GENİŞLETİLEBİLİR YAZMAÇ YENİDEN ADLANDIRMA YÖNTEMİ TASARIMI

GÖRKEM AŞILIOĞLU

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

AĞUSTOS 2011

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Ünver KAYNAK

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU

Anabilim Dalı Başkanı

Görkem AŞILIOĞLU tarafından hazırlanan GENİŞLETİLEBİLİR YAZMAÇ YENİDEN ADLANDIRMA YÖNTEMİ TASARIMI adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Yrd. Doç. Dr. Oğuz ERGİN

Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Yrd. Doç. Dr. A. Murat ÖZBAYOĞLU _____

Üye : Yrd. Doç. Dr. Oğuz ERGİN _____

Üye : Doç. Dr. Coşku KASNAKOĞLU _____

Üye : Doç. Dr. Soner ÖNDER _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

……..……..

Görkem AŞILIOĞLU

Üniversitesi : TOBB Ekonomi ve Teknoloji Üniversitesi

Enstitüsü : Fen Bilimleri

Anabilim Dalı : Bilgisayar Mühendisliği

Tez Danışmanı : Yrd. Doç. Dr. Oğuz ERGİN

Tez Türü ve Tarihi : Yüksek Lisans – Temmuz 2011

Görkem AŞILIOĞLU

GENİŞLETİLEBİLİR YAZMAÇ YENİDEN ADLANDIRMA YÖNTEMİ TASARIMI

ÖZET

Yazmaç yeniden adlandırma güncel çok yollu işlemcilerde gerçek olmayan veri bağımlılıklarını ortadan kaldırmak için sıklıkla kullanılan bir tekniktir. Bu teknik mimari tasarımda belirtilen yazmaçların işlemciye gelen buyrukların çözülmesi sırasında fiziksel yazmaçlara atanması ile gerçekleştirilir. Bu atamalar bir eşleştirme tablosunda tutulur.

Çok yollu işlemciler dallanma tahmini gibi teknikler kullandığında işlemci hatalı bir tahmin sonucunda olmaması gereken bir duruma düşer. Yanlışlıkla işlenilmeye başlanan buyrukların yazmaçlarının yeniden adlandırmaları bir şekilde geri alınmalı ve doğru duruma dönülmelidir.

Güncel işlemcilerde bu geri dönüşümü yapan teknikler ya geri dönüşüm hızından, ya da donanım karmaşıklığı yönünden taviz vermektedir. Bu çalışma donanım karmaşıklığı yönünden daha basit olan, bunun yanında en yavaş halinda iki saat vuruşunda yeniden adlandırma tablosunu eski haline getirebilen ve rahat genişletilebilen bir yeniden adlandırma sistemi önermektedir.

Önerilen yapı her mimari yazmaç için farklı boylarda İGİÇ kuyrukları kullanarak her mimari yazmaç için farklı miktarda kopya tutmayı hedefliyor. Bu çalışmanın sonuçları bazı özel durumlar dışında önerilen sistemin donanımla sınırlı yapılardan başarımının daha iyi olduğunu gösteriyor.

Bu çalışmanın yanında, işlemcide kullanılan alanı en aza indirmek için İGİÇ kuyruklarının boylarının başarımı çok etkilemeden en aza indirilmesi üzerine bir çalışma daha yapıldı. Bu çalışmada bir genetik algoritma kullanarak alan kullanımı ve başarımı en uygun şekilde birleştirmeyi başardık.

Anahtar Kelimeler: Bilgisayar mimarisi, çok yollu işlemciler, yazmaç yeniden adlandırma, genetik algoritmalar

University : **TOBB University of Economics and Technology**

Institute : **Institute of Natural and Applied Sciences**

Science Programme : **Computer Engineering**

Supervisor : **Asst. Prof. Oğuz ERGİN**

Degree Awarded and Date : **M. Sc. – July 2011**

Görkem AŞILIOĞLU

COMPLEXITY-EFFECTIVE RENAME TABLE DESIGN FOR RAPID SPECULATION RECOVERY

ABSTRACT

Register renaming is a commonly used technique to remove false data dependencies in contemporary superscalar processors. This is done by assigning physical registers to registers defined in architectural design during the decoding process of the instructions in the processor. These assignments are kept in an alias table.

When superscalar processors use techniques such as branch prediction the processor may reach a state it should not be in as a result of a misprediction. Instructions fetched mistakenly need to restore the rename assignments and return to a correct state.

In contemporary processors the techniques which restore the rename table either sacrifice restore speed or hardware complexity. This study shows an extendable technique which has less hardware complexity, yet can restore the rename table in at most two clock cycles.

The design proposes the use of differently sized FIFO queues for each architectural register to hold checkpoints. This study shows that the proposed structure performs better than existing techniques except in a few exceptional cases.

Besides the rename table design, a study was also done on determining the optimum FIFO queue size for each architectural register without losing performance. This study proposes the use of genetic algorithms to successfully balance area usage and performance in a reasonable amount of time.

Key Words: Computer Architecture, Superscalar Processors, Register Renaming, Genetic Algorithms

TEŞEKKÜR

Yüksek lisans ve lisans çalışmalarım boyunca bana yardım eden, yönlendiren, desteğini esirgemeyen değerli hocam ve tez danışmanım Yrd. Doç. Dr. Oğuz ERGİN'e, derslerde ve diğer konularda deneyimlerinden yararlandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği öğretim üyelerine, tez konusu ile ilgili çalışmalarım sırasında büyük katkıları olan Mehmet KAYAALP ve Vehbi Eşref BAYRAKTAR'a, yüksek linsans çalışmalarım boyunca burs ve diğer maddi destekler sağlayan TÜBİTAK'a, Z10 Laboratuvarında çalışan tüm lisans ve yüksek lisans öğrencilerine ve beni asla yalnız bırakmayan, en büyük destekçim olan aileme teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

ÖZET iv

ABSTRACT v

TEŞEKKÜR vi

İÇİNDEKİLER vii

ÇİZELGELERİN LİSTESİ ix

ŞEKİLLERİN LİSTESİ x

KISALTMALAR xi

SEMBOL LİSTESİ xii

1. GİRİŞ 1

2. TEMEL KONULAR 4

2.1. Boru Hattı 4

2.2. Çok yollu İşlemciler 6

2.3. Sıra Dışı İşleme ve Yeniden Sıralama Belleği (YSB) 7

2.4. Yazmaç Yeniden Adlandırma 8

2.5. Dallanma Tahmini 10

3. ÖNERİLEN YENİDEN ADLANDIRMA YAPISI 12

3.1. Dallanma Tahmini Hatalarından Kurtulma 12

3.2. İGİÇ Kuyrukları Kullanan Yapı 15

3.3. Donanım Yapısı 19

3.4. Tamamı Dairesel İGİÇ Kuyrukları ve Çift Yazmaç Tekniği 22

3.5. Kopya Alma İle Birlikte İGİÇ Kuyrukları Kullanımı İçin Geliştirme 24

4. İGİÇ KUYRUK İLE YAZMAÇ YENİDEN ADLANDIRMA DENEYLERİ 26

4.1. Denektaşı Programları 26

4.2. PTLSim ve M-Sim Benzetim Parametreleri 28

4.3. x86 İGİÇ Benzetimi 30
4.4. Alpha İGİÇ Benzetimi – Yazmaçların Kullanım Oranlarının Bulunması .. 33
4.5. Alpha İGİÇ Benzetimi – Temel Deneyler 35
4.6. Alpha İGİÇ Benzetimi – Kuyruk Boyları İstatistiksel Ayarlanmış Yapılandırma 36
4.7. Alpha İGİÇ Benzetimi – Denektaşı Programlarının Dallanma Nitelikleri . 37
4.8. Alpha İGİÇ Benzetimi – Çift Yazmaç Tekniği 40
4.9. Alpha İGİÇ Benzetimi – Kopya Alma Durumunda İyileştirme 41
5. GENETİK ALGORİTMA KULLANARAK PARAMETRE TESPİTİ 43
5.1. Genetik Algoritmalar 44
5.2. Kullanılan Algoritma ve Hesaplamalar 45
5.3. Sonuçlar 48
6. SONUÇLAR 51
6.1. İGİÇ Yazmaç Yeniden Adlandırma Sonuçları 51
6.2. Genetik Algoritma ile Parametre Arama Sonuçları 51
KAYNAKLAR 53
ÖZGEÇMİŞ 55

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

Tablo 2.1. Boru Hattı Çalışması 5

Tablo 2.2. Bağımsız Buyruklar 6

Tablo 2.3. Sırasız İşlemede Hata Olabilecek Buyruk Grubu 8

Tablo 3.1. Aynı Anda İGİÇ Kuyruğuna İki Yazma Yapan Buyruklara Örnek 25

Tablo 4.1. Spec 2000 Tam Sayı Denektaşı Programları 26

Tablo 4.2. Spec 2000 Kayan Nokta Denektaşı Programları 27

Tablo 4.3. PTLSim Benzetim Parametreleri 28

Tablo 4.4. M-Sim Benzetim Parametreleri 28

Tablo 4.5. x86 Genel Amaçlı Yazmaç Kullanım Verileri 32

Tablo 4.6. Belirlenen İGİÇ Kuyruk Boyları 34

Tablo 5.1. Genetik Algoritma Sözde Kod 45

Tablo 5.2. Seçilen En İyi 25 Sonuç 49

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil 2.1. Temel RISC Boru Hattı 4

Şekil 2.2. Sanal Veri Bağımlılığı Örnekleri 8

Şekil 2.3. Yeniden Adlandırma Sonucu 10

Şekil 3.1. Önerilen Yeniden Adlandırma Kuyruk Yapısı 16

Şekil 3.2. Önerilen Kopya Alma Yapısı 18

Şekil 3.3. Kopya Alma Yapısının Donanımı 20

Şekil 3.4. İGİÇ Kuyruklarının Donanımı 21

Şekil 4.1. PTLSim Başarım Grafiği 31

Şekil 4.2. x86 Genel Amaçlı Yazmaç Kullanım Grafiği 32

Şekil 4.3. Yazmaçların En Fazla Kullanımı 34

Şekil 4.4. Geri Yürüme İle Başarım Farkı 35

Şekil 4.5. İGİÇ Kuyruk Boylarını İstatistiksel Yapılandırma Başarım Değişimi 37

Şekil 4.6. Çevrim Başına En Fazla Dallanma 38

Şekil 4.7. Çevrim Başına Ortalama Dallanma 39

Şekil 4.8. Dallanmalar Arası Ortalama Buyruk Sayısı 39

Şekil 4.9. İGİÇ Tamsayı - Kayan Nokta Eşleşmiş Dinamik Yazmaçlarda Başarım Değişimi 41

Şekil 4.10. Kopya Alma Tablosu İle Birlikte Çalışan İyileştirme Başarımı 42

Şekil 5.1. Genetik Algoritma Genel Görünüş 44

Şekil 5.2. Kullanılan Genetik Algoritmanın Görsel Gösterimi 46

Şekil 5.3. Kullanılan Genetik Algoritmanın Çiftleşme Aşaması 47

Şekil 5.4. Farklı Yapılandırmalarla Elde Edilen ÇBB Değerleri 48

KISALTMALAR

Kısaltmalar	Açıklama
FIFO	First In First Out
SRT	Speculative Rename Table
CRT	Commit Rename Table
RT	Rename Table
RAT	Register Alias Table
YSB	Yeniden Sıralama Belleği
RISC	Reduced Instruction Set Computer
CISC	Complex Instruction Set Computer
İGİÇ	İlk Giren İlk Çıkar
SPEC	Standard Performance Evaluation Corporation

SEMBOL LİSTESİ

Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklamalar**
$M_{Ortalama}$	Yazmaçların en fazla kullanım değerlerinin ortalaması
$M_{Standart\ Sapma}$	Yazmaçların en fazla kullanım değerlerinin standart sapması

1. GİRİŞ

Güncel çok yollu mikroişlemciler başarımı artırmak için sıra dışı buyruk yürütme (out-of-order execution) ve dinamik zamanlama (dynamic scheduling) gibi teknikler kullanırlar. Sıra dışı buyruk yürütme tekniği donanımda buyrukları program sırasına göre değil de, bağımlılıklarının tamamlanma sırasına göre işlenebilmesine olanak sağlar [1]. İdeal durumda hiç bir buyruk bir önceki buyruğa bağlı değilse, her buyruk kaynaklar eline ulaştığında (bellek vb. kaynaklardan) işlenebilir hale gelmektedir. Ancak gerçekte böyle bir durum söz konusu değildir. Buyruk arası bağımlılıklar başarım düşüşüne neden olan en önemli nedenlerden biridir.

Bu bağımlılıklar *denetim bağımlılıkları* ve *veri bağımlılıkları* olarak iki alt konuya ayrılabilir. Denetim bağımlılıkları bir buyruğun işlenmesi bir önceki buyruğun sonucuna bağlı ise gerçekleşir. Güncel işlemciler bu bağımlılıktan kurtulmak için dallanma buyruklarının sonuçlarını tahmin eden teknikler kullanmaktadır. Bu teknikler başarımı oldukça artırmakla birlikte hatalı bir tahmin durumunda işlemciyi doğru duruma geri getirmek için de çeşitli tekniklerin geliştirilmesini gerektirir.

Veri bağımlılıkları ise gerçek veri bağımlılıkları ve sanal veri bağımlılıkları olarak ikiye ayrılabilir. Sanal veri bağımlılıkları işlemcilerde yazmaçların yetersiz olmasından ya da programcı veya derleyicinin var olan yazmaçları tutarsız kullanmasından kaynaklanan bağımlılıklardır. Bu bağımlılıkları yok etmek için daha fazla yazmaç eklemek gerekir, ancak bir işlemcinin buyruk kümesi tanımlandıktan sonra daha fazla yazmaç eklemek mümkün olmayabilir. Ayrıca, yazmaç eklemek varolan derleyicilerin yazmaç kullanma yapılarını değiştirmez ve kullanıcıların yanlış kullanımlarını düzeltmez.

Bu sorunu çözmek için işlemci içinde daha büyük bir yazmaç öbeği oluşturup işlemci mimarisinde belirlenmiş yazmaçları bu yeni yazmaç öbeğindeki yazmaçlara yeniden adlandırabiliriz. Bu yönteme yazmaç yeniden adlandırma denir. Bu teknik kullanılarak sanal veri bağımlılıkları ortadan kaldırılabilir. Bu teknikte sonuç üreten

her buyruğa yeni bir fiziksel yazmaç atanır. Bu tekniğin detayları Bölüm 2'de bulunabilir.

Yazmaç yeniden adlandırma tekniğinde yeniden adlandırılan yazmaçlar yazmaç adlandırma tablosu (register alias table, RAT) isimli bir yapı kullanılarak takip edilir. RAT işlemcinin o anki kesin durumunu belirtir ve bir özel durum ya da dallanma tahmini hatası sonucu kurtarılmalıdır. Bu kurtarma işleminin başarımı en az etkilemesi için mümkün olan en az sayıda saat vuruşunda yapılması önemlidir. Literatürde dallanma tahminini geri almak için kullanılan teknikler arasında yazmaç adlandırma tablosunu kopyalamak, yeniden adlandırma belleğinde (reorder buffer, ROB) ileri ya da geri yürümek ya da hata yapan dallanma buyruğu commit edene kadar beklemek başlıca öne çıkar. Bu tekniklerin her birinin devre karmaşıklığı ve dallanma hatası kurtarma süresi yönünden farklı iyi ve kötü tarafları vardır.

Bu tez yeniden adlandırma tablosunu her mimari yazmaç için bir adet İGİÇ kuyruk kullanarak yeniden oluşturarak dallanma tahmini hatalarından kısa sürede kurtulmayı öneren sistemin detaylarını anlatır. Bu kuyruklar her yazmaç için ayrı bir kopya alma yapısı gibi davranacaktır. Normal çalışma sırasında kuyruk işaretçileri (tail pointer) bir tabloda saklanacak ve dallanma hatası durumunda bu işaretçiler eski hallerine döndürülerek hatadan kurtulmak mümkün olacaktır. İGİÇ tasarımı tüm tablonun kopyalarının alındığı tasarımla karşılaştırıldığı zaman devre karmaşıklığı yönünden daha basittir, ve YSB üzerinde yürüyerek RAT'i düzeltmeye çalışan tekniklere göre daha hızlı sonuç vermektedir.

İGİÇ kuyrukları kullanarak tüm mimari yazmaçlar için ayrı kopyalar oluşturmak demek her mimari yazmaç için tutulacak kopya sayısının farklı olabileceğini anlamına gelir. Az kullanılan yazmaçlara daha küçük İGİÇ kuyrukları atanarak işlemci üzerindeki alandan ve güç tüketiminden kazanılabilir. Ancak hangi yazmaca ne boyda kuyruk atanacağı verilmesi kolay bir karar değildir; yalnızca yazmaç kullanım istatistiklerine bakmak ideal sonuç vermemektedir. Farklı kuyruk boylarının denenmesi gerekmektedir, ancak her test kümesinin tam benzetimi eldeki kaynaklara da bağlı olarak saatler ya da günler alabilmektedir. Bu durumda çok

sayıda farklı İGİÇ kuyruk boyu düzenlerinin denenmesi gerçekçi olmaktan çıkmaktadır. Bu tez verilen İGİÇ kuyruklarının alması gereken boyları bulmak için genetik algoritma tabanlı bir yapı önerir. Bu algoritma kullanılarak ideal İGİÇ kuyruk boylarına tüm kuyruk boyu düzenlerini denemeden yakınsanması amaçlanmaktadır. Sonuçlarımız bu algoritmanın tatmin edici başarım gösteren ve alan olarak istatistiki hesaplamalarla bulunan kuyruk boylarından daha az alan kapladığını gösterir.

2. TEMEL KONULAR

Bu kısım mikroişlemcilerle ilgili bazı temel konularla ilgili bilgi vermeyi amaçlar. Burada sözü geçen konular tezde anlatılan yapıların anlaşılmasında büyük rol oynayacaktır.

İşlemci bir programcı tarafından verilen buyruklar doğrultusunda hesaplama yapan ve bir bilgisayarı denetleyen aletlere verilen genel isimdir. Günümüzde işlemciler yalnızca masaüstü ve diz üstü bilgisayarlarda değil, cep telefonlarından oyun konsollarına kadar pek çok alette kullanılmaktadır.

2.1. Boru Hattı

İlk işlemciler her saat darbesinde tek bir buyruk işleyecek şekilde tasarlanmıştır. Ancak bu durumda saat darbesi hızı en yavaş çalışan buyruk ile sınırlı kalmak zorundadır. Bu sorunu çözmek için bir buyruğun işlenmesini aşamalara bölerek her aşamada birden fazla buyruğun farklı aşamalarının işlenmesi amaçlanmıştır. Bu yönteme boru hattı yöntemi denir. Temel bir RISC (reduced instruction set computing) makinesinde beş aşamalı, Şekil 2.1'de görülen aşamalara sahip bir boru hattı bulunmaktadır.

