IEEE International Symposium DDECS 2017, Dresden, Germany.

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

HACETTEPE ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
Bilgisayar Müh. ANABİLİM DALI BAŞKANLIĞI'NA

Tarih: 8/6/17

Tez Başlığı / Konusu: Güvenilirlik Odaklı Tümleşik Sistem Tasarım Yöntemi

Yukarıda başlığı/konusu gösterilen tez çalışmamın a) Kapak sayfası, b) Giriş, c) Ana bölümler ve d) Sonuç kısımlarından oluşan toplam 66 sayfalık kısmına ilişkin, 6/6/17 tarihinde şahsım/tez danışmanım tarafından turnitin adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % 3 'tür.

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

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

Gereğini saygılarımla arz ederim.

Tarih ve İmza

Adı Soyadı: Tohid Taghizad Gogjeh Yoran
Öğrenci No: N13222426
Anabilim Dalı: Bilgisayar Müh
Programı:
Statüsü: ☑ Y.Lisans ☐ Doktora ☐ Bütünleşik Dr.

DANIŞMAN ONAYI

UYGUNDUR.

Doç. Dr. Süleyman Tosun
(Unvan, Ad Soyad, İmza)