MİKROİŞLEMCİLERDE YAZMAÇ ÖBEĞİNİN ENERJİ TASARRUFU İÇİN BÖLÜMLENMESİ

MELTEM ÖZSOY

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

TEMMUZ 2009

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Yücel ERCAN

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan Doğdu

Anabilim Dalı Başkanı

Meltem ÖZSOY tarafından hazırlanan MİKROİŞLEMCİLERDE YAZMAÇ ÖBEĞİNİN ENERJİ TASARRUFU İÇİN BÖLÜMLENMESİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Yrd. Doç. Dr. Oğuz ERGİN

Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Yrd. Doç. Dr. Hamza KURT _____

Üye : Yrd. Doç. Dr. Bülent TAVLI _____

Üye : Yrd. Doç. Dr. Oğuz Ergin _____

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.

Meltem ÖZSOY

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 2009**

Meltem ÖZSOY

MİKROİŞLEMCİLERDE YAZMAÇ ÖBEĞİNİN ENERJİ TASARRUFU İÇİN BÖLÜMLENMESİ

ÖZET

Bu tezde günümüz işlemcilerinde önemli bir yere sahip olan elemanlardan yazmaç öbeğinde enerji tasarrufu sağlayacak iyileştirmeler üzerinde durulmuştur. Yazmaç öbeği işlemci içindeki en yakın ve erişilmesi en kolay bellek görevini görür. Bu nedenle hızlı erişimi sağlanacak şekilde normal belleklere göre çok daha fazla enerji tüketimine sebep olacak şekilde üretilmiştir. Bu çalışmada da yazmaç öbeğinin harcadığı enerjinin dar değerler kullanılarak azaltılması sağlanmıştır. Dar değerler veriyolu genişliğinden daha az sayıda bitle gösterilebilen değerlerdir. Daha az sayıda bit ile gösterilen değerler hem durağan hem de devingen olarak enerjiyi daha verimli kullanırlar. Çalışma sonunda tasarlanan yazmaç öbeğinde, veriler kendi genişliklerine göre farklı yazmaç öbeği bloklarına yazılırlar. Bu farklı yazmaç öbeği bloklarının boyutları da işlemcinin başarımının düşmemesi için devingen olarak değiştirilebilir şekilde tasarlanmıştır. Çalıştırılan programın ihtiyaçlarına göre darlık gruplarının büyüklüğünü devingen olarak artırıp azaltan bu yöntem ile enerjiden yaklaşık %65 verim elde edilirken başarımdan neredeyse hiç kayıp olmamıştır.

Anahtar Kelimeler: Mikroişlemciler, Dar Değerler, Yazmaç Öbeği, Enerji Verimi

University : **TOBB University of Economics and Technology**
Institute : **Institute of Natural and Applied Sciences**
Science Programme : **Computer Engineering**
Supervisor : **Associate Professor Dr. Oğuz ERGİN**
Degree Awarded and Date : **M.Sc. – July 2009**

Meltem ÖZSOY

REGISTER FILE PARTITIONING FOR ENERGY EFFICIENCY IN MICROPROCESSORS

ABSTRACT

This thesis is basically about energy efficient register file techniques. Register file is one of the important parts inside microprocessors. It is the smallest memory used by all the instructions for at least one time. That is the reason why they are constructed with energy consuming fast materials. Reducing energy dissipation of register file by using narrow values is the subject of this work. Narrow values method is expressing data with fewer bits than actual data width of the processor. By using narrow values, register file stores fewer bits and needs less static and dynamic energy. By the end of this work, designed register file stores data in narrow value groups and values are written to those groups according to their widths. Size of narrow value groups can be set dynamically with the needs of the program for having the same performance as normal processor. The register file which has dynamically changing narrow value groups saved 65% of energy with negligible performance loss.

Keywords: Microprocessors, Narrow Values, Register File, Energy Efficiency

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Yrd. Doç. Dr. Oğuz ERGİN'e yine kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine, bana yardımda sınır tanımayan arkadaşlarım Mehmet Kayaalp ve Yusuf Onur Koçberber'e, yüksek lisansım boyunca adeta benim ailem olup her öğle yemeğini benimle paylaşan yüksek lisans arkadaşlarıma, bana her zaman güvenen ve yanımda olan aileme ve yüksek lisans çalışmalarım boyunca bursumun karşılanmasını sağlayan TUBITAK' a teşekkürü bir borç bilirim.

İÇİNDEKİLER

ÇİZELGELERİN LİSTESİ

ŞEKİLLERİN LİSTESİ

KISALTMALAR

Kısaltmalar	Açıklama
SRAM	Static Random Access Memory
CISC	Complex Instruction Set Computer
RISC	Reduced Instruction Set Computer
BDP	Buyruk seviyesinde paralellik
YSO	Yazdıktan Sonra Okuma
YSY	Yazdıktan Sonra Yazma
OSY	Okuduktan Sonra Yazma
YSB	Yeniden Sıralama Belleği
SPEC	Standart Performance Evaluation Corporation
ILP	Integer Linear Programming

BÖLÜM 1

1. GİRİŞ

İşlemci, bilgisayarla birlikte insanların işlem yapma ihtiyacıyla doğmuş ve günümüze kadar büyük bir gelişim göstermiştir. Bu gelişimin temel taşları transistörün bulunması, tümleşik devreler ve devre teknolojisinde silikon kullanımı olarak sayılabilir. İlk bilgisayarların kişisel kullanım için olamayacağı açıktı ve beklentiler düşüktü ancak bu temel ilerlemeler gerçekleştikçe bilgisayarlar daha çok kişisel kullanıma yöneldi ve hayatın her alanında yaygınlaştı. Büyük boyutlu olan bilgisayarların devreleri de büyüktü ve güç tüketimleri fazlaydı ancak silikon teknolojisi çok daha küçük güçlerde devrelerin sürülebileceğini gösterdi ve beklentilerin daha da artmasına sebep oldu. Gelişen teknolojiyle birlikte daha hızlı işlem yapacak birimler tasarlandı ve işlemci mimarileri de değişip karmaşıklaştı. Bu yenilikler hız getirdiği gibi daha hızlı bilgisayarlar daha çok güç tüketimine sebep oldu ve güç tüketimi hız ve başarımın önünde aşılamaz bir duvar olarak ortaya çıktı. Bu tezin amacı ise mevcut başarımı etkilemeden işlemci üzerinde enerji tüketimini azaltacak bir düzenleme olarak yazmaç öbeğinde iyileştirmeler yapmaktır.

1.1. İşlemcilerde Enerji Tüketimi

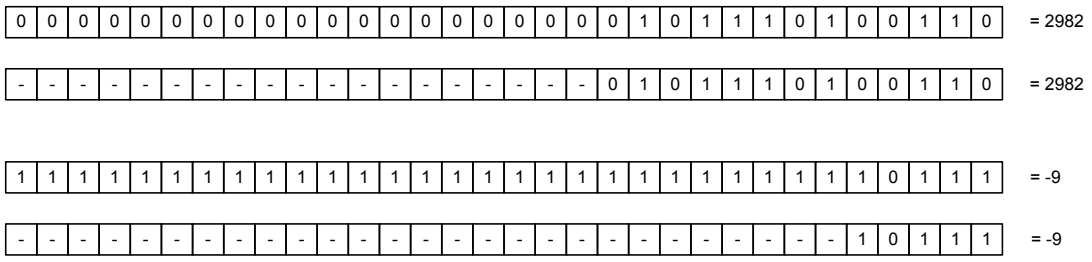
İşlemcilerin tasarımı sırasında tasarımcının pek çok kısıtı vardır. Bunlardan en önemlisi alan kısıtıdır. İşlemcinin çok küçük bir alana sığdırılması gerekir. Mevcut teknoloji ile nanometre boyutlarında çalışmak mümkündür ancak boyut bu kadar küçükken yalnızca devre elemanlarının sığdırılması değil başka sorunlar da ortaya çıkmaktadır. Bunlardan bir tanesi tüketilen enerji ile birlikte ortaya çıkan sıcaklık ve bu sıcaklığın işlemciden uzaklaştırılması problemidir. Ayrıca oluşacak olan ısı işlemlerin hatalı olması ve işlemcinin daha kolay bozulması sorunlarını da beraberinde getirir. Sıcaklık probleminin altında yatan ise aslında enerji tüketimidir. Tüketilen enerjinin ortaya çıkaracağı ısı hatalara ve tasarım problemlerine sebep olur. İşlemcinin kullanabileceği enerji sınırlıdır ve iç içe yan yana bu kadar çok devrenin

sadece bulunuyor olması bile enerji harcanmasına neden olur. Bu nedenle işlemci içinde enerjinin tasarruflu kullanılması gereklidir.

İşlemcide toplam enerjinin yaklaşık %50 si arka arkaya dizilmiş kuyruk ve dizi gibi yapılar tarafından harcanır [1]. Bunlar ilerleyen bölümlerde bahsedilecek olan yükleme saklama kuyruğu, yayın kuyruğu, yeniden sıralama belleği ve yazmaç öbeğidir. Bu yapılarda yapılacak olan iyileştirmeler işlemcinin tümünün enerji tüketiminde önemli bir gelişme sağlayacaktır. Bir matris şeklinde arka arkaya ve yan yana bağlanmış kapılardan oluşan yazmaç öbeği de işlemcinin tümüne oranla yaklaşık %20 enerji tüketir [17]. Yazmaç öbeğinde tüketilen enerji yazmaç öbeğinin satırlarına okuma ya da yazma amacı ile erişildiği anlarda olur. Ancak işlemci bir değer yazıp okumadığı halde yazmaç öbeği durağan haldeyken de enerji tüketir, buna *durağan enerji tüketimi* denir. Yazmaç öbeğini oluşturan SRAM bit hücreleri bitleri saklamak için değerlerini tazeleyen bir döngü içerisinde bulunurlar. Bu döngüde bulunan transistörler sızıntı akımı geçirerek hiçbir iş yapılmadığı anlarda da enerji tüketimine sebep olurlar. Diğer dizi yapıları da aynı şekilde durağan halde de enerji tüketirler. İşlemcinin tümüne bakıldığında özellikle enerji tüketiminin tek bir birimde yoğunlaşmadığı tüm birimler arasında dağıldığı görülebilir [18].

1.2. Dar Değerler

İşlemcilerde tüm işlemler ikilik düzende yürütülmektedir ve tasarım tamamen bu esas üzerine kurulmuştur. Eksi sayıların gösterilmesi için ikiye tümleyen gösterimi kullanılmaktadır. İkiye tümleyen gösteriminde işareti bildiren bitin de sayı değeri vardır ve bir bitin gereksiz yere boşa harcanması veya hem eksi hem de artı sıfır olması gibi sorunlar önlenmiştir. İkiye tümleyende eksi sayılar 1 ile pozitif sayılar ise 0 ile başlayarak gösterilir. Sayının değerine gelene kadar solundan sağına doğru işareti bildiren 0 ya da 1 ler kopyalanır. Örneğin 32 bitlik bir veriyolunda kullanılan tüm değerlerin anlamlı bitleri 32 değildir. Gerçek hayatta da işlemcilerde veriyolunun genişliğinin tümü her zaman kullanılmamaktadır. Aşağıda dar değerlere örnek olarak 2982 ve -9 sayıları gösterilmiştir. Görüldüğü gibi ikisinin de gösterilmesi için 32 bitten daha az sayıda bit kullanılması yeterlidir. 2982 için 32 yerine 13, -9 için de 32 yerine 5 bit kullanılması yeterlidir.



Şekil 1.1 – Dar değerlere örnek olarak 5 bitlik ve 13 bitlik değerler

Ancak mutlaka veriyolunda onun tüm genişliğini kullanan veriler de olacaktır. Bu nedenle dar değerler saklanırken onun dar olduğunu bildiren bir bit de bulunur. Değer okunurken bu bitin varlığı nedeniyle ön bitleri işaretle genişletilerek yani veriyolu genişliğine getirilerek kullanılır.

1.3. Amaçlanan Çalışma

İşlemcinin enerji tüketimi başlığında da bahsedildiği gibi işlemcinin enerji tüketimi hem sınırlara bağlı kalınması açısından hem de işlemci ısısını kontrol etmek açısından çok önemlidir. Devredeki büyük sorunlardan birisi olan sızıntı akımından kaynaklanan durağan enerji tüketimini azaltmaya yönelik çalışmalar yapılmıştır. Devreleri kullanılmadıkları anda açma kapatma da bu tür yöntemlerdendir [26]. Yine yazmaç öbeği gibi dizi yapılara uygulanabilecek kullanılmayan satırların kapatılması yöntemi [19] çalışmasında gösterilmiştir. Diğer başka yazmaç öbeğinde enerji tasarrufu yöntemi ise yazma okuma portlarından bazılarını iptal etmek [12] ve yazmaç öbeğini bloklara bölmektir [20], [21], [15], [16], [5]. Yazmaç öbeğinde dar değerler kullanılarak dar değerlerin tek bir satıra yazılması [6] ve yine yazmaçlarda oluşabilecek hatalara karşı dayanıklılık için dar değerlerin tekrar edilerek yazılması [14] yapılan çalışmalardan bazılarıdır. Bu çalışmada yapılan yazmaç öbeğinde veriler saklanırken yazılma ve okunma enerjisinden olduğu gibi hem de durağan enerjiden tasarruf etmek için yazmaç öbeğinin darlık gruplarına ayrılması ile yazılıp okunmayan kısımların kapatılması veya iptal edilmesi kullanılır. Yazmaç öbeğinin sabit satır sayısında gruplara bölünerek değerler hangi darlık grubundaysa o bloğa yazılması yöntemi yazmaç öbeğinin *sabit bölümlenme*sidir. Bunun yanı sıra

programın ihtiyaçlarına cevap verecek şekilde ve enerjiden en fazla tasarrufu sağlamak amacıyla darlık gruplarının satır sayılarının (boyutlarının) zaman içinde değiştirilebildiği yöntem ise yazmaç öbeğinin *değişken bölümlenme*sidir.

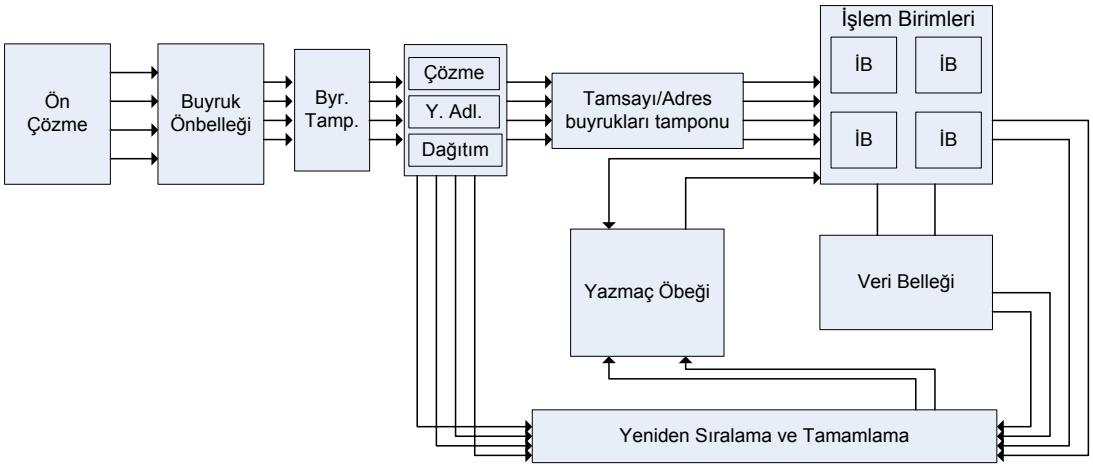
Bu tezde dar değerler kullanılarak yapılan sabit ve değişken bölümleme tekniklerinin nasıl gerçekleştirildiği ve hangisinin başarımı etkilemeden enerjiden daha iyi tasarruf sağladığı gösterilmiştir. İlerleyen bölümlerde ilk olarak 2. Bölümde işlemcilerin içyapısından ve yazmaç öbeğinin işlemci içindeki görevinden, daha sonra 3.bölümde işlemci veriyolu genişliği ile ilgili yapılan çalışmalardan ve işlemcideki dar değerlerin kullanım sıklığından bahsedilecektir. 4.bölümde ise yazmaç öbeğinin sabit ve değişken olarak bölümlenmesi yöntemleri ayrıntılı olarak anlatılacaktır. Son olarak 5.bölümde kullanılan tekniklerin benzetimlikler üzerinde yürütülmesi ve elde edilen sonuçlar işlenirken 6.bölümde son görüşler belirtilecek ve gelecekte bu işe bağlı olarak yapılabilecek olan diğer çalışmalar üzerinde durulacaktır.

BÖLÜM 2

2. MODERN İŞLEMCİLERİN YAPISI

İşlemcilerim bilgisayardaki temel görevi çalıştırılması istenen programdaki buyrukların sırasıyla işletilmesidir. Buyruklar programı oluşturan küçük işlemler olarak tanımlanabilir. Her buyruk veriyolundan geçirilerek gerekli sonuç hesaplanır ve işlemciden atılır, bu şekilde istenen programın çalışması sağlanır. Buna göre her buyruğun geçtiği belirli aşamalar vardır. Derleyici yardımıyla Java gibi üst düzey bir dilde yazılmış olan kodlardan buyruklar oluşturulur ve bunlar bellekte saklanırlar. Bellekte sırayla durmakta olan buyruklar belirli bir noktadan başlanarak işlemci içine alınırlar. 1 ve 0 lar halinde bulunan buyruğun, bitlerin dizilimine göre hangi işi yapacağı belirlenir. Daha sonra buyruğun ihtiyaç duyduğu veriler bellekten ya da yazmaçlardan getirilerek işlemler gerçekleştirilir ve sonuç değeri hesaplanır. Son adımda da buyruğun sonucunun yazılmasıyla işlem tamamlanmış olur. Tüm buyruklar genel olarak bu aşamalardan geçerek işlenirler. İşlemcide buyruğun sırasıyla geçtiği bu aşamalar veriyolunu oluştururlar. İlk işlemcilerde tüm bu işlemlerin tek bir saat vuruşunda tamamlanması daha sonra da hızlanma amacıyla daha kısa aralıklarla birden fazla saat çevriminde tamamlanmasına dayanan teknikler geliştirilmiştir. İşlemci tasarımında boruhattı tekniği günümüz işlemcilerine ulaşmada en büyük adım olmuştur. Boruhattı tekniğinde veriyolu küçük ve hızlı aşamalardan oluşmaktadır ve hiçbir aşama boş kalmayacak şekilde buyruklar veriyoluna arka arkaya alınmaktadır. Bu şekilde boruhattı dolduktan sonra her vuruşta bir buyruğun sonuçlanması sağlanır. Günümüz işlemcilerinin temelinde de boruhattı tekniği yatmaktadır ancak boruhattının sağladığı hızlandırma da yetersiz kalmıştır. Modern işlemciler çok yollu (superscalar) ve sırasız yürütüm(out of oreder execution) yapan işlemcilerdir. Bir boruhattı yerine birden fazla boruhattının paralel işlemesiyle daha çok buyruk daha kısa sürede işlenebilir. Çok birimli işlemci bir bakıma birden fazla boruhattı ya da birden fazla birim kullanılması anlamına gelir. Örneğin veriyolundaki işlem birimlerinin çoğaltılması ile buyrukların kendisinden önceki buyrukları bekleme süreleri azalır ya da birimler boşsa bekleme yapmasına

gerek kalmaz. Birden fazla birimin paralel çalışıyor olması da veriyoluna bir buyruk yerine daha çok buyruk alınmasına olanak verir. İçeriye alınan buyrukların sırayla işletilmesi ile yine sonuçta birden çok buyruğun sonucu elde edilmiş ve hızlanma sağlanmış olur. Ancak doğal olarak bir programı oluşturan buyruklar arasında bağımlılıklar vardır. Örneğin döngü gibi yapılar devamlı bir indeksi okumak zorundadır ya da kontroller yapacak buyruklar program akışında var olacaktır. Bu tür bağımlılıklar ve program akışını değiştiren buyruklar nedeniyle birden çok buyruğun paralel yürütülmesi her zaman uygun değildir. Bu nedenle sırasız yürütüm denilen ve bağımlılıkları kontrol ederek arada sorunu olmayan buyrukların yürütülmesini ve buyruklar çıktıkça içeriye buyruk alımı devam etmesini sağlayan sırasız yürütüm tekniği geliştirilmiştir. Sırasız yürütüm çok yollu bir işlemcinin yüksek başarım elde etmesi için en önemli yöntemlerden birisidir. Bu sayede çok yollu işlemci 2 katına kadar hızlanabilir [22]. Ancak işlemcide sırasız yürütümü düzenleyecek birimlere ihtiyaç vardır. Bunlar yazmaçların arasındaki sözde bağımlılıkları kaldıran yazmaç yeniden adlandırma birimi, programın akışını değiştiren buyrukların devam edeceği noktayı tahmin eden dallanma tahmin birimi, buyrukların ihtiyacı olan yazmaçların listesini tutarak kaynakları hazır olan buyrukları işlem birimine gönderen yayın kuyruğu ve sırası değişmiş olarak işlenen buyrukların yeniden sırayla işlemci dışına çıkarılmasını sağlayan yeniden sıralama belleği yapılarıdır. Sırasız, çok yollu yürütümü sağlayan yeniden adlandırma, dallanma tahmini, yayın kuyruğu ve yeniden sıralama belleği gibi yapılardan ve nasıl çalıştıklarından aşağıdaki kısımlarda bahsedilecektir. Yine işlemcide çok yolluluk ve sırasız yürütümü sağlayan birimler dışında normal bir işlemcide bulunması gereken yazmaç öbeği, program sayacı ve işlem birimleri de vardır.



Şekil 2.1 – Çok Birimli Sırasız Yürütüm Yapan İşlemci İçyapısı

2.1. Buyruk Yakalama ve Çözme

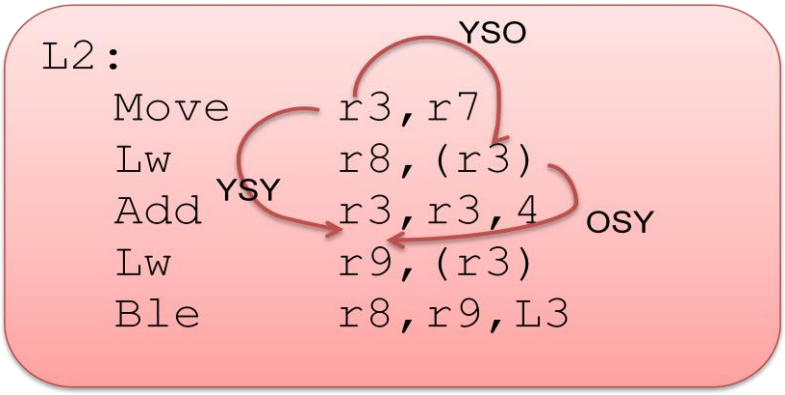
İşlemcide buyruklar genellikle onlara özel ayrılmış olan buyruk belleğinde tutulurlar. Buyruk belleğinden birkaç buyruk aynı anda çekilerek işlemciye getirilirler. Buyrukların birlikte çekilebilmesini kolaylaştırmak için arka arkaya çekilecek buyruklar bir sıra ya da blok halinde bulunurlar. Buyrukların durduğu bellek aynen bir önbellek mantığında çalışır, ana bellekten getirilen buyruklar önbellekte saklanırlar ve gerektiğinde program sayacının gösterdiği değere ve veriyolunun o anda alabileceği kapasiteye göre birkaç tane olmak üzere içeri alınırlar. İçeri alınacak buyruk sayısının o işlemcinin yürütme genişliği olup belirli bir üst sınırı vardır genellikle 4 ya da 8 buyruk bir seferde işlemci içine alınır. Buyruklar çekildikten sonra daha çözme işlemine geçmeden aralarında programın akışını denetleyecek dallanma buyrukları varsa dallanma yapıp yamayacağı ve gideceği adresin ne olduğuna dair tahmin ve hesaplamalar yapılır. Çünkü bir vuruş sonraki buyruğun çekileceği pozisyon kendinden önceki dallanma buyruğuna bağlıdır. Dallanma buyruğu geldiği zaman eğer dallanma yönüne dair tahmin yapılmazsa gideceği nokta belli oluncaya kadar buyrukların çekilmesi mümkün olmaz ve işlemci beklemek zorunda kalır. Dallanma yapıp yapmayacağına dair tahmin, genellikle buyruğun geçmişine bağlıdır yani buyruk geçmişte dallanma yapmışsa tekrar aynı şekilde davranması beklenir. Dallanma tahminleri ayrıca programın genel gidişatına bağlı

olarak da yapılabilir [23]. Birden çok tahmin edicinin bir araya getirilmesi ve daha iyi tahminde bulunan birimin kullanılması da dallanma tahmininde bir başka yöntemdir [23]. Eğer buyruklar dallanma buyrukları değillerse dallanma tahmini aşamasına uğramadan ilerleyip direkt çözme aşamasına gelirler. Çözme aşamasında buyrukların önceden belirlenmiş bir kısmına bakılarak hangi işlemi yapacağı anlaşılır ve buna göre veriyolu içerisinde ilerleyeceği birimler için denetim işaretleri hazırlanır. RISC (Reduced Instruction Set Computer) türü mimarilerde sabit uzunlukta buyruklar olduğu için her buyrukta aynı kısımdaki bitler kullanılabilirken x86 gibi CISC (Complex Instruction Set Computer) mimarilerde buyruk boyutları değişkendir ve bu durum çözme işleminin karmaşıklaşmasına ve uzamasına neden olmaktadır.