Şekil 2.1. Temel RISC Boru Hattı

Şekil 2.1'de görülen boru hattı yapısında Getir aşaması bellekten buyrukların işlemciye gelmesini ifade eder. Çöz aşamasında ise buyruğun işlenmesi için yapılacak işlemler, kullanılacak yazmaçlar ve hangi işlem birimlerinin kullanılacağı belirlenir. Buyruğun işlenenlerinin hazır olup olmadığı ve bağımlılıklar da bu aşamada belirlenir. Yürüt aşamasında buyruk için gerekli olan hesaplama işlemleri yapılır. Bellek aşamasında buyruk bellek erişimi gerektiyorsa bu yapılır. Yaz

aşamasında ise üretilen sonuçlar buyruğun belirttiği sonuç yazmacına yazılır. Bu boru hattı temel bir boru hattının örneği olup, farklı işlemcilerde bu aşamalar birden fazla aşamaya bölünmüş ya da sıraları hafifçe değişmiş olarak karşımıza çıkabilir.

Boru hattı kullanıldığı zaman işlemci içindeki tüm birimler boru hattı kullanılmayan durumlara göre daha etkin çalışır. Boru hattı kullanılmadığı durumlarda da buyruklar bu aşamalardan geçmek durumundadır, ancak her aşama işlenirken diğer aşamaların devreleri boş durmaktadır. Birbirinden bağımsız buyruklar geldiği sürece boru hattı kullanan bir işlemci tüm parçalarını kullanabilmektedir. Birbirinden bağımsız buyruklar geldiğinde boru hattı Tablo 2.1'deki gibi görünür.

Tablo 2.1. Boru Hattı Çalışması

Buyruk No.	**Boru Hattı Aşaması**								
Saat Darbesi	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
B1	G	Ç	Ü	B	Y				
B2		G	Ç	Ü	B	Y			
B3			G	Ç	Ü	B	Y		
B4				G	Ç	Ü	B	Y	
B5					G	Ç	Ü	B	Y

Görüldüğü gibi yeşil ile işaretlenmiş saat darbesinden sonra gelen buyruklarla birlikte işlemcinin tüm birimleri sürekli olarak çalışır halde olacaktır. Ancak bu hiçbir buyruğun diğerine bağlı olmadığı, ideal durumda görülebilecek bir görüntüdür. Belleğe erişmek isteyen bir buyruk kendinden sonra gelen buyrukları bellekten verisi gelene kadar bekletebilir, bir önceki buyruğun verisine bağımlı diğer bir buyruk o buyruğun sonlanmasını beklemek zorunda olabilir ya da bir dallanma buyruğunun belli bir aşamasına kadar boru hattına girecek bir sonraki buyruğun adresi bilinmiyor olabilir. Dallanma buyruklarının bekleme yapmaması için 2.5 numaralı bölümde anlatılan dallanma tahmini yöntemi kullanılmaktadır. Diğer iki sorunun çözülmesi için ise 2.2 numaralı bölümde anlatılan çok yollu işlemciler ve 2.4 numaralı bölümde anlatılan yazmaç yeniden adlandırma çözüm olmaktadır. Ancak dikkat edilmelidir ki yazmaç yeniden adlandırma yalnızca sanal bağımlılıkları yok etmeye yarar. Gerçek

veri bağımlılıkları çözümlenmeden o veriyi bekleyen diğer buyruklar tam anlamıyla çalışamaz.

Arka arkaya gelen buyruklardaki veri bağımlılıkları daha basitçe boru hattı içinde yönlendirme yapılarak çözülebilir. Tablo 2.1'de gösterilen tüm buyrukların bir önceki buyruğun ürettiği veriye bağımlı olduğunu düşünelim. Normalde, her buyruk yürütme aşamasına geçemeden önce bir önceki buyruk yaz aşamasına gelene kadar beklemek durumunda kalır. Veri yönlendirmesi boru hattının bir aşamasının çıktısının diğer bir aşamanın girdisi olarak kullanılmasına olanak sağlar. Yukarıdaki örnek boru hattında yürütme aşamasının sonundan başına yönlendirme olduğu düşünülürse, her buyruk bir önceki buyruğun ürettiği veriyi yürütme aşamasının başında elde edebilir hale gelir. Bu durumda Tablo 2.1'de gösterilen yapının daha fazla zaman harcanmadan korunması sağlanmış olur.

2.2. Çok yollu İşlemciler

Çok yollu işlemciler bir saat darbesinde birden fazla buyruk üzerinde işlem yapabilen işlemciler olarak tanımlanabilir [1]. Bu durum günümüzde kullanılan çok işlemcili ya da çok çekirdekli yapılarla karıştırılmamalıdır. Tek bir işlemci içinde buyruk düzeyindeki paralellikten yararlanarak birden fazla buyruğu aynı anda işlemeye çalışır. Buyruk düzeyindeki paralellik birbirinden bağımsız olan buyrukların aynı anda işlenebilmeleri olarak açıklanabilir. Tablo 2.2'de görülen buyruklar bağımsız buyruklara bir örnek teşkil etmektedir. B1'den B5'e kadar olan buyrukların hiç biri diğerinin sonucuna bağlı değildir. Bu durumda bu buyrukların arka arkaya çalışması gerekmez, aynı anda da çalışabilirler.

Tablo 2.2. Bağımsız Buyruklar

B1	ADD R1, R2, R3
B2	SUB R4, R3, R2
B3	MUL R5, R6, R2
B4	DIV R7, R3, R10
B5	SUB R8, R6, R12

Bu buyrukları aynı anda işlemek için çok yollu işlemciler işlemci içinde aynı birimden birden fazla bulundurur. Bölüm 2.1'de anlatılan boru hattı yapısındaki beş aşama için işlemci içerisinde bir yapı bulunur. Bu yapılar iki katına çıkarılırsa, aynı anda birbirinden bağımsız iki buyruk işlenebilir.

Yalnızca buyruk getiren, çözen, işleyen, belleğe giden ve yazan yapıların ikilenmesi yeterli değildir. Bunun yanında buyrukların bu aşamalara birden fazla gelebilmesi için bağımlılıkları denetleyen ve bu buyrukları uygun kısımlara yönlendiren bir yapı gerekmektedir.

Ancak bu yöntemle de bir buyruğun diğer bir buyruğa ya da bellekten gelecek bir veriye bağımlılığı olursa işlemci bu buyruğu beklemek durumunda kalacak.

2.3. Sıra Dışı İşleme ve Yeniden Sıralama Belleği (YSB)

Buyruk düzeyindeki paralellikten daha etkin yararlanmak için birbirinden bağımsız buyrukları sıralarına bakmaksızın işlemeye sıra dışı işleme denir. Sıra dışı işleme yapmak için boru hattı aşamaları 2.1 numaralı bölümde anlatılandan biraz daha farklı olmaktadır. Sıra dışı işlem yapılacağı zaman işlemci Getir aşamasında işlemcinin *getirme genişliği* kadar bellekten buyruk getirir. Bu aşamadan sonra bir saat darbesinde *çözme genişliği* kadar buyruk çözülür (bitlerine bakılarak kullanılacak yazmaçları, bağımlı olduğu buyruklar belirlenir). Yazmaç yeniden adlandırma da bu aşamada yapılır. Bu aşamada bir sonraki buyrukların nereden getirileceği belirlenir. Dallanma buyruğu varsa bu adres dallanma tahmini yapılarak belirlenir. Çözülen buyruklar bir buyruk kuyruğuna gönderilir. Bu kuyruğa aynı zamanda yayın kuyruğu ya da bekleme istasyonları da denir. Buyruklar işlenenleri hazır olana kadar bu kuyrukta bekler. Bu kuyruktaki buyruklar işlenenleri hazır olduğu anda, program sırasından bağımsız olarak işleme birimlerine gidebilirler.

Buyrukların ürettiği sonuçlar başka bir kuyrukta tutulur. Bu kuyruğa yeniden sıralama belleği (re-order buffer, ROB) denir. Bu kuyruktaki buyruklar program sırası ile sonuçlarını ilgili yazmaçlara ya da belleğe yazarlar. Buyrukların

sonuçlarının yazıldığı bu aşamaya *işleme* ya da *emekli olma* aşaması denir. Bu düzenleyici belleğin olmaması durumunda çeşitli sorunlar ortaya çıkabilir.

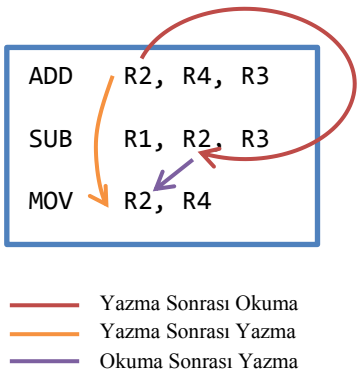
Tablo 2.3. Sırasız İşlemede Hata Olabilecek Buyruk Grubu

B1	ADD R1, R2, R3
B2	DIV R1, R3, R2
B3	SUB R1, R6, R2

Tablo 2.3'de görülen buyrukların sıra dışı işlem yapan bir işlemcide işlendiğini düşünelim. Programcı bu buyrukların işlenmesi bittiğinde R1 yazmacında B3 buyruğunun ürettiği sonucun durmasını amaçlamaktadır. Ancak YSB benzeri, buyrukların sonuçlarını yazma sıralarını düzenleyen bir yapı olmazsa bu üç buyruk sonuçlandığında R1 yazmacında bu buyruklardan hangisinin sonucunun yazdığı belirsiz olacaktır.

2.4. Yazmaç Yeniden Adlandırma

Çok yollu, sıra dışı işlem yapabilen bir işlemci işlenecek buyrukların birbirlerine ya da belleğe veri bağımlılıkları yoksa her saat darbesinde tasarımın desteklediği sayıda buyruk işleyebilir. Ancak işlemcinin desteklediği buyruk kümesinde belirtilen yazmaç sayısı yeterli değil ise, ya da derleyici veya programcı bu konuyu göz ardı ettiyse bu yazmaç yetersizliğine bağlı olarak sanal veri bağımlılıkları ortaya çıkabilir.



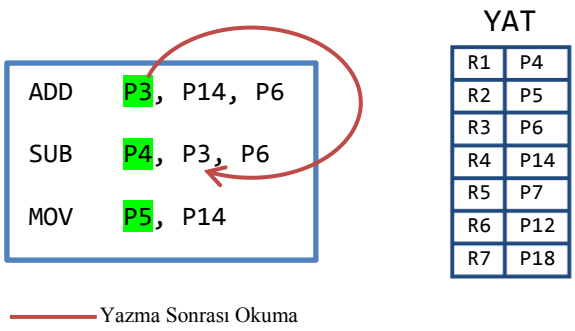
Şekil 2.2. Sanal Veri Bağımlılığı Örnekleri

Şekil 2.2 veri bağımlılıklarının örneklerini gösterir. İlk buyruk R4 ve R3 yazmaçlarını toplayıp R2 yazmacına sonucu yazar. İkinci buyruk ise R2 yazmacından R3 yazmacını çıkarır ve sonucu R1 yazmacına yazar. Üçüncü buyruk ise R4 yazmacındaki veriyi R2 yazmacına yazar. Bu buyrukların sırası işlendiğini düşünürsek, ortaya çıkan üç temel sorun görürüz. Yazma Sonrası Okuma (YSO) bir yazmaca yazıldıktan sonra o yazmacın değeri kullanılarak bir işlem yapıldığında oluşan bağımlılıktır. Bu örnekte SUB buyruğu ADD buyruğunun ürettiği değeri kullanarak işlem yapacaktır, ancak bu işlemi yapması ADD buyruğu bu sonucu üretmeden mümkün değildir. Bu bağımlılığın çözülmesi için beklemek gerekmektedir. Bekleme süresi yönlendirme ve benzeri yöntemler kullanılarak azaltılabilir. Diğer bir bağımlılık ise Yazma Sonrası Yazma (YSY) bağımlılığıdır. Bu bağımlılık ADD ve MOV buyrukları arasında görülmektedir. Bu iki buyruk sonuçlarını aynı yazmaca (R2) yazmaktadır. Ancak ADD buyruğu R2'ye yazmadan önce MOV buyruğu yazarsa aradaki SUB buyruğu (veya diğer olabilecek buyruklar) yanlış değeri kullanabilir. Bu bağımlılık MOV buyruğu farklı bir yazmaç kullanırsa çözülebilecek bir bağımlılıktır. Örnekte görülen son bağımlılık ise Okuma Sonrası Yazma (OSY) bağımlılığıdır. Bu bağımlılık MOV buyruğu SUB buyruğunun okuduğu bir yazmaca yazdığı için ortaya çıkar. Eğer MOV buyruğu SUB buyruğundan önce işlerse, SUB buyruğu yanlış bir değer kullanarak hesap yapacaktır. Bu bağımlılık da MOV buyruğunun farklı bir yazmaç kullanması ile çözülebilir.

Olabilecek üç bağımlılıktan iki tanesini farklı bir yazmaç kullanılarak çözülebilmektedir. Ancak bunu buyruk kümesinde yapmak için yeterli mimari yazmaç tanımlanmamış olabilir, ya da program yazılırken olan yazmaçlar seçilmemiştir. Her iki durumda da işlemci daha büyük bir fiziksel yazmaç öbeği kullanıp mimari yazmaçları yeniden adlandırırsa bu sorunu çözebilir.

İşlemci sonuç üreten her buyruğun sonucunu yazacağı yazmacı yeniden adlandırır. Bu adlandırmalar Yazmaç Adlandırma Tablosu (Register Alias Table, RAT) adı

verilen bir tabloda tutulur. Buyruklar kullanacakları kaynak yazmaçlarının değerlerini bu tablodan okuyarak kullanırlar.



Şekil 2.3. Yeniden Adlandırma Sonucu

Şekil 2.3'de görülebileceği gibi yeniden adlandırma sonucunda YSY ve OSY tipi bağımlılıklar ortadan kalkmıştır. Geriye kalan tek bağımlılık gerçek bağımlılık olarak adlandırabileceğimiz YSO bağımlılığıdır. Yazmaçların yeniden adlandırılması buyruklar çözülürken yapılır ve buyrukların bağımlılıkları yeniden adlandırmanın sonucunda kullanacakları yazmaçlara göre belirlenir.

2.5. Dallanma Tahmini

Kod içerisindeki dallanmalar işlemcinin beklemesine yol açabilecek başka bir sorun ortaya çıkarır. Çok yollu, sıra dışı işlem yapabilen işlemciler bir saat darbesinde bellekten birden fazla buyruk getirmektedir. Peki işlemci koşullu bir dallanma ile karşılaştığı zaman dallanma buyruğundan sonraki buyruklar hangi bellek adresinden gelecektir? Bu sorunun en basit çözümü dallanma buyruğunun atlayıp atlamayacağı belirlenene işlemciye yeni buyruk getirmemektir. Ancak dallanma buyrukları boru hattının ileri aşamalarından birinde sonuçlanıyorsa bu boru hattının o süre boyunca boş duracağı anlamına gelir. Döngülerin parçası olan dallanma buyruklarında bu bekleme süresi katlanarak artmaktadır.

Bu beklemeyi önlemek için güncel işlemciler dallanmaların atlayıp atlamayacağını tahmin ederler. Bu tahminler genelde işlemcideki dallanmaların geçmişine bağlıdır.

Programın gidişine göre tahmin etmek, ya da birden çok tahmin ediciyi bir araya getirip en iyi tahmini yapan birimi kullanmak da kullanılan seçenekler arasındadır [2].

Dallanma tahmini doğru yapıldığında başarımı artıracağı açıkça görülmektedir. Ancak tahminin yanlış olması bir grup buyruğun işlenmemesi gerekirken işlenmiş olduğu anlamına gelir. YSB benzeri yapı kullanan işlemcilerde dallanmadan sonraki buyrukları tahmin edilen dallanma tahmin sonucunu doğrulamadan işlemeyecek şekilde ayarlayarak bu buyrukların tahminin yanlış olduğu durumlarda sonuçlarını yazmaması sağlanabilir. Ancak yazmaç yeniden adlandırma işlemi çözme aşamasında yapılıyor. Buyruklar YSB'ye girmeden çok önce RAT'e yeniden adlandırılan yazmaçlarını yazmış oluyorlar. Bu yapı tahminin hatalı olduğu durumlarda tahmin yapılan buyruğun anındaki haline bir şekilde dönmek zorunda.

Bu hatalardan kurtulmanın yolları Dallanma Tahmini Hatalarından Kurtulma3.1 numaralı bölümde detaylı olarak anlatılmaktadır. Hatalı tahminden kurtulmanın bu tezde önerilen genişletilebilir diğer bir yolu ise 3.2 numaralı bölümde açıklanmaktadır.

3. ÖNERİLEN YENİDEN ADLANDIRMA YAPISI

Önceki bölümlerde anlatıldığı gibi yazmaç yeniden adlandırma buyruk seviyesinde paralelliği artırmak için yapılan, Yazma Sonrası Yazma ve Okuma Sonrası Yazma sanal veri bağımlılıklarını ortadan kaldıran bir tekniktir. Bu teknik buyruk seviyesinde paralelliği artırmakla birlikte dallanma tahmini gibi tahmin tabanlı çalışan bazı tekniklerin uygulanmasını zorlaştır.

Yazmaçlar çözme aşamasında yeniden adlandırılır. Bu aşamadan sonra buyruklarda belirtilmiş mimari yazmaçlar etiket olarak kullanılır, asıl veri ise fiziksel yazmaçlarda tutulur. Hangi mimari yazmacın içeriğinin herhangi bir anda hangi fiziksel yazmaçta tutulduğu yeniden adlandırma tablosunda tutulur. Bu tablo herhangi bir anda işlemcinin kesin durumunu belirtir. Bu tabloda herhangi bir nedenle herhangi bir hata olması işlemcinin hatalı işlemler yapacağını ve büyük olasılıkla da kısa süre içinde yeniden başlatılana kadar çalışamaz hale geleceği anlamına gelir. Tahmin yapılan tekniklerle birlikte kullanıldığı zaman, yanlış yapılan bir tahmin nedeni ile işlemciye alınan ve çözülen buyruklar bu yeniden adlandırma tablosunu değiştirmiş olacak. Bu buyruklar YSB yardımıyla sonuçlarını yazmadan işlemciden gidebilir ve yayın kuyruğundan atılabilir, ama çözme aşamasında yapılmış olan yeniden adlandırmanın işlemci durumu üzerinde kalıcı bir etkisi olmuş oluyor. Ancak yazmaç yeniden adlandırmanın bize getirdiği başarım artışından da vazgeçmek istemiyoruz. Bu nedenle, tahmin hatalarından kurtulmak için çok yollu, sıra dışı işlem yapan işlemciler çeşitli yöntemler kullanır.

3.1. Dallanma Tahmini Hatalarından Kurtulma

Günümüzde kullanılan tüm işlemciler bir tür dallanma tahmini yapmaktadır. Tahmin hatalı olduğunda bu buyrukları yeniden düzenleme belleğinden ve yayın kuyruğundan atmak yeterli olmamaktadır. Bu buyrukların yeniden adlandırma tablosu üzerindeki etkileri de aynı zamanda geri alınmalıdır. Bu amaçla literatürde ve gerçek işlemcilerde kullanılan çeşitli yöntemler bulunmaktadır [3] [4].

Bu yöntemler ikinci bir yeniden adlandırma tablosuna bağlı olup olmadığı şeklinde ikiye ayrılabilir. Kullanılan bu ikinci yeniden adlandırma tablosuna emeklilik yeniden adlandırma tablosu ya da tamamlanmış yeniden adlandırma tablosu denir. Bu yapının kullanıldığı sistemlerde normal yeniden adlandırma tablosuna ayırt etmek için tahmini yeniden adlandırma tablosu ya da spekülatif yeniden adlandırma tablosu denebilir. Tamamlanmış yeniden adlandırma tablosu bir mimari yazmacın tamamlanan buyruklar tarafından yazılmış en son fiziksel yazmaç karşılığını tutar. Bu tablo boru hattının tamamlama aşamasında, buyruklar yeniden sıralama belleğinden çıkarken güncellenir.

Tamamlanmış yeniden adlandırma tablosu kullanan teknikler *bekleme* ve *ileri yürüme*, tamamlanmış yeniden adlandırma tablosu kullanmayan teknikler ise *geri yürüme* ve *kopya alma* olarak belirlenebilir.

Bekleme: Bu teknikte hata yapan dallanma buyruğu yeniden sıralama belleğinden çıkana kadar beklenir. Bu buyruk yeniden sıralama belleğinden çıktığı andan sonraki buyruklar zaten hatalı tahmin nedeniyle işlenmiş demektir. Bu noktadan sonra yalnızca işlenmesi gereken buyruklar yeniden adlandırmalarını tamamlanmış yeniden adlandırma tablosuna yazdıkları için bu tabloyu tahmini yeniden adlandırma tablosu üzerine kopyalayıp doğru adresten buyruk almaya başlamak sorunu çözer. Bu teknik dallanmanın tamamlanması uzun sürebileceği ve birden fazla dallanma olması durumunda çok fazla bekleme olabileceği nedeniyle çok tercih edilmez.

İleri yürüme: Dallanma hatası olduğu zaman bu teknik bekleme tekniği gibi beklemek yerine yeniden sıralama belleğinde hata yapan dallanmaya kadar olan tüm buyrukları sözde bitirir. Bu sözde bitirme işlemi tamamlanmış yeniden adlandırma tablosu üzerine sanki o buyruklar tamamlanmış gibi yeniden adlandırma kayıtlarını yazar. Bu buyrukların çalışacağı kesin olduğu için bu sözde bitirme işleminin sonunda tamamlanmış yeniden adlandırma tablosu hata yapan dallanma buyruğundan işlemenin tekrar başlamasına izin verecek bir duruma gelir. İşlemenin gerçekten başlayabilmesi için bu tablo tahmini yeniden adlandırma tablosu üzerine kopyalanır. Bir saat darbesinde sözde bitirilen buyrukların sayısı genelde işlemcinin

tamamlama genişliğine eşittir. Eğer hata yapan dallanma yeniden sıralama belleğinin başından ne kadar uzaksa dallanma tahmin hatası cezası o denli fazla olur.

Geri yürüme: Bu yapı tamamlanmış yeniden adlandırma tablosu kullanmaz. Her buyruk çözülme aşamasında yeniden adlandırma tablosunda yaptığı değişiklikleri yeniden sıralama belleğinde saklar. Bir dallanma tahmini hatası olduğu zaman işlemci yeniden sıralama belleğinin en sonundan başlayarak hata yapan dallanma buyruğuna kadar işlenmiş tüm buyrukların yeniden adlandırma tablosunda yaptıkları değişiklikleri geri alır. Hata yapan dallanma yeniden sıralama belleğinin sonundan ne kadar o kadar uzun sürede geri alınabilir. Bir saat darbesinde işlemcinin geri alabileceği buyruk sayısı tamamlama genişliğine eşittir.

Kopya alma: Bu teknikte yeniden adlandırma tablosunun devresi bu tablonun bir grup gölge kopyasını tutacak şekilde değiştirilir. Bir dallanma buyruğu yeniden adlandırma aşamasını geçtiği anda yeniden adlandırma tablosunun güncel halinin bir kopyası alınır. Dallanmanın hata yaptığı ortaya çıkarsa, bu dallanmaya karşılık gelen kopya yeniden adlandırma tablosunun üzerine geri kopyalanır. Eğer tahmin hatası olmadan bir dallanma işlemciden çıkarsa bu dallanmaya karşılık gelen yazmaç yeniden adlandırma tablo kopyası boş olarak işaretlenir ve tekrar kullanılabilir. Bu teknik yeniden adlandırma tablosunun tek bir saat darbesinde kurtarılmasını sağlar. Ancak belirli bir saat darbesi hızı için kullanılabilecek kopya sayısı devre karmaşıklığı nedeniyle sınırlıdır. Bu sınır işlemci içine aynı anda alınabilecek dallanma sayısını da belirler. Eğer kopya tablosu doluyken bir dallanma gelirse bu dallanma bir kopya satırı boşalana kadar yeniden adlandırma aşamasında beklemek zorundadır.

Kopya alma sistemi dışarıda bırakılırsa kullanılan bu yapılar içerisinde başarımı en yüksek olan yapının (Spec 2000 denektaşı program grubu kullanılarak) geri yürüme olduğu gösterilmiştir. Kopya alma sistemi geri yürümeden daha yüksek başarım göstermektedir ancak devre düzeyindeki karmaşıklığı kullanılmasına engel olabilmektedir.

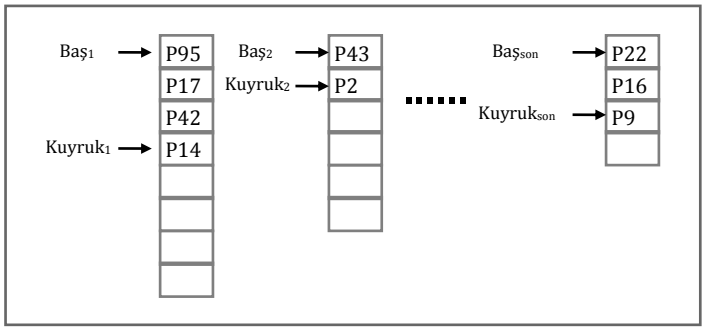
3.2. İGİÇ Kuyrukları Kullanan Yapı

3.1 numaralı bölümde anlatılan kurtarma yapıları güncel işlemcilerde kullanılmaktadır. İşlemci tasarımı yaparken dallanma hatalarından kurtulmanın en kısa sürede olması tercih edilir. Bunu da kopya alma tekniği sağlamaktadır. Ancak bu tekniğin devre karmaşıklığı kullanılmasını engelleyebilir ya da başarımı düşürebilir. Başka bir deyişle, bir işlemci tasarımcısı kopya almanın başarımını basit bir devre kullanarak elde etmek ister. Bu amaçla, her mimari yazmaç için ayrı ayrı geçmiş tutan bir yapı öneriyoruz.

Önerilen yapıda her dallanma geldiğinde tüm yeniden adlandırma tablosunun bir kopyasını almak yerine her yazmaç için o yazmacın geçmişini tutan bir dairesel İGİÇ (İlk giren İlk çıkar – First In First Out, FIFO) kuyruğu öneriyoruz. Bu teknik herhangi bir mimari yazmaca sonuç yazan bir buyruk yeniden adlandırma aşamasına gelip bir fiziksel yazmaç ayırdığı zaman, ayrılan bu yazmacın etiketi İGİÇ kuyruğuna kuyruk işaretçisinin gösterdiği noktadan yazılır. Bu kuyruktaki kuyruk işaretçisi sürekli mimari yazmacın güncel fiziksel yazmaç karşılığını gösterir. Bu yazmaca bağımlı tüm diğer buyruklar kullanmaları gereken fiziksel yazmacı bu kuyruk işaretçisine bakarak okurlar.

İşlemci içerisindeki yeniden adlandırma yapısına göre İGİÇ kuyrukları içerisindeki girdilerin çıkarılması değişebilir. P6 mimarisinde yeniden sıralama belleği aynı zamanda fiziksel yazmaç olarak hizmet eder ve mimari yapı ayrı bir mimari yazmaç öbeğinde tutulur. Bu durumda baş işaretçisi bu mimari yazmacı kullanan bir buyruk tamamlandığı zaman bir ileri gider (kuyruğun en sonundaysa başa döner). Bunun aksine Pentium 4, Alpha 21264 ve MIPS R10000 gibi fiziksel ve mimari yapıyı tek bir yazmaç öbeğinde tutan yapılarda bir fiziksel yazmaç ancak o fiziksel yazmacın gösterdiği mimari yazmacı kullanan başka bir buyruk tamamlandığı zaman serbest bırakılır. Bu durumda önerilen yapıda bir mimari yazmacın değerini tutan bir fiziksel yazmaç serbest bırakıldığı zaman o mimari yazmaca karşılık gelen İGİÇ kuyruğunun baş işaretçisi ilerletilir.

Önerilen yapıda her mimari yazmacın kendine özel bir dairesel İGİÇ kuyruğu olduğu için her İGİÇ kuyruğunun alabileceği girdi sayısı farklı olabilir. Bir buyruk bir mimari yazmaca yazmak istiyorsa ve bu yazmaca karşılık gelen İGİÇ kuyruğunda hiç yer yoksa, bu buyruk yeniden adlandırma aşamasında bu yazmacın İGİÇ kuyruğunda boş bir yer açılana kadar beklemek zorunda kalır. Bu beklemeler yüzünden olan başarım düşüşünü en aza indirmek için İGİÇ kuyruklarının boyları uygun seçilmelidir. Bu İGİÇ kuyruklarının boyları programların ve derleyicilerin ortak davranışları göz önüne alınarak belirlenebilir. Ancak bu belirlemenin en iyi sonucu vereceği garanti değildir. İGİÇ boylarını belirlemek için yapılan deneysel çalışma 5 numaralı bölümde anlatılmaktadır.



Şekil 3.1. Önerilen Yeniden Adlandırma Kuyruk Yapısı

Bir İGİÇ'deki girdilerin sayısı en fazla $(fiziksel\ yazmaç\ sayısı - mimari\ yazmaç\ sayısı)$'na eşit olabilir. Çünkü her mimari yazmacın bir fiziksel yeri olmalıdır ve aynı fiziksel yazmaç aynı anda birden fazla mimari yazmaca atanamaz.

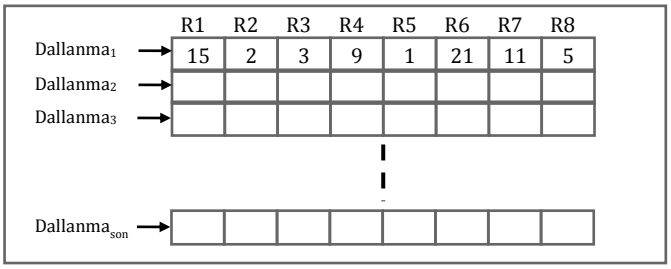
Önerilen bu yapı dallanma tahmini hatalarından kurtulmayı kolaylaştırır. Tüm tahmini yeniden adlandırma tablosu değerleri doğrudan eşlemeli bir SRAM bit hücresi öbeğinde bulunur. Dallanma hatası olduğu durumlarda yeniden adlandırma tablosunu kurtarmak için İGİÇ kuyrukların kuyruk işaretçilerini uygun hallerine getirmek yeterlidir. Bu yapı kullanılarak yukarıda anlatılan tüm dallanma tahmini hatasından kurtulma teknikleri uygulanabilir. Bekleme tekniği kullanan bir işlemcide tamamlanmış yeniden adlandırma tablosu kullanmaya gerek kalmaz: Hata yapan

dallanma buyruğu yeniden sıralama belleğinin en başına ulaştığı zaman İGİÇ kuyruklarının baş işaretçileri işlemci için doğru durumu gösterir hale gelecektir. Hatta İGİÇ kuyruklarının baş işaretçilerinin gösterdiği değerler birleştirilirse tamamlanmış yeniden adlandırma tablosunu oluşturmak mümkündür. Bu özellik güncel işlemcilerde kullanılan tamamlanmış yeniden adlandırma tablosunu ayrıca tutma ihtiyacını ortadan kaldırmaktadır.

İleri ve geri yürüme teknikleri geri dönme işlemleri sırasında İGİÇ kuyruklarında hiçbir verinin üzerine yazılmadığı için, bu yapıyı kullanarak daha basit şekilde gerçekleştirilebilir. İleri yürüme tekniği için sözde tamamlanma işlemi o buyruğun yeniden adlandırdığı mimari yazmaca karşılık gelen İGİÇ kuyruklarının baş işaretçisi bir ilerletilirse kolayca gerçekleştirilebilir. Geri yürüme işleminde ise işlemci yalnızca dışarı atılacak buyruğun yeniden adlandırdığı mimari yazmaca karşılık gelen İGİÇ kuyruğunun kuyruk işaretçisini bir azaltması yeterlidir. Bu şekilde hatalı tahmin edilen dallanma buyruğuna ulaşıldığında kuyruk işaretçileri olması gereken değerlerine geri dönmüş olacaktır.

Kopya alma sistemi de önerilen İGİÇ kuyruğu yapısı ile devre düzeyinde basitleşmektedir. Kopya alma tekniğinde normal durumda her dallanma buyruğu çözüldüğünde tüm yeniden adlandırma tablosunun bir kopyası alınır. Bu işlem yeniden adlandırma tablosunun her bit bitini kayan yazmaç olarak gerçekleştirmeyi gerektirir. Aynı anda işlemci içerisinde alınabilecek dallanma buyruklarının sayısı kopya alma sisteminin tutabildiği kopya sayısıyla sınırlıdır. Bu nedenle mümkün olduğunca fazla kopya satırı olması tercih edilir bir durumdur. Ancak kopya satırlarının sayıları arttığı zaman bir kayma yazmacının mantık derinliği ve bununla birlikte de yeniden adlandırma tablosunun gecikmesi artar. Bu da aynı zamanda işlemcinin saat frekansını kısıtlar. Önerilen İGİÇ kuyruğu yapılarında her dallanma geldiğinde tüm yeniden adlandırma tablosunu kopyalamak gerekmez. Yeniden adlandırma aşamasına gelen bir dallanma buyruğu olduğu zaman tüm İGİÇ kuyruklarının o andaki kuyruk işaretçilerinin saklanması yeterlidir.

Şekil 3.2'de kuyruk işaretçilerinin saklanabileceği örnek bir tablo gözükmektedir. Her girdinin indisi gelen dallanma buyruklarının tanımlayıcıları olarak belirlenir. İç içe gelen dallanma buyrukları durumunda birden fazla dallanma buyruğunun aynı anda tablodan çıkarılması gerekebileceği için bu tablo da dairesel İGİÇ kuyruğu olarak gerçekleştirilmelidir. Tablodaki her girdi bir kuyruk işaretçisi tutar. İGİÇ kuyruk boyları birbirinden farklı olabileceği için her kuyruk işaretçisi aynı sayıda bit ile gösterilmeyebilir. Bu durumda en büyük olası boya bakabiliriz. Her İGİÇ kuyruğu en fazla işlemcideki fiziksel yazmaç sayısı kadar büyük olabilir. Bu durumda her kuyruk işaretçisi $log_2(fiziksel\ yazmaç\ sayısı)$ sayıda bit ile gösterilebilir. Eğer işlemcide 256 fiziksel yazmaç ve 8 adet mimari yazmaç varsa, kuyruk işaretçileri 8 bit ile gösterilebileceği için kopya alma tablosunun her bir satırı 64 bit olmalıdır. 64 bitlik yazmaç boyları varsayıldığında bu yapının gecikmesinin yazmaç öbeğine benzer olacağı görülebilir. Tabii ki bu gecikme işlemci içine alınacak buyruk sayıları artırıldığında artacaktır.



Şekil 3.2. Önerilen Kopya Alma Yapısı

Bir dallanma tahmini hatası olduğu durumda işlemci bu kopya alma tablosuna dallanmanın indisi ile erişerek İGİÇ kuyruklarının kuyruk işaretçilerini kopya alma tablosunda yazan işaretçilerle değiştirir. Bu işlem İGİÇ kuyruklarını dallanma buyruğu gelmeden hemen önceki düzgün duruma geri döndürür. Bu yapının devre karmaşıklığına ve işlemcinin saat frekansına bağlı olarak bu geri dönüşüm bir ya da iki saat darbesinde yapılabilir. Öngörülen donanım tasarımında yeniden adlandırma devresi yükselen ve alçalan saat darbelerinde farklı işler yapmaktadır. Yükselen saat darbesinde doluluk karşılaştırılmaları yapılmakta, düşen saat darbesinde de işaretçiler güncellenmektedir. Kopya alma tablosundan yükselen darbede okunan veri düşen

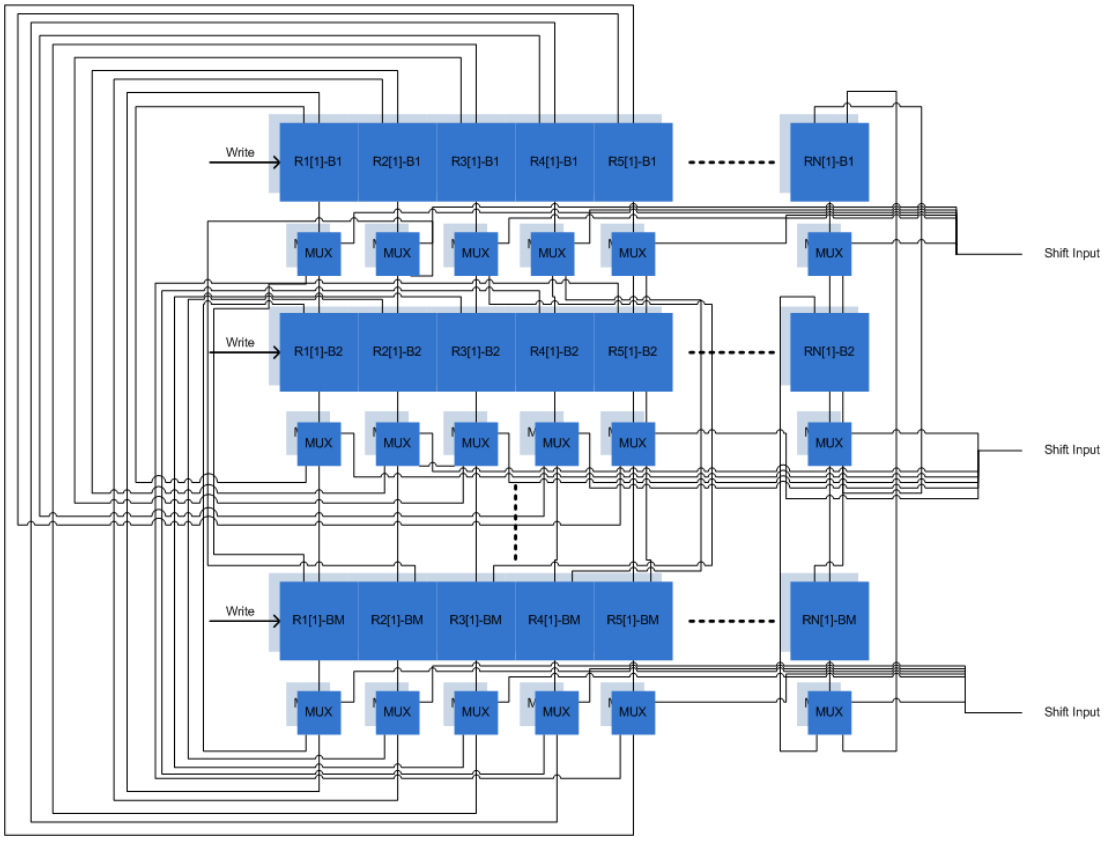
darbeye kadar yeniden adlandırma donanımına ulaşırsa tek saat darbesinde dallanma tahmini hatasından kuyruk işaretçileri güncellenerek geri dönülebilir. Bu mümkün değilse bu işlem iki parçaya bölünüp (yükselen ve düşen darbeleri kullanmak yerine) iki saat darbesinde (ilk saat darbesinde kopya alma tablosundan kuyruk işaretçileri okunur, ikinci saat darbesinde İGİÇ kuyruklarındaki işaretçilerin üzerine yazılır) dallanma tahmini hatasından kurtulmak mümkün olur.