2.2. Yazmaç Yeniden Adlandırma

Buyruklar yürütülürken işlemci içine birden fazla buyruk almak beraberinde problemler de getirir. Örneğin 4 buyruğun içeri birlikte alındığı bir mimaride, aralarında veri bağımlılıkları bulunması durumunda 4 buyruğun paralel olarak işlemesi mümkün değildir. Buyrukların paralel olarak işleyebilmesine Buyruk Düzeyinde Paralellik (BDP) denir. Buyruklar arasındaki bağımlılıklar azaldıkça BDP artar. Yazmaç yeniden adandırma aşaması ise gerçekte var olmayan bağımlılıkların çözülmesi için mimari yazmaçların işlemci içinde farklı yazmaçlarla eşlenerek işlemlerin bu yeni yazmaçlarla yazılmasıdır. Örneğin Intel P6 işlemcisinde mimari olarak 8 yazmaç olmasına rağmen içeride bunlara karşılık 40 yazmaç bulunmaktadır [24]. Mimari yazmaçlara karşılık gelen diğer daha fazla sayıdaki yazmaçlara fiziksel yazmaç öbeği denir. Fiziksel yazmaçlar ile mimari yazmaçların eşleşmesi de yazmaç yeniden adlandırma tablosunda tutulur. Gelen buyruklar işlem yapacakları yazmaçları bu tablodan alarak tam olarak doğru fiziksel yazmaçlarla işlem yapabilirler. Buyruklar arasındaki bağımlılıklar Şekil 2.2de gösterilmiştir. Burada işlenecek buyruklar `move`, `lw`, `add` ve `ble` buyruklarıdır. `move` buyruğu yazmaçlar arasında veri taşıma işlemi yapar. Şekil 2.1deki örnekte r7 deki değeri r3 yazmacına yazması istenmektedir. `lw` buyruğu ise bellekten belirli bir adresteki değeri okuyarak hedef yazmacına okuduğu değeri yazar. Şekilde r3 yazmacında bulunan değeri adres olarak kullanıp bellekten veri okuyacak ve getirdiği veriyi r8 yazmacına atacaktır.

`add` buyruğu adı üstünde toplama işlemi yapmaktadır, r3 e 4 ekleyip tekrar r3 üzerine değeri yazacaktır. `ble` ise program akışını etkileyen dallanma buyruklarından biridir, r8 ve r9 yazmaçları arsasındaki eşitlik ya da küçüklük durumuna göre L3 ile gösterilen etikete dallanma yapacaktır. Bu nedenle kaynak olarak hem r8 e hem de r9 a ihtiyacı vardır.



Şekil 2.2 – Bağımlılıkları olan buyruklar

Şekil 2.1deki kod parçasında YSO (Yazdıktan sonra okuma), YSY (Yazdıktan sonra yazma) ve OSY (Okuduktan sonra yazma) bağımlılıkları vardır. Bunlardan yalnızca YSO bekleyerek çözümlenebilecek doğru bir veri bağımlılığıdır. Diğerleri yazmaç yeniden adlandırma ile çözümlenirler. YSO bağımlılığı `move` ve `lw` buyrukları arasında r3 üzerinde gerçekleşir. `move` dan sonra gelen `lw` buyruğu mutlaka `move` un yazdığı r3 de bulunan değeri okumak zorundadır. Bu nedenle `move` değeri yazana kadar yükleme saklama kuyruğunda bekleyecektir. Bu tür bir bağımlılığın çözümü için yazmaç yeniden adlandırma yeterli olmaz. YSY bağımlılığı ise `move` ve `add` buyrukları arasında yine r3 üzerinde gerçekleşir. `add` den sonra gelen `lw` buyruğunun `add` in değiştirmiş olduğu r3 değerini okuması gerekir. Ancak sırasız yürütüm sırasında buyruklar bir arada işlemciye girdikleri zaman `add` yerine `move` dan sonra hesaplanmış olan r3 ü okuyabilir. Bu durumdan korunmak için `move` ve `add` arasındaki bağ koparılarak `add` buyruğunun işlemesi için r3 e farklı bir fiziksel yazmaç atanır. Yine benzer bir durum `add` ve bir önceki `lw` arasında gerçekleşir. r3 yazmacı üzerinde OSY bağımlılığı oluşur. `Lw` ve `add` in birlikte veriyoluna alındığı

bir sırada `add` buyruğu `lw` den önce işletilirse `lw` ün okuduğu değer bir üstteki `move` yerine `add` in sonuçlandıracağı değer olabilir. Bundan kurtulmak için `lw` ve `add` arasındaki bağ da koparılarak r3 e yeni bir fiziksel yazmaç atanır. Buna göre elimizde A, B, C, D fiziksel yazmaçları olduğu kabul edilirse Şekil 2.2deki şekilde yeniden adlandırma yapılır.

```
L2:
   Move    A,r7
   Lw      C,(A)
   Add     B,A,4
   Lw      D,(B)
   Ble     C,D,L3
```

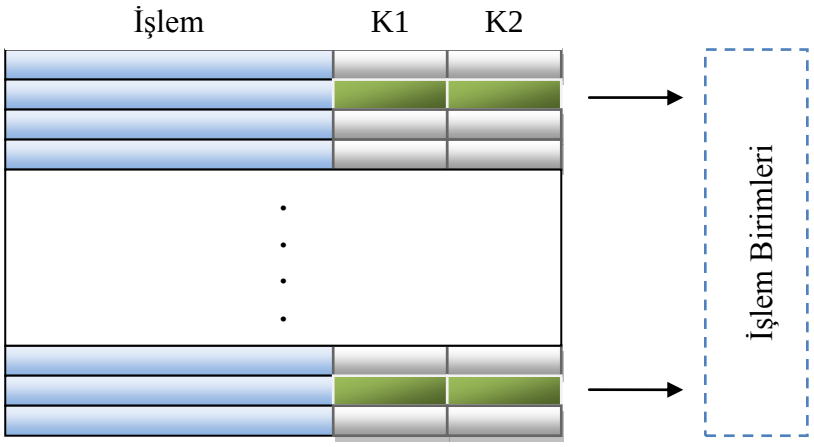
Şekil 2.3 – Yazmaçları yeniden adlandırılmış buyruklar

İlk `move` ve `lw` arasında r3 yerine A yazmacı kullanılırken, `add` ve ondan sonra gelen `lw` arasında B yazmacı kullanılır. Yeniden adlandırılmış olan buyruklar yazmaç yeniden adlandırma tablosunda tutulurlar. Her gelen buyruk kendi kaynağının hangi fiziksel yazmaç ile eşlendiğini tutarak kendi kaynakları hazır olduğunda işleme geçer. İlk dört buyruk işlemci içine alındığı zaman `lw` den sonra gelen `add` kaynak olarak A fiziksel yazmacını bekleyecek ancak kendisinden sonra gelenler ise r3 e erişmek isterlerse B yazmacını bekleyeceklerdir. İlk 4 buyruk yeniden adlandırma aşamasını geçtikten sonra yeni gelen buyruk r3 değerini B fiziksel yazmacından okur. Çünkü yazmaç yeniden adlandırma tablosu r3 mimari yazmacı için bu değeri gösterir. Fiziksel yazmaçların durumları işlemler devam ederken sürekli olarak değişir, bağlı olduğu buyruk tamamlandığı zaman yazmaç da mimari düzeye geçer ve fiziksel yazmaç artık serbest bırakılır. Bundan sonra yeni gelen buyruklar için kullanılır. Veriyolunda yazmaç yeniden adlandırma aşamasından sonra işlemler arasındaki aslında var olmayan bağımlılıklar sonlanır ve buyruklar paralel şekilde yürütülebilirler. Sırasız yürütümün önemli noktalarından birisi de yazmaç yeniden adlandırmadır. Yazmaç yeniden adlandırma evresi bir

bakıma her buyruğa belirli bir yerin rezerve edilmesi anlamına gelir. Bu çalışmada yapılan, yazmaç öbeğinin bölümlenmesi işinin de ilk adımı yazmaç yeniden adlandırma evresinde buyruğa verilen yazmacın türünün belirlenmesi ile başlar.

2.3. Yayınlama ve Yürütme

Buyrukların veriyolu içine alınıp kaynakları hazır olanların derhal çalıştırılması *sırasız yürütüm* anlamına gelir. Sırasız yürütümün etkin kullanılabilmesi için de birden çok işlem birimi olması gerekir ve işlem birimleri de birbirlerinin sürelerine bağımlı olmamaları için boruhattına konulmuş olmalıdır [29]. Bunu sağlamak için de buyruğun gerçekleştirmek istediği işlemin yapıldığı yürütme aşamasında önce yayınlanma ve değerlerin okunması aşamaları yerleştirilmiştir. Yapısal ve veri bağımlılıklarına sahip olan buyruklar bunlardan kurtuluncaya kadar bekletilmek zorundadırlar. Yazmaç yeniden adlandırma ile bağımlılık sorunları çözüldükten sonra veri bağımlılıklarından da diğer buyrukların çalışması sonucu kurtulan ve hazır olan buyruklar işlem birimlerine gönderilirler. İşlemcilerde yayınlama işlemi genellikle bir kuyruk yardımıyla yapılır. Buna rezervasyon durağı ya da yayın kuyruğu denir. Çözme ve yazmaç yeniden adlandırma kısmından gelen buyruklar artık işlem, kaynaklar ve sonucun yazılacağı yerler belirlenmiş şekilde yayın kuyruğuna gelirler. Yayın kuyruğunda bekleyen buyrukların kaynak yazmaçları hazırlandıkça – hesaplandıkça – buyruklar işlenmeye hazır hale gelirler ve yayın kuyruğundan atılırlar. Bir yazmacın değerinin hazır olduğunun yayın kuyruğuna bildirilmesi değişik yöntemlerle yapılabilir. Bunlar belirli periyotlarla kontroller yapmak ya da sonuçlanan işlemin yayın kuyruğuna haber vermesi şeklinde olabilir. Günümüz işlemcilerinde yayın kuyruğuna hazır olan sonuçların bildirilmesi dağıtma denilen ara bir aşama ile gerçekleştirilir. Dağıtma sırasında sonucu hesaplanan yazmaçlar yayın kuyruğunun kaynaklarının tümü ile karşılaştırılmasını sağlayan portlardan gönderilirler bu sırada kaynakların ve hesaplanmış değerlerin etiketleri karşılaştırılır ve aynı olanlar hazır olarak işaretlenirler.



Şekil 2.4 – Yayın Kuyruğu

Yayın kuyruğunun uzunluğu işlemcinin çalışma penceresini belirleyen önemli değişkenlerden birisidir. Diğer birimlerin boyutlarına ve içeriye alınabilecek buyruk sayısına göre ihtiyaca cevap verecek büyüklükte yayın kuyruğuna ihtiyaç vardır. Aksi takdirde yayın kuyruğu dolu olduğu için onun çıkmasını bekleyen buyruklar nedeniyle veriyolu dolacak ve yeni buyruklar alınamayacaktır. Yayın kuyruğu, her ne kadar kuyruk diye isimlendirilse de ilk giren ilk çıkar mantığıyla işletilmez, kaynakları hazırlanan buyruklar işlem birimlerine gönderilirler. Sırasız yürütüm için gereken mimarinin düzenlenmesi yani buyrukların içeri alındıktan sonra sırasız yürütüme hazırlanması için bağımlılıkların bulunması ve çözümlenmesi için Tomasulo Algoritması [7] gibi yöntemler kullanılır. Yayın kuyruğu ise hem Tomasulo algoritmasında hem de sırasız yürütüm için kullanılan diğer yöntemlerde de önemli noktalardandır. Yayın kuyruğu tüm işlemci için bir tane olabileceği gibi farklı işlem birimleri için bölünmüş olarak da kullanılabilir. Tasarım aşamasında bölünmüş yayın kuyrukları düzenleyici gibi görünse de donanımın gerçekleştirilmesi sırasında fazladan karmaşaya neden olur. Ancak yükleme ve saklama buyruklarının bellekle ilişkisi olması ve dolayısıyla tamamlanma sürelerinin diğer buyruklara göre çok daha uzun olması nedeniyle yükleme ve saklama buyruklarının tutulduğu özel bir rezervasyon durağı da işlemcilerde bulunmaktadır. Temelde yayın kuyruğu ile aynı mantıkta yani kaynakları hazır olmayan buyrukların beklemesi – yükleme saklama buyrukları için kaynaklar bellekten gelecek verilerdir – ve kaynakları hazır

olanların işlenmesi için kurulmuş bir yapıdır. Yükleme saklama kuyruğunda bekleme sürelerini kısaltmak için veri bağımlılıkları kuyruk içindeki saklama buyruklarının veriyi daha belleğe yazmadan onu okuyacak olan yükleme buyruğuna değerin yönlendirilmesi ile de çözülebilir.

Yürütme işlemi ise yayın kuyruğundan gelecek olan hazır sinyali ile ya da tasarıma bağlı olarak verilerin kendilerinin yayın kuyruğundan alınmasıyla birlikte kaynak değerlerle bildirilen işlemin yapılmasıyla gerçekleştirilir. Çok birimli işlemcilerde işlem birimleri yapacağı işe özelleşmiş ve paralel çalışır durumda bulunurlar. Örneğin bu çalışmada kullanılacak olan işlemcide 6 tamsayı 2 kayan nokta işlemi yapan birim vardır. Yayın kuyruğundan çıkan buyruklar hangi işlem yapılacaksa oraya yönlendirilir. Ancak işlem birimleri işlerini tek bir vuruşta bitirmek zorunda değillerdir. Uzun süren bölme gibi işlemler yapan birimler kendi içinde boruhattına konularak saat sıklığını etkilememesi sağlanmıştır. Bu tür birimlerin sonuçları daha geç belli olur ve onu bekleyen sonraki buyruklar da hazır olduğunda değerleri alabilirler. Sırasız yürütümün başarıma etkisi böyle uzun işlemler için özellikle ortaya çıkar. Sonradan gelen kısa sürecek buyruklar kendinden öncekileri beklemeden birimler arasında ilerleyip sonucunu yazabilirler.

2.4. Sonuçların Yazılması ve Tamamlama

İşlem birimlerinde yapılan hesaplamalar sonucunda çıkan değerler fiziksel yazmaç öbeğine yönlendirilirler, bu aşamada yazmaç öbeğine yazılan değerler daha sonra buyrukların türüne göre bellek adres göstericisi ya da dallanma buyruğunun atlayacağı adresi bildiren değerlerdir. İşlemci içine alınan buyrukların tümünün aynı anda yazılabilmesini sağlamak için yazmaç öbeği aynı sayıda yazma portuna sahiptir. Yine kaynakların okunmasında sorun olmaması için yazma sayısının 2 katı kadar da okuma portu bulunur. Değerler işlem biriminde alındıktan sonra sonuçları yazmaç öbeğinde yazmaç yeniden adlandırma aşamasında belirlenmiş olan adrese yazılırlar. Yine işlemcideki durumları bildiren bayraklar da yazma evresinde ayarlanırlar. Bir buyruğun sonucunun yazılmış olması onun artık işlemcinin veriyolunda gezinmeyeceği anlamına gelir. Buyruğun artık tek bağlı kaldığı yer yeniden sıralama belleğidir. Buyruklar işlemciye ilk girdiği anda o buyruğa ait bir etiket yeniden sıralama belleğine kaydedilir, daha sonra buyruklar sonuçlandıkça YSB inde işlerini

bitirdiklerine dair işaretlenirler. İşlemini tamamlamış buyruklar da yine birkaç buyruk birlikte YSB den çıkarılırlar. İşlemcinin bir vuruşta tamamlayabileceği en fazla buyruk sayısı tasarım sırasında belirlenmiştir. Ancak bu kadar sayıda buyruk hazır değilse, hazır olan sayıda buyruk YSB den atılır. Sırasız yürütümün işlemci içindeki karmaşasından dışarıda programın etkilenmemesini sağlayan bu yapıdır. Gelen buyruklar sırayla içeri alınır ve içeriden sırayla çıkarılırlar.

BÖLÜM 3

3. İŞLEMCİLERDE VERİ GENİŞLİĞİ

Sayısal değerler elektronik dünyada ikilik düzende yani 1ler ve 0lar kullanılarak ifade edilirler. Bunun nedeni elektronik devrelerin iki voltaj düzeyini ayırabilecek şekilde tasarlanmasıdır. Genelde "1" 1 volt ile ve "0" da 0 volt ile gösterilir. İşlemci bilgisayarın içine yerleştirilen ve yerleştirildikten sonra da müdahalesi neredeyse imkansız olan bir parçadır. Bu nedenle değiştirilemeyecek bu parça üzerinde tasarım sırasında verilen kararlar çok önemlidir. Bu kararlar buyruk ve veri tutan belleklerin ayrı mı birleşik mi olacağından başlayıp en alt düzeyde birimler arasındaki tellerin uzunluğuna kadar çok geniş bir çerçevede verilir. İşlemci tasarımında en önemli kararlardan birisi de işlemcideki kelime uzunluğunun kaç bit olacağıdır. İşlemcideki kelime uzunluğu kaç bit ise birimler arasındaki yollar bu genişlikte tasarlanacak ve tüm işlem birimleri de buna uyumlu olarak yapılacaktır. Hatta bellekte bulunacak buyruklar dolayısıyla bellek de kelime uzunluğundan etkilenecektir. Kelime uzunluğu ya da veriyolu genişliği genelde ikilik düzene uyum ve kelimenin kendi içinde de kolay adreslenebilmesi açısından ikinin bir kuvveti olacak şekilde seçilir. Buna göre 8 bit yani 1 bayta dayanan temel genişliğin birkaç katı olarak seçilebilir. Kelime uzunluğun artması büyük değerlerle işlem yapmayı mümkün kılar ve kolaylaştırırken, küçülmesi de işlemciyi hızlandıracak ancak yapılabilecekleri kısıtlayacaktır. Bu nedenle hem işlemcinin kabul edilebilir hızlarda çalışmasını hem de büyük değerlerle de işlem yapılabilmesini sağlayacak veriyolu genişlikleri seçilmelidir. Bu konuda değişik birimlerde değişik genişlikte değerlerin kullanılması gibi yöntemlere de gidilerek tasarım değiştirilip rahatlatılabilir. Modern işlemcilerde özellikle günlük kullanım için olanlarda kelime uzunluğu genelde 4 ya da 8 bayt yani 32 ya da 64 bit olarak seçilir. Bizim kullanacağımız işlemcide veriyolu genişliği 64 bit olarak belirlenmiştir.

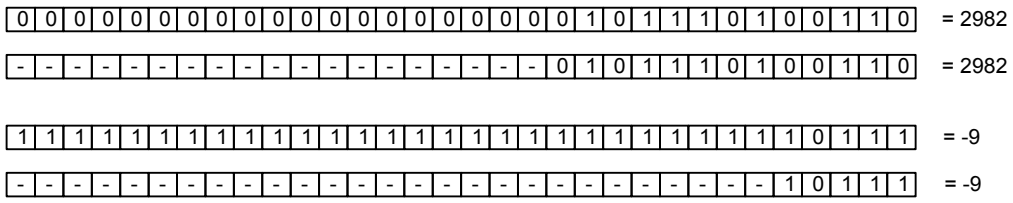
3.1. Dar Değerler ve İlgili Diğer Çalışmalar

Veriyolunda kullanılan değerler ikilik düzende ve ikiye tümleyen gösterimi ile ifade edilirler. İkiye tümleyen gösterimi eksi sayıların ifadesini kolaylaştırmak için

kullanılır. Bu gösterimde ilk bit değerin işaretini gösterir hem de bir değer ifade eder. Örneğin 3 bit kullanıldığında

000	0	001	1	010	2	011	3
100	-4	101	-3	110	-2	111	-1

Gösterir. -3 sayısı 3 bit ile gösterilebilirken 5 bit ile gösterildiği zaman 11101 şeklinde, yine 10 bit ile gösterilmek istendiğinde 1111111101 şeklinde gösterilir. Değerin işaret biti yeni eklenen kısma kopyalanır. Sayı pozitif olduğunda da 1 yerine 0 kopyalanır. Bu küçük örnekten de görüleceği gibi aslında 3 bit ile gösterilmesi mümkün olan -3 sayısı veriyolunda 64 bit ile gösterilecektir. 32 bitlik veriyolunda -9 ve 2982 sayısının gösterimi aşağıdaki şekildedir.



Şekil 3.1 – 9 bitlik ve 13 bitlik değerlerin 32 bit veriyolunda gösterimi

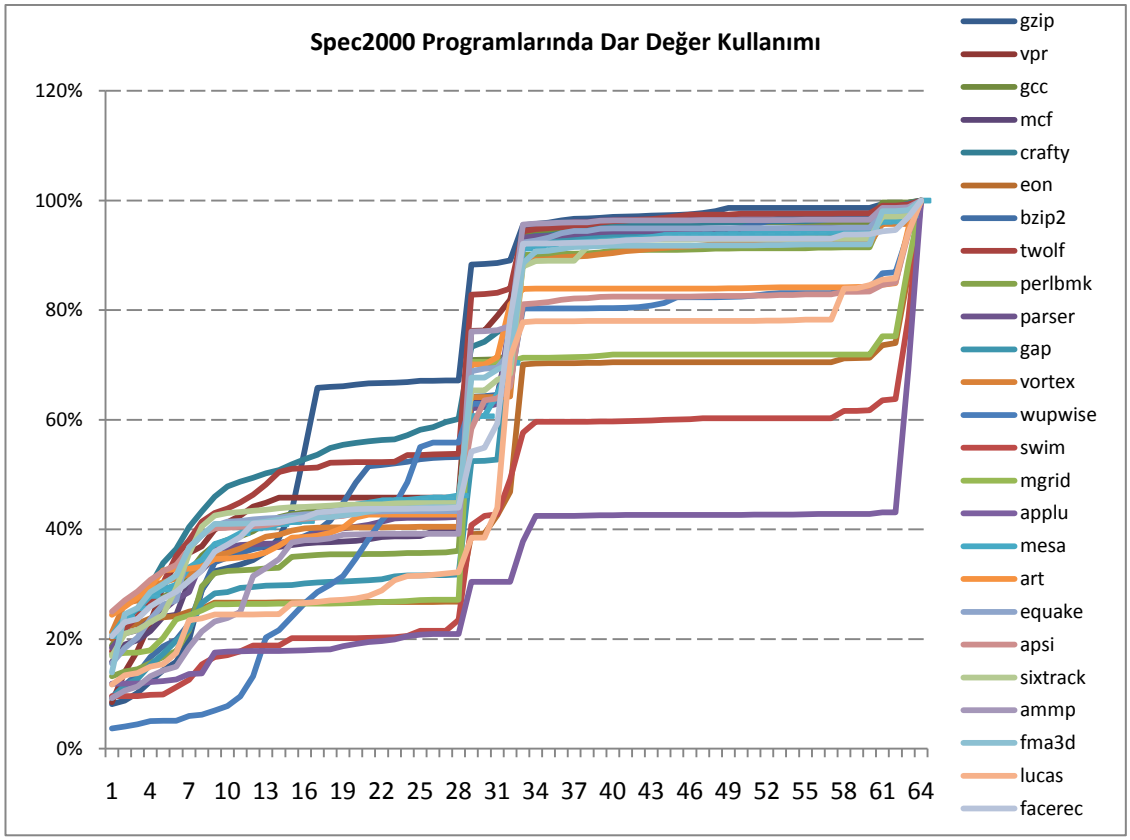
2982 sayısı için 13 bit ve -9 için 5 bit yeterli iken veriyolunda bunlar için 32 bit kullanılarak gösterilmiştir. İfade edildiği bit sayısından daha az sayıda bit ile ifade edilebilen değerlere *dar değer* denir. Dar değerler veriyolu genişliğinin boşa harcanmasını önlemek için fark edilerek değerlendirilir.