3.3. Donanım Yapısı

Yeniden adlandırma tabloları, işlemci içerisindeki herhangi bir bellek yapısı gibi SRAM bit hücreleri kullanılarak gerçekleştirilir. Bekleme ya da ileri/geri yürüme yöntemini kullanan mimarilerde sade SRAM bit hücreleri yeterli olmaktadır, ancak kopya alma sistemini kullanan yapılarda her bit hücresi kayan yazmaç özelliklerine sahip olmalıdır. İşlemcinin içerisinde 16 adet kopya satırı bulunması isteniyorsa yeniden adlandırma tablosunun her bit hücresi 16 bitlik bir kayan yazmaç olmalıdır. N adet mimari yazmacı olan bir işlemcideki kopya alma yapısında kullanılacak donanımın basitleştirilmiş bir örneği Şekil 3.3'de görülmektedir.

Önerilen yapı kopya alma donanımının aksine kayan yazmaç gibi yapılara ihtiyaç duymaz ve bu karmaşıklığı bit hücresi seviyesinden kaldırır. Tüm İGİÇ kuyrukları basit SRAM bit hücreleri ile gerçekleştirilir. Her İGİÇ kuyruğu aslında yeniden adlandırma tablosu ile aynı bit genişliğine sahip ancak daha fazla girdisi olan ufak bir rasgele erişilebilir bellektir.

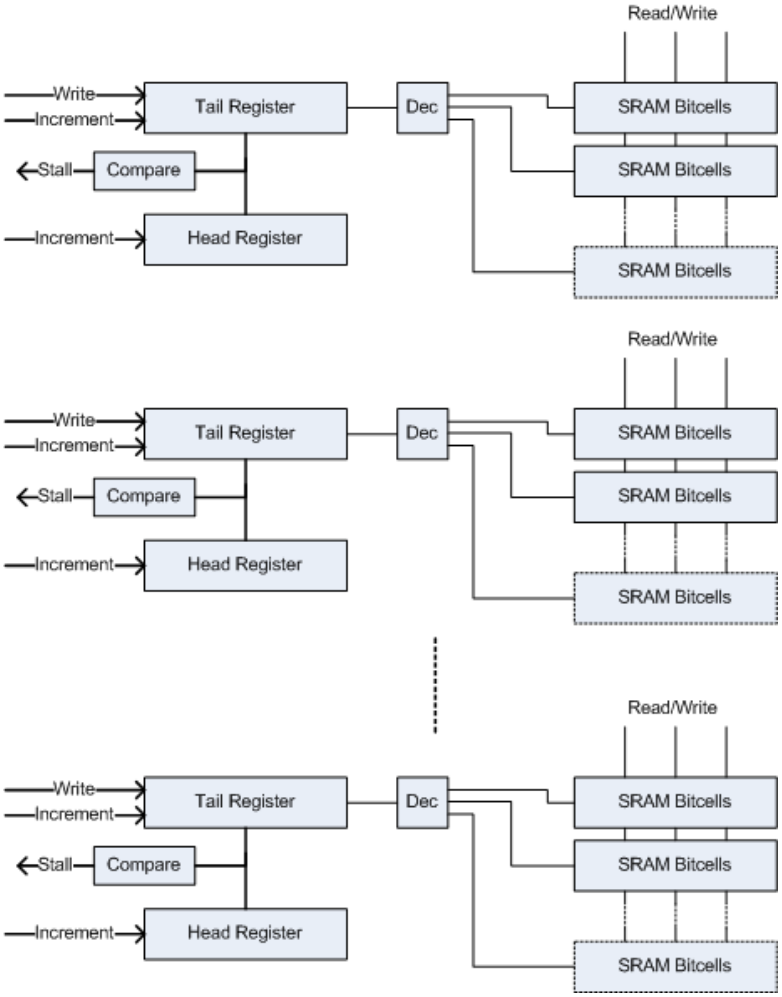
Temel 4-yollu bir yapıda yeniden adlandırma tablosunun 4 adet yazma kapısı (4 ayrı buyruk 4 ayrı mimari yazmacı yeniden adlandırırsa) ve 8 adet okuma kapısı (her buyruğun birbirinden farklı kaynak yazmaçları olursa) vardır. Bu yapı normalde bağımlılığı kontrol eden devre bir saat darbesinde olabilecek birden fazla yeniden adlandırmayı ortaya çıkardıktan sonra kullanılır. Önerilen yapıda bağımlılık kontrolü yapan devre değişmez, ancak yazmaç yeniden adlandırmalarının tutulması için yeni bir yapı sunar.



Şekil 3.3. Kopya Alma Yapısının Donanımı

Şekil 3.4'de İGİÇ kuyruklarının donanım yapısı görülebilmektedir. N adet mimari yazmaca sahip olan bir mimaride önerilen yapı N adet SRAM bit hücrelerinden oluşan İGİÇ kuyruğu kullanılması gerektiğini söyler. Her bir kuyruk rasgele erişim için normal çözücü devreleri bulunan normal bir yük RAM'idir. Ancak bu çözücülerin girdileri gelen buyruklar veri almak yerine doğrudan ilgili İGİÇ kuyruğunun kuyruk işaretçisine bağlıdır. Yeniden adlandırılan buyruklar fiziksel yazmaç etiketlerini normal bit satırlarını kullanarak kuyruk işaretçisi tarafından otomatik olarak seçilen hücrelere yazarlar. Göz önünde bulundurulması gereken başka bir nokta ise 4 yollu bir makinede her bir İGİÇ kuyruğuna 4 değere kadar değer yazılması mümkün olmasına rağmen yalnızca tek bir değer okunur. Dolayısıyla, her İGİÇ yapısı 4 adet yazma kapısı ve 1 adet okuma kapısı içerir. Ayrıca, yapı herhangi bir yazmaç öbeği gibi rasgele erişime açık olmasına rağmen erişim rasgele değildir. Her saat darbesinde kuyruk işaretçisinin gösterdiği değer okunur ve buna karşılık gelen mimari yazmaç yeniden adlandırıldıysa bu noktadan

sonraki en fazla dört değere veri yazılır. Tüm yeni yazmaç atamaları yazıldıktan sonra kuyruk işaretçisi güncellenir. Tüm İGİÇ kuyrukları dairesel belleklerdir. Yani kuyruk ve baş işaretçileri kuyruğun en son değerinden bir sonra en baş değerini gösterir.



Şekil 3.4. İGİÇ Kuyruklarının Donanımı

İGİÇ kuyruklarının baş işaretçileri de her kuyruk için ayrı yazmaçlarda tutulur ve boru hattının bitirme aşamasında güncellenir. Bir buyruk bitirilip yeniden sıralama belleğinden çıkarıldığı zaman sonucunu yazdığı mimari yazmaca karşılık gelen İGİÇ kuyruğunun baş işaretçisini bir artırır. Buyruklar boru hattının bitirme aşamasından İGİÇ kuyruklarındaki veri yüküne ulaşmaz – buradan yalnızca baş işaretçilerine erişirler.

Yeniden adlandırılması gereken bir mimari yazmaç için baş işaretçisi kuyruk işaretçisinden bir ileri ise işlemci yeniden adlandırma aşamasında takılır. Bu yazmaçlar dallanma hatası gibi olağan dışı durumlar haricinde yalnızca artırılırlar, bu nedenle bu işaretçilerin tutulduğu yazmaçların paralel veri yükleme yeteneğine sahip sayaçlar şeklinde gerçekleştirilmesi gerekir.

Kuyruk işaretçilerinin bir dallanma yeniden adlandırma aşamasına ulaştığındaki kopyalarını saklayarak dallanma tahmini hatası olması durumunda geri getirilmesini sağlayacak kopya alma tablosu ise basit bir saklama belleği yapısıdır. Bu yapı SRAM bit hücreleri kullanılarak gerçekleştirilir ve kopya satırı sayısı aşırı miktarda artırılmadığı durumlarda yazmaç öbeğinden daha küçük olacaktır. Bu da bu yapıya bir saat darbesinden daha kısa sürede erişilebileceği anlamına gelir.

Önerilen yapıda karmaşıklık işaretçi yazmaçlara kaydırılmıştır. Bu yazmaçlar sayaç olmakla birlikte anında veri yüklemek için de bir yapıya sahip olmaları gerekir. Bunun yanında baş ve kuyruk işaretçilerinin karşılaştırılması için de bir devre bulunması gerekir.

3.4. Tamamı Dairesel İGİÇ Kuyrukları ve Çift Yazmaç Tekniği

Önerilen yeniden adlandırma yapısında bazı geliştirmeler üzerinde de çalışıldı. Önceki bölümlerde önerilen İGİÇ kuyruk yapılarının her yazmaç için aynı olmayacağından bahsedildi. Normal durumda, bu yazmaçlara ait İGİÇ kuyruk boyları belirlenirken ortalama değerlere bakılarak karar verilmesi gerekiyor. Ancak ortalamada başarımı artıran değerler her sonuç alınırken göz önünde bulundurulan her denektaşı programı için en iyi sonucu vermiyor. Benzer şekilde tam sayı denektaşı programları kayan nokta yazmaçlarını hiç kullanmıyor, ancak kayan nokta denektaşı programları hem tam sayı yazmaçlarını, hem de kayan nokta yazmaçlarını kullanıyor. İGİÇ kuyruk boylarının işlemcinin çalışması sırasında dinamik olarak belirlenmesinin bu duruma yardımcı olup olmayacağı araştırıldı. Yapılan denemeler ve gözlemler sonucu tamamen dinamik bir İGİÇ kuyruğu yapısının gerçekçi olmayacağı anlaşıldı.

Her İGİÇ kuyruğunun dinamik olarak boyunun değişebilmesi için bütün işaretçilerin toplam İGİÇ kuyruklarının toplam boyu (ya da her İGİÇ kuyruğunun boyuna bir alt sınır konulduysa İGİÇ kuyruklarının toplam boyu ile yazmaç sayısının bu alt sınır ile çarpımının farkı) kadar girdiyi gösterebilecek boya çıkması gerekti. Çünkü boyların dinamik olarak değişmesi durumunda tüm İGİÇ kuyruk girdileri tek bir mimari yazmaç için kullanılabilir hale gelmesi söz konusu. Kuyrukların dinamik olarak boylarını değiştirebilmesi için her kuyruğun tek bir büyük bellek yapısının parçası olması da gerekir. Bu yapının yazma kapıları önceki tasarımdaki herhangi bir kuyruktaki yazma kapıları kadar olsa yeterlidir. Dört yollu bir işlemci için bir saat darbesinde en fazla dört yeni yazmaç yeniden adlandırılabilir, bu da bir İGİÇ kuyruğunun yazma kapıları sayısı ile aynıdır. Ancak bu yapının tek bir İGİÇ kuyruğunun aksine daha fazla okuma kapısı olması gerekir. Dört yollu bir işlemci için çözülen dört buyruğun her birinin iki kaynak yazmacı olduğu durumda aynı anda sekiz değer okunmalıdır.

Baş ve kuyruk işaretçilerine ise ayrı kontrol yapıları eklenmelidir. Kuyruk ve baş işaretçilerinin başa dönecekleri nokta ve dönecekleri baş noktası ayrı bir yazmaçta belirlenmelidir – her yazmaç için bir baş noktası belirlenir ve dairesel bir şekilde sıradaki yazmacın baş noktası bir önceki yazmacın son noktası olur. Bunun yanında bu yazmaçlar arasında karşılaştırma yapılması da gerekir, bu da devre karmaşıklığını artırır. Bir İGİÇ kuyruğunun büyütülmesi için bu sınır yazmaçları üst sınır için azaltılmalı, alt sınır için ise artırılmalıdır. Ancak İGİÇ kuyrukları kendi içlerinde de dairesel oldukları için bu değiştirme işlemi ancak baş işaretçisi kuyruk işaretçisinden daha küçük bir değer gösterdiği zaman mümkün olmaktadır. Bu koşulun yanında bu sınırlar değiştiği zaman kopya alma tablosunda bu değişimden etkilenen yazmaçların girdileri olmaması gerekir – bir dallanma tahmini hatası olduğunda kuyruk yazmaçları düzeltilirse kuyruk yazmaçları hatalı bir şekilde artık o İGİÇ kuyruğuna ait olmayan bir noktayı gösterir hale gelebilir.

Bu yapının çalışması için hangi yazmacın daha çok kullanıldığı işlemci içerisinde belirlenmeli ve buna göre yazmaçların boyları ayarlanmalıdır. Bir önceki paragrafta

bahsedilen zorluklar bu boy değiştirme sırasında sıkıntılara yol açar. Koşturulan programın bir aşamasında X yazmacı çok sık kullanılarak İGİÇ kuyruğunun boyunu artırmak ister. Ancak etrafındaki iki yazmaçtan alabileceği boy sınırlıdır. Bu boyu artırmak için diğer yazmaçlardan da bloklar alınması gerekir: bu da ancak X yazmacına ait İGİÇ kuyruğunun etrafındaki kuyrukların X yazmacının kuyruğuna bir girdi verip, aynı anda da komşu diğer yazmaçtan bir girdi edinerek en az alabilecekleri boylarını korumalarıyla mümkün olur. Bu hamle tek bir yazmaca ait İGİÇ kuyruğunun etrafındaki iki yazmacın kuyruklarından girdi almasına göre daha çok koşulu olan bir işlemdir: X yazmacının kuyruğuna girdi verecek olan Y yazmacının kuyruğu hem X'e girdi verebilecek, hem de diğer komşusundan girdi alabilecek durumda olmalıdır.

Bu koşullar altında Denektaşı Programları bölümünde detaylandırılan denektaşı programlarıyla yapılan deneylerde yazmaçların kuyrukları programın başında bir yöne sarkıp, program süresince kısa süreli kullanımı artan yazmaçların kuyruk boylarını yeterince hızlı artıramaz hale geldi. Yukarıda bahsedilen zorluklarla birlikte bu sonuçlar bu şekilde dinamik kuyruk boyu düzenlemenin kullanılabilir bir yöntem olmadığını kanıtladı.

Ancak tüm yazmaçları tek bir dairesel büyük kuyruk kullanarak yönetmek yerine yazmaçları ikili ikili gruplamak uygulaması mümkün bir sistem oluyor. Yukarıdaki donanım karmaşıklıkları bu durumda da geçerli. Ancak bu sefer İGİÇ kuyrukları bir tam sayı yazmacınınki bir kayan nokta yazmacı ile eşleşecek şekilde sıralandı. Bu şekilde devre karmaşıklığının artması karşılığında önerilen yapının yarısı kadar yazmaç kullanarak benzer bir başarım elde edilmiş oldu.

3.5. Kopya Alma İle Birlikte İGİÇ Kuyrukları Kullanımı İçin Geliştirme

İGİÇ kuyrukları ile kopya alma tekniğinin birlikte kullanılacağı durumlarda İGİÇ kuyrukları üzerinde başarımı artıracak bir ek yapılabilir. İGİÇ kuyruklarına normal durumda o mimari yazmacı hedefleyen herhangi bir buyruk yeniden adlandırıldığında yeni bir girdi yazılır. Ancak kopya alma kullanılacağı durumlarda

geri dönülecek noktaların bilinmesi yeterlidir – diğer bir deyişle herhangi bir dallanmanın geldiği anda son durum İGİÇ kuyruklarında bulunursa bu bizim amacımız için yeterli olur.

Bunu sağlamak oldukça basittir: Buyruklar yeniden adlandırıldıklarında sonuçlarını yazacakları mimari yazmaca karşılık gelen İGİÇ kuyruğuna normalde olduğu gibi yeniden adlandırma sonucu elde ettikleri fiziksel yazmaç etiketini yazabilirler. Ancak normalin aksine kuyruk işaretçileri bu yazma sonucunda ilerlemez. Kuyruk işaretçileri yalnızca yeni bir dallanma buyruğu çözme aşamasından geçtiğinde ilerler. Böylece İGİÇ kuyrukları yalnızca dallanma olduğu zaman doluluğunu artırmış olur. Bir dallanma buyruğu gelmediği sürece aradaki herhangi bir duruma dönmek gerekmeyeceği için kopya alma ile birlikte İGİÇ kuyruklarının kullanımında bu yöntem kullanarak başarım artışı sağlanabilir.

Bu yöntem kullanılırsa İGİÇ kuyruklarının ihtiyaç duyduğu yazma kapılarının sayısı da azalacaktır. Örnek olarak 4 yollu bir işlemcide tek bir çevrimde bir İGİÇ kuyruğuna 4 değil, 2 girdi yazılır.

Tablo 3.1. Aynı Anda İGİÇ Kuyruğuna İki Yazma Yapan Buyruklara Örnek

```
ADD R1, R2, R2
JLE R3, R2
ADD R1, R3, R3
JGE R2, R3
```

Tablo 3.1'de görülen buyruklar R1 yazmacına ait İGİÇ kuyruğuna iki defa yazma yapar. İki tane aynı yazmaca yazan buyruk ve aralarında dallanmalar gelmediği durumda bir İGİÇ kuyruğuna bir saat darbesinde iki defa yazmak gerekmez. Daha geniş yollu işlemciler için bir İGİÇ kuyruğuna bu yöntem kullanılarak en fazla kaç girdi yazılabileceği (3.5.1) denklemi ile gösterilebilir.

$$En\ Fazla\ Yazma = \left\lfloor \frac{yol\ genişliği}{2} \right\rfloor \qquad (3.5.1)$$

4. İGİÇ KUYRUK İLE YAZMAÇ YENİDEN ADLANDIRMA DENEYLERİ

Önerilen yeniden adlandırma yapısının test edilmesi için PTLSim [5] ve M-Sim [6] benzetimcileri kullanılarak denektaşı programlar kullanılarak deneyler yapıldı. PTLSim C++ programlama dili ile yazılmış, x86 ve x86-64 [7] buyruk kümesini çok yollu olarak çalıştırabilen bir benzetimcidir. M-Sim ise SimpleScalar [8] benzetimcisi üzerine geliştirilmiş, Alpha [9] işlemcilerinin buyruk kümesini çok yollu olarak çalıştırabilen bir benzetimcidir.