Dar değerlerin kullanılmayan bitlerinin ifade edilmemesi ve o değerin dar olduğunu ifade eden 1 bitlik alanın değere eklenerek okuma yapılacağı zaman değerin genişletilmesi ile işlemcide yüksek enerji tasarrufları elde etmek mümkündür. Bu konuda şimdiye kadar pek çok çalışma yapılmıştır. Çalışmalar işlemcinin farklı birimlerinde enerji tasarrufu ve dar değerlerin fark edilmesi gibi konularda gerçekleştirilmiştir. Dar değerlerin kullanımında da enerji harcayacak kısımlar değerin dar olduğunun fark edilmesi ve dar olarak kaydedilmiş değerin işareti

kopyalanarak veriyolu genişliğine getirilmesidir. Ancak bu gibi kısımlar enerjiden tasarruf edilmesini engellemeyecek kadar az güç tüketen kısımlardır. Dar değerler işlemcide enerji tasarrufu için kullanıldığı gibi kimi zaman başarımı artırmak için de kullanılmıştır. Enerji tasarrufu için veriyolunda dar değerlerin fark edildiği anlarda işlem birimlerinin veri genişliğini azaltılmasıyla birlikte birkaç dar değerin aynı anda işlem yapmasını sağlayacak yapılar geliştirilmiştir [2]. Yine enerji tasarrufu için [25] çalışmasında önbellekteki arka arkaya sıfır olan dar değerlerin sıfır kısımlarının sıkıştırılması kullanılmıştır. Yazmaç öbeğinde dar değerlerin kullanılması ise verinin saklandığı bir yer olduğu için üzerinde daha çok durulan bir geliştirmedir. Genellikle yazmaç öbeğinin dar ve normal değerler için farklılaştığı yapılar tasarlanmıştır. [17] çalışmasında yazmaç öbeğini farklı darlıklarda bloklara ayırarak gelen değerleri kendine ait olan bloğa yazılmasını sağlar. Yazmaç öbeği 2 ye bölünerek dar olanların dar tarafa diğer normal olanlarında her iki tarafa birden yazılması [13] ya da yazmaçların hepsini ikiye bölerek normal yazmaç sayısını 2 katına çıkarıp dar olanları tek yazmaca normal genişlikte hatta uzun olanları da birkaç yazmaca birlikte yazmak [8] kullanılan yöntemlerdir. Dar değerlerden yararlanarak yeniden adlandırma tablosunda dolaylı olarak erişilecek yazmaç adresi tutmak yerine eğer değer darsa değerin kendisini tutarak gereksiz yazma ve okuma işlemi yapılması önlenmiştir [9]. Yazmaç öbeğinden okunacak değerleri buyrukların durumuna göre birisi dar birisi geniş ya da ikisi de dar veya geniş olmak üzere farklı okuma portlarıyla tasarlayarak okuma ve yazma enerjisinden tasarruf sağlanmıştır [10]. Dar değerleri yeniden adlandırma aşamasında tahmin edip yazmaç öbeği satırlarının bir kısmını kullanmalarını daha sonra yeni gelecek olan dar değerlerin de yazmaçların diğer yarılarına yazılmaları ile başarımdan kazanç sağlanır [11]. Bu şekilde hem daha fazla boş yazmaç kalır hem de yazmaç yeniden adlandırma aşamasında beklemeler azalır.

3.2. Dar Değerlerin Kullanımı ve Değişim Evreleri

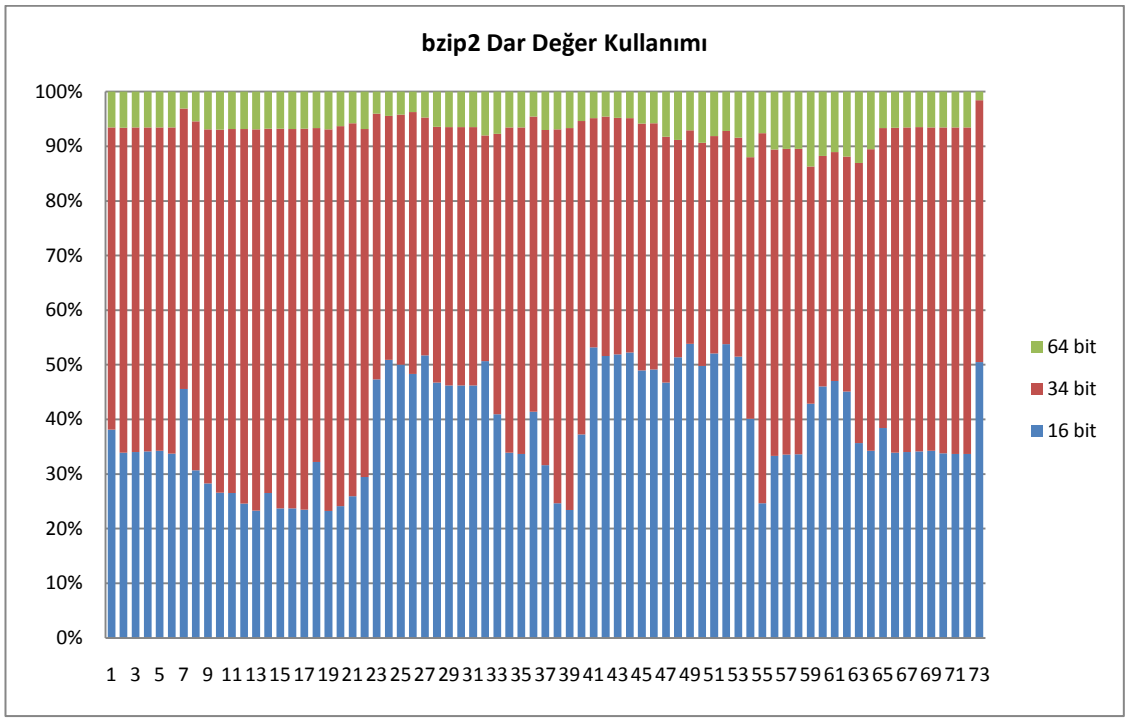
İşlemcinin geneline bakıldığında yazılan değerlerin çok yüksek bir oranda dar olduğu fark edilmiştir. Özellikle tam sayılarla işlem yapılan tamsayı yazmaç öbeğine yazılan değerlerin yaklaşık % 95 i 34 bit ile gösterilebilir.



Şekil 3.2 – Spec2000 Programlarında Dar Değerlerin Kullanımı

Denektaşı programları olarak SPEC (Standart Performance Evaluation Corporation) şirketi tarafından hazırlanmış Spec2000 deneme programları grubu kullanılmıştır. Programların hepsi çalıştırıldığında işlemcinin hemen hemen tüm farklı buyruk türleri birleşimleri için çalışması denenmiş olur. Denektaşı programlar işlemcilerin arasında başarım değerlendirmesi yapılması ve araştırmaların denenebilmesi için hazırlanmıştır. 5.bölümde programların neler olduğu açıklanacaktır. Programların hepsinde dar değerlerin oranı çok yüksektir. Değerlerin büyük bir kısmı 34 bit ile gösterilebildiği için 64 bitlik veriyolunda geriye kalan 30 bit bir sürü çevrim boyunca boşu boşuna sızıntı akımına yani enerji kaybına sebep olacaktır. Ayrıca okuma yazma yapılırken de yine 34 bitlik değerler yazılabileceği halde 64 bitlik değerlerin yazılıp okunması enerji kaybına sebeptir.

Dar değerlerin programın çalışması sırasında hangi aşamalarda ne kadar yoğun geldiği ise aşağıdaki grafikte gösterilmiştir.



Şekil 3.3 – Bzip2 Dar Değer Kullanımı

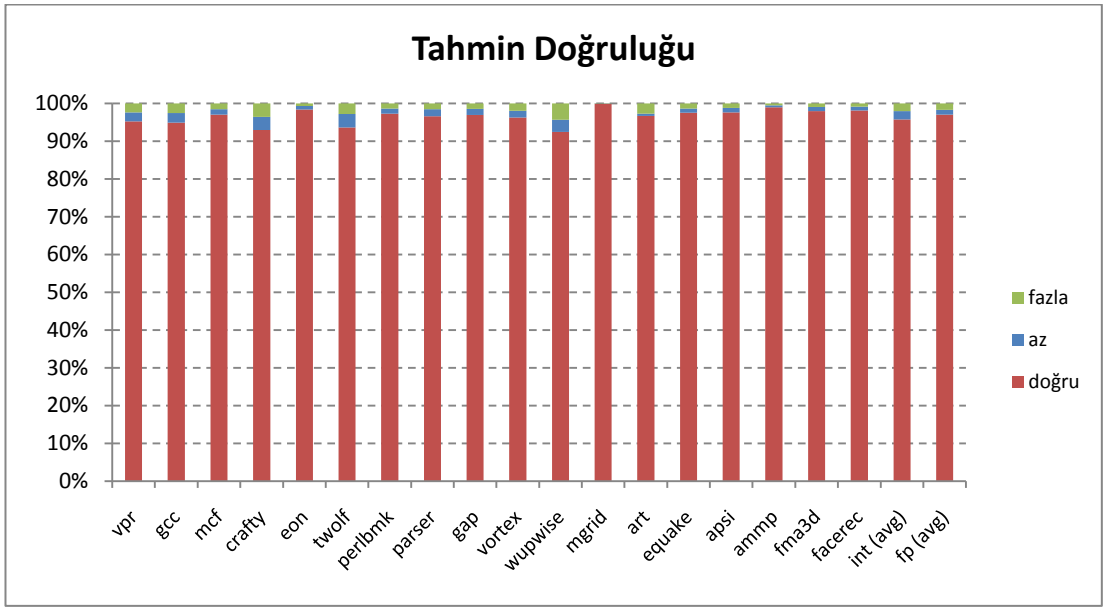
Şekil 3.3de değerler 3 gruba bölünerek 10 milyon çevrimde birbirlerine oranları gösterilmiştir. Görüldüğü gibi ilk 20 devirde 34 bitlik değerlerin oranı % 50 -70 arasında değişmekte iken daha sonraki devirde % 40 a inmektedir. Örnek olarak Bzip2 programı gösterilmiştir ancak 5.bölümde her bir denektaşı programı için darlık kullanımlarının nasıl değiştiği gösterilecektir. Dar değerlerin geliş evreleri her program için farklı şekillerde olmaktadır. 33 bit bellek adreslemesi için kullanılan değerin genişliğidir, bu nedenle programların belleğe eriştikleri sıralarda 33 – 34 bit darlıklar artış gösterecektir. Yine programın yaptığı işe göre de gelen darlıklar farklı kullanım sıklıkları gösterir.

3.3. Dar Değerlerin Belirlenmesi ve Darlık Tahmini

Önceki kısımda belirtildiği gibi dar değerlerin kullanılmasında enerjiden tasarruf sağlanacaktır. Ancak dar değerlerin kullanımı için de veriyolunda yani işlemcide

değişiklikler yapılması gereklidir. Bunlardan en önemlisi dar değerlerin belirlenmesidir. Dar değerlerin belirlenmesi işlemi yan yana olan birlerin ve sıfırların belirlenmesine dayanır. Dar olarak ifade edilebilen değerin sıkıştırılacak olan bitleri hepsi aynı olan işaret bitleridir. Bitlerin arka arkaya 0 ya da 1 olması durumunda değerin dar olduğunu bildiren bir ya da iki bit eklenir. Kaç farklı darlık grubu kullanılacağına bağlı olarak eklenecek bit sayısı değişir. Dar değerlerin belirlenmesi arka arkaya bulunan bitleri fark eden yine arka arkaya bağlanmış N tipi transistörlerden oluşan devre ile yapılır. Devre [13] çalışmasındaki şekilde tasarlanmıştır. Sıfırları ve birleri fark eden farklı devreler yapılmıştır. Transistörlerin gatelerine incelenecek değerin bitleri bağlanır ve sıfırların inceleneceği devrede değerlerden birisinin 1 olması ile sıfıra doğru akış gerçekleşir ve değerin dar olmadığı anlaşılır. Aynı durum 1lerin kontrol edildiği devrede de gerçekleşir. Veriyolundaki 64 bitlik değerler birkaç parçaya bölünerek paralel olarak darlık belirleyici devreler kullanıldığı zaman enerjiden ve zamandan neredeyse hiç kayıp yaşanmadan dar değerin belirlenmesi sağlanır.

Yeniden adlandırma aşamasında buyruklara yazmaçlar atanacağı için buyrukların hangi darlıkta olduğu dolayısıyla hangi türden yazmaç atanacağının belirlenmesi gerekir. Buyruk işlenmeden sonucu yani darlığı anlaşılamayacağı için gereken buyruğun darlığını tahmin etmek için darlıkları tahmin eden bir birime ihtiyaç vardır. Darlık tahmini farklı yöntemlerle yapılabilir [11]. Bizim kullandığımız yöntemde her buyruğun en son hesapladığı değerin darlığı ve buyruğun hangi işlemi yaptığı kaydedilir, bu sonucu bildiren bir alan da buyrukların okunduğu belleğin satırlarına eklenir. Her yeni çekilen buyruğun önceki darlık değerine bakılır ve eğer aynı işlem yapılıyorsa tahmin olarak aynı değer alınır. Yazma aşamasında eğer tahmin yanlışsa değer yeni yazmaca atılır ve bellekteki tahmin değeri yenilenir. Eğer buyruğa dair bir tahmin bulunamamışsa en geniş olan gruptan yer verilir. Darlıkların hem adresi hem de yaptığı işleme göre gruplanıp ona göre darlıklara bakılması tahmin doğruluğunun yüksek olmasını sağlar Bu yöntem ile tahmin edilen değerlerin doğruluk oranı aşağıdaki grafikte gösterilmiştir.



Şekil 3.4- Darlık Değeri Tahmin Doğruluğu

Görüldüğü gibi tahmin edilen değerlerin ortalama % 95 i doğrudur. Geriye kalan % 5 lik kısmın da yarısı yine sorun çıkarmayacak şekilde fazla yer verilmiş olan yazmaçları ifade eder. Bunların tahmin değerlerinin düzeltilmesi dışında yazmacının yenilenmesi gerek yoktur. Kalan % 2,5 lık kısımda yanlış tahmin yapılmış yani gerekenden az yer verilmiştir. Bu durumda hem tahminin yenilenmesi hem de yazmacın yeni bir yazmaca taşınması ve bunun yayın kuyruğuna bildirilmesi gerekir. Ancak bu durum çok az gerçekleşeceğinden işlemcide sorun yaratmayacaktır.

BÖLÜM 4

4. YAZMAÇ ÖBEĞİ BÖLÜMLEME

Yazmaç öbeği bölümlenmesi bu tezde sabit ve değişken olmak üzere 2 şekilde gerçekleştirilmiştir. Sabit bölümlemede yazmaç öbeği önceden belirlenmiş satır sayısında ve bit genişliğinde parçalara bölünmüştür. Bu şekilde yazılmayan ve okunmayan bitler üzerinden enerji tasarrufu yapılması planlanmıştır. Değişken bölümleme ise programların ihtiyaçlarına göre değişebilecek bir yazmaç öbeği yapısını gerçekleştirmek için tasarlanmıştır. Değişken bölümlemede dar yazmaç bloklarının büyüklükleri program akışı sırasında değiştirilebilmektedir. İki yöntem için de ihtiyaçların benzerlik ve farklılık gösterdiği noktalar vardır. Örneğin yazmaç atama yöntemleri benzerken, yapılması gereken fiziksel değişiklikler sabit bölünmüş yazmaç öbeğinde daha fazladır.

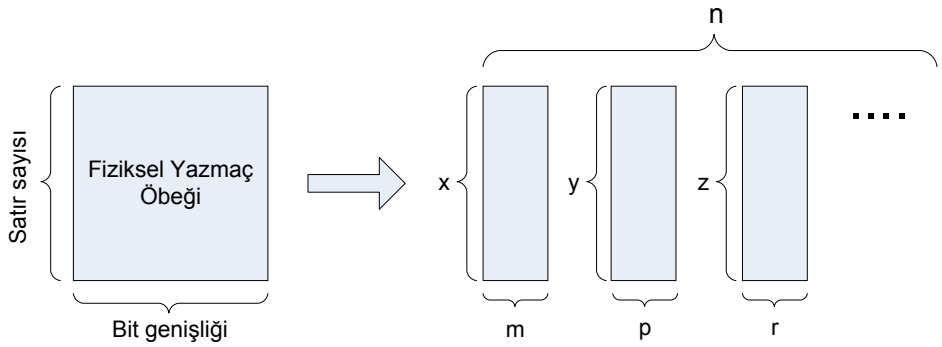
4.1. Sabit Bölümleme

Yazmaç öbeği sabit olarak bölündüğü zaman tek bir parça halinde bulunan yazmaç öbeği birkaç ayrı yazmaç öbeği olarak kullanılır. Bu bölümlemenin en önemli özelliği değişmesi mümkün olmayacak şekilde yapılmış bir tasarım olmasıdır. Bunun için blokların darlık bit sayıları ve büyüklükleri hemen her ihtiyacı karşılayacak şekilde tasarlanmalıdır. Ayrıca normal şartlarda buyruklar için yazmaçlar verilirken hangi fiziksel yazmacın verildiği önemli değildir. Sıradaki boş yazmaç yeni buyruk için kullanılırken, sabit olarak bölümlenmiş bir yazmaç öbeğinde yer verme işi de daha farklı adımlardan oluşacaktır. İşlemci tasarımı sırasında içerisine farklı boyutlarda yazmaç öbeği blokları yerleştirilecektir. Buna göre yazmaç öbeği devresi birbirinden bağımsız olarak tüm satırlara erişilebilen yapılar olmalıdır.

4.1.1. Bölümleme Yöntemi

Sabit yazmaç bölümleme yönteminde amaçlanan yazmaç öbeğinin darlık değerlerine bağlı belirli sayıda parçaya bölünmüş olarak kullanılmasıdır. İşlemcideki mevcut yazmaç öbeği ile aynı görevi yapacak olan birim, kullanılmayan bitler dolayısıyla enerjiden kazanç sağlayacaktır. Böyle bir sistemde ise yapılan yazmaç öbeğinin dar

yazmaç blokları nedeniyle yer bulma sorununa yol açmaması gereklidir. Yazmaç öbeğinin farklı bloklara bölünmesi için problemde bazı parametreler mevcuttur. Şekil 4.1de bu parametrelerin neler olacağı gösterilmiştir.



Şekil 4.1. Fiziksel Yazmaç Öbeğinin Sabit Bölümlenmesi

Dar blokların toplam satır sayısının fiziksel yazmaç öbeğinin satır sayısı ile aynı kalması enerjinin verimli kullanılması için gereklidir. Aksi takdirde hem yazmaç öbeğine fazladan satır ilavesi yapılacak hem de istenen verim sağlanamayacaktır. Bu nedenle yazmaç öbeğinin bölüneceği blok sayısı her blok bir satır olmak üzere toplam satır sayısı kadar bloğa kadar gidebilir. Bu çalışmada kullanılan mimaride 256 yazmaç olduğu için toplamda 256 farklı blok konulabilir. Ya da her bir darlık grubu için faklı bir blok tasarlanabilir. Bu durumda da kullanılan mimariye göre toplam 64 tane blok bulunacaktır. Her bir blok erişilecek başka bir ayrı tablo anlamına geldiği için ve her bir tabloda gelen adresi işlemek için kod çözücü gibi donanımlara ihtiyaç olduğu için satır sayısı kadar bloğun bulunması verimli olmayacaktır. Yine benzer şekilde her değerin darlık grubunun belirlenmesi işlemi daha fazla darlık sayısı olması ile uzun sürecektir ve hem enerji hem zamandan kayba neden olacaktır. Bu yüzden her bir bit için farklı blok yapılması mümkün değildir. Yazmaç öbeğinin sabit olarak bölümlenmesi birçok parametreye sahip olan

ve başarım üzerinde doğrudan etkisi olan bir problemdir. Başarım üzerindeki olumsuz etkisi yeniden adlandırma aşamasında buyrukla yazmacın eşleştirilmesi sırasında hangi bloğa yazılacağının anlaşılması için harcanacak süre kaybıyla ortaya çıkar. Problemin daha çözülebilir olması ve fiziksel açıdan makul olacak boyutlara getirilmesi için blok sayısı sabit bir sayı olarak 3 seçilmiştir. 3 farklı darlıktaki yazmaç bloğuna gelen değerlerin yerleştirilmesi sırasında karar vermek için geçecek süre işlemcideki hiçbir aşamayı etkilemeyecek kadar az olacaktır. Ayrıca 3 farklı yazmaç bloğunun fiziksel olarak getireceği yük de sınırlıdır. Bu şekilde hem durağan hem de devingen enerjiden kazanç sağlanabilecektir. Ancak 3 yazmaç bloğu olması kararına rağmen hala başarımı ve enerji tasarrufunu etkileyecek olan kararlar olarak yazmaç bloklarının satır sayısı ve bit genişlikleri vardır. Bu karaların verilmesinde ise bu çalışmada izlenen yöntem birkaç farklı satır – sütun birleşiminin sonucunda elde edilen değerlerin üzerinden doğrusal tamsayı programlama (ILP - **I**nteger **L**inear **P**rogramming) kullanarak en verimli satır sütun birleşimlerini bulmaktır. Temel olarak alınan değerler ve birleşimler aşağıdaki Çizelge 4.1 de gösterilmiştir.

Çizelge 4.1. Benzetim Yapılmış olan Sabit Bölümleme Satır Sütun Birleşimleri

	Blok 1		Blok 2		Blok 3	
Birleşim	Satır Sayıları	Bit Sayısı	Satır Sayıları	Bit Sayısı	Satır Sayıları	Bit Sayısı
1	86	9	85	23	85	64
2	100	9	86	23	70	64
3	120	9	96	23	40	64
4	145	9	81	23	30	64
5	103	9	103	23	50	64

Bit genişlikleri hep aynı değerde tutularak satır değişimlerine göre sonuçlar alınmıştır. Bu şekilde alınan değerlerde eşit ve eşite yakın bölümlemeler daha iyi sonuçlar vermiştir. Ancak bu şekilde ilerlemek mümkün olamayacağı için problemin çözümünde ILP kullanılmıştır.

ILP Problemi

Integer Linear Programming ya da Integer Programming NP-complete bir problemdir yani polinomsal bir zamanda çözümünün bulunması mümkün değildir [27]. ILP probleminin tanımı belirli şartlar altında istenen bir fonksiyonun değerini maksimize veya minimize edecek tam sayı değerleri bulmaktır.

Aşağıdaki

$$z = \sum_{j\epsilon E} c_j x_j , \qquad c_j \epsilon R \tag{4.1}$$

Fonksiyonunun

$$x_j \epsilon Z, \ \forall j \epsilon E \tag{4.2}$$

$$0 \leq x_j \leq \alpha_j , \ j\epsilon E, \alpha_j \ bir \ tam \ sayı \ ya \ da + \infty \tag{4.3}$$

Kısıtları altında çözümü bir $[x_j]$, $j \in E$ vektörüdür. ILP probleminin tanımı onun tam olarak yazmaç bölümlemesi sorununa uygulanabileceğini gösterir. Yazmaç bölümlenmesinde minimize edilmek istenen fonksiyon kullanılan enerjinin hesaplandığı fonksiyon ve kısıtlar da donanımsal sınır değerler olarak alınır. Yazmaç öbeğinin tüketeceği enerjinin hesaplanması harcanan devingen enerji ve durağan enerjinin toplamı ile yapılır. Durağan enerji kazanımı kullanılan bitler üzerinden silikon düzeyinde benzetim yapılabilen Cadence [28] şirketinin tasarladığı Virtuoso aracı ile hesaplanırken yine aynı şekilde hesaplanan okuma ve yazma enerjileri de devingen olarak harcanan enerjinin hesaplanması için kullanılır. Buna göre harcanan enerji değeri okuma sayısının okuma enerjisi ile çarpımının, yazma enerjisinin yazma enerjisi ile çarpılması ile bulunur. Her darlık grubu için de hesaplanacak değerlerin toplamı ile yazmaç öbeğinin harcadığı toplam enerji elde edilir.

Yazmaç Öbeği Bölümleme ILP Çözümü

Yazmaç öbeği bölümlenmesi sırasında problemin minimize edilmesi amaçlanan fonksiyonu enerji değeri olduğu gibi, problemin değişkenleri de fonksiyona bağlı olarak okuma, yazma sayıları ve bir okuma ya da yazma sırasında harcanan enerji değerleridir. Çizelge 4.1de gösterildiği şekilde hazırlanmış birleşimlerle okuma, yazma enerjileri ve sayıları alınmıştır. Ancak sadece veriler belirli noktaları göstermektedir. Bu nedenle ara değer kestirimi ile bitlerin ve satır değerlerinin tüm birleşimlerini gösteren bir tablo elde edilmiştir. Yazma ve okuma enerjileri sabit sayılar olmasına rağmen yazmaç öbeğine kaç defa yazılma ve okunma yapıldığına dair sayılar yanıltıcı olabilmektedir. Her deneme programının 1,5 milyar buyruk çalıştırması beklenirken bazı programların buyruk sayıları bu değerin altında kalmaktadır ve bu nedenle sonuçlar hesaplanırken daha çok buyruk yürüten programların sonuçlarda baskın çıkacağı görülmektedir. Herhangi bir programın sonuçlar üzerindeki baskısını kaldırmak için okuma ve yazma sayılarının normalize edilmesi yani her bir darlık grubunun okuma, yazma sayısının tüm okuma yazma sayılarının toplamına oranlanması gereklidir. Bu şekilde kullanılan değişken yazma sayısı değil yazma oranı olacaktır. Ayrıca programların okuma sayıları da yazma sayılarının bir katı olarak hesaplanmıştır. Bunun için okuma katsayısı değeri hiç değişiklik yapılmamış normal bir işlemciden alınan değerler ile hesaplanmıştır. Bu hesaplama okuma sayılarının yazma sayılarına oranlanmasıdır.

Yazmaç öbeği bölümlenmesi probleminde değişkenlerin normalize edilmesi aşağıdaki şekilde hesaplanır.

$$Yazma\ Oranı_i = \frac{\sum_{k=1}^{i} Yazma\ \ Sayısı_k}{\sum_{k=1}^{n} Yazma\ \ Sayısı_k} \qquad n = 64 \qquad (4.4)$$

Bir darlık grubu için o darlık grubuna ait yazma sayısı tüm darlık gruplarına ait yazma sayılarının toplamı yani toplam yazma sayısına bölünerek enerji hesabında kullanılacak olan yazma oranı elde edilir.

$$Okuma\ Katsayısı_j = \frac{\sum_{i=1}^{j} Okuma\ \ Sayısı_i \Big/ \sum_{i=1}^{n} Okuma\ \ Sayısı_i}{\sum_{i=1}^{j} Yazma\ \ Sayısı_i \Big/ \sum_{i=1}^{n} Yazma\ \ Sayısı_i} \qquad n = 64 \qquad (4.5)$$

Yazmaç öbeğinden değer okuma sayılarının hesaplanması sırasında yine değişiklik yapılmamış işlemcideki okuma ve yazma sayıları yerine okuma ve yazma oranları kullanılarak okuma katsayısı hesaplanır. Yazma oranı ile okuma katsayısının çarpımı ile o programın çalışmasındaki okuma sayıları hesaplanır. Dar değerlerin yazılacağı blokların satır ve sütun sayılarına göre gerçek okuma ve yazma sayısı değerlerine oturtulmuş olan bir tablo hazırlanmıştır. Tablonun boş olan kısımları ara değer kestirimi ile doldurulmuş haldedir. Aynı şey satır sayıları ve bit genişliğine bağlı olarak enerji değerleri için de hesaplanmıştır. Ancak boyutu değişen yazmaç öbeklerinin daha farklı sıklıklarda yazıp okuduğu gözlemlenmiştir. Örneğin 1 satıra düşürülmüş bir yazmaç öbeğine 10 satır olandan çok daha sık erişim yapılacaktır. Bu durumu da hesaplamaya katabilmek için her dar yazmaç öbeği bloğunun aktifliğini bildiren bir oran hesaplanır. Bu oran yazma sayısının değişiklik yapılmamış olan yazılma oranına bölünmesi ile bulunur.

$$Aktiflik\ Oranı_{(i,j)} = \frac{Yazma\ \ Oranı_{i,j}}{Yazma\ \ Oranı_{i,temel}} \qquad i = satır\ sayısı, j = bit\ sayısı \qquad (4.6)$$

Yazmaç öbeği probleminin değişkenleri ve minimize edilmesi gereken fonksiyonu aşağıda sırasıyla gösterilmiştir.

Çizelge 4.2. Blok Boyutu Değişkenleri

x	Blok 1 bit genişliği
m	Blok 1 satır sayısı
y	Blok 2 bit genişliği

n	Blok 2 satır sayısı
z	Blok 3 bit genişliği
k	Blok 3 satır sayısı

Çizelge 4.3. Enerji Değişkenleri

A	Blok 1 yazma sayısı
B	Blok 2 yazma sayısı
C	Blok 3 yazma sayısı
OK	Okuma Katsayısı
E_O	Okuma Enerjisi
E_Y	Yazma Enerjisi

Çizelge 4.2 ve Çizelge 4.3de sıralandığı gibi değişkenler enerji için ve blokların boyutu için olan değişkenler olmak üzere 2 grupta toplanmıştır. Değişkenlerin sağlaması gereken kısıtlar ve birbirlerine göre durumları ise aşağıda verilmiştir.

$$m + n + k = 256 \quad (4.7)$$

$$x < y < z \quad (4.8)$$

$$z = 64 \quad (4.9)$$

Tüm satır sayılarının toplamı temel alınan yazmaç öbeğinin boyutunu geçmeyecek şekilde 256 değerini bulmalıdır. Ayrıca grupların darlıkları aralarında fark olan değerler olarak seçilmelidir ve mutlaka en büyük darlık grubu veriyolu genişliğini sağlayacak şekilde yani bizim mimarimizde 64 bit olmalıdır.

Bir blok için enerji fonksiyonu ise toplam kullanılan yazma enerjisi ve okuma enerjisidir. Toplam yazmaç öbeği enerjisi ise 3 blok için hesaplanan enerjilerin toplamıdır.

$$E = N_Y . E_Y + N_O . E_O \qquad (4.10)$$

$$E_{Toplam} = \sum_{i=1}^{n} E_i \quad n = 3 \qquad (4.11)$$

Enerji değerlerinin belirtilen değişkenler üzerinden hesaplanması aşağıda gösterilmiştir. Buna göre temel alınan yazma sayıları, yazma enerjisini hesaplamak için, yazma sayısı üzerinden hesaplanan okuma sayıları ise okuma enerjilerini hesaplamak için kullanılmıştır.

$$E_{Toplam} = \sum_{i=1}^{n} (N_{Y,i} . E_{Y,i} + N_{O,i} . E_{O,i}) \quad n = 3 \qquad (4.12)$$

Enerji hesabına aktiflik oranı katıldığında ve gerekli sadeleştirmeler yapıldığında aşağıdaki son 2 fonksiyon elde edilir. Minimize edilmesi istenen fonksiyonlar da bunlardır.

$$E_{Toplam} = \sum_{i=1}^{n} (N_{Y,i}) . \left(E_{Y,i} + k_{prog,i} . E_{O,i}\right) \quad n = 3 \quad aktiflik\ katsayısız \quad (4.13)$$

$$E_{Toplam} = \sum_{i=1}^{n} (N_{Y,i} . k_{akt}) . \left(E_{Y,i} + k_{prog,i} . E_{O,i}\right) \quad n = 3 \quad aktiflik\ katsayısı\ ile \quad (4.14)$$

Bizim ILP problemimizde uzayımız satır ve sütunlara bağlı olarak çıkarılmış olan enerji ve kullanım matrislerinin birleşimi ve bulunması gereken değişkenler de minimize edilmesi istenen enerji fonksiyonunda dolaylı olarak dahil olan enerji ve kullanım değerlerini etkileyecek satır ve sütün sayılarıdır. Bu haliyle normal bir ILP probleminden çözümü çok daha kolaydır. Yazmaç öbeği bölümlenmesi sırasında ortaya çıkan bu basit ILP problemi veri kümesinin küçüklüğü ve kısıtların azlığı

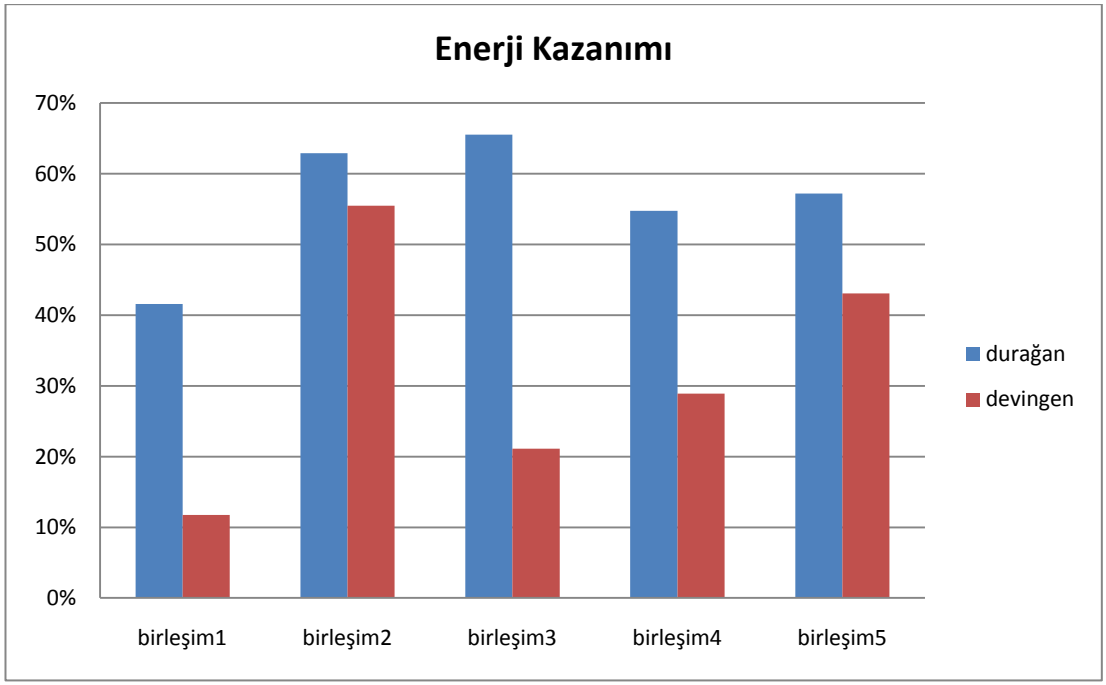
nedeniyle ILP çözümüne yönelik metotlarla değil, Java dilinde yazılan bir deneme programı ile çözülmüştür. Bu program enerji ve kullanım istatistiklerini dışarıdan aldıktan sonra tüm olasılıkları hesaplayarak en az enerjiyi sağlayacak satır - sütun birleşimlerini sunmuştur. İşlemcinin benzetiminin aldığı süreye kıyaslandığında deneme yapacak olan programın birkaç dakikalık süresi sonuçlara ulaşmayı hızlandırır. Hesaplamalar yapılırken gerçekte alınan en küçük değerlerin üzerinde ve aralarında en azından 10 bit fark bulunan darlık değerleri grupları seçilmiştir. Buna göre en tasarruflu enerji kullanacak olan yazmaç öbeği bloklarının aşağıdaki birleşime sahip olması gerekir.

Çizelge 4.4. Sabit Bölümleme Satır Sütun Birleşimleri

	Blok 1		Blok 2		Blok 3	
Yazmaç Öbeği Birleşimleri	Satır Sayısı	Bit Genişliği	Satır Sayısı	Bit Genişliği	Satır Sayısı	Bit Genişliği
1	85	56	85	60	86	64
2	85	16	85	34	86	64
3	85	10	85	32	86	64
4	50	16	65	32	141	64
5	56	16	76	32	124	64

Satır ve sütun birleşimleri hesaplanırken, her ne kadar ILP ye başvurulmuş olsa da hesaplamalarda ara kestirim kullanıldığı için yanlışlar doğabileceğinden ötürü darlıkların kullanımında gözlemlenen 16 ve 34 bit darlığa sahip bloklar da denenmiştir. Yine benzer bir şekilde satır sayıları eşitlenmiş olarak sabit tutulup ILP çözümü ile hangi darlıkların kullanılması gerektiği incelenmiştir. Buna göre çıkan sonuçlardan 5 tanesi seçilerek gerçek benzetimi yapılmış ve enerji tasarrufu sonuçları elde edilmiştir. Buna göre sabit olarak yazmaç öbeğinin darlık gruplarına bölünmesi ile ortalama %55 e kadar yazmaç öbeği enerjisinden tasarruf etmek mümkündür.

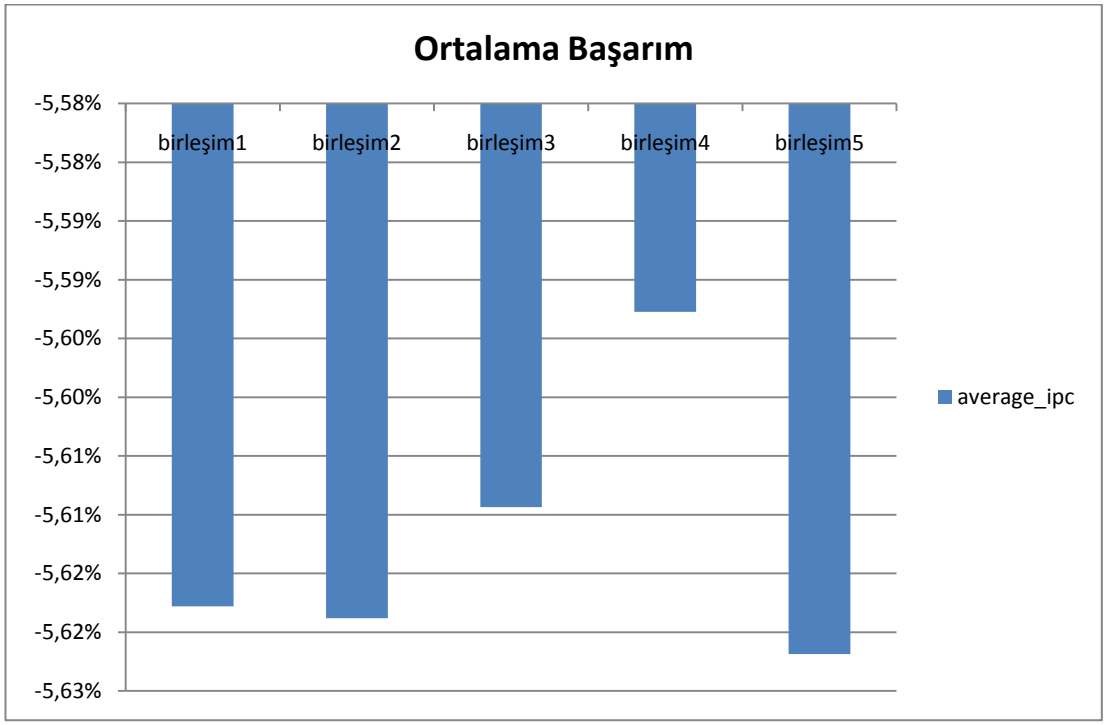
Yine dar yazıldığı için kullanılmayacak olan bitlerin yazmaç öbeği satırlarında gösterilmemesi ile alandan da ortalama %24,5 kazanılmıştır.



Şekil 4.2 – Sabit Bölümleme ile Enerji kazanımı

Şekil 4.2de yalnızca ortalama olarak kazanılan enerjiler gösterilmiştir. Deneme programının durumunu gösteren grafikler Bölüm 5 de anlatılan benzetim ve sonuçlar kısmında değinilmiştir.

Enerji tasarrufu sağlayacak olan bu yöntem yazmaçların atanması yani yazmaç yeniden adlandırma sırasında da beklemelere neden olacaktır ve başarımı düşürecektir. Bir çevrim başına düşen buyruk sayısının değişimi ile işlemci başarımının da nasıl etkilendiği gözlemlenebilir. Yine bu grafikte verilen değerler ortalama değerlerdir. 5.Bölümde her bir deneme programının ayrıntılı değişimleri verilecektir.

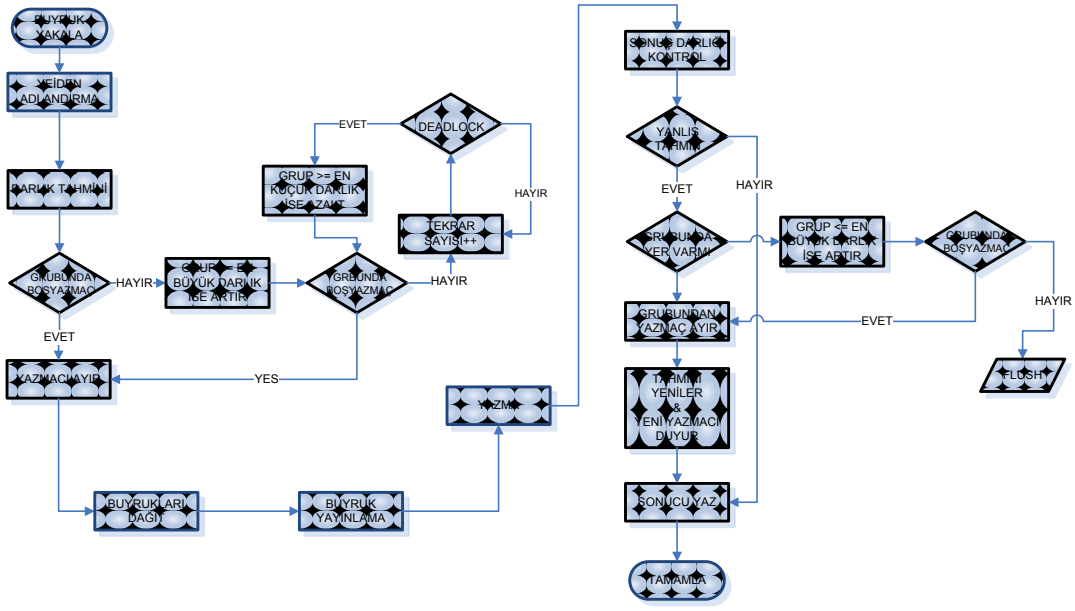


Şekil 4.3 – Sabit Bölümleme Satır Sütun Birleşimleri ÇBB Değişimi

4.1.2. Yazmaç Ataması

İşlemcide yazmaç öbeğinin bölümlenmesi ile birlikte yeni gelen buyruklara hangi sırada yazma yeri verileceği sorunu da ortaya çıkar. Normal durumda her biri eş olan ve aynı sürede erişilen yazmaç öbeği satırlarından sıradaki boş olan yazmacın verilmesi [3] ile ilerleyen yeniden adlandırma evresi, bu yeni yazmaç öbeği düzeninde farklı olarak işlenecek değerlerin darlıklarına göre yerleştirilmesi gerekir. Yazmaçların atanması sırasında izlenecek olan yöntem işlemci başarımında belirleyici olacaktır. Yeniden adlandırma aşamasında buyruğun türüne ve yerine göre buyruğun çıkaracağı sonucun hangi darlıkta olduğu tahmin edilir. Tahmin edilen darlık değeri buyruğun yerleşmek istediği yazmaç öbeğini gösterir. Yazmaçların ataması yapılırken amaç, mümkün olduğunca doğru tahminler yapmak ve yazmaçları kendi gruplarına yerleştirmektir. Bu şekilde doğru yere yerleşen buyruklar boş yere yazmadıkları bitlerin harcayacağı enerjiyi tasarruf etmiş olacaklardır. Sonuç darlıklarının doğru tahmin edilmesi problemi 3.Bölümde gösterildiği üzere büyük ölçüde halledilmiş durumdadır. Yeniden adlandırma birimi buyruklara yer verirken

en az %90 tam doğru ve yaklaşık %5 olması gerekenden büyük yer vermektedir. Bu oran başarımın birinci koşulu tahmin doğruluğu için iyi bir değerdir. Yanlış tahmin durumları ise yazma aşamasında düzeltilerek, düzeltilmiş yazmaç adresi yayın kuyruğuna bildirilmektedir. Tahmin edilen yazmacın yanlış olması durumunda yapılan işlem de birkaç çevrim boyunca yayın kuyruğunu meşgul edecektir. Hangi yazmaçtan kaynak değerlerin okunacağını gösteren etiketlerin yenilenmesi sırasında normalde yayın kuyruğuna girecek değerler yerine yeni yazmaç adresleri yayın kuyruğu içerisine alınır. Bu durumda da yanlış tahminler de birkaç çevrim kayba neden olacaktır. Ancak tahminlerin doğruluk yüzdeleri yeterince yüksek olduğu için bu birkaç çevrim önemsenmeyecek kadar az bir değer olacaktır. Yazmaç öbeği sabit olarak bölündüğü zaman işlemcinin buyruklara sonuç yazmacı ataması aşağıdaki akış şemasında gösterildiği gibidir.



Şekil 4.4 – Sabit Yazmaç Blokları Yazmaç Ataması Akışı

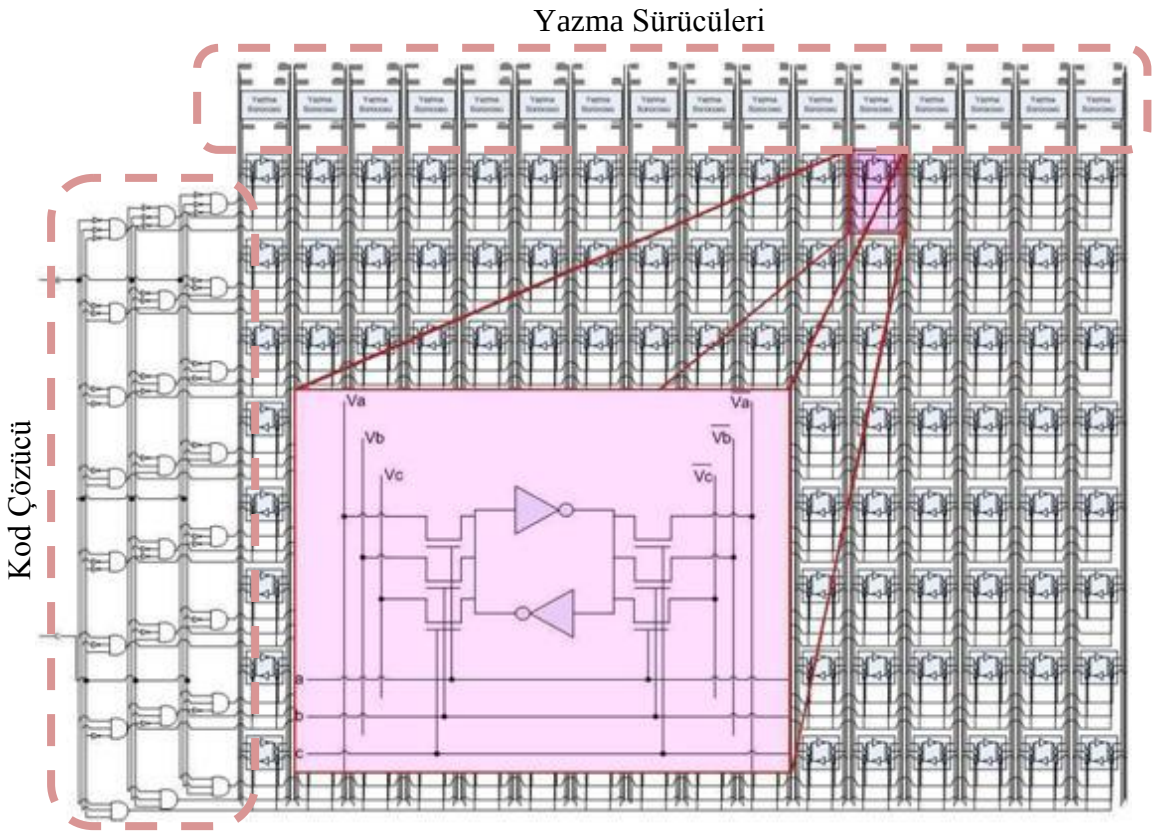
Buyruk bellekten alındıktan sonra yeniden adlandırma birimine getirilir ve burada buyruğun üreteceği sonuca dair tahmin yapılır. Bu tahmin buyruğun bulunduğu konuma ve yaptığı işe bağlı olarak yapılır. Hesaplanacak sonucun darlık tahmini

yapıldıktan sonra yazmacın yerleştirilmesi beklenen yazmaç öbeğinde boş yer olup olmadığı kontrol edilir. Daha önce de belirtildiği gibi amaç her buyruğa kendi grubundan yer vererek enerji tasarrufu sağlamaktır. Bu nedenle ilk aranacak yazmaç öbeği kendisine ait olandır. Eğer yer yoksa darlık grubu yükseltilerek yani daha geniş bir yazmaca yazması kabul edilerek tekrar yer aranır. Yer bulununcaya kadar giderek genişleyen yazmaç öbekleri aranır. Hiçbirinde yer bulamadığı durumda bir sonraki çevrimde tekrar yazmaç bulmak için yeniden adlandırma birimine gelir. Yazmaç öbeğinde yer bulunamaması durumu normal bir işlemcide de her zaman gerçekleşebilir. Amaç yeniden düzenlenmiş yazmaç öbeğinin bu sayıyı artırmamasıdır. Belirli sayıda çevrim boyunca eğer buyruk yazmaç bulamamışsa bu sefer daha dar yazmaçlardan birisi verilerek yeniden adlandırma aşamasındaki kördüğüm önlenir ve ilerleme sağlanır. Buyruk artık yer bulduktan sonra sırasıyla buyruk dağıtma ve yayınlama aşamalarında geçerek yazmaç öbeğine yazılmak üzere yazma aşamasına gelir. Yazma aşamasında buyruk hesaplanan sonucunu kendisi için ayrılmış olan yazmaca yazacaktır. Bu aşamada buyruk için yapılan doğruluk tahmini tekrar kontrol edilerek hatalı tahminler yenilenir ve yazmaç öbeğinde aynen yeniden adlandırma aşamasında olduğu gibi yer bulmaya çalışılır. Ancak eğer yazmaç bulunamazsa yazma aşamasına kadar girmiş olan bir buyruğun düzeltilmesi yazma aşamasını gereksiz yere meşgul edeceği için ve yapılması yeni donanım gerektirdiği için anlamsızdır. Bunun yerine yer bulunamaması halinde veriyolu boşaltılır. Eğer doğru gruptan yer verilmişse yani tahmin doğruysa yeniden adlandırma aşamasında belirtilen yazmaca sonuç yazılır.