PTLSim günümüzde ev bilgisayarlarında sıkça kullanılan bir buyruk kümesini çalıştırır. Ancak altyapı olarak üzerinde çalıştığı işlemciyi kullandığı için her çalışmada tutarlı sonuç vermemektedir. Bunun aksine M-Sim kullanımı daha kolay bir benzetimci olup, daha kısa sürede daha tutarlı sonuçlar verebilmektedir. Bu nedenle çalışmaya başlanmasından kısa süre sonra M-Sim benzetimcisi kullanılmaya başlanmıştır.

4.1. Denektaşı Programları

Sistemin denenmesi için Spec 2000 [10] denektaşı program grubu kullanılmıştır. Bu program grubu işlemcilerin başarımını ölçmek için çeşitli tam sayı ve kayan nokta denektaşı programları içerir. Bu denektaşı programlarının isimleri ve açıklamaları Tablo 4.1 ve Tablo 4.2'de verilmiştir.

Tablo 4.1. Spec 2000 Tam Sayı Denektaşı Programları

Program	Yazıldığı Dil	Açıklama
gzip	C	Veri sıkıştırma
vpr	C	FPGA Devre Yerleştirme ve Düzenleme
gcc	C	C Programlama Dili Derleyicisi
mcf	C	Kombinasyonal İyileştirme

crafty	C	Satranç Oyunu
parser	C	Kelime İşleme
eon	C++	Sanal Gerçeklik
perlbmk	C	PERL Programlama Dili
gap	C	Grup Teorisi, Yorumcu
vortex	C	Nesneye Dayalı Veritabanı
bzip2	C	Veri Sıkıştırma
twolf	C	Yerleştirme ve Düzenleme Benzetimliği

Tablo 4.2. Spec 2000 Kayan Nokta Denektaşı Programları

Program	Yazıldığı Dil	Açıklama
wupwise	Fortran 77	Fizik/Kuantum Chromo Devingenleri
swim	Fortran 77	Akışkan Modelleme
mgrid	Fortran 77	3 Boyutlu Problem Çözümü
applu	Fortran 77	Parabolik / Eliptik Kısmi Diferansiyel Denklemler
mesa	C	3 Boyutlu Grafik Kütüphanesi
galgel	Fortran 90	Akışkanlar Dinamiği
art	C	Görüntü Tanıma / Yapay Sinir Ağları
equake	C	Sismik Dalga Benzetimliği
facerec	Fortran 90	Görüntü İşleme: Yüz Tanıma
ammp	C	Sayısal Kimya
lucas	Fortran 90	Sayı Teorisi
sixtrack	Fortran 77	Yüksek Enerjili Nükleer Fizik İvmelendiricisi Tasarımı
apsi	Fortran 77	Meteoroloji: Hava Kirliği Dağılımı

4.2. PTLSim ve M-Sim Benzetim Parametreleri

PTLSim ve M-Sim benzetimcileri ile yapılan deneylerde benzetimcilerin benzettikleri makinelerin parametreleri aşağıdaki tablolarda verilmiştir.

Tablo 4.3. PTLSim Benzetim Parametreleri

Parametre	**Yapılandırma**
Makine Genişliği	4 buyruk getir, 4 buyruk yayınla, 4 buyruk bitir
Pencere Boyu	32 satır yayın kuyruğu 48 satır yükle kuyruğu 32 satır sakla kuyruğu 128 satır YSB
İşlem Birimleri (Sayı/Gecikme)	Tamsayı AMB (2/1) Yükleme birimi (2/2) Kayan nokta işlem birimi (2) Saklama birimi (2/2)
Gecikmeler	AMB Çarpım (4) AMB Bit Taraması (3) AMB Bölme (32) Kayan Nokta Aritmetik (6) Kayan Nokta Vektör Aritmetiği (1)
L1 Buyruk Önbelleği	32 KB, 4 yollu kümeli ilişkili, 64 bayt satır, 1 çevrim isabet zamanı
L1 Veri Önbelleği	16 KB, 4 yollu kümeli ilişkili, 64 bayt satır, 1 çevrim isabet zamanı
L2 Tümleşik Önbellek	256 KB , 16 yollu kümeli ilişkili, 64 bayt satır, 6 çevrim isabet zamanı
L3 Tümleşik Önbellek	4 MB, 32 yollu kümeli ilişkili, 64 bayt satır, 16 çevrim isabet zamanı
Dallanma Tahmini	Çift durumlu ve iki aşamalı tahmin edici
Bellek	140 çevrim gecikme

Tablo 4.4. M-Sim Benzetim Parametreleri

Parametre	**Yapılandırma**
Makine Genişliği	4 buyruk getir, 4 buyruk yayınla, 4 buyruk bitir
Pencere Boyu	32 satır yayın kuyruğu 48 satır yükle kuyruğu 32 satır sakla kuyruğu

	128 satır YSB 128 tamsayı yazmaç öbeği 128 kayan nokta yazmaç öbeği
İşlem Birimleri (Sayılar)	Tamsayı AMB (4) Tamsayı Çarpma/Bölme (1) Bellek Kapıları (2) Kayan Nokta AMB (4) Kayan Nokta Çarpma/Bölme (1)
L1 Buyruk Önbelleği	64 KB, 2 yollu kümeli ilişkili, 64 bayt satır, 1 çevrim isabet zamanı
L1 Veri Önbelleği	128 KB, 4 yollu kümeli ilişkili, 64 bayt satır, 1 çevrim isabet zamanı
L2 Tümleşik Önbellek	512 KB, 16 yollu kümeli ilişkili, 64 bayt satır, 10 çevrim isabet zamanı
Branch Predictor	Çift Durumlu Tahmin Edici
Bellek	İlk grup için 300 çevrim, takip eden gruplar için 2 çevrim isabet zamanı, istek başına 4 grup.

Denektaşı programların benzetimi beş yüz milyar buyruk koşturularak yapılmıştır. Her program için SimPoint [11] kullanılarak benzetime başlanmadan önce belirli bir noktaya kadar ilerlenmiştir. Bu noktadan sonra PTLSim ve M-Sim kullanılarak beş yüz milyar buyruk koşturulmuş ve aksi ifade edilmedikçe bütün sonuçlar bu şekilde elde edilmiştir.

İlk aşamada PTLSim kullanılarak denektaşı programları x86 mimarisinde koşturulmuştur. Olumlu sonuçlar alındıktan sonra daha fazla yazmacı olan Alpha işlemcisinin benzetimini yapan M-Sim benzeticisi kullanılmaya başlanır. Başarım metriği olarak Çevrim Başına Buyruk (ÇBB) kullanılmıştır.

Önerilen İGİÇ kuyruklarını kullanan yapıdan önce PTLSim içerisinde gerçekleştirimleri bulunan kopya alma ve geri yürüme algoritmaları denenmiştir. Daha sonra İGİÇ kuyrukları x86 mimarisindeki genel amaçlı sekiz yazmaç için PTLSim içerisinde bulunan kopya alma algoritması değiştirilerek gerçekleştirilmiş ve denenmiştir. Bu yazmaçlara atanan İGİÇ kuyruklarının boyları ise yazmaçların program çalışması sırasındaki kullanımları incelenerek belirlenmiştir.

PTLSim sonuçları alındıktan sonra Alpha işlemcisi benzetimine geçilmiştir. M-Sim PTLSim'in aksine Alpha işlemcisinin benzetimini yapar. Alpha işlemcisinde 32 adet tam sayı yazmacı, 32 adet de kayan nokta yazmacı bulunmaktadır. Bu yazmaçlardan iki tanesi (bir tanesi tam sayı, bir tanesi de kayan nokta olmak üzere) donanımda toprağa bağlıdır (değerleri sürekli sıfır) ve bu nedenle yeniden adlandırılmaz. Bu nedenle M-Sim ile çalışırken toplamda 62 adet mimari yazmaç için uygun İGİÇ kuyrukları oluşturmayı amaçladık.

Başlangıç için Alpha işlemcisi için de yazmaçların kullanım oranları bulunmuştur. Bu işlemcide de sırasıyla geri yürüme, kopya alma, 2 çevrimde dallanma hatasından dönebilen İGİÇ kuyruğu ve 1 çevrimde dallanma hatasından dönebilen İGİÇ kuyruğu teknikleri çalıştırılmıştır. Daha sonra başlangıçta bulunan İGİÇ kuyruk boyları denenmiş ve başarım üzerine etkileri incelenmiştir.

Önceki benzetmelerin sonuçlarında karşılaşılan beklenmedik başarım düşüşlerine cevap aramak için denektaşı programlarının dallanma nitelikleri araştırılmıştır. Sonuç olarak YSB'de bulunan ortalama dallanma sayısı, YSB'de bulunan en fazla dallanma sayısı ve dallanma buyrukları arasında bulunan ortalama buyruk sayısı bulunmuştur. Bu veriler kullanılarak başarım düşüşleri ve diğer bazı nitelikler açıklanmıştır.

Son olarak 3.4 numaralı bölümde anlatılan çift yazmaç tekniği ve 3.5 numaralı bölümde anlatılan geliştirme uygulanmış ve sonuçları bu tekniklerin kullanılmadığı İGİÇ kuyrukları ile karşılaştırılmıştır.

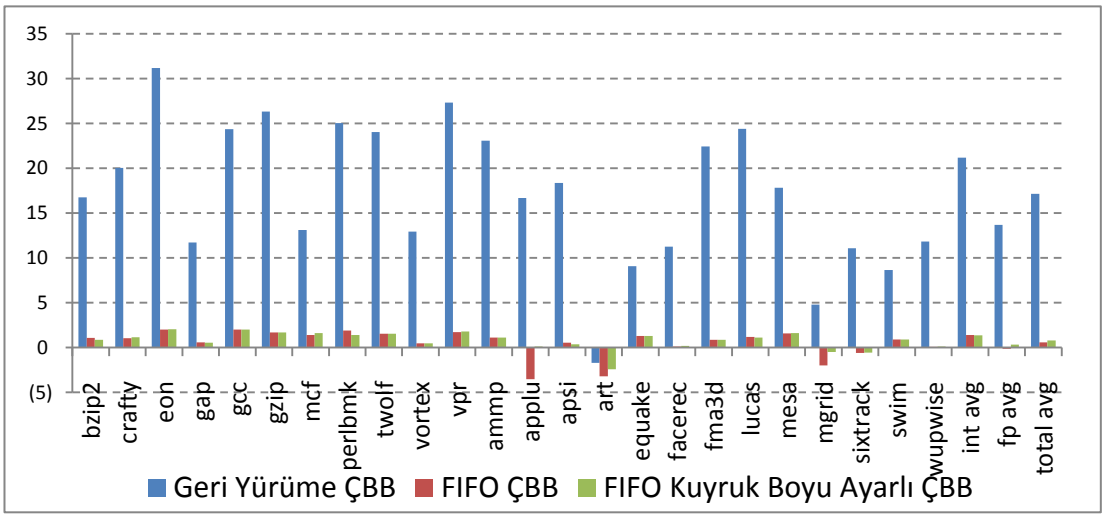
4.3. x86 İGİÇ Benzetimi

Şekil 4.1 PTLSim kullanılarak ölçülmüş sonuçları gösterir. Şekil 4.1 denektaşı programları için kopya alma yöntemine oranla geri yürüme, İGİÇ kuyruğu ve İGİÇ kuyruk boyu ayarlı sistemlerin yüzde kaç daha yavaş olduğunu gösterir. Kopya alma yönteminde kopya satırı sayısı deneme amaçlı sonsuz seçilmiştir. Geri yürüme yönteminde PTLSim dallanma hatası olduğunda bir çevrimde bitirme genişliği kadar buyruğu geri alabilecek şekilde ayarlanmıştır. İGİÇ ÇBB şeklinde işaretli grup ise

her İGİÇ'nun boyu 16 olacak şekilde ayarlıdır. İGİÇ Kuyruk Boyu Ayarlı ÇBB şeklinde işaretli grupta ise (4.3.1) denklemindeki formül ile hesaplanmıştır.

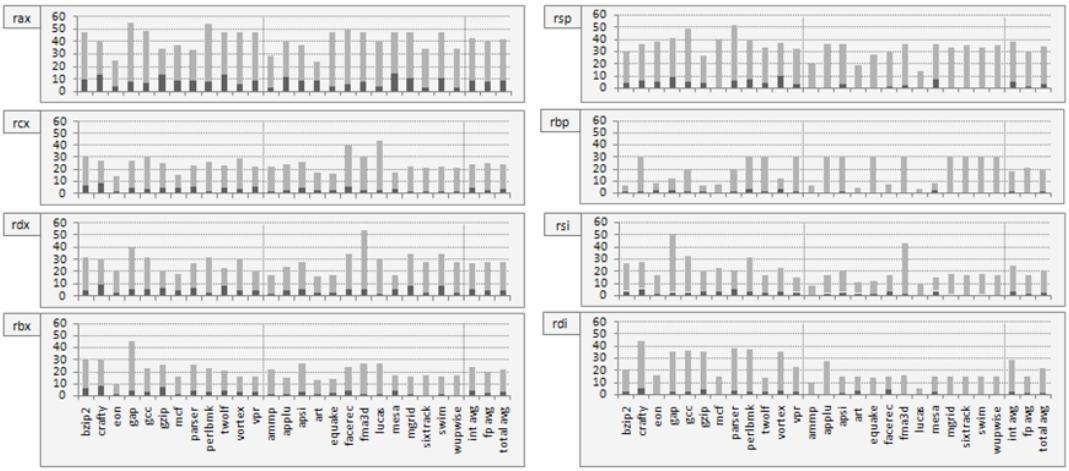
$$\text{İGİÇ } kuyruk\ boyu = 16 * Yazmaç\ Kullanım\ Faktörü \qquad (4.3.1)$$

(4.3.1)'deki Yazmaç Kullanım Faktörü yazmaçların ortalama ne kadar kullanıldığı kullanılarak hesaplanmış bir değerdir. Yazmaçlara en çok kullanılandan en az kullanılana kadar 1 ve 2 arasında değerler atanmıştır. Bu durumda en çok kullanılan yazmacın İGİÇ kuyruğunda 32 satır, en az kullanılan yazmacın İGİÇ kuyruğunda ise 16 satır bulunacaktır.



Şekil 4.1. PTLSim Başarım Grafiği

Şekil 4.2'da ise x86 buyruk kümesindeki sekiz genel amaçlı yazmacın kullanım oranları görülmektedir. Bu grafikte açık gri çubuklar en fazla yazmaç kullanımını, koyu gri çubuklar ise ortalma yazmaç kullanımını temsil eder. (4.3.1) denklemindeki Yazmaç Kullanım Faktörünün hesaplanması bu veriler kullanılarak yapılmıştır. Yapılan hesaplama, Yazmaç Kullanım Faktörü ve sonuç olarak ortaya çıkan İGİÇ Kuyruk Boyu Tablo 2.1'de görülmektedir.



Şekil 4.2. x86 Genel Amaçlı Yazmaç Kullanım Grafiği

Tablo 4.5. x86 Genel Amaçlı Yazmaç Kullanım Verileri

	En Fazla	**Ortalama**	**Yazmaç Kullanım Faktörü**	**İGİÇ Kuyruk Boyu**
RAX	41.64	7.98	2	32
RBX	21.44	3.50	1.372549	22
RCX	24.28	5.12	1.5994398	26
RDX	27.20	2.88	1.2857143	21
RSP	33.20	3.26	1.3389356	21
RBP	18.20	0.84	1	16
RSI	20.28	2.24	1.1960784	19
RDI	21.36	1.94	1.1540616	18

Şekil 4.1'deki sonuçları değerlendirmeden önce bu işlemlerin PTLSim için yalnızca tam sayı yazmaçlarına uygulandığı göz önünde bulundurulmalıdır. Bunu göz önünde bulundurarak Şekil 4.1'deki sonuçlara bakıldığında İGİÇ kuyrukları kullanan tekniğin kopya alma tekniğinden başarımının %1 - %3 arası daha az olduğu görülüyor. Bunun karşılığı olarak devre yapısı oldukça basitleştirilmiş bir sistem

kullanılmış oluyor. Tam sayı ortalamalarına baktığımızda ise boyları ayarlanmış İGİÇ kuyrukları ile tümünün boyu 16 olan İGİÇ kuyrukları arasında çok fazla başarım farkı görülmüyor. Bu aşamadan sonra araştırmalara M-Sim ile devam edildiği için PTLSim ile yapılan çalışma yalnızca önerilen sistemin çalışıp çalışmadığını gösteren bir deney olmuştur ve bu doğrultuda amacına da ulaşmıştır.

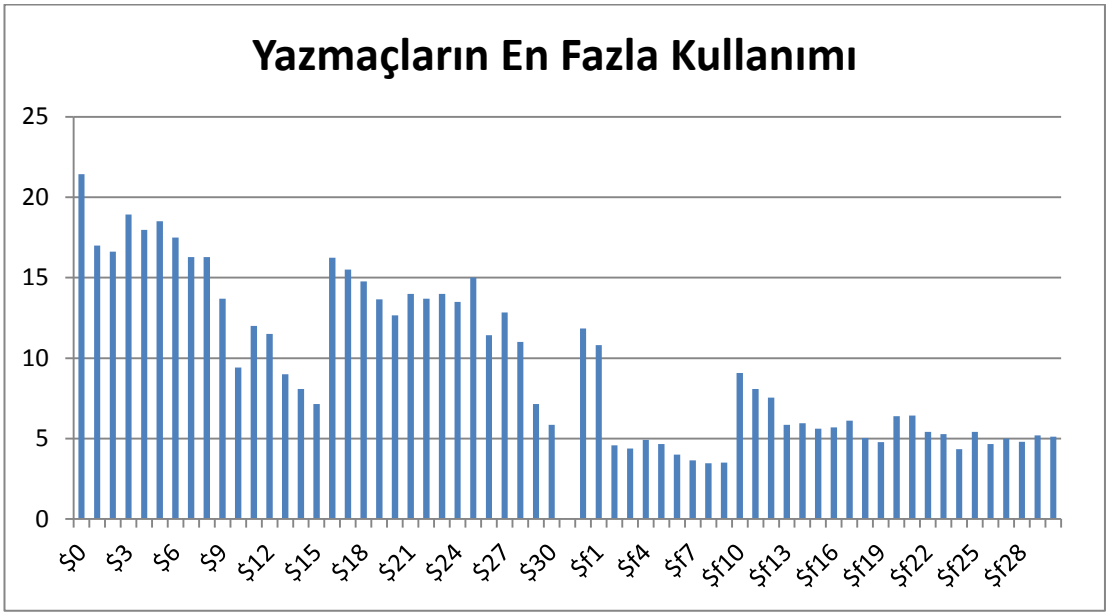
4.4. Alpha İGİÇ Benzetimi – Yazmaçların Kullanım Oranlarının Bulunması

M-Sim ile çalışmaya başladığımızda ilk amacımız PTLSim benzeticisinde olduğu gibi hangi yazmaçların diğerlerine oranla daha fazla kullanıldığını bulmak oldu. Bu amaçla denektaşı programları çalıştırıldı ve mimari yazmaçların en fazla ne kadar kullanıldığı bulundu. İGİÇ kuyruk boyları genel olarak seçileceği için bizim için bu değerlerin ortalaması önemli – her denektaşı programı için özel İGİÇ kuyruk boyları belirlememiz mümkün değil.