4.1.3. Yazmaç Öbeği Tasarımı

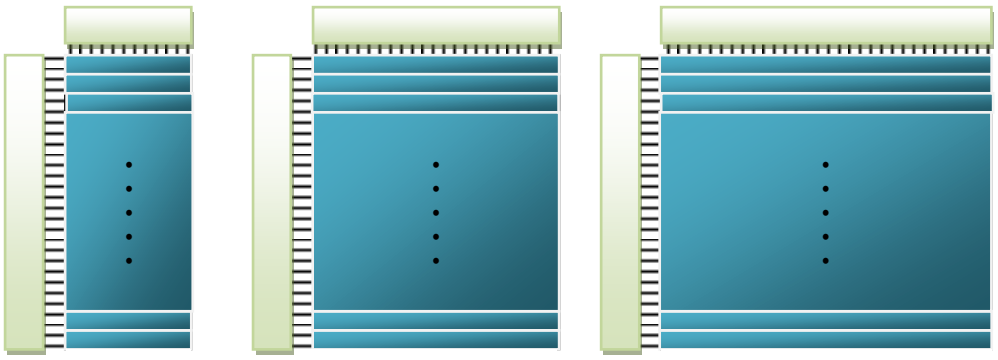
Yazmaç öbeği SRAM hücrelerinden oluşan bir matris şeklinde tasarlanmıştır. SRAM hücreleri veri tutmak için en pahalı ancak erişim süresi açısından en hızlı veri tutucu birimlerdir. Bu nedenle veriyolunun en hızlı erişmesi gereken bellek olan yazmaç öbeği yapımında da SRAM kullanılır. Şekil 4.5 bir seferde tek bir değerin yazılabileceği 16 bitlik veriler tutabilen 8 yazmaçlık bir yazmaç öbeğidir. 8 yazmacın ifade edilmesi için ($2^3 = 8$) 3 bite ihtiyaç vardır. Bu 3 bitin hangi satırı ifade ettiği ise yazmaç öbeği satırları önüne konulmuş olan bir kod çözücü ile belirlenir. Hangi satırın seçildiği belirlendikten sonra her bir bit için o satıra yazılmasını sağlayan

yazma sürücüleri devreye girer. Doğru bitlerin etkinleştirilmesi ile yazma ve okuma işlemleri yapılır. Yazmaç öbeğinin sabit olarak 3 bloğa bölünmesinde ise bu yazmaç öbeği bloklarından 1 tane büyük yerine 3 tane farklı boyutlarda küçük yazmaç öbekleri olacaktır.



Şekil 4.5 – Yazmaç Öbeği 16 bitlik 8 yazmaç

Yazmaç öbeğinin bölümlenmesi durumunda ortaya çıkan Şekil 4.6daki yapıda yerden tasarruf edilmesine rağmen her bir blok için ayrı bir kod çözücü ve her bit için ayrı yazma sürücüsü gerekli olacaktır. Ancak bunlar da hesaba katılmış olmasına rağmen farklı yazmaç öbekleri tasarlama fikrinin getireceği tasarruf %55 olacaktır.



Şekil 4.6 – Sabit Bölümlenmiş 3 Farklı Darlıktaki Yazmaç Öbeği

Yazmaç öbeği tasarımı yapılırken işlemcinin tüm değişkenleri de doğal olarak göz önünde bulundurulur. Bunlardan en önemlisi ise veriyoluna bir seferde getirilen buyruk sayısıdır. Birden fazla buyruğun veriyolunda paralel ilerleyebilmesi için yazmaç öbeğinin de aynı anda o kadar sayıda yazma ve okuma işlemini yürütebilmesi gereklidir. Bu nedenle genelde işlemciler birkaç girişe ya da erişim noktasına sahip olacak şekilde tasarlanırlar. Aynı şekilde kullanılan mimaride de 4 tane buyruk aynı anda veriyoluna alındığı ve bir buyruk 2 değer okuyup 1 sonuç yazdığı için 4*3 = 12 girişli yazmaç öbekleri bulunmaktadır.

4.2. Değişken Bölümleme

Yazmaç öbeğinin değişken bölümlenmesi, yazmaç öbeğinin kapladığı alanı değiştirmeden belirlenmiş sayıda bit dizilerinin kapatılıp açılması ile programın yazmaç ihtiyaçlarını kolayca karşılarken hem de enerji tasarrufu sağlanmasıdır. Yazmaç öbeği birkaç darlık grubuna bölünür ve yazmaç öbeği satırları istenildiği zaman açılacak ve kapatılabilecek şekilde tasarlanır. Bu şekilde kapatılmış olan bitlerden hem durağan hem de okuma yazma enerjisi olarak kazanç sağlanacaktır. Ayrıca programın ihtiyaçlarına göre yazmaçların açılması ve tüm yazmaç öbeğinin kullanılabilir olması da sağlanır. Ancak yazmaç atanması işlemi yine durağan bölümlemede olduğu gibi işlenecek değerin ait olduğu yazmaç öbeği grubundan

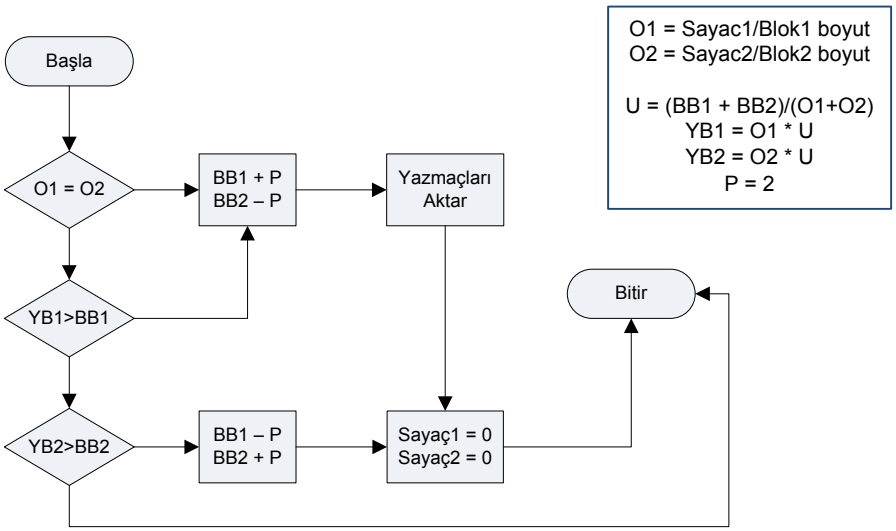
verilmek zorundadır. Ayrıca grupların boyutlarının değiştirilmesi için belirli aralıklarla programın gidişatı incelenmeli ve buna cevap verecek yeni boyutlar ayarlanmalıdır. Fiziksel olarak yazmaç öbeğinin değişken bölümlenmesi sırasında alandan kazanç sağlanmayacaktır. Esnekliğin sağlanması için yazmaç öbeğinin tümü işlemci içinde bulunur. Yazmaç öbeğinin satırları ise karar verme devresinin göndereceği değerler yardımıyla sürülür. Bu şekilde devingen olarak darlık gruplarının büyüklüklerinin değiştirilmesi için gereken enerji azaltılmış olur.

4.2.1. Bölümleme Yöntemi

Yazmaç öbeğinin değişken bölümlenmesi sırasında da bir başka problem ortaya çıkmaktadır. Sabit bölümlemedeki değişmesi mümkün olmayan bloklar problemi değişken bölümlemede değiştirilmesi mümkün olan ancak hangi sırayla ve ne şekilde değiştirilmesi daha çok tasarruf sağlayacağı bilinmeyen bloklar sorunu haline gelmiştir. Değişken yazmaç öbeği bölümlenmesinde satır sayıları ihtiyaca göre değiştirilebilir ancak yatayda da kapatılıp açılacak bit sayısının tasarım sırasında bilinmesi gereklidir. Çünkü dikey olarak yazma sürücülerinin kontrolünü yapacak anahtarlar bit sayısına göre yerleştirileceklerdir. Bu durum da darlık gruplarının tasarım sırasında belirlenmesini zorunlu kılar. Değişken bölümleme için de 3 farklı darlık grubu kullanılmıştır. Bunlar en sık kullanılan darlık değerleri olan 16, 34 ve 64 bitlik değerlerdir. Darlıkların belirlenmesi ile birlikte bir diğer değişken olarak devingen değişim sırasında bir seferde kaç satırın hareket ettirileceği ortaya çıkar. Yazmaç öbeği tasarımı sırasında her iki satırın bir tane ortak voltajı kullanmasından ötürü bir seferde açılıp kapatılacak birimlerin 2 ya da 2 nin bir katı olması işlemleri pratikleştirecektir. Devingen olarak satırları dağıtacak olan algoritmanın da [19] çalışmasında olduğu gibi belirli periyotlarda kullanımları kontrol etmesi gereklidir. Bunun için seçilecek periyot süresi de bir başka değişken olarak problemde karşımıza çıkmaktadır. Bunların dışında yer vermek için kullanılacak yöntem de başarım üzerinde etkili olacaktır. Belirli istatistikler toplandıktan sonra mevcut duruma göre karar verilmesiyle birlikte yazmaç öbeğinin satır sayıları değiştirilecektir. Devingen olarak yazmaç öbeğinin boyutunun değiştirilmesi için temel olarak bir seferde kaç satırın değiştirileceği, ne kadar zaman aralıkları ile değiştirme yapılacağı ve kullanım istatistikleri gerekli olacaktır.

4.2.2. Blokların Devingen Değiştirilmesi

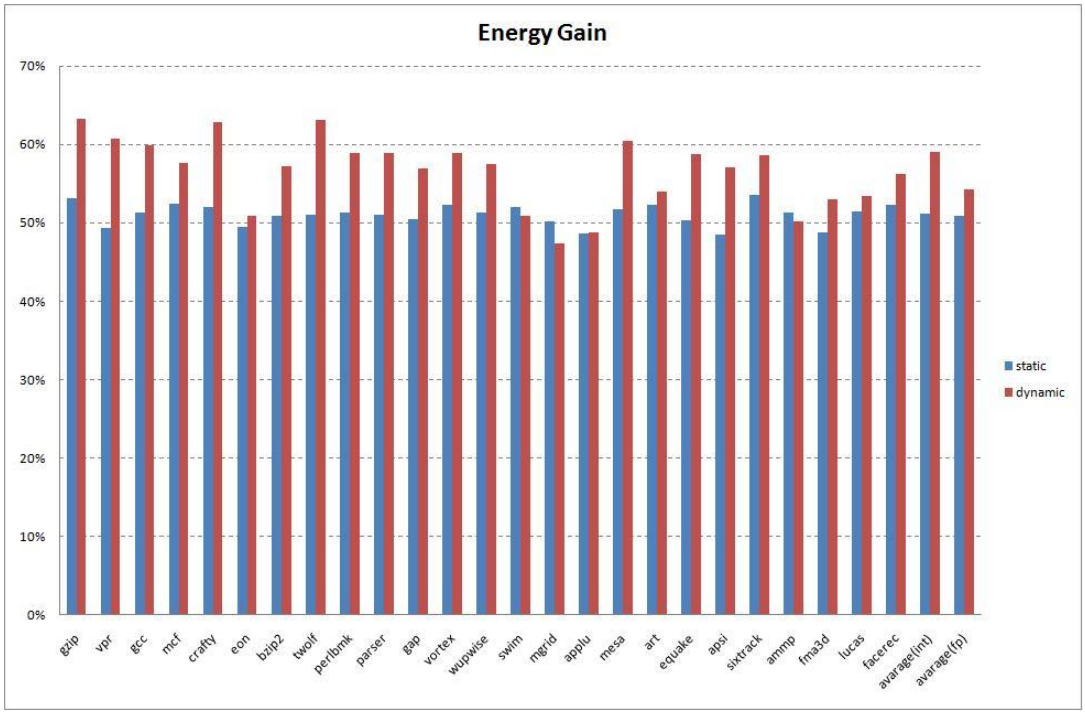
Yazmaç öbeği değişken bölümlendiğinde yine 3 farklı blok bulunacaktır. Blokların büyüklüklerinin devingen değiştirilmesi ise satırların açılıp kapatılması yani darlıklarının değiştirilmesi ile mümkün olacaktır. Devingen değiştirme işlemi belirli aralıklarla toplanmış olan istatistiklerin incelenmesi ve mevcut duruma göre yapılması gereken değişikliklerin belirlenmesidir. Değişiklikler yazma aşamasında belirlenir ve yine aynı yazma aşmasında yapılır. Yazmaç öbeğinin açılıp kapanması işlemi bir çevrim sürdüğü için bir sonraki aşamada yeni yazmaç bloğu boyutlarıyla işlemler devam ettirilir. Devingen yer verme kararı alınması için program işlediği sırada bazı kullanım istatistikleri tutulması gereklidir. Daha çok ihtiyaç olan gruptan daha fazla yazmaç açılması gereklidir ve bu da ancak nasıl bir darlık kullanım grafiği olduğu ile bulunabilir. Buyruklar yeniden adlandırma aşamasına geldiklerinde yazmaç öbeğinden sonuçlarını yazmak için yer ayıracaklardır. Bu sırada dar değer tahmini sonucu dar değer üreteceği düşünülen buyruklara istediği darlıktaki yazmaç öbeği bloğundan yer ayrılır. Buyruklara yazmaç atama sırasında buyruğun kendi grubundan yer bulamaması bölümlemenin başarısız olduğu anlamına gelir. Eğer buyruk yer bulamazsa aynen sabit bölümlemede olduğu gibi bir üst gruptan aramaya devam ederek yeniden adlandırma işlemini yapar. Ancak yazmacın yer bulamaması o yazmaç bloğuna ait bir sayacı artırır. Bu şekilde tüm programın çalışması boyunca hangi buyruk girmek istediği darlık grubundan yer bulamadıysa bu değerler sayaçlarda gözlemlenebilir. Belirli periyotlarda bu sayaçlar kontrol edilerek nasıl bir bölümleme yapılacağına tekrar karar verilir. Periyot olarak seçilen değer kullanılan mimaride 1024 olarak seçilmiştir. Bir periyotta 1. ve 2. darlık grubu arasında, sonraki periyotta da 2. ve 3. Darlık grupları arasında yazmaç sayısı dengelenmesi işlemi yapılır. Her bir yer değiştirme işleminde bir “parça” diğer öbeğe geçirilir. Bu şekilde özyinelemeli olarak ihtiyacı fazla olan gruba tekrar tekrar yer verilir. Bir parça değeri olarak tasarımda 2 satır seçilmiştir. Her kontrol sonunda 2 yazmaç grubunu değiştirebilir.



Şekil 4.7 – Devingen Yer Verme Kararı Akış Şeması

Kullanım istatistikleri ve mevcut durumun ışığında karar verilecek olan aşama yazma aşamasıdır. O anda yazma aşamasında olan ve yazmaç öbeğini kullanan değerler işlemlerini bitirdikten sonra yazmaç öbeğinin yeni durumu için karar verilir. Kendi yazmaç bloğunda yer bulamayan buyrukların sayılarını gösteren sayaçlar blokların o andaki boyutları (BB1, BB2) ile normalize edilir. Bunun sebebi 1 satır olduğu halde 200 tane buyruğu kaçıran bir bloğun 100 satır olduğu halde 200 satır kaçıran bloğa oranla yazmaca ihtiyacının daha fazla olmasıdır. Yani yazmaç öbeğine girememe oranı daha fazladır. O anda karşılaştırılmakta olan 1. ve 2. grubun buyruk kaçırma oranları (O1, O2) üzerinden 2 grubun toplam yazmaçları paylaştırılır. Buyrukları kaçırma oranlarına göre yazmaç öbeğinin bu kısmının yeni dağılımı hesaplanır. Bu noktadan sonra istenirse yeni boyutu sağlayacak şekilde yazmaçlar gruplar arasında transfer edilebilir. Bizim tasarımımızda yalnızca 2 yazmaç yer değiştirecektir. Blokların oranları karşılaştırılırken eşit olmaları ve sıfır olmaları durumları da göz önünde bulundurulmuştur. Eğer değerlerin birisi 0 ise sıfıra bölünme gerçekleşeceği için sorun yaratacaktır, bu durumda değere etkilemeyecek kadar küçük bir sayı eklenerek sorun ortadan kaldırılır. Oranların eşit olması durumunda ise blokların eşit olarak ikiye bölünmesi söz konusu olacağından bu durumda daha dar olan gruba yazmaç aktarılır. Ancak bu durumda en geniş olan grubun fazla yazmaç kaybetmesi

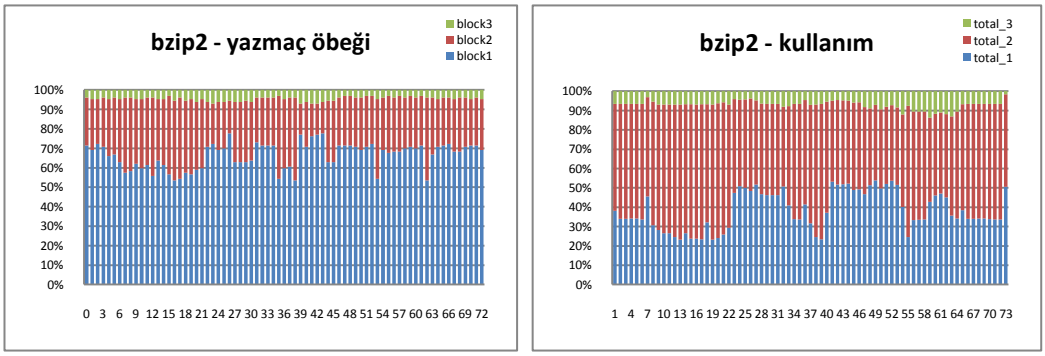
söz konusu olacağından eşit oran durumunda belirli sayıda boş yazmaç kalması da gözetilir. Tasarımda bu sınır 10 yazmaç olarak belirlenmiştir. Daha geniş olan grupların yazmaç bulmakta zorlanması başarımı etkileyeceği hatta programın çalışmasını durdurabileceği için 2. veya 3. darlık grubundan yani veriyolu genişliğinde olan gruptan yazmaç bulunamadığı her durumda da kontrol periyodu gelmemiş olsa bile devingen değiştirme kararı verme algoritması çalıştırılır. Bu şekilde yazmaçlar dağıtıldığı zaman kullanılmayan yazmaçlar en dar olan gruba geçerler ve başarım da en alt seviyede etkilenecek şekilde yazmaç öbeğinde yüksek enerji tasarrufu sağlanmış olur. Aşağıdaki grafikte durağan ve devingen enerjiden ne kadar kazanç sağlandığı her bir deneme programı için gösterilmiştir.



Şekil 4.8 – Değişken Bölümleme ile Enerji Kazanım Grafiği

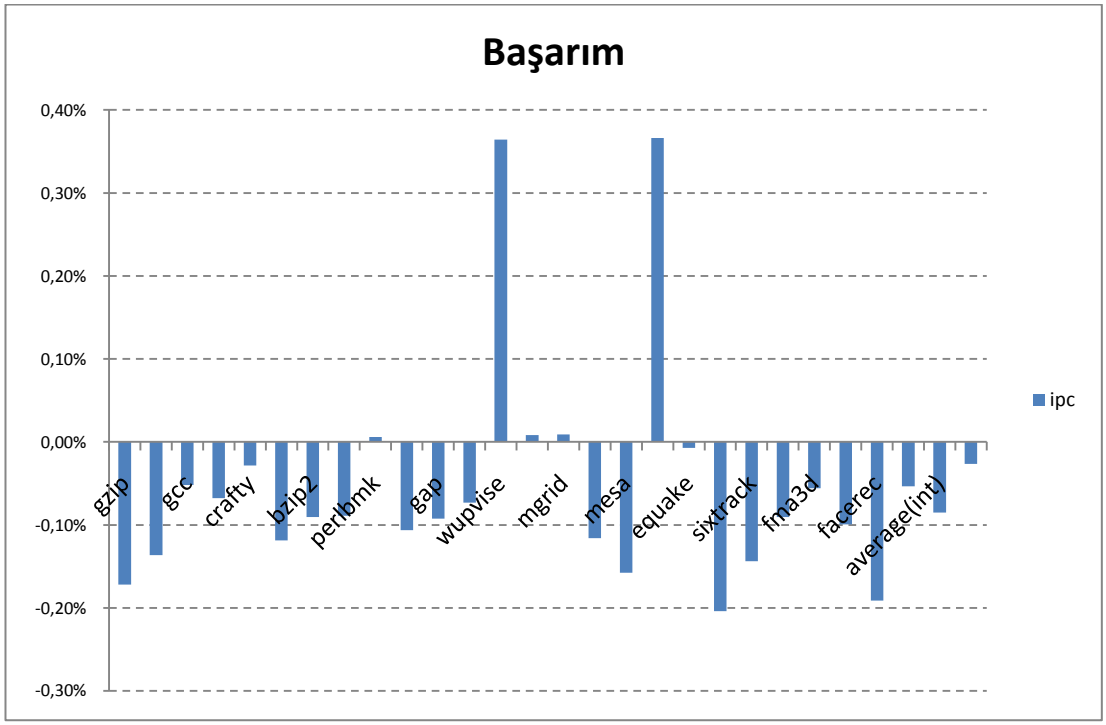
Grafikte enerji tasarrufu programların sonuçlarına göre gösterilmiştir. Buna göre en az tasarruf sağlayan applu ve mgrid programlarında bile yaklaşık %50 durağan ve devingen olarak enerjiden kazanç sağlanmıştır. Programların çalışması sırasında 10 milyon çevrimde bir alınan değerlerle devingen olarak da enerjiden ne kadar kazanç

sağlandığı 5. Bölümde gösterilecektir. Yaklaşık %50 tasarruf edilmesi devingen değiştirme yönteminin iyi çalıştığı anlamına gelmektedir. Devingen yer verme kararının doğruluğunu test etmek için 10 milyon çevrimde darlık gruplarının değişiklik yapılmamış işlemcide dağılımı ve yazmaç öbeğinin o andaki blok dağılımları incelenmiştir.



Şekil 4.9 – Değişken Yazmaç Öbeği Dağılımı ve Dar Değer Kullanımı

Şekilde yalnızca bzip2 programı için grafikler gösterilmiştir. Diğer programlar için çizilmiş olan grafikler 5.bölümde gösterilecektir. Şekilde de görüldüğü gibi programın işleyişi boyunca ilk anlarda 2. Gruptan gelen değerlerin sayısı arttıkça yazmaç öbeğindeki 2.grubun boyutu artmıştır. Ayrıca 3.gruptan çok az sayıda değer geldiği için en çok enerji tüketen 3.gruptan en az sayıda yazmaç bulunmaktadır. Birinci grubun boyutunun bu kadar fazla olmasının nedeni ise kullanılmayan yazmaçların en dar yani en az enerji tüketen 1.gruba geçiriliyor olmasıdır. Bu şekilde programın tüm yazmaç ihtiyaçlarına cevap verecek şekilde tasarlanmış ve yazmaç bulmadığı durumlarda da yeniden adlandırma aşamasındaki sorunları çözümleyip programı ilerleten bir yapıda başarımdan düşüş neredeyse yok denecek kadar azdır.



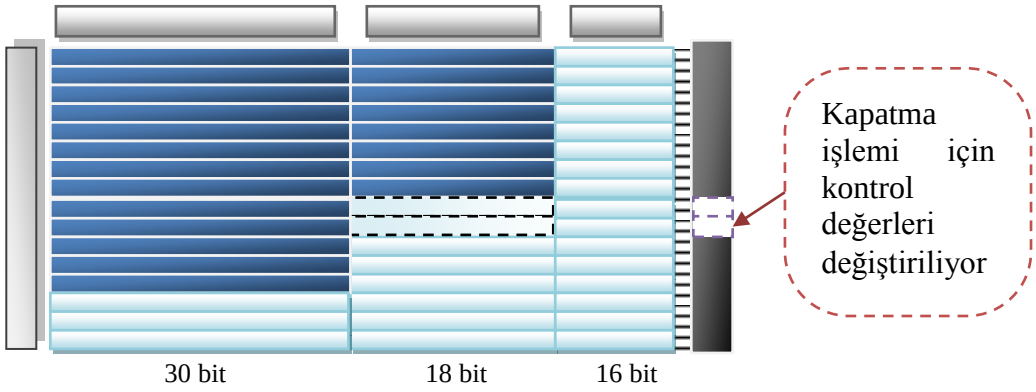
Şekil 4.10 – Çevrim Başına Buyrukta oluşan Başarım Kaybı

Ortalama %0,2 olan başarımdan yani çevrim başına işlenen buyruk sayısındaki kayıp önemsenmeyecek bir miktardır. Bazı programlardaki ÇBB artışı benzetimliğin hata payı içerisindedir. Zaten bu kazanım da yaklaşık % 0,4 olup önemsenmesi gerekli değildir.

4.2.3. Yazmaç Öbeği Tasarımı

Yazmaç öbeği blok boyutlarının artması ve azalması satırların dar değer için kapatılmış olan kısımlarının açılması ve kapatılması ile gerçekleşir. Satırın açılması dar olan bir gruptan daha geniş bir gruba geçiş yapması, satırların kapatılması ise geniş bir gruptan dar bir gruba geçişi ifade eder. Yazmaçlar açıldığı sırada yapılan işlem yalnızca dar olmak üzere kapatılmış bit grubundan 2 satırın etkinleştirilmesidir. Yazmaçların kapatılması sırasında ise eğer daha dar gruba geçecek olan yazmaçlar dolu ise kapatma işleminin yapılması mümkün değildir. Çünkü yazılmış olan değerin kullandığı bitler kapatılacak ve değerin değişmesine neden olacaktır. Bu yüzden kapatılacak olan satırlarda değer bulunuyorsa bu değerlerin gelecek bir kaç çevrim içerisinde yazmaç öbeğinin başka satırlarına taşınması gerekir. Taşıma işlemi

yeniden adlandırma aşamasında taşınacak yazmaçlar için yer ayrılması ile başlar ve birkaç çevrim sonra geçilebilecek olan yazma aşamasında taşınan yazmaçların yazılması ile son bulur. Taşıma işlemi gerçekleştikten sonra bloğun satırları kapatılarak dar olan gruba geçirilir. Yazmaç öbeği satırlarının açılıp kapatılması değişken bölümleme kontrolü yapan devrenin çıkış olarak vereceği her satırın durumunu bildiren bitler ile gerçekleştirilir.

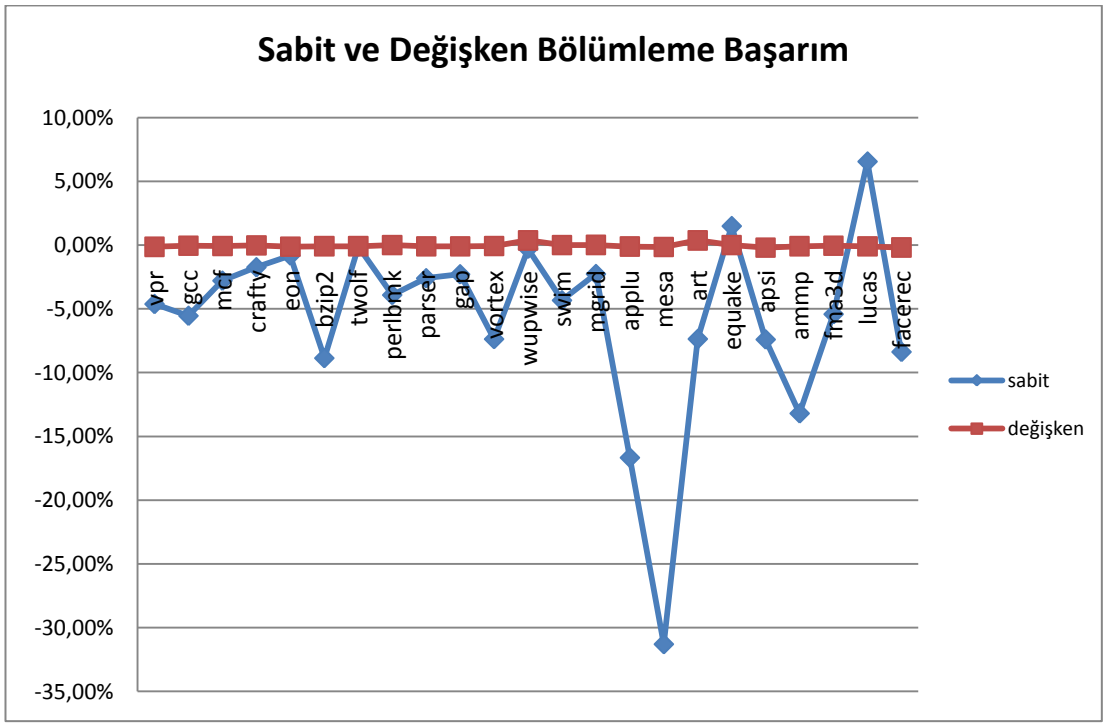


Şekil 4.11 – Değişken Bölümlenmiş Yazmaç Öbeği

Yazmaç öbeğinin devingen boyut değiştirmesi için değişim kararını veren kontrol biriminden tüm yazmaçların durumunu bildirecek bir bit dizisi alınır ve bu dizi yardımıyla satırlar sürülür. Örneğin satırın durumu 1.bloktan 2.bloğa geçtiğini gösteriyorsa 16 bitin yanındaki 18 bitlik kısım 1 bloktaki son 2 satır için etkinleştirilir. Kontrol biriminin işlemci üzerine getireceği yük işlemcide hali hazırda var olan bölme ve çarpma birimlerinin kullanılması ile azaltılmış durumdadır. İşlemciye eklenmesi gerekenler ise istatistiklerin tutulduğu sayaçlar ve satırları kontrol edecek olan kontrol bitleridir. Ancak işlemcinin yazmaç öbeğinden kazanacağı enerjiye oranla önemsenmeyecek kadar azdırlar. Değişken olarak bölümlenmiş yazmaç öbeği üzerinde temel yapısını değiştirecek müdahale olmadığı için kontrol birimi devreden çıkarıldıktan sonra normal veriyolu genişliğinde değerler tutan bir yazmaç öbeği olarak kullanılabilir.

4.3. Sabit ve Değişken Bölümlemenin Karşılaştırılması

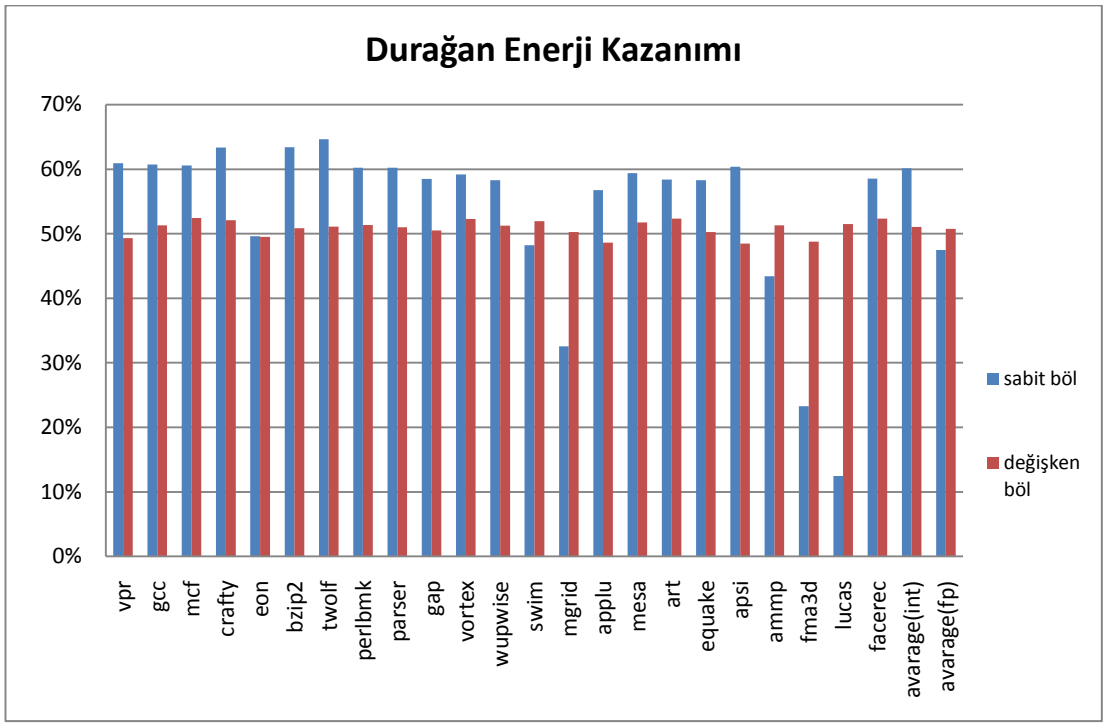
Yazmaç öbeğinin sabit olarak ya da değişken olarak bölümlenmesi ile deneme programları üzerinden görüldüğü kadarıyla çok yakın değerlerde tasarruf edilmiştir. Ancak başarımın değişken bölümleme yönteminde daha yüksek olduğu ÇBB değerlerini gösteren başarım grafiklerinden de gözlemlenebilir.



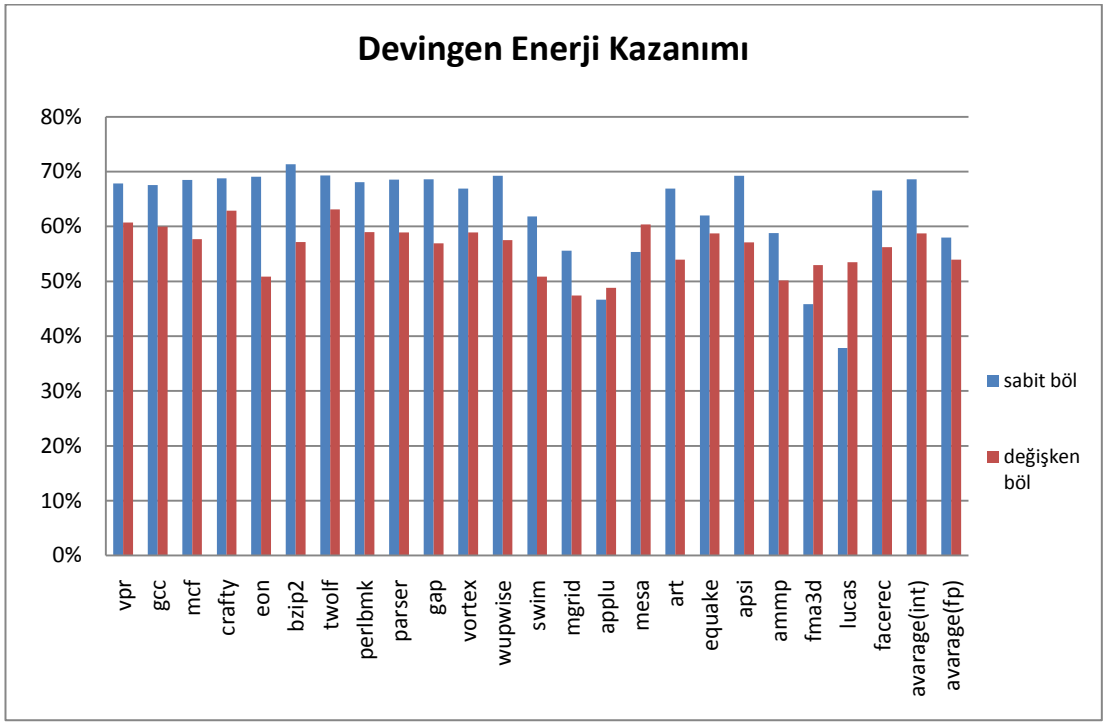
Şekil 4.12 – Sabit ve Değişken Bölümleme ÇBB Karşılaştırması

Yazmaç öbeği sabit olarak bölündüğünde bazı programlarda başarım neredeyse hiç etkilenmezken bazılarında ise %30 ye varan düşüşler gözlemlenmiştir. Doğal olarak bunun sebebi her programın yazmaç ihtiyaçları aynı olmadığı halde öbeklerin esnek bir şekilde programa uyum sağlayamamasıdır. Sabit ve değişken bölümlemenin karşılaştırılması için eşit bölümlenmiş bir yazmaç öbeği kullanılmıştır. Değişken olarak dağıtılan yazmaçların da ilk dağılımı eşit bölümlemedir. Daha sonra programın ihtiyaçlarına göre yazmaç öbeği farklı dağılımlar gösterecektir. Hatta bazı durumlarda tekrar eşit bölümlemeyi de bulabilir. Sabit bölümleme başarım açısından düşüşe sebep olmasına rağmen enerji kazanımı bakımından dinamiğe göre daha iyi

bir sonuç vermiştir. Sonuçta aynı darlıklarda bulunan birimlerden kapatılmış olanlara göre tamamen iptal edilmiş olanların daha az enerji tüketeceği rahatça gözlemlenebilir. Ancak grafikten de görülebileceği gibi aradaki fark çok azdır.



Şekil 4.13 – Sabit ve Değişken Bölümlemenin Sızıntı Enerjisinden Kazanımları



Şekil 4.14 – Sabit ve Değişken Bölümlemenin Devingen Enerji Kazanımları

Devingen açma ve kapama işleminde var olan yazmaç öbeğinin çevresine satırların kontrolünü sağlayacak ek devreler yerleştirilir. Herhangi bir istenmeyen durumda enerji tasarrufunun kapatılıp tüm yazmaç öbeğinin açılması yalnızca devrenin yazmaç öbeği ile bağlantısının kesilmesi ile mümkün olur. Bu şekilde yazmaç öbeğinin bir tasarruf eden bir de normal modu olur ve istenildiği zaman geçiş yapılabilir. Ancak yazmaç öbeği sabit olarak öbeklere ayrıldığında hiçbir durumda müdahale etmek mümkün olmayacaktır. Program ne kadar ihtiyacı olursa olsun ne yazmaç öbeğinin tümünü ne de biraz daha geniş olan birkaç tane yazmaç kullanamayacaktır. Ayrıca yine durağan bölümleme sırasında yazmaç atama yöntemi dolayısıyla en geniş olan gruba hiçbir yere giremeyen yazmaçların birçoğu gelmekte ve yazmaç öbeğinin geniş satırlarında yer tutmaktadırlar. Programın çalışması sırasında da sorun çıkmaması için en büyük darlık grubundan çok sayıda yazmaç bulunur. Bu durumda da dar değerler bile gerekenden büyük yerlere yazıldığı için yeterli enerji tasarrufu yapılması mümkün olmaz. Buna rağmen diğer bölümleme türünde kapatılan yazmaç satırları iptal edilmiyor ve gereksiz yere boş duran yazmaçlar hemen en dar olan gruba geçirilerek enerji tasarrufu sağlanıyor.

BÖLÜM 5

5. BENZETİMLİK ORTAMI VE DENEYSEL SONUÇLAR

Yapılan çalışmada kullanılan işlemci benzetimliği x86 mimarisini çok yollu olarak benzetimliğini yapan C++ dilinde yazılmış olan PTLsim [4] dir.

5.1. PTLsim

PTLsim hem x86 mimarisi benzetimliği hem de x86-64 buyruk kümesini yürüten bir sanal makinedir. Pentium 4 üstü ve Athlon 64 ve benzeri işlemcilerin benzetimliğini yapan açık kaynaklı tek araçtır. Açık kaynaklı olduğu için tezde bu araç tercih edilmiştir.

5.2. Benzetim Parametreleri ve Denek Taşı Programlar

PTLsim ile benzetimi yapılmış olan işlemcinin donanım parametreleri Çizelge 5.1de sıralanmıştır.

Çizelge 5.1. Benzetim Parametreleri

Parametre	Yapılandırma
Makine Genişliği	4 komut getir, 4 komut yayınla, 4 komut sonlandır
Pencere Boyu	32 satır yayın kuyruğu, 80 satır yükle/sakla kuyruğu, 128 satır YSB, 256 satır tamsayı yazmaç öbeği
İşlem Birimleri ve Gecikmeler	Tamsayı AMB (6/1), yükle/sakla birimi (2/2), tamsayı çarpma (2/4), tamsayı bölme (1/32), kayan nokta toplama (2/6), kayan nokta çarpma (2/6), kayan nokta bölme (2/6). Toplam 6 tamsayı, 2 kayan nokta işlem birimi
L1 Komut 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, 2 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, 14 çevrim isabet zamanı
BTB	1024 satır, 4 yollu kümeli ilişkili
Dallanma Tahmin	64K satır çift durumlu ve iki aşamalı birleşik
Bellek	140 çevrim gecikme

PTLsim ile yapılacak denemelerde kullanılacak olan denek taşı programların listesi ve açıklamaları aşağıdaki Çizelge 5.2 ve Çizelge 5.3de verilmiştir.

Çizelge 5.2. Tamsayı 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

Çizelge 5.3. 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
fma3d	Fortran 90	Sonlu Element Çarpışı Benzetimliği
sixtrack	Fortran 77	Yüksek Enerjili Nükleer Fizik İvmelendiricisi Tasarımı
apsi	Fortran 77	Meteoroloji: Hava Kirliliği Dağılımı

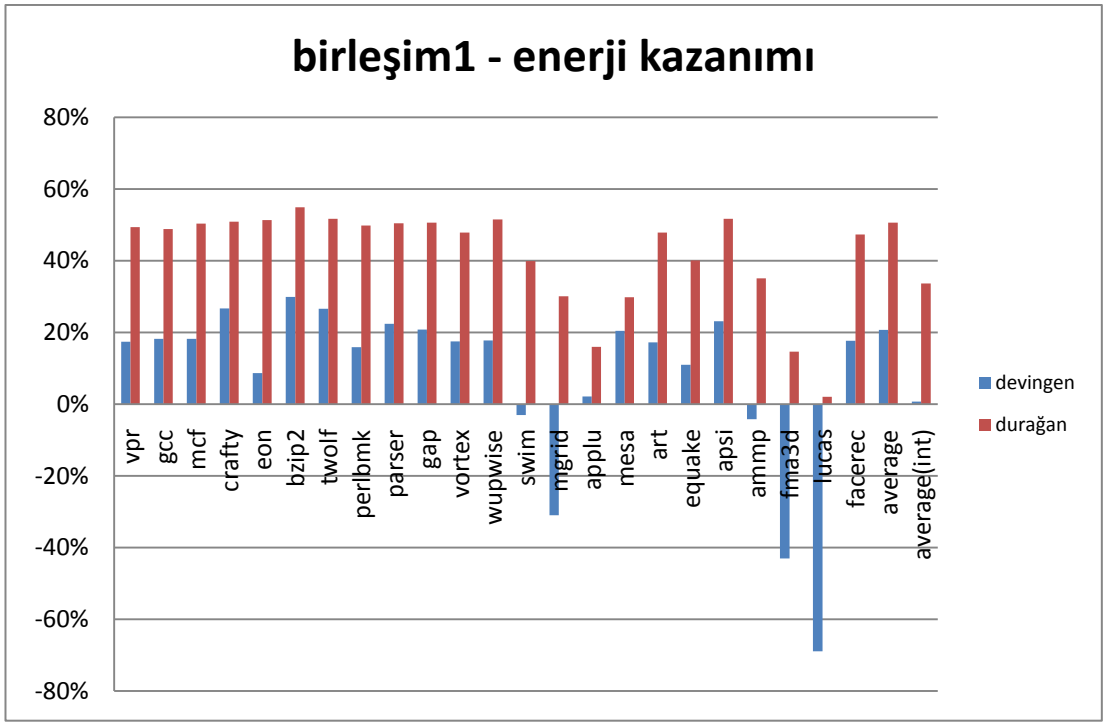
5.3. Benzetimlik Sonuçları

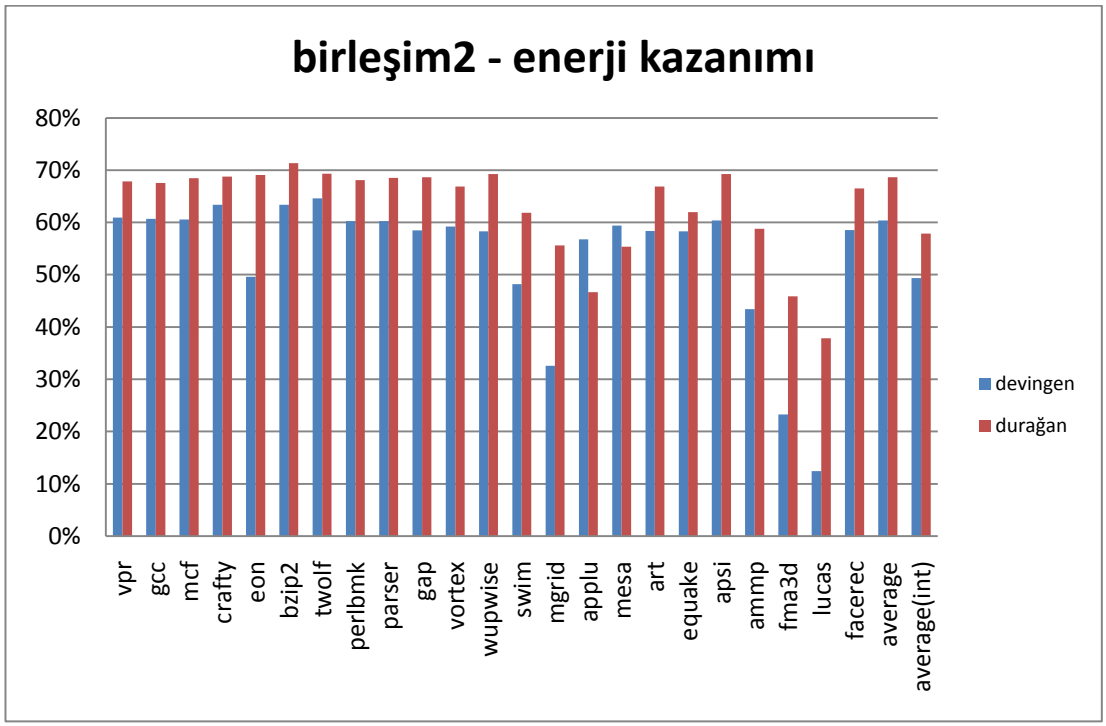
Yukarıda verilen donanımın PTLsim de yapılan değişikliklerle benzetimliği yapılması sonucunda alınan sonuçlar bu bölümde verilecektir. 4. Bölümde anlatıldığı gibi sabit ve değişken bölümleme ile ilgili farklı tasarımlar dolayısıyla farklı benzetimler yapılmıştır. Tasarımların enerji kazanımları anlatılırken PTLsim yardımıyla alınan kullanım istatistiklerine dayanılarak Cadence devre benzetimliğinde tasarlanmış olan fiziksel enerji değerleri kullanılmıştır. Cadence ile yapılan tasarımlarda 90nm devre teknolojisi kullanılmıştır. Yapılan tasarım başarımdan kayıp yani ÇBB (çevrim başına işlenen buyruk) ve enerjiden kazanç olarak değerlendirilmiştir. Enerji değerleri durağan ve devingen enerji olarak 2 kategoridedir.

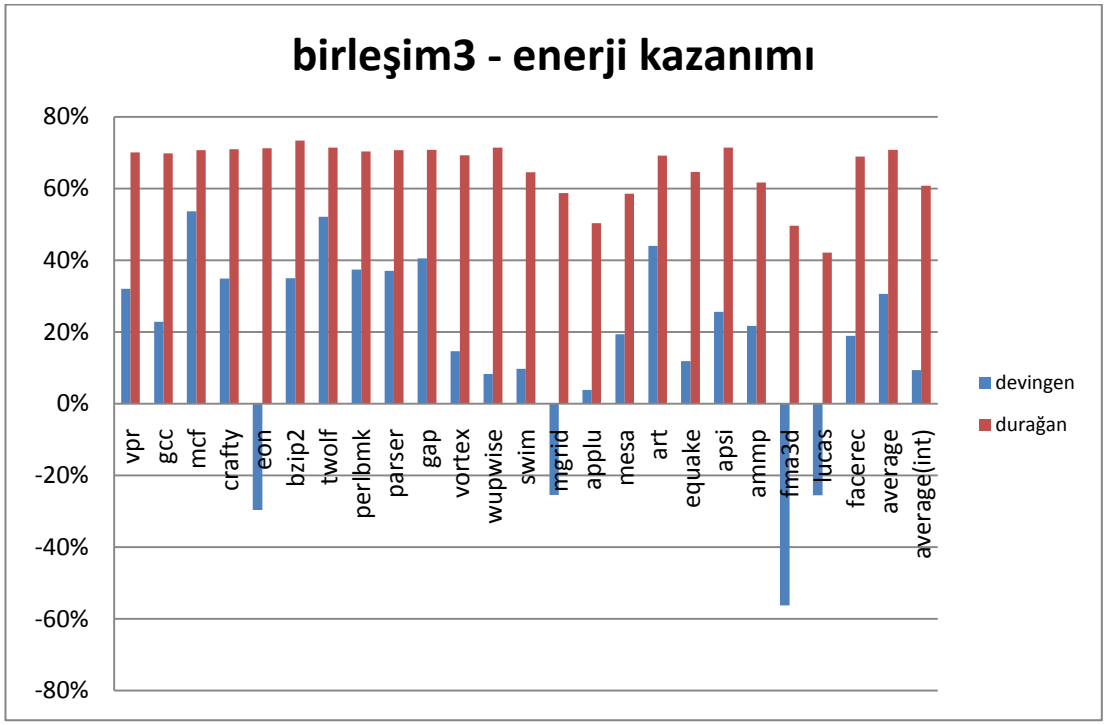
5.3.1. Yazmaç Öbeği Sabit Bölümleme Sonuçları