Şekil 4.3'de görüldüğü gibi her yazmaç aynı miktarda kullanılmamaktadır. Özellikle kayan nokta yazmaçlarının denektaşı programlarının yarısında hiç kullanılmadığı ve tam sayı yazmaçlarının her türlü programda sıkça kullanıldığı göz önüne alınırsa tam sayı yazmaçlara İGİÇ kuyruk boylarında öncelik vermek daha uygun gibi görünüyor. İGİÇ kuyruk boylarının belirlenmesi için bu verilerin de yardımıyla (4.4.1) denkleminde gösterilen metrik geliştirilmiştir.

$$\text{İGİÇ } Kuyruk\ Boyu = M_{Ortalama} + M_{Standart\ Sapma} \qquad (4.4.1)$$

$M_{Ortalama}$ yazmacın en fazla kullanım değerinin denektaşı programları arasında ortalamasını belirtir. Aynı şekilde $M_{Standart\ Sapma}$ ise en fazla kullanım değerlerinin standart sapmasıdır. Bu metrik ile belirlenen İGİÇ kuyruk boyları ile yapılan denemeler yanında sabit boydaki İGİÇ kuyrukları ile de deneyler yapılmıştır. Belirlenen İGİÇ kuyruk boyu değerleri Tablo 4.6'da verilmiştir.



Şekil 4.3. Yazmaçların En Fazla Kullanımı

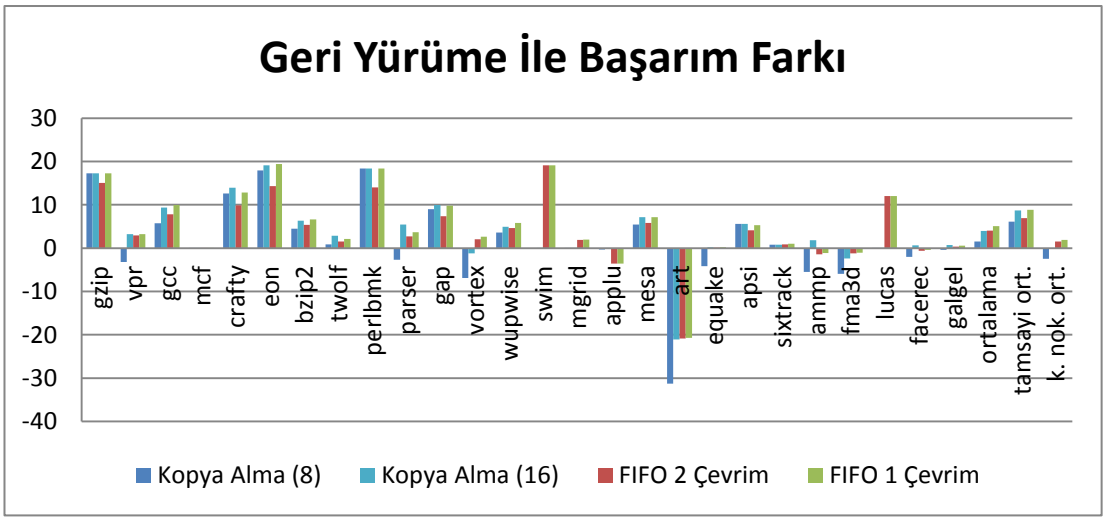
Tablo 4.6. Belirlenen İGİÇ Kuyruk Boyları

Yazmaç Numarası	İGİÇ Kuyruk Boyu	Yazmaç Numarası	İGİÇ Kuyruk Boyu
$0	36	$f0	20
$1	29	$f1	18
$2	28	$f2	10
$3	31	$f3	10
$4	30	$f4	12
$5	32	$f5	11
$6	29	$f6	9
$7	28	$f7	8
$8	26	$f8	8
$9	22	$f9	8
$10	17	$f10	16
$11	21	$f11	16
$12	20	$f12	15
$13	17	$f13	12
$14	13	$f14	12
$15	13	$f15	12
$16	27	$f16	11

$17	25	$f17	13
$18	25	$f18	10
$19	23	$f19	9
$20	22	$f20	13
$21	23	$f21	15
$22	26	$f22	11
$23	25	$f23	11
$24	25	$f24	9
$25	26	$f25	11
$26	19	$f26	10
$27	21	$f27	10
$28	25	$f28	10
$29	13	$f29	10
$30	10	$f30	11
$31	0	$f31	0

4.5. Alpha İGİÇ Benzetimi – Temel Deneyler

Başlangıç için geri yürüme, kopya alma ve sabit İGİÇ kuyruk boyları ile benzetimler yapıldı.



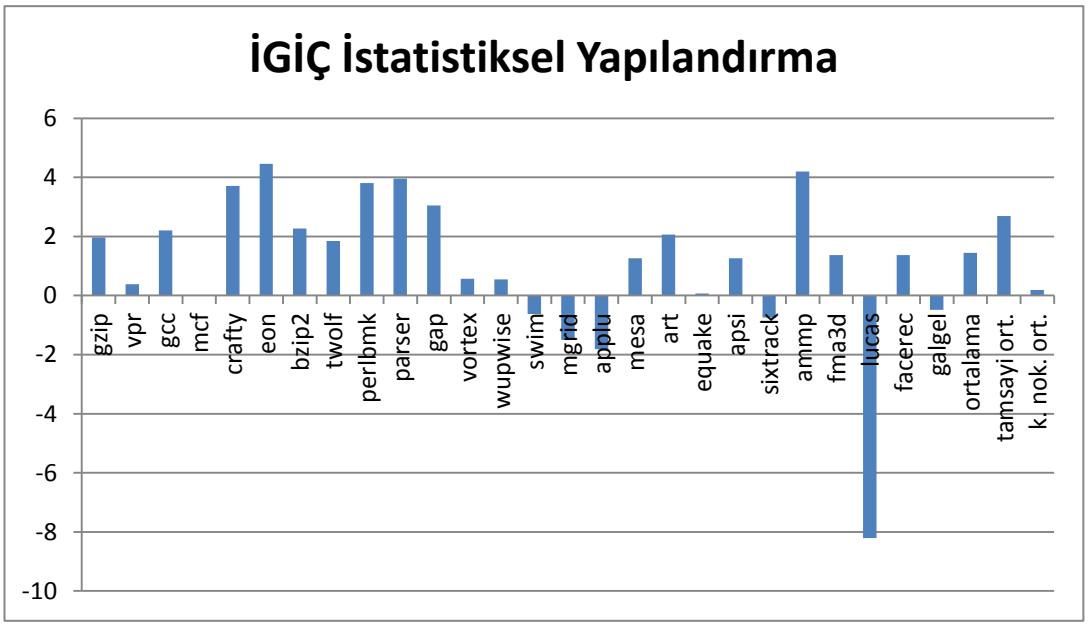
Şekil 4.4. Geri Yürüme İle Başarım Farkı

Şekil 4.4 bize kopya alma tekniğinin çeşitli yapılandırmalarını ve İGİÇ kuyruklarının sabit boylar kullandıkları durumdaki başarımlarını geri yürüme ile karşılaştırarak gösterir. Kopya alma tekniğinde 8 kopya satırı kullanıldığında başarımın genel ortalamalarda düştüğü, kayan nokta denektaşı programlarında ise ortalamada geri yürümeden başarımının daha düşük olduğu görülüyor. Kopya satırı sayısı 16'ya çıktığında bile kayan nokta denektaşı programlarında ortalamada ÇBB'nin geri yürümeden daha düşük olduğu göze çarpıyor. Buna yol açan *art* ve *fma3d* gibi başarımı kopya alma tekniği kullanılınca oldukça düşmüş olan bazı denektaşı programlar. Bu düşüşün nedenini incelemeden önce İGİÇ kuyruk tekniğinin başarımına da bir göz atmakta fayda var. İGİÇ 2 Çevrim olarak işaretli çubuklar tüm İGİÇ kuyruklarının boyunun 16 olarak belirlendiği, devre karmaşıklığıyla sınırlanmadığı için kullanabileceği kopya satırı sayısında pratik bir sınır olmayan ve dallanma tahmini hatası olması durumunda yeniden adlandırma tablosunu 2 saat darbesinde kurtarabilecek İGİÇ kuyruk tekniğinin başarımını gösterir. Genel ortalamada 16 kopya satırı kullanan kopya alma tekniğiyle eşdeğer bir başarım sunmaktadır. Kayan noktada ise kopya alma tekniklerinden daha iyi başarım gösterir. Tek çevrimde kuyruk işaretçilerini düzeltebilecek bir teknik kullanıldığında ise tüm ortalamalarda şimdiye kadar gördüğümüz en iyi başarım değeri alınmış olur. Bu sonuçlar işlemcide bir anda bulunabilecek dallanma sayısının sınırlanmasının başarıma olumsuz etkisi olduğunun bir göstergesidir. Bu sonucu destekleyen başka bir bulgu ise kopya alma tekniğinin sınır kullanılmadan uygulandığında elde edilen sonuçlarıdır. Bu deneyde geri yürümeye göre ortalamada %60'dan daha fazla başarım artışı gözlenmiştir. Bazı denektaşı programlarında bu artış %700'ün üstünde olduğu için aynı grafikte göstermek pratikte mümkün olmamıştır.

4.6. Alpha İGİÇ Benzetimi – Kuyruk Boyları İstatistiksel Ayarlanmış Yapılandırma

Bir sonraki deney aşamasında Tablo 4.6'da belirtilen İGİÇ kuyruk boyları kullanılarak M-Sim çalıştırılmıştır. Bu değerler kullanıldığında tüm kuyruk

boylarının 16 olduğu durumdan başarımın ne kadar farklı olduğunu aşağıdaki tabloda görebiliriz.



Şekil 4.5. İGİÇ Kuyruk Boylarını İstatistiksel Yapılandırma Başarım Değişimi

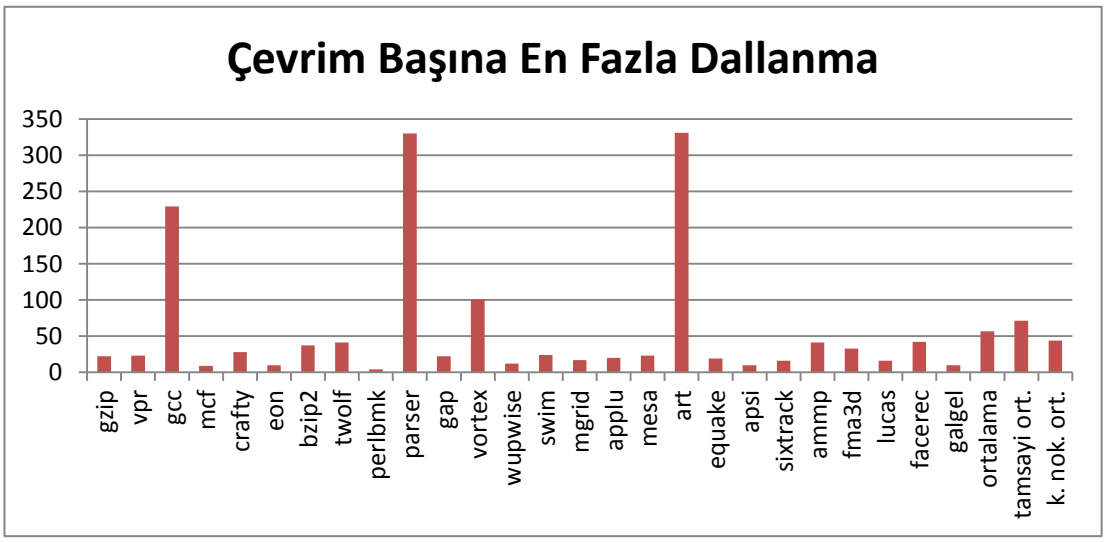
Şekil 4.5'de görüldüğü gibi bu özel değerlerin kullanımı bazı denektaşı programlarında başarım kaybına neden olsa da ortalamada başarımı artırıyor. Ancak bu başarım artışı ile birlikte kullanılan yazmaç sayısı da artmış oluyor. Tüm İGİÇ kuyruklarının boyları 16 iken toplam İGİÇ kuyruk satır sayısı 992 oluyor. Ancak bu değerler kullanıldığında toplam İGİÇ kuyruk satır sayısı ise 1088'e çıkmış oluyor. Başarımda artış sağlamış olmamıza rağmen işlemci üzerinde kullanılan alanda da bir artış oluyor. Alan ve başarımı düzenlemek için uygun İGİÇ kuyruk boyları seçimi ile ilgili çalışma 5'inci bölümde detaylı olarak açıklanmıştır.

4.7. Alpha İGİÇ Benzetimi – Denektaşı Programlarının Dallanma Nitelikleri

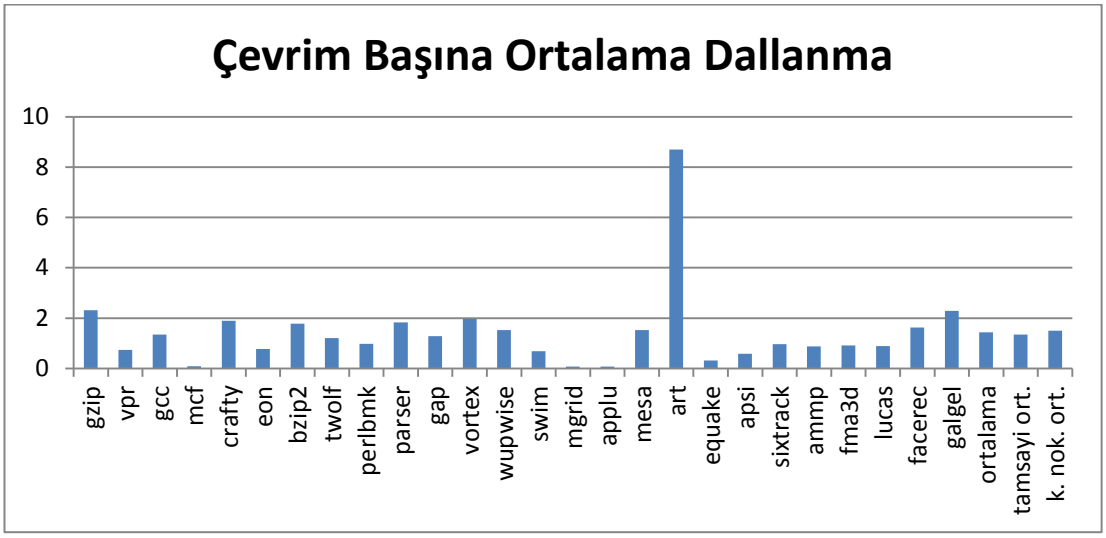
Bu sonuçlardan *art* ve *fma3d* gibi denektaşı programlarındaki başarım düşüşünün dallanma tahmini hatalarından ve arka arkaya işlemciye çok fazla dallanma gelmesinden, bu nedenle işlemcinin kopya alma tabloları boşalana kadar beklemek zorunda kalmasından kaynaklandığı hipotezi ortaya çıkar. Bunu doğrulamak için

işlemci herhangi bir şekilde sınırlanmazsa (YSB ya da fiziksel yazmaç sınırları gibi) "Aynı anda kaç dallanma geliyor?" ve "Ortalamada bir vuruş içerisinde YSB'de kaç dallanma aktif olarak bulunuyor?" sorularının cevapları arandı.

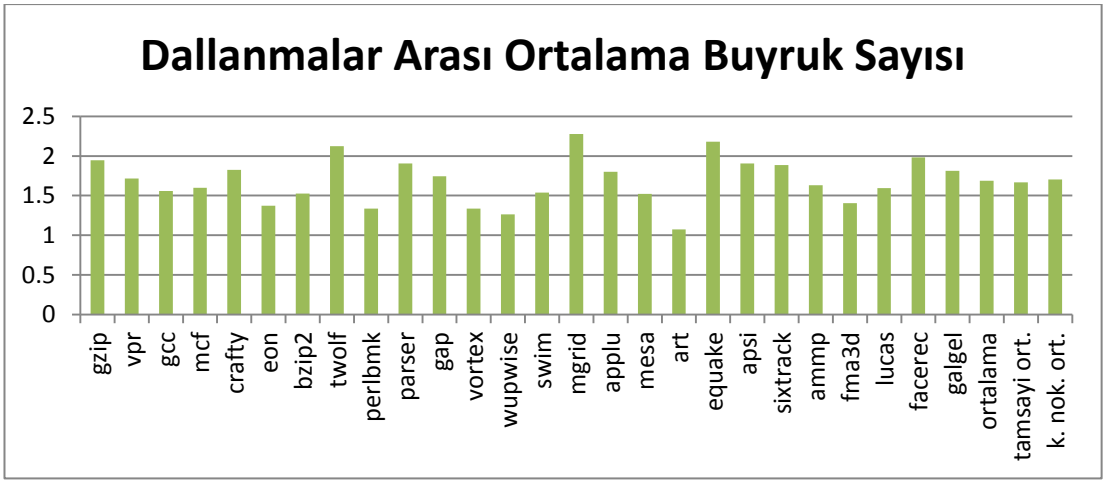
Şekil 4.6, Şekil 4.7 ve Şekil 4.8'de sırası ile denektaşı programlarında yukarıdaki paragrafta bahsedilen koşullarda çevrim başına YSB'de aktif en fazla dallanma, çevrim başına YSB'de ortalama dallanma ve dallanma buyrukları arasındaki ortalama buyruk sayısı bulundu. Başarımı en çok etkilenmiş olan *art* denektaş programı açıkça çok fazla koşullu dallanma buyruğunu çok sık olarak içerdiği için kopya satırlarını dolduruyor. Yeni bir dallanma buyruğu gelirse bu dallanmalar bitirilene kadar işlemciye yeni buyruk alınamıyor – işlenen bu pek çok dallanma düzenli ilerlemiyorsa dallanma tahmin hatası olasılığını da artırıyor. Ancak bu şekilde Şekil 4.6'te fazla dallanma içerdiği görünen tamsayı denektaşı programları bu durumdan etkilenmemiş gözüküyor.



Şekil 4.6. Çevrim Başına En Fazla Dallanma



Şekil 4.7. Çevrim Başına Ortalama Dallanma



Şekil 4.8. Dallanmalar Arası Ortalama Buyruk Sayısı

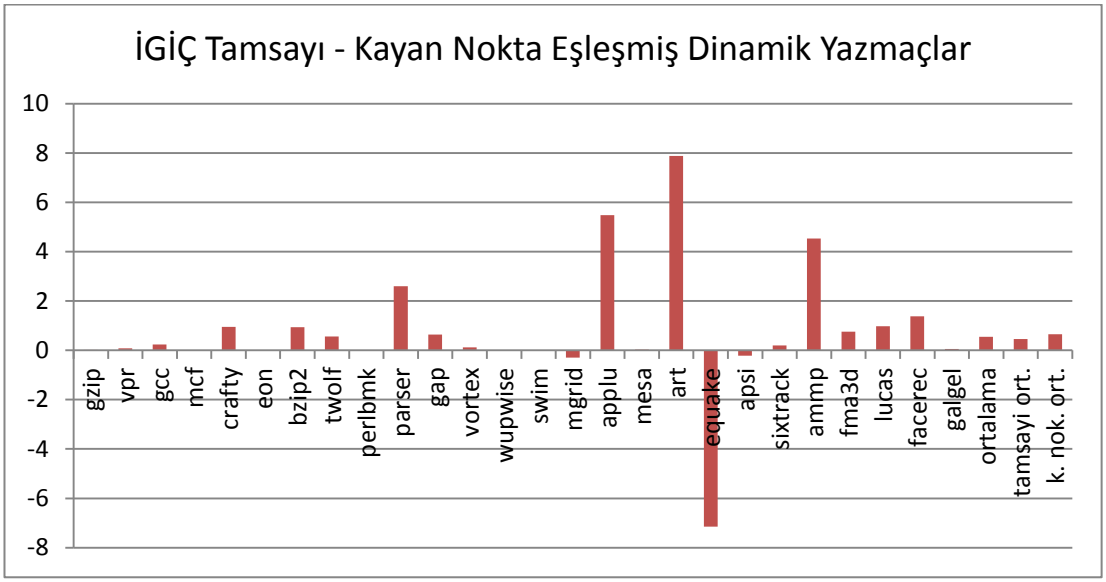
Burada görülen sayılara bakıldığında Şekil 4.6'te kayan nokta denektaşı programlarının YSB'de bulunan en fazla dallanmaları 20 üzerine çıktığında kopya alma ve İGİÇ kuyruğu tekniklerinde başarımda düşüş yaşandığı gözlemlenebiliyor. Çevrim başına ortalama dallanma sayısı ve yapılan işlemlerin uzunluğu gibi diğer faktörler de başarımdaki diğer oynamaları açıklayabiliyor. Başka bir göz önünde bulundurulması gereken nokta da kayan nokta denektaşı programlarının gerçek ÇBB değerleri. Bu değerler doğal olarak düşük olduğu için (1'den küçük) ufak oynamalar çok büyük yüzde değişikliklerine neden olabiliyor. Dallanmalar arası ortalama buyruk sayıları da başka bir resim çizmekte. Grafikten görüldüğü gibi ortalamada hiç

bir denektaşı programı iki dallanma arasında 2.5'tan fazla buyruk işlemiyor. Bu durumda geri yürüme yapan dört yollu bir işlemci dallanma tahmini hatası olduğunda bu buyrukları tek saat darbesinde geri alabilir denilebilir. Geri yürümenin başarımının daha iyi olduğu durumlar tahmin hatalarının dallanma buyrukları zincirinin sonunda çıkması ve geri alınacak buyrukların program boyunca daha az olması ile açıklanabilir. Ancak geri yürüme sırasında bir dallanmanın hata yaptığının anlaşıldığı durumda dallanmalar dahil olmak üzere işlemcide bulunan tüm buyruklar beklemeye yol açar, bu nedenle bu argüman yalnızca özel durumlarda geçerli olur. Bunun yanında, denektaşı programlarında İGİÇ kuyruklarındaki başarım düşüşleri ortalamanın çok üstünde yazmaç kullanımları ile de açıklanabilir. Tüm bunlara rağmen genel resme bakıldığında temel İGİÇ kuyruk tekniğinin kopya alma tekniğinden daha az devre karmaşıklığı ile denk başarım sağladığı görülebilmektedir.

4.8. Alpha İGİÇ Benzetimi – Çift Yazmaç Tekniği

M-Sim ile yapılan diğer bir deney ise 3.4 numaralı bölümde anlatılan, her tam sayı yazmacının bir kayan nokta yazmacı ile eşleştirilerek tek bir kuyruk, iki baş ve kuyruk işaretçisi ve bunların aralarındaki bağlantıyı ayarlayan diğer devreler ile gerçekleştirimini konu alır. Bu deneyde tüm tam sayı yazmaçlarına ait İGİÇ kuyruk boyları 24, tüm kayan nokta yazmaçlarına ait İGİÇ kuyruk boyları ise 8 olarak başlar. Kuyruk boyları 4'ün altına inemez. Bir mimari yazmacın İGİÇ kuyruğu 5 çevrim boyunca yarısından fazlası dolu ise boyunu artırmaya çalışır. İlk deneyde tam sayı yazmaçlar numaraları karşılıklı gelen yazmaçlarla eşleşmiştir ($0 ve $f0, $1 ve $f1 vb.).

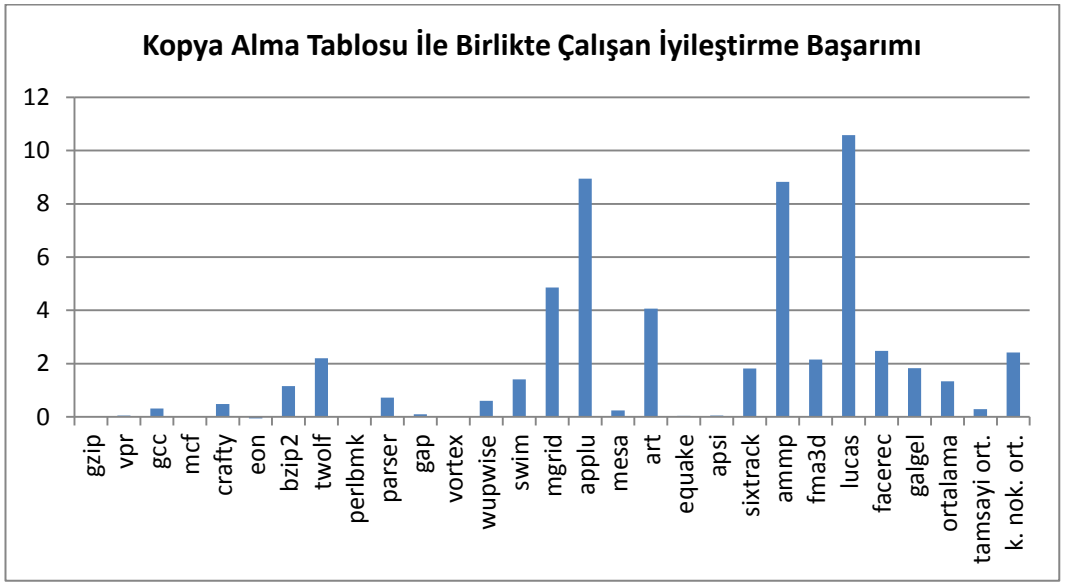
Şekil 4.9'de görüldüğü üzere bu teknik uygulandığında ortalama başarımda çok küçük de olsa bir artış görülmektedir. Burada dikkate alınması gereken başka bir konu da az da olsa başarım artışının tüm İGİÇ kuyruklarının boylarının 16 olduğu durumla aynı sayıda yazmaç kullanarak elde edilmiş olmasıdır.



Şekil 4.9. İGİÇ Tamsayı - Kayan Nokta Eşleşmiş Dinamik Yazmaçlarda Başarım Değişimi

4.9. Alpha İGİÇ Benzetimi – Kopya Alma Durumunda İyileştirme

3.5 numaralı bölümde bahsedilen İGİÇ kuyrukları ile kopya alma tekniğinin birlikte kullanılması ile kullanılabilir hale gelen iyileştirme de denendi. Bu iyileştirme sonucunda ortalamada %1.3'e yakın bir artış görüldü. Bu artışın bu kadar düşük olmasının nedenini belki de en iyi Şekil 2.1 açıklar. Eğer ortalama olarak dallanma buyrukları arasında bu denli az buyruk varsa, yalnızca bir dallanma geldiğinde İGİÇ kuyruklarının kuyruk işaretçilerini (dolaysıyla boyunu) artırmak çok da büyük bir etki yaratmayacaktır. Bu benzetimin sonuçları Şekil 4.10'da görülebilir. Kayan nokta denektaşı programları dallanmalar arasında sık miktarda buyruk olmamasına rağmen başarımda bu yöntemin kullanımıyla artış göstermiştir. Bu teknik ortalamadaki başarım artışı azlığına rağmen genel olarak bakıldığında başarım artışları daha küçük olan ya da başarım azalışı olan kayan nokta denektaşı programlarında başarım kazanmak için uygulanabilir.



Şekil 4.10. Kopya Alma Tablosu İle Birlikte Çalışan İyileştirme Başarımı

5. GENETİK ALGORİTMA KULLANARAK PARAMETRE TESPİTİ

İGİÇ kuyrukları kullanan yeniden adlandırma tekniği için İGİÇ kuyruklarının boyları yapılandırılırken ya sabit değerler seçildi, ya da (4.3.1) denklemi kullanılarak Tablo 4.6'daki değerler elde edildi ve kullanıldı. Bu değerler kabul edilebilir başarım değerleri gösterdi. Ancak belirli bir aralıkta olabilecek tüm değerler denenmeden önce bu kullanılan değerlerin en iyi başarımı sağladığı söylenemez. Ya da benzer şekilde bu değerlerden daha az güç çekebilecek ve benzer başarım gösterebilecek bir İGİÇ kuyruk boyu grubu olup olmadığı deneyleri yapmadan iddia edilemez.

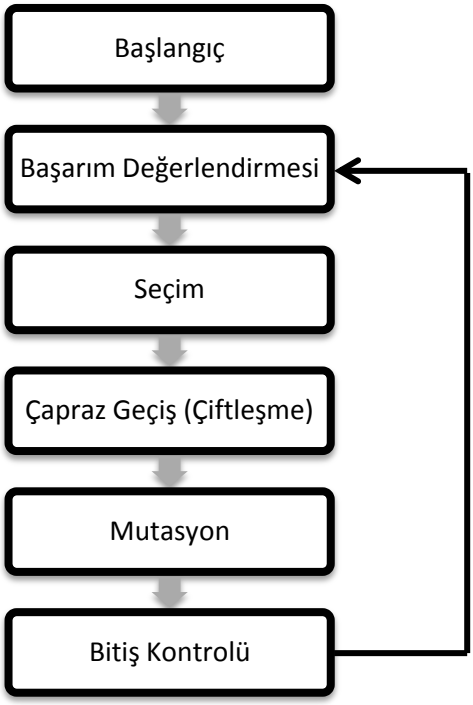
Ancak pratik olarak bakıldığında İGİÇ kuyrukları için mümkün olan her değerin denenmesi mümkün değil. Her denektaş programı elimizdeki donanım yapısı ile yaklaşık 6 saatte sonlanıyor. Bu süreyi kısaltmak için denektaşı programlarının ileri sarma noktalarında işlemcinin tüm yapısını (bellek, önbellekler, yazmaç öbeği ve işlemci durumunu belirten diğer tüm veri yapıları) saklayarak o noktadan devam edilse ve koşan buyruk sayıları çok küçültülse bile tüm bir denektaşı program grubunun çalışması en az 1 saat alıyor. Bunun yanında İGİÇ boylarının 8'den küçük ve 32'den büyük olamayacağına karar verilerek değer arama uzayında da 32 yerine 25 değer bırakılmıştır. Bu şekilde tüm arama uzayının denenmesi (5.1) denklemindeki birim zamanın bir saat olduğu düşünüldüğünde 10^{82} yıl sürecektir.

$$(25^{62}) = 4.70 \times 10^{86}\ birim\ zaman \qquad (5.1)$$

Zamanın sınırlı bir kaynak olduğu da göz önünde bulundurularak bu doğrulamayı yapmak için alternatif bir yöntem arandı. Buradaki problem 62 adet İGİÇ kuyruğunun en uygun boylarının ne olduğunu bulmak. Ancak elimizde başarımının kabul edilebilir olduğunu bildiğimiz bir değer zaten var. Bu değer kullanılarak genetik algoritmalar yardımı ile tüm arama uzayını denemek zorunda kalmadan kullandığımız değerleri doğrulamaya ya da başarımı veya alanı daha iyi olan yeni değerler bulmaya çalışıldı.

5.1. Genetik Algoritmalar

Genetik algoritmalar büyük parametre uzaylarını evrim ve doğal seleksiyondan gelen fikirleri kullanarak arayabilen algoritmalardır. Geniş uygulama alanları bulunur. Önceki bazı literatür çalışmaları ekonomi ve finansal başarısızlıkların tahmini üzerine kuruludur [12] [13] [14]. Genetik algoritmalar özellikle parametre uzayı için karmaşık bir uygunluk fonksiyonu tanımlanması gerekiyorsa başarılı olur. Bu algoritmalar başlangıç, seçim, çapraz geçiş ve mutasyon adımları ile çalışır [15] [16] [17]. Bu algoritmaların basit bir görünümü Şekil 5.1'de görülebilir.



Şekil 5.1. Genetik Algoritma Genel Görünüş

Başlangıç aşamasında çözüm uzayından bir grup rasgele çözüm seçilir. Bazı önceden belirlenmiş çözümler rasgele çözümlerin arasındaki başarımlarına bakmak adına bu aşamada elle yerleştirilebilir. Seçim adımında rasgele üretilen çözümlerden kullanıcı tarafından belirlenen bir uygunluk fonksiyonuna göre en iyiler çiftleşme için seçilir. Uygunluk fonksiyonunun tasarımı ve çiftleşme adaylarının seçimi algoritma tasarımcısına bırakılır ve algoritma sonucu üzerinde büyük etkisi vardır. Çapraz geçiş

aşamasında iki çözüm arasında çiftleşme gerçekleşir. Bu çözümlerin belirli kısımlarını değişmesi ve yeni çözümler oluşturması ile gerçekleşir. Bu aşama çeşitli şekillerde gerçekleştirilebilir. Mutasyon aşaması ise çözümlerin bayatlamasını önlemek adına rasgele bir çözümün rasgele bir değerini değiştirir. Bizim algoritmamıza mutasyonlar belirli olasılık sınırları içerisinde yalnızca çocuklara uygulanmaktadır.

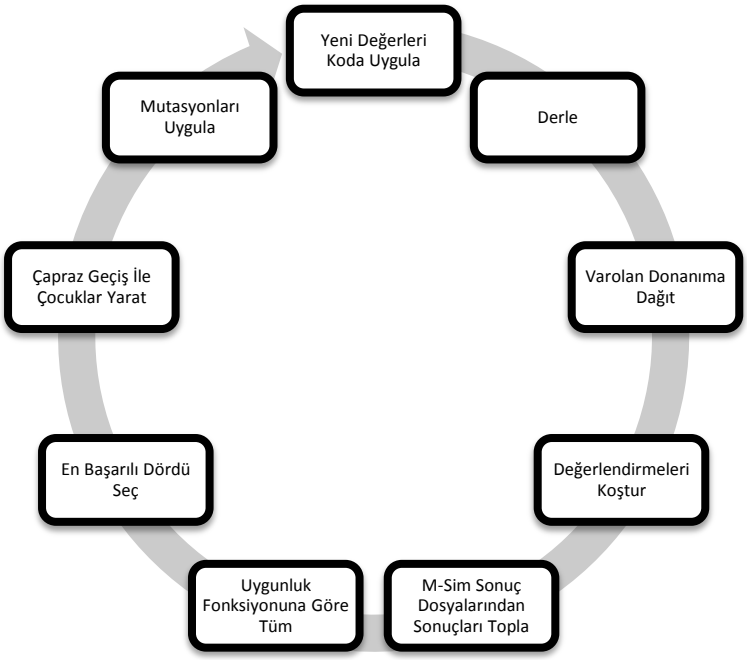
5.2. Kullanılan Algoritma ve Hesaplamalar

Sonuçların hesaplanması için genetik algoritmaların bir varyasyonu kullanıldı. Algoritmanın sözde kod halinde yazılmış, basitleştirilmiş bir hali Tablo 5.1'da görülebilir.

Tablo 5.1. Genetik Algoritma Sözde Kod

1. Her elemanı 8 ve 32 arasında olan 14 tane rasgele değer kümesi yarat.
2. Önceden hesaplanmış çözümü ekle.
3. Her küme için tüm denektaşı programlarını çalıştır.
4. Test kümelerini aşağıdaki gibi hesaplanmış uygunluklarına göre sırala
    a. İki test kümesi arasındaki yüzde ÇBB farkını hesapla.
    b. Test kümelerinin toplam boylarının yüzde farkını hesapla.
    c. Denklem (5.2.1)'i kullanarak uygunluğu hesapla.
    d. Başarım değerlendirmesi sıfırdan büyükse, ilk test kümesi diğerinden büyüktür. Aksi durumda, ikinci test kümesi birinciden büyüktür.
5. En başarılı dört test kümesini çiftleşme için seç.
6. En başarılı test kümesini takip eden üç test kümesi ile aşağıdaki algoritmayı kullanarak çiftleştir:
    a. Rasgele bir bölme noktası yarat.
    b. İlk çocuk bölme noktasına kadar ilk test kümesinin değerlerini, bölme noktasından sonra ikinci test kümesinin değerlerini alır.
    c. İkinci çocuk bölme noktasına kadar ikinci test kümesinin değerlerini, bölme noktasından sonra birinci test kümesinin değerlerini alır.
    d. Oluşan çocuklardaki değeleri her biri için %10 şansla 8 ve 32 arasında rasgele bir sayıya değiştir (mutasyon).
7. Oluşan 6 çocuğu değerlendir ve diğer test kümeleriyle birlikte sırala.
8. Çocuklardan hiç biri en üst dörde giremediyse algoritma sonlanır. Eğer algoritma önceden belirlenmiş bir sayıda koştuysa algoritma sonlanır.
9. 5. Adıma git.

Bu algoritmayı uygulayan parametre test uygulaması Ruby programlama dilinde yazılmıştır. Programın görsel bir gösterimi de aşağıda gösterilmiştir. Program buradaki çıkış koşullarından herhangi biri sağlanana kadar çalışmaya devam eder.