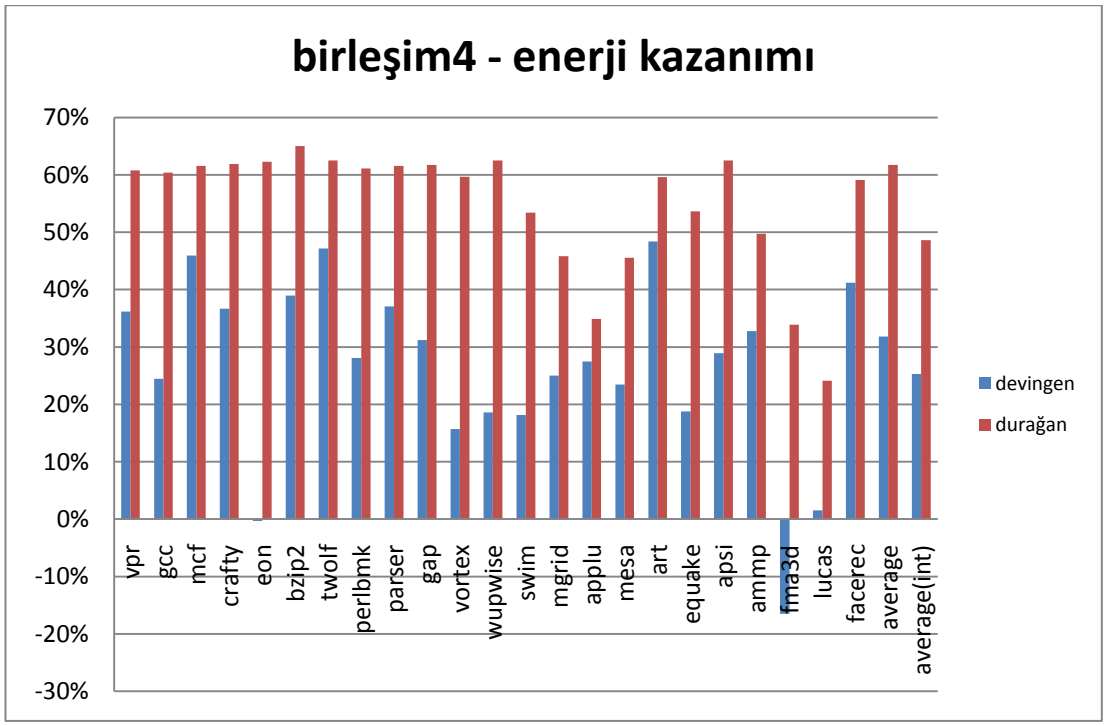
Sabit bölümleme sonucunda 5 farklı satır ve bit birleşimi elde edilmiştir. Bunlar 2.bölümdeki Çizelge 4.4de gösterilmiştir. Bu farklı birleşimlerin enerji değişimlerini bildiren grafikler gösterilmiştir. Grafikler her bir bölümlemenin temel işlemciye göre

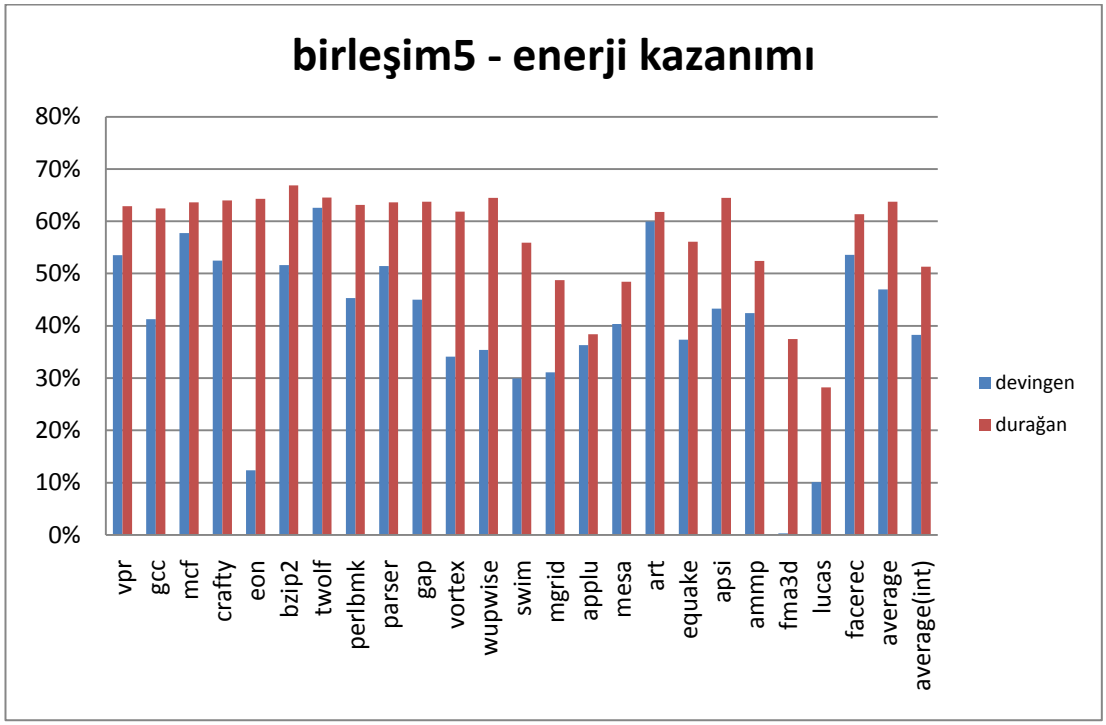
ne kadar enerji kazancı sağladığını gösterir. Durağan ve devingen enerji kazanımları farklı çubuklarla ifade edilmiştir.





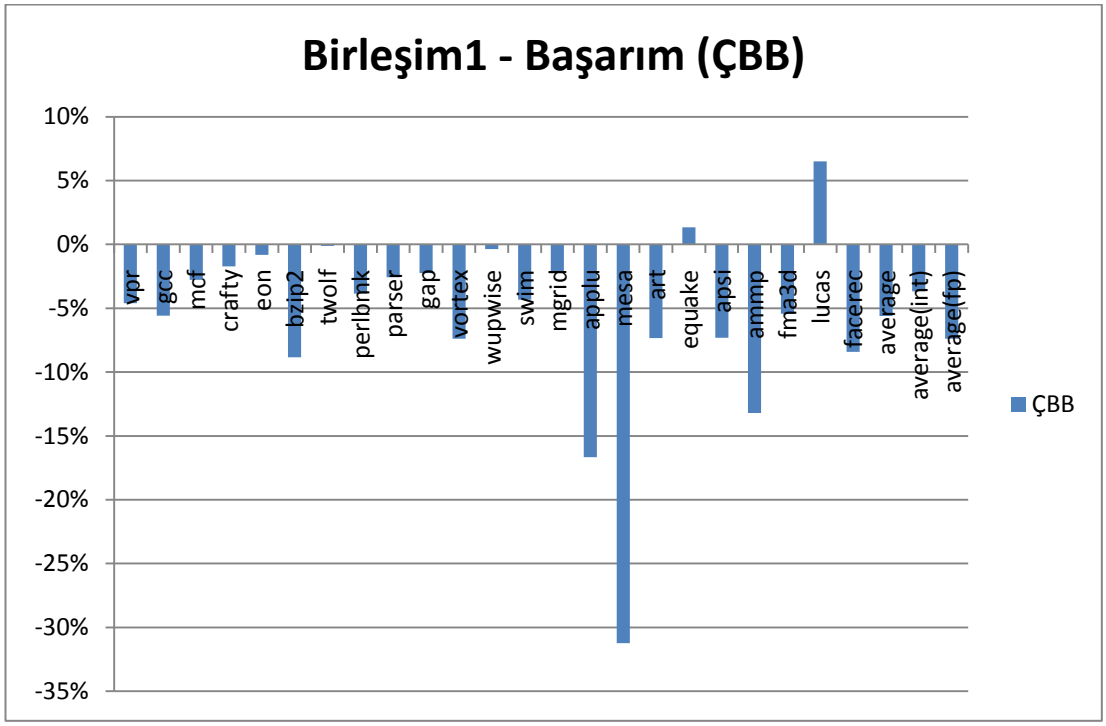

birleşim3 - enerji kazanımı
80%
60%
40%
20%
0%
-20%
-40%
-60%
-80%
vpr
gcc
mcf
crafty
eon
bzip2
twolf
perlbmk
parser
gap
vortex
wupwise
swim
mgrid
applu
mesa
art
equake
apsi
ammp
fma3d
lucas
facerec
average
average(int)
devingen
durağan

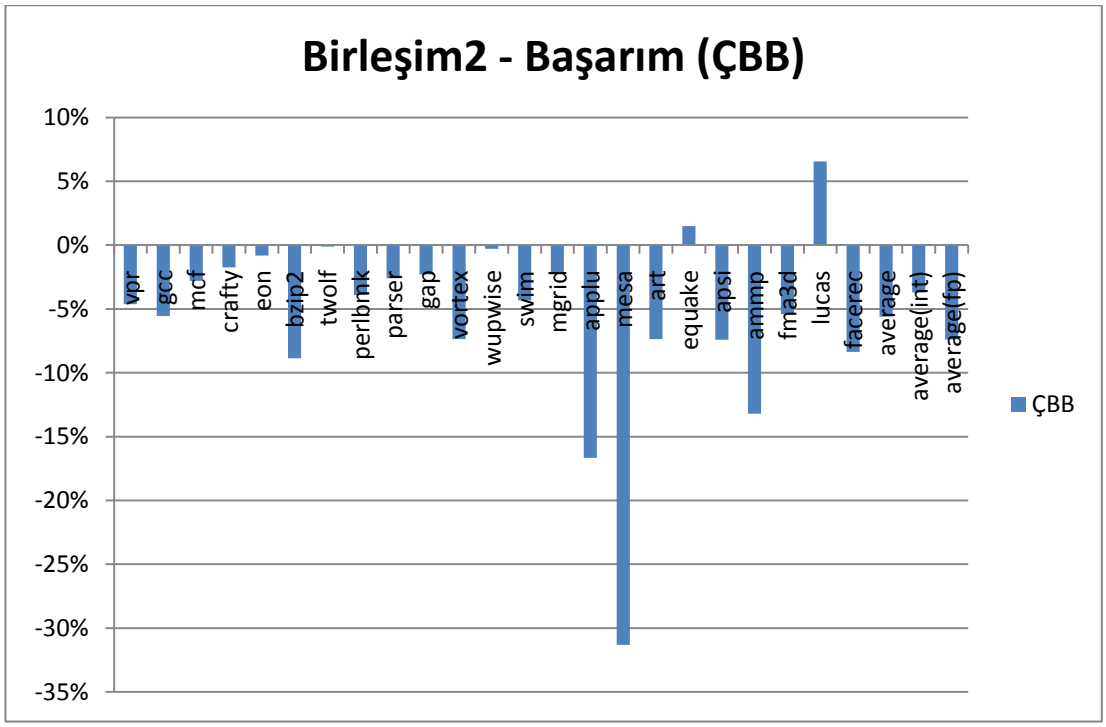

birleşim4 - enerji kazanımı
70%
60%
50%
40%
30%
20%
10%
0%
-10%
-20%
-30%
vpr
gcc
mcf
crafty
eon
bzip2
twolf
perlbmk
parser
gap
vortex
wupwise
swim
mgrid
applu
mesa
art
equake
apsi
ammp
fma3d
lucas
facerec
average
average(int)
devingen
durağan

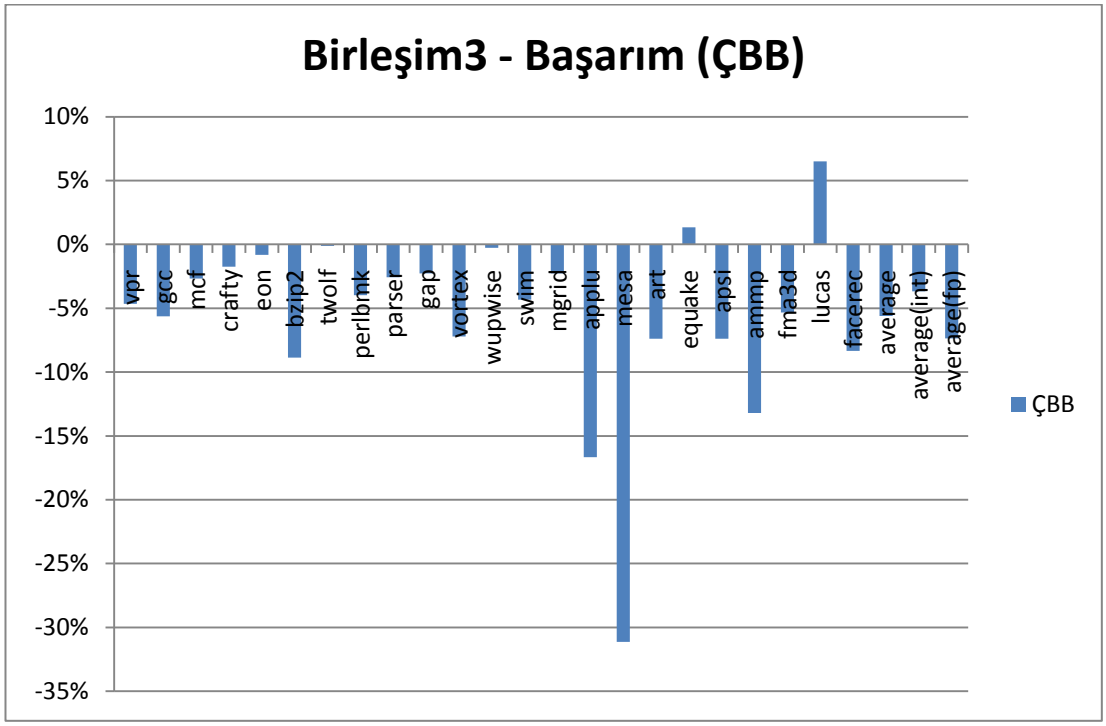


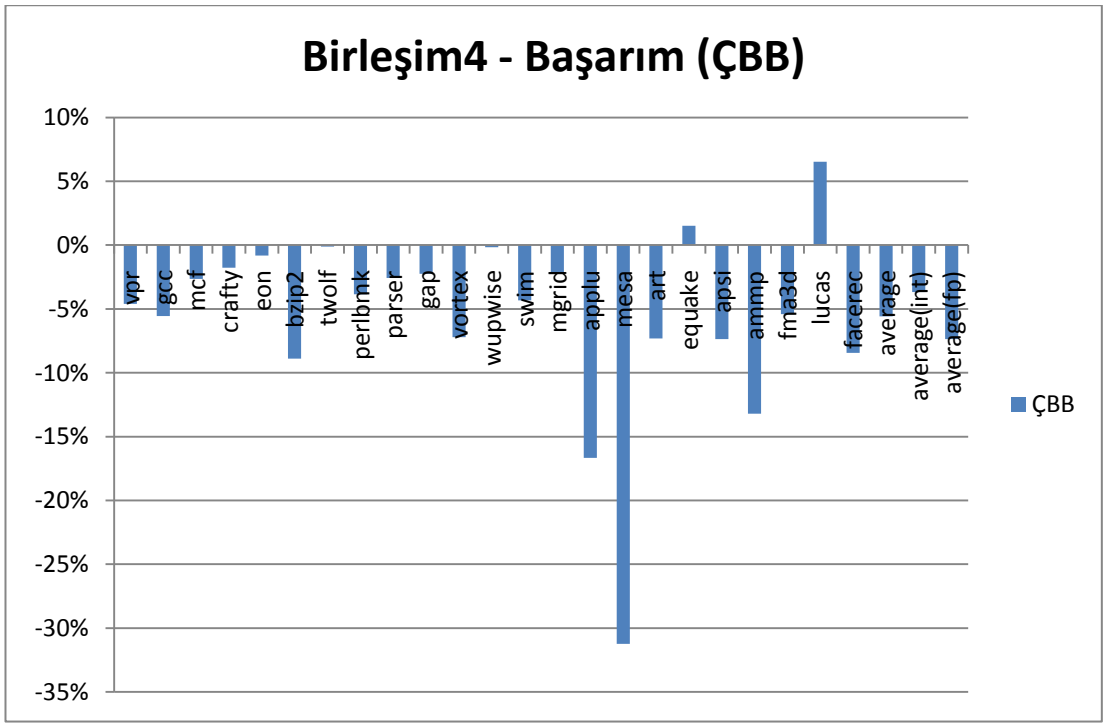
Enerji kazanımlarının eksi olduğu durumlar özellikle birinci birleşim için neredeyse normal yazmaç öbeğinin aynısı olan bölümlemede eklenen kod çözücülerin sebep olduğu farklardır.

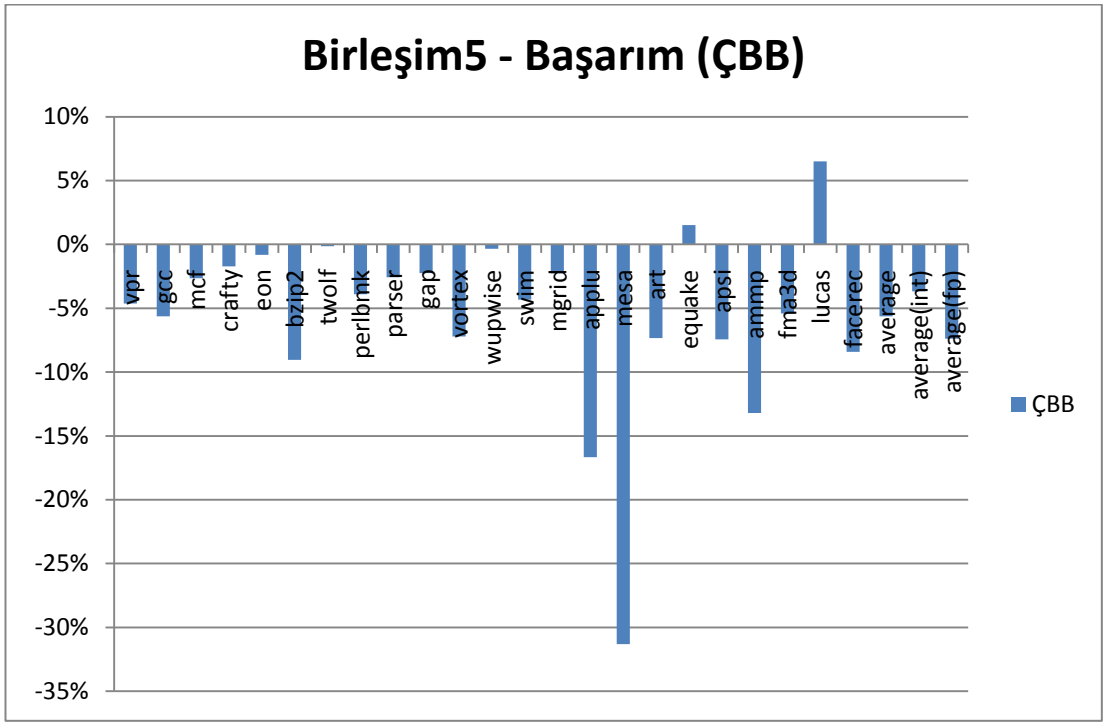
Sabit bölümlemenin 5 farklı birleşimi için başarımın durumunu gösteren ÇBB grafikleri aşağıda verilmiştir.


Birleşim1 - Başarım (ÇBB)
10%
5%
0%
-5%
-10%
-15%
-20%
-25%
-30%
-35%
vpr
gcc
mcf
crafty
eon
bzip2
twolf
perlbmk
parser
gap
vortex
wupwise
swim
mgrid
applu
mesa
art
equake
apsi
ammp
fma3d
lucas
facerec
average
average(int)
average(fp)
ÇBB


Birleşim2 - Başarım (ÇBB)
10%
5%
0%
-5%
-10%
-15%
-20%
-25%
-30%
-35%
vpr
gcc
mcf
crafty
eon
bzip2
twolf
perlbmk
parser
gap
vortex
wupwise
swim
mgrid
applu
mesa
art
equake
apsi
ammp
fma3d
lucas
facerec
average
average(int)
average(fp)
ÇBB


Birleşim3 - Başarım (ÇBB)
10%
5%
0%
-5%
-10%
-15%
-20%
-25%
-30%
-35%
vpr
gcc
mcf
crafty
eon
bzip2
twolf
perlbmk
parser
gap
vortex
wupwise
swim
mgrid
applu
mesa
art
equake
apsi
ammp
fma3d
lucas
facerec
average
average(int)
average(fp)
ÇBB

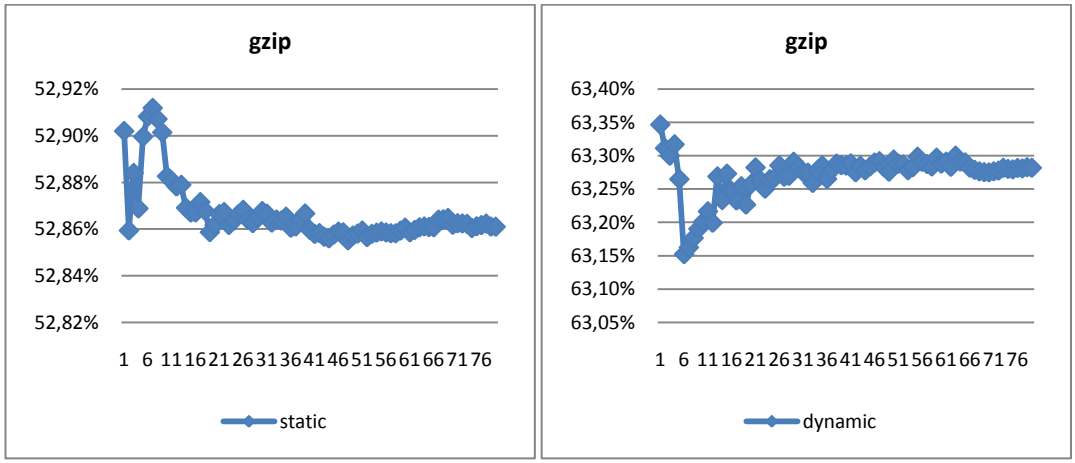

Birleşim4 - Başarım (ÇBB)
10%
5%
0%
-5%
-10%
-15%
-20%
-25%
-30%
-35%
vpr
gcc
mcf
crafty
eon
bzip2
twolf
perlbmk
parser
gap
vortex
wupwise
swim
mgrid
applu
mesa
art
equake
apsi
ammp
fma3d
lucas
facerec
average
average(int)
average(fp)
ÇBB

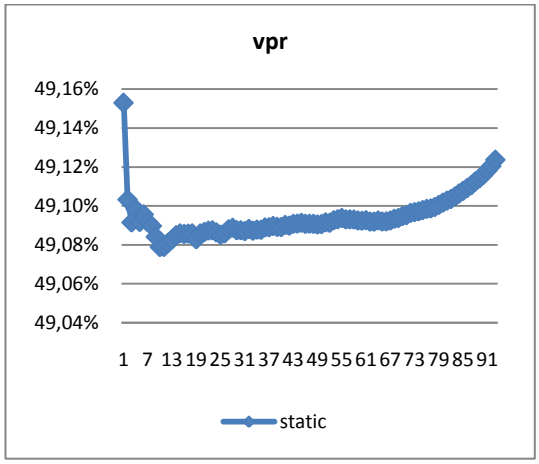


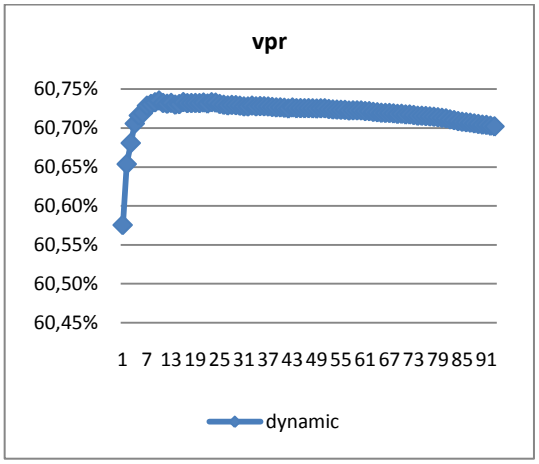
Sabit bölümleme yapıldığında hemen hemen tüm birleşimlerde aynı başarım kaybı yaşanmıştır.

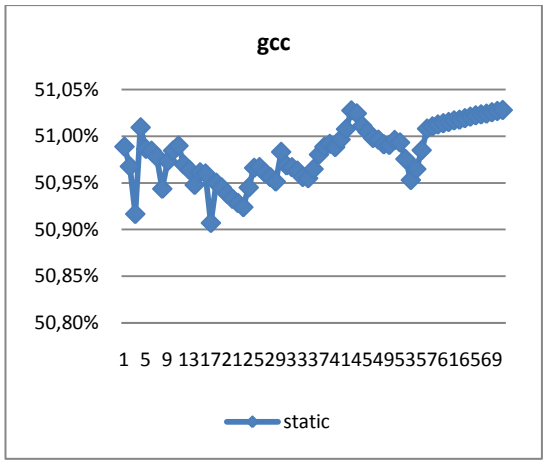
5.3.2. Yazmaç Öbeği Değişken Bölümleme Sonuçları

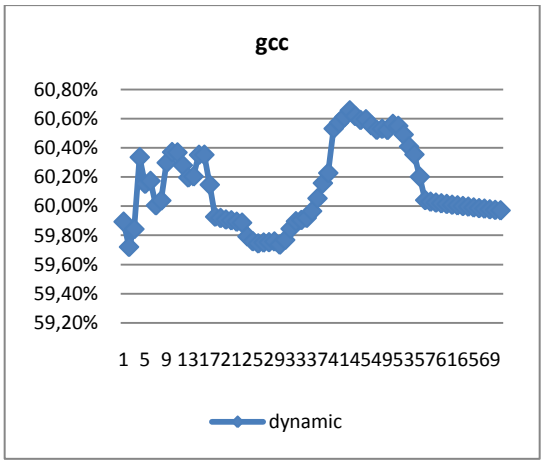
Değişken bölümlemenin enerjisi incelenirken geliştirilen tasarımın adım adım durumunun görülebilmesi için 10 milyon çevrimlik periyotlarda durağan ve devingen enerji kazanımları gösterilmiştir. Her bir denektaşı program için farklı bir grafik oluşturulmuştur.