Şekil 5.2. Kullanılan Genetik Algoritmanın Görsel Gösterimi

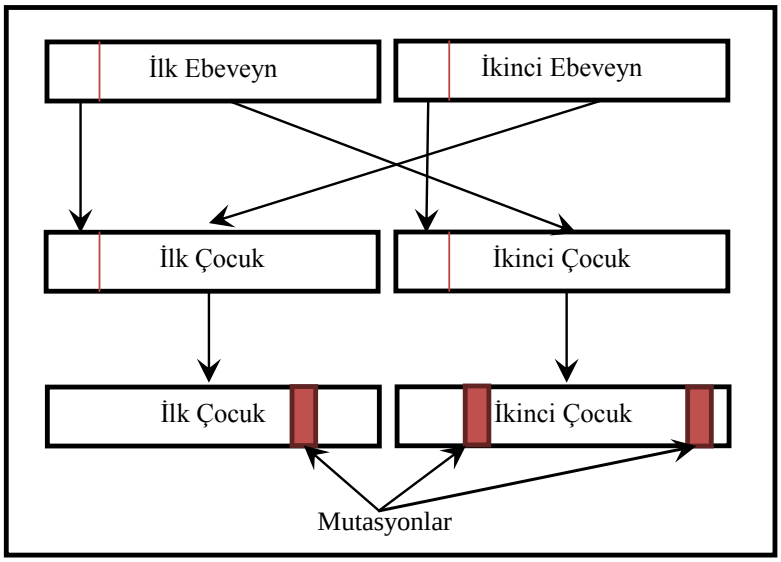
$$uygunluk = \left(4 \times \text{ÇBB}_{\text{yüzde}}\right) - Boy_{toplam_{\text{yüzde}}} \qquad (5.2.1)$$

Başlangıç için program değerleri 8 ve 32 arasında rasgele seçilmiş olan 14 adet değer kümesi yaratır. Burada 8 ve 32 deneysel olarak seçilmiştir: 8'den küçük İGİÇ kuyruk boyları başarımı olumsuz etkilemekte ve 32'den büyük İGİÇ kuyruk boyları da kayda değer bir artış sağlamamaktadır. Daha sonra (4.4.1) denklemi ile belirlenmiş sabit değer kümesi de bu 14 değer kümesine eklenir – program başladığında toplam değer kümesi sayımız 15 olur.

Parametre test uygulaması M-Sim kodu içerisindeki uygun yeri her küme için uygun test değerleri ile değiştirir. Kodu değiştirdikten sonra uygulama M-Sim'i yeniden derler ve denektaşı programlarını benzetim için uygun olan bilgisayarlara dağıtır. Denektaşı programları hızlı ölçüm yapmak adına yalnızca 1 milyon buyruk çalıştırır.

Ayrıca M-Sim ileri alınmış bir kontrol noktasından başlayarak her denemede tekrar ileri alma gereğini ortadan kaldırır.

Her test kümesinin ÇBB'si (Çevrim Başına Buyruk) ve boyu hesaplanır. Her denektaşı programı test kümesi değerleri ile çalıştırılır. Test kümesinin ÇBB değeri tüm denektaşı programlarının ÇBB'lerinin ortalaması alınarak hesaplanır. Genel amaçlı bir işlemci tasarladığımız için burada ortalama alıyoruz. Eğer tam sayıya ya da kayan noktaya yönelik bir işlemci tasarlayacak olsaydık ona göre ağırlık verilebilirdi. Boyları ise M-Sim'de kullandığımız İGİÇ kuyruk boyu değerleri toplanarak elde edilir. Veri kümeleri iki temel kritere göre sıralanır: ÇBB (büyük daha iyi) ve toplam kuyruk boyu (küçük daha iyi). Bu sıralamada ÇBB kuyruk boyuna göre dört kat daha fazla önemli olarak alınır. Bunun nedeni başarımın alandan daha değerli olarak görülmesidir. Arzu edilirse bu sabitler eldeki deneye uyacak şekilde değiştirilebilir. Bu sıralama ile en üst dört veri kümesi belirlenir.



Şekil 5.3. Kullanılan Genetik Algoritmanın Çiftleşme Aşaması

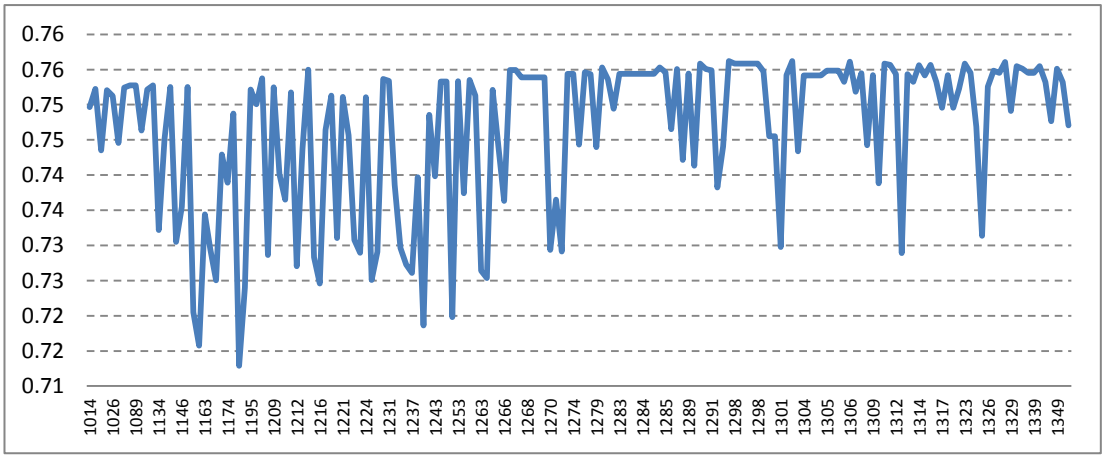
Başlangıçtaki 15 kümeden seçilen 4 küme şu şekilde çiftleştirilir: En başarılı veri kümesi diğer üç veri kümesiyle çiftleştirilerek toplam 6 çocuk elde edilir. Çocuklar her küme için rasgele bir birleşme noktası seçilerek Şekil 5.3'de görüldüğü gibi çapraz geçiş ile yaratılır. Bir çiftin ilk çocuğu birleşme noktasına kadar ilk

ebeveynin, daha sonra diğer ebeveynin değerlerini içerir. Diğer çocuk bunun tersidir. Çapraz geçişten sonra çocukların her bir değeri için 0 ve 1 arası rasgele bir sayı yaratılır. Eğer bu sayı 0.9'dan daha büyükse o değer 8 ve 32 arası rasgele bir değere değiştirilerek mutasyon sağlanmış olur.

Buradan oluşan çocuklar diğer veri kümeleri gibi değerlendirmeye alınır. Eğer herhangi biri en üst dörde girebilirse çiftleştirme tekrarlanır ve sonuçlara yeniden bakılır. Eğer hiç biri en üst dörde giremezse algoritmayı bitirir ve yeni rasgele değerlerle baştan başlarız. Belli bir sayıda çiftleşme tamamlandığı zaman algoritma daha iyi bir çocuk olup olmasına bakmaksızın biter. Bizim amaçlarımız için algoritmayı 5 defa çalıştırdık ancak algoritmanın kendisi çok defa çalıştı.

5.3. Sonuçlar

Şekil 5.4'de sonuçların bir grafiği görülebilir. Burada gözüken ÇBB değerleri Spec 2000 denektaşı programlarının tümünün ÇBB'lerinin ortalamasıdır. Algoritma her defasında farklı test kümeleri ile yeniden çalıştırılmıştır. Her çalışma sırasında algoritma (4.4.1) denklemi ile önceden belirlenmiş değerlerden daha iyi başarım gösteren bir değer kümesi bulmayı başardı. Kuyrukların işlemcide kaplayacakları alana bakmaktan vazgeçtiğimiz anda ise başarımı artan pek çok değer görüyoruz.



Şekil 5.4. Farklı Yapılandırmalarla Elde Edilen ÇBB Değerleri

Aşağıdaki tablo ise bazı test sonuçlarını gösteriyor. Bu tabloyu elde etmek için kullanılan etkinlik ölçüsü (5.3.1) denklemi ile tanımlanabilir.

$$Etkinlik\ Ölçüsü = \frac{ÇBB}{Yapıda\ Kullanılan\ Satır\ Sayısı} \qquad (5.3.1)$$

Bu etkinlik ölçüsü yatırılan birim donanım miktarı için kazanılan ÇBB'yi gösterir. İdeal sonucumuz bu liste içerisinde tepeye yakın sonuçlardan biri olmalıdır, çünkü işlemci içerisinde en az yer kaplayan ve başarımı da buna göre en fazla olan bir veri kümesi arıyoruz. Önceden belirlenmiş değerlerimiz (Tabloda satır 8) ile bulunan en iyi sonuç arasındaki başarım farkı yalnızca %0.46'dır. En iyi sonucun bizim 1088 yazmacımıza karşılık 1298 yazmacı olduğunu göz önüne alınca bu başarım artışı alanda %18 artış için çok küçük kalıyor. Bu sonuçlara bakıldığında mühendislik açısından daha mantıklı bir sonuç tablodaki 2. Satır olacaktır. Bu değer ÇBB'yi yalnızca %0.06 azaltır, ancak alan kullanımını da aynı anda %6.6 azaltmış olur. İşlemcinin içindeki yapıların enerji dağılımı ve gecikmesi içerisindeki satırlara bağlı olduğundan satır sayısını az tutmak saat frekansının artırılmasında soğutma maliyetinin azaltılmasında büyük rol oynar. Bu nedenle hemen hemen aynı başarım verildiğinde yapıdan 69 satır çıkarmak genetik algoritma çözümümüz için başarı kabul edilebilir: Aynı başarım daha az maliyet, daha az enerji dağılımı ve daha az erişim gecikmesi ile sağlanmıştır.

Tablo 5.2. Seçilen En İyi 25 Sonuç

Sıra	ÇBB	Toplam Kuyruk Satır Sayısı	Etkinlik Ölçüsü
1	0.749654	1014	0.0007393
2	0.752262	1019	0.0007382
3	0.752054	1024	0.0007344
4	0.751238	1026	0.0007322
5	0.743508	1023	0.0007268
6	0.744554	1027	0.0007250
7	0.752454	1043	0.0007214
8	**0.752746**	**1088**	**0.0006919**
9	0.752746	1089	0.0006912
10	0.752169	1092	0.0006888
11	0.752746	1097	0.0006862
12	0.746331	1091	0.0006841
13	0.752538	1141	0.0006595
14	0.752538	1150	0.0006544

15	0.745154	1140	0.0006536
16	0.732185	1134	0.0006457
17	0.735469	1146	0.0006418
18	0.730508	1141	0.0006402
19	0.742923	1166	0.0006372
20	0.734400	1163	0.0006315
21	0.752138	1195	0.0006294
22	0.738908	1174	0.0006294
23	0.753762	1199	0.0006287
24	0.748723	1191	0.0006287
25	0.729462	1165	0.0006261

Bu genetik algoritma benzetim zamanımızdan büyük miktarda tasarruf yapmamıza yardımcı oldu. 25 farklı değer alabilecek 62 parametre için uygulanacak kaba kuvvet bir algoritma denklem (5.1)'de de gösterildiği gibi tamamlanması mümkün olmayan bir zaman dilimi sunar. Her veri kümesinin denenmesi 1 saat sürüyor kabul edilirse bu (5.38×10^{82} yıl)'a denk geliyor. Bizim algoritmamızın kullanılması ise kaba kuvvet kullanan algoritmanın aksine yalnızca $5 \times (15 + 5 \times 6) = 225$ birim zaman tutuyor. Bu da 225 saate karşılık gelir. Çok kısa olmamasına rağmen diğer çözüme göre oldukça etkin. Burada kaba kuvvet uygulamasının en iyi sonucu bulacağının kesin olduğuna dikkat edilmeli – ancak bunu mantıklı bir sürede yapması mümkün değil. Burada kullanılan genetik algoritmayı çalıştırılarak en uygun çözüme yakınsamak bu durum için daha uygun bir çözüm.

6. SONUÇLAR

Bu tez genişletilebilir bir yazmaç yeniden adlandırma yapısı ve bu yapı üzerinde yapılabilecek çeşitli değişiklikler ve geliştirmeler önermiştir. Bu yapının parametrelerinin belirlenmesi için ayrıca genetik algoritmalar kullanılarak bir en iyiye yakınsama çalışması yapılmıştır.

6.1. İGİÇ Yazmaç Yeniden Adlandırma Sonuçları

İGİÇ yazmaç yeniden adlandırma tekniği kopya alma tekniğinin aksine devre karmaşıklığı az olan ve gerçekleştirilmesi daha kolay olan bir sistemdir. Bu tezde uygun şartlar altında kopya alma yapısından dahi daha iyi başarım gösterebileceği belirtilmiştir. Bunun yanında diğer belirgin teknik olan geri yürüme tekniğine karşı da denemeler yapıldı ve ona karşı da başarım artışı sağlandı.

İlk önerilen İGİÇ kuyruk yapısına ek olarak yazmaçları çifter çifter gruplayarak yapılan denemelerde başarım artışı görülmemesine rağmen devre karmaşıklığının artması karşılığında kullanılan İGİÇ kuyruk satır sayısı yarıya indirildi. Bunun yanında İGİÇ kuyruk yapısının kopya alma tekniğiyle kullanılması durumunda bazı değerlerin İGİÇ kuyruğuna yazılmasının gerekli olmadığı gösterildi ve bunun getirdiği başarım artışı gözlendi.

İGİÇ kuyrukları kullanan yeniden adlandırma tekniğinin bu tezde yayınlanan sonuçlar göz önüne alınarak gerçekleştirilmesinin başarımı artırmak adına olumlu bir adım olduğu söylenebilir. Bu gerçekleştirme için devre yapıları da çizilip bu yapının diğer tekniklere oranla ne kadar alan kapladığı ve ne kadar güç tükettiği de araştırılması gereken konular arasındadır.

6.2. Genetik Algoritma ile Parametre Arama Sonuçları

İGİÇ kuyruklarının boylarının belirlenmesi için ortaya atılan istatistiksel sonucu doğrulamak ve çok uzun süre harcamadan yeni kuyruk boyları bulabilmek için

genetik algoritmalardan faydalanıldı. Kullanılan genetik algoritma tutarlı olarak istatistiksel olarak bulunan sonuçtan daha başarılı veri kümeleri üretmeyi başardı. Buna rağmen deney sonunda yapılan sıralamada istatistiksel olarak bulunan sonuç üst sıralarda kaldı – bu da kullanılan formülü doğrulamış oluyor.

Önerilen genetik algoritma yapısı yalnızca mimari araştırmada değil, çok geniş parametre uzaylarının aranması gereken her durumda kullanılabilir. Bu tezde önerilen algoritmanın geniş parametre uzaylarında en iyi sonuca yakınsadığı ve benzetim zamanında büyük kazançlar sağladığı gösterilmiştir.

KAYNAKLAR

[1] Johnson, W., Super-Scalar Processor Design, *Technical Report CSL-TR-89-383*, Stanford University, 1989.

[2] McFarling, S., Combining Branch Predictors, *Technical Note WRL-TN-36*, Western Research Labratory.

[3] Akkary, H., R., R., Srinivasan, S., Checkpoint Processing and Recovery: Towards Scalable Large Instruction Window Processors, IEEE/ACM International Symposium on Microarchitecture (MICRO), 2003.

[4] Aasarai, K., Moshovos, A., Toward a Viable, Out-of-Order Soft Core: Copy-Free, Checkpointed Register Renaming, IEEE International Conference on Field Programmable Logic, August 2009.

[5] Yourst, M., PTLSim: A cycle accurate full system x86-64 microarchitectural simulator, IEEE Symposium on Performance Analysis of Systems and Software (ISPASS), 2007.

[6] Sharkey, J., Ponomarev, D., Ghose, K., M-Sim: A Flexible, Multithreaded Architectural Simulation Environment, *Technical Report CS-TR-05-DP01*, Department of Computer Science, State University of New York, Binghamton, 2005.

[7] Hinton, G.., The Microarchitecture of the Pentium 4 Processor, Intel Technology Journal, 5(1), February 2001.

[8] Burger, D., Austin, T., The SimpleScalar tool set: Version 2.0, *Technical Report*, Department of CS, University of Wisconsin-Madison, June 1997.

[9] Kessler, R.E., The Alpha 21264 Microprocessor, IEEE Micro, vol. 19, 24-36, March 1999.

[10] Henning, J., SPEC CPU2000: Measuring CPU Performance in the New Millenium, 28-35, 2000.

[11] Hamerly, G., Perelman, E., Lau, J., Calder, B., SimPoint 3.0: Faster and More Flexible Program Analysis, Workshop on Modeling, Benchmarking and Simulation, June 2005.

[12] Shin, K., Han, I., Case-based reasoning supported by genetic algorithms for corporate bond rating, Expert Systems with Applications, 16, 85-95, 1999.

[13] Min, S., Lee, J., Han, I., Hybrid genetic algorithms and support vector machines for bankruptcy prediction, Expert Systems with Applications, 31(3), 652-660, 2005.

[14] Back, B., Laitinen, T., Sere, K., Neural networks and genetic algorithms for bankruptcy predictions, Expert Systems with Applications, 11(4), 407-413, 1996.

[15] Davis, L., Handbook of Genetic Algorithms, *Van Nostrand Reinhold*, New York, 1991.

[16] Goldberg, D., Genetic Algorithms in Search, Optimization and Machine Learning, *Addison-Wesley*, Reading, MA, 1989.

[17] Holland, J., Adaptation in natural and artificial systems, *The University of*

Michigan Press, Ann Arbor, 1975.

[18] Hwu, W., Checkpoint repair for out-of order execution machines, 14th International Symposium on Computer Architecture (ISCA), 18-26, June 1987.

[19] Sima, D., The Design Space of Register Renaming Techniques, IEEE Micro, vol. 20, 70-83, September/October 200.

[20] Yeager, K., The MIPS R10000 Superscalar Microprocessor, IEEE Micro, vol. 16, 28-40, April 1996.

[21] G., A., M., K., Ergin, O., Complexity-Effective Rename Table Design For Rapid Speculation Recovery, Lecture Notes in Computer Science, vol. 5974, 15-24, 2010.

[22] Shin, K., Lee, Y., A genetic algorithm application in bankruptcy prediction modeling, Expert Systems with Applications, 23(3), 321-328, 2002.

[23] Zhou, P., Onder, S., Carr, S., Fast branch misprediction recovery in out-of-order superscalar processors, roceedings of the 19th annual international conference on Supercomputing, 41-50, New York, NY, USA, 2005.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, Adı	: AŞILIOĞLU, Görkem
Uyruğu	: T.C.
Doğum Tarihi ve Yeri	: 02.05.1988
Medeni Hali	: Bekâr
Telefon	: 0 (536) 399 7282
Faks	: 0 (312) 292 4290
E-Posta	: gasilioglu@etu.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Y. Lisans	TOBB ETÜ Bilgisayar Mühendisliği	2011 (beklenen)
Lisans	TOBB ETÜ Bilgisayar Mühendisliği	2009

İş Deneyimi

Yıl	Yer	Görev
2009 – 2011	TOBB ETÜ	Ders Asistanlığı
2009 – 2009	Kasırga Bilişim Elektronik Ltd.	Yazılım Programlama
2008 – 2008	TOBB ETÜ	Yazılım Programlama
2007 – 2007	KaleData	Bilgi İşlem

Yabancı Dil

İngilizce (ileri seviye)

Yayınlar

1. Konferanslar

1. G. Aşılıoğlu, E. M. Kaya, O. Ergin. Complexity-Effective Rename Table Design for Rapid Speculation Recovery, ARCS 2010, pp. 15-24, 2010 (Lecture Notes in Computer Science volume 5974, Springer)