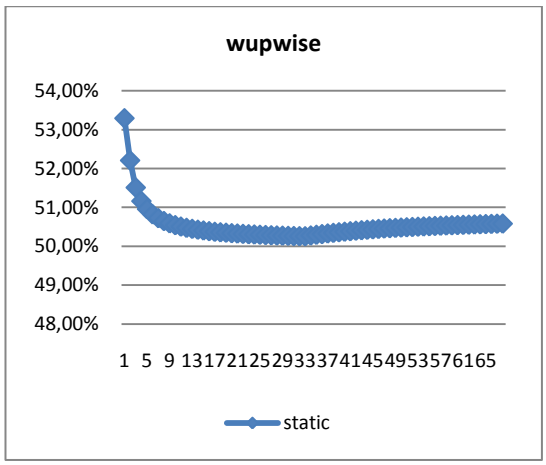


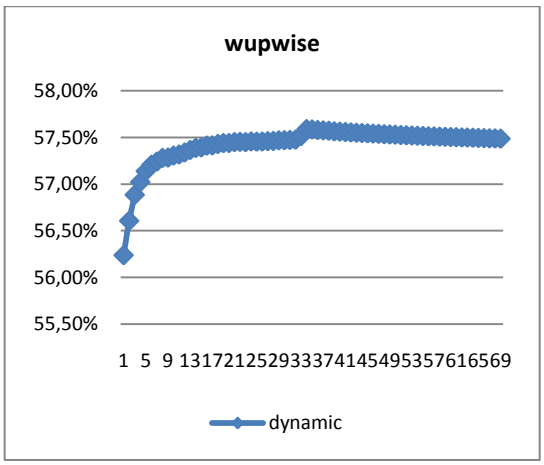

vpr
49,16%
49,14%
49,12%
49,10%
49,08%
49,06%
49,04%
1 7 13 19 25 31 37 43 49 55 61 67 73 79 85 91
static

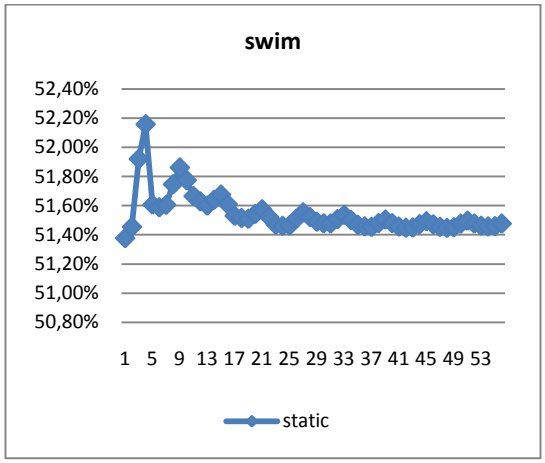

vpr
60,75%
60,70%
60,65%
60,60%
60,55%
60,50%
60,45%
1 7 13 19 25 31 37 43 49 55 61 67 73 79 85 91
dynamic

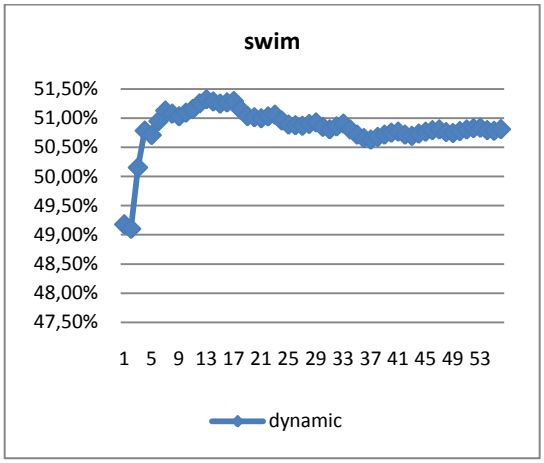

gcc
51,05%
51,00%
50,95%
50,90%
50,85%
50,80%
1 5 9 13172125293337414549535761 6569
static

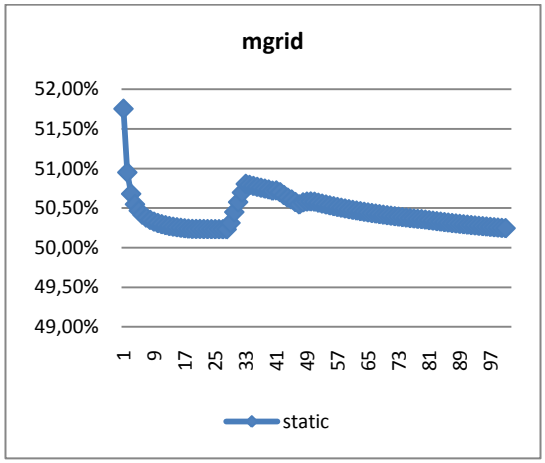

gcc
60,80%
60,60%
60,40%
60,20%
60,00%
59,80%
59,60%
59,40%
59,20%
1 5 9 13172125293337414549535761 6569
dynamic

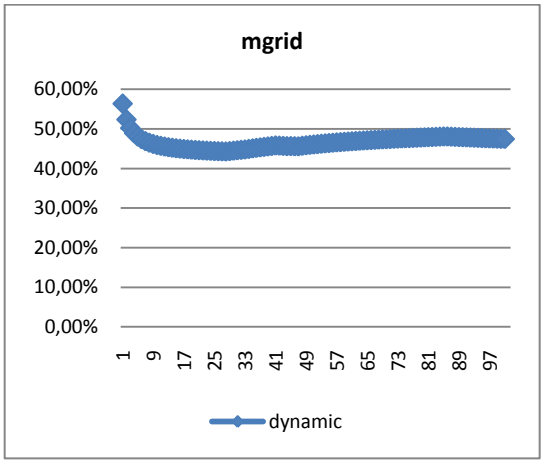

wupwise
54,00%
53,00%
52,00%
51,00%
50,00%
49,00%
48,00%
1 5 9 13172125293337414549535761 65
static

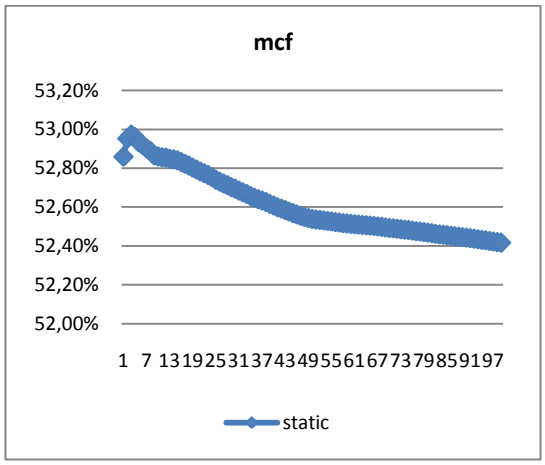

wupwise
58,00%
57,50%
57,00%
56,50%
56,00%
55,50%
1 5 9 13172125293337414549535761 6569
dynamic

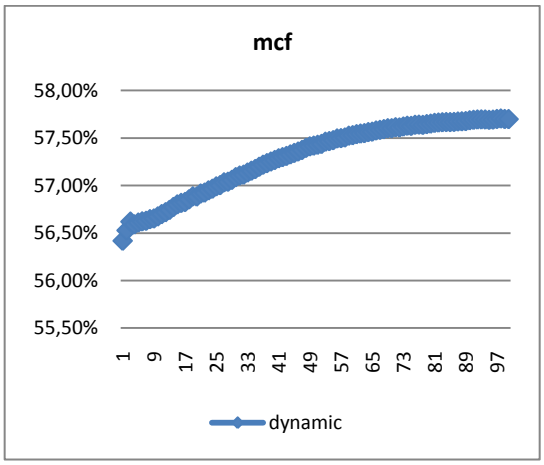

swim
52,40%
52,20%
52,00%
51,80%
51,60%
51,40%
51,20%
51,00%
50,80%
1 5 9 13 17 21 25 29 33 37 41 45 49 53
static

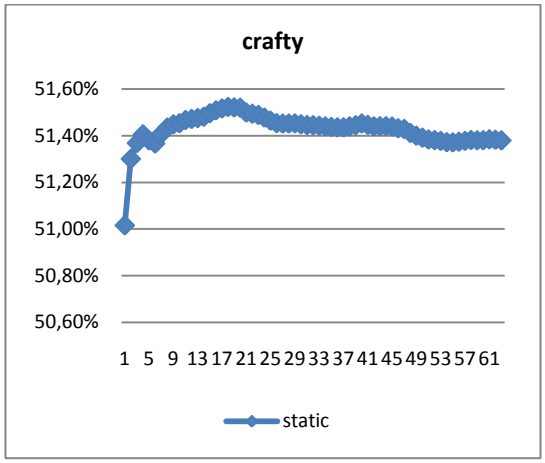

swim
51,50%
51,00%
50,50%
50,00%
49,50%
49,00%
48,50%
48,00%
47,50%
1 5 9 13 17 21 25 29 33 37 41 45 49 53
dynamic

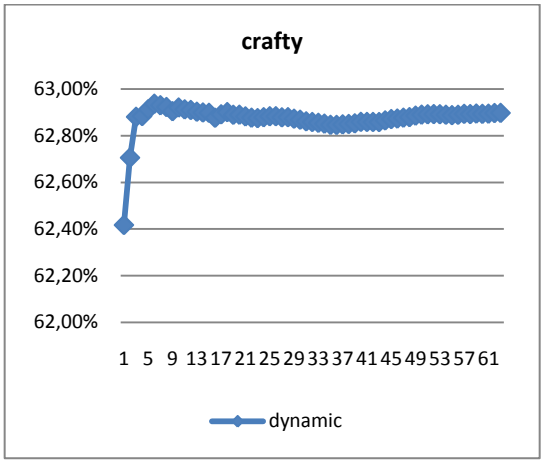

mgrid
52,00%
51,50%
51,00%
50,50%
50,00%
49,50%
49,00%
1 9 17 25 33 41 49 57 65 73 81 89 97
static

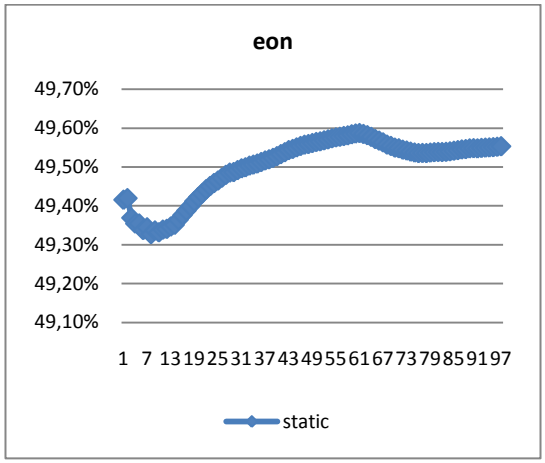

mgrid
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
1 9 17 25 33 41 49 57 65 73 81 89 97
dynamic

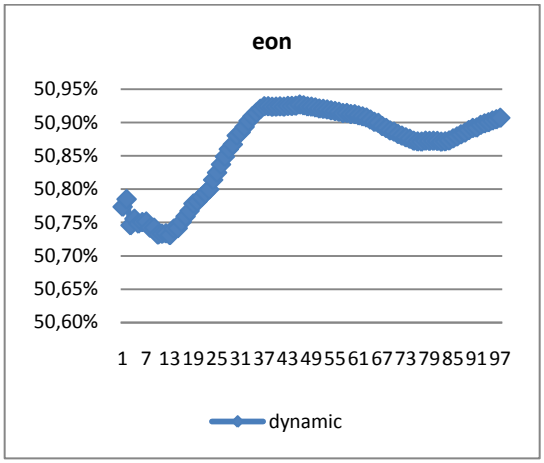

mcf
53,20%
53,00%
52,80%
52,60%
52,40%
52,20%
52,00%
1 7 13192531374349556167737985 9197
static

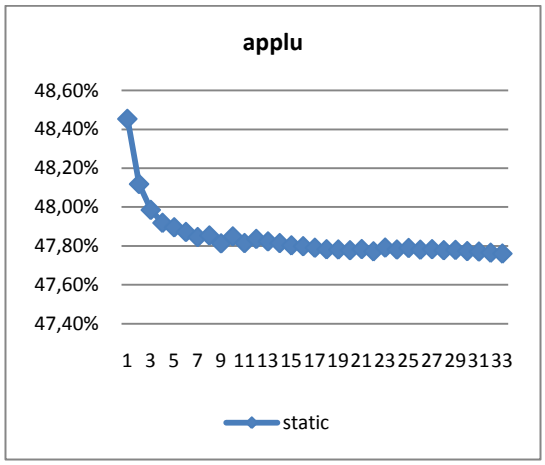

mcf
58,00%
57,50%
57,00%
56,50%
56,00%
55,50%
1 9 17 25 33 41 49 57 65 73 81 89 97
dynamic

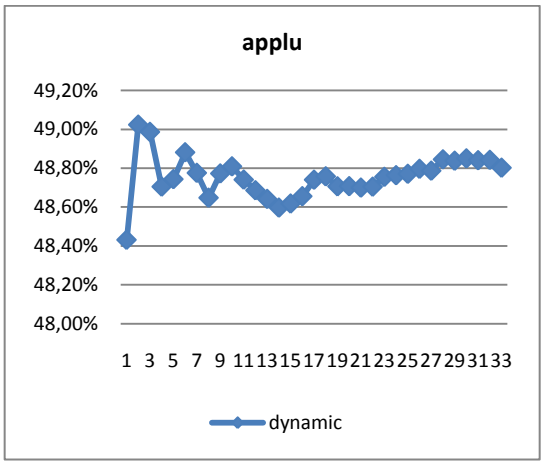

crafty
51,60%
51,40%
51,20%
51,00%
50,80%
50,60%
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61
static

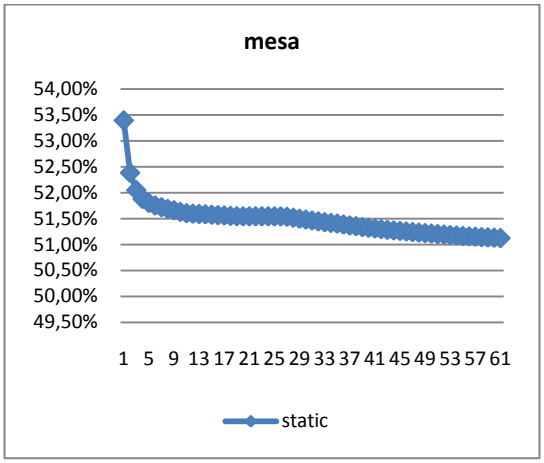

crafty
63,00%
62,80%
62,60%
62,40%
62,20%
62,00%
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61
dynamic

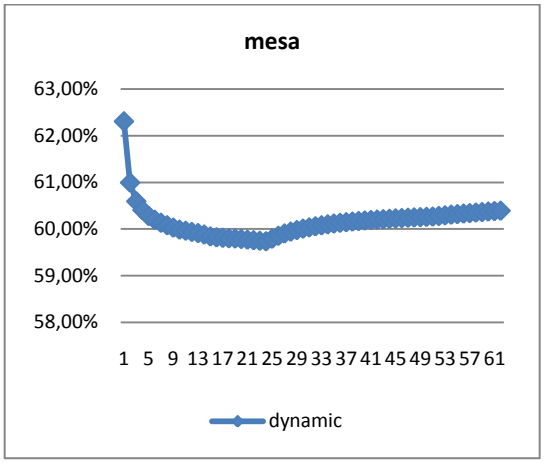

eon
49,70%
49,60%
49,50%
49,40%
49,30%
49,20%
49,10%
1 7 13 19 25 31 37 43 49 55 61 67 73 79 85 91 97
static

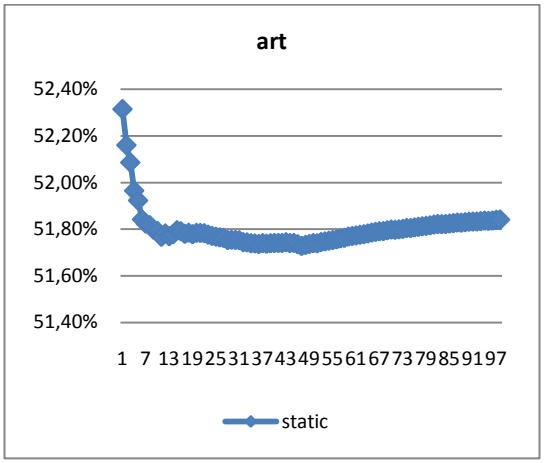

eon
50,95%
50,90%
50,85%
50,80%
50,75%
50,70%
50,65%
50,60%
1 7 13 19 25 31 37 43 49 55 61 67 73 79 85 91 97
dynamic

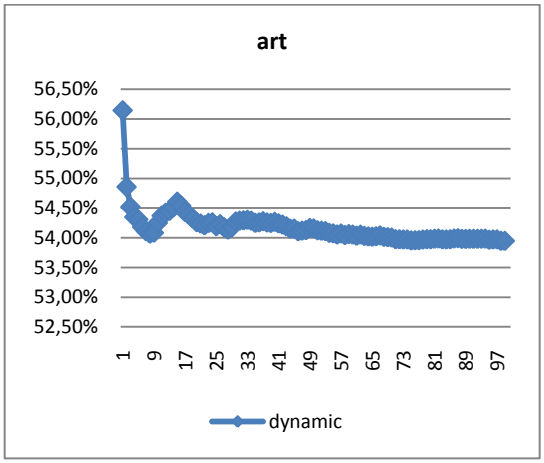

applu
48,60%
48,40%
48,20%
48,00%
47,80%
47,60%
47,40%
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33
static

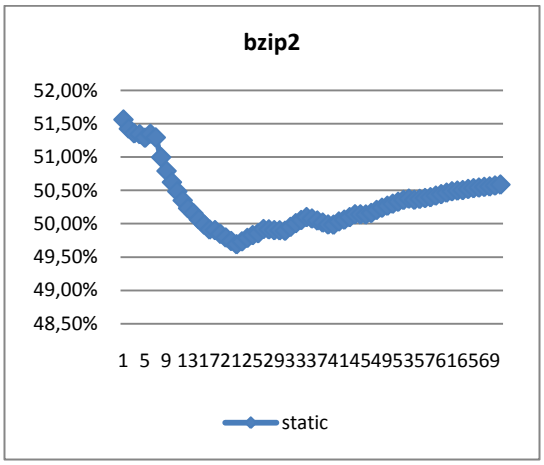

applu
49,20%
49,00%
48,80%
48,60%
48,40%
48,20%
48,00%
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33
dynamic

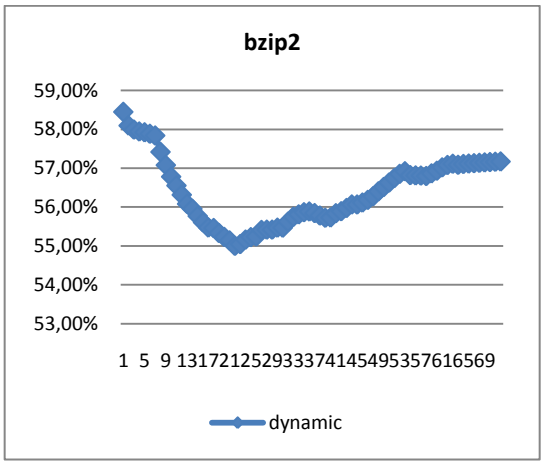

mesa
54,00%
53,50%
53,00%
52,50%
52,00%
51,50%
51,00%
50,50%
50,00%
49,50%
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61
static

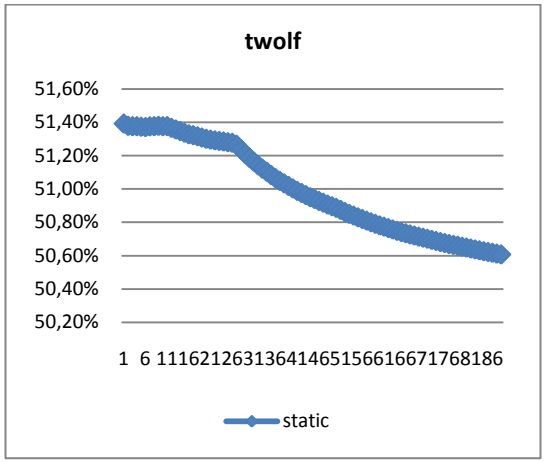

mesa
63,00%
62,00%
61,00%
60,00%
59,00%
58,00%
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61
dynamic

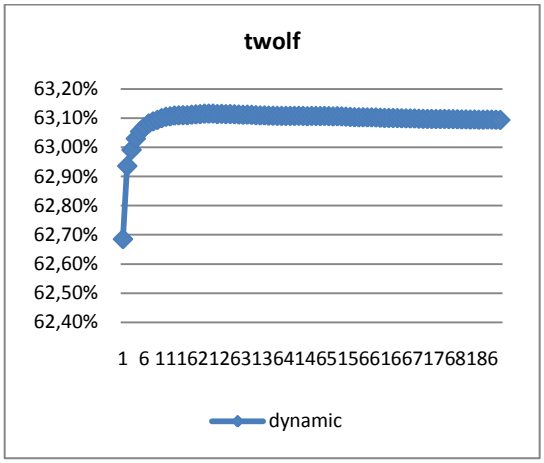

art
52,40%
52,20%
52,00%
51,80%
51,60%
51,40%
1 7 13 19 25 31 37 43 49 55 61 67 73 79 85 91 97
static

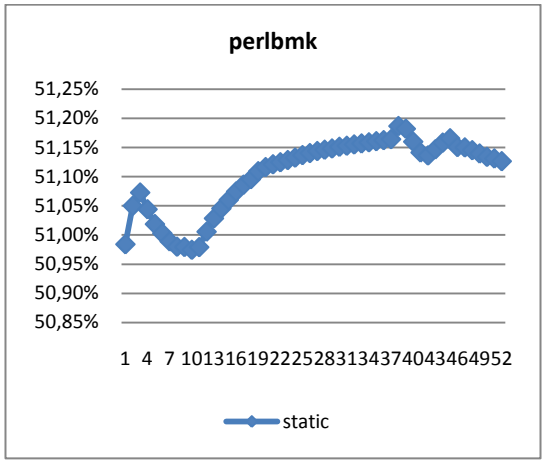

art
56,50%
56,00%
55,50%
55,00%
54,50%
54,00%
53,50%
53,00%
52,50%
1 9 17 25 33 41 49 57 65 73 81 89 97
dynamic

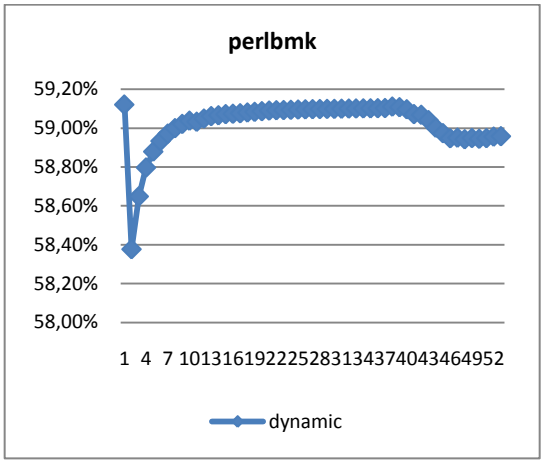

bzip2
52,00%
51,50%
51,00%
50,50%
50,00%
49,50%
49,00%
48,50%
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69
static

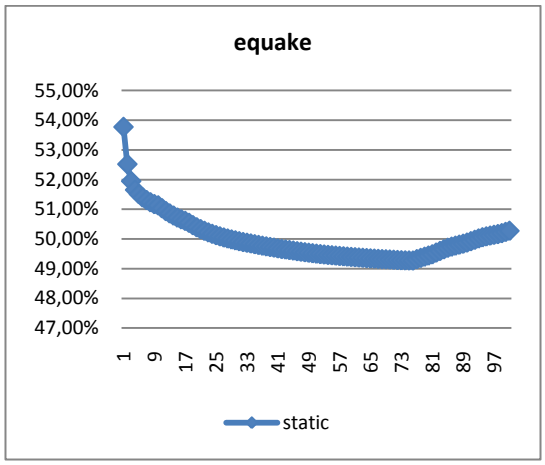

bzip2
59,00%
58,00%
57,00%
56,00%
55,00%
54,00%
53,00%
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69
dynamic

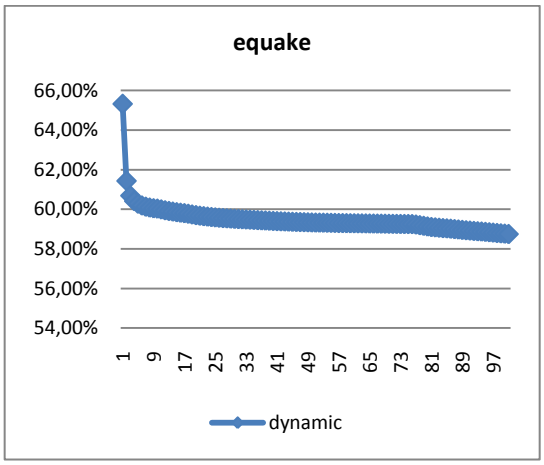

twolf
51,60%
51,40%
51,20%
51,00%
50,80%
50,60%
50,40%
50,20%
1 6 1116212631364146515661667176818 6
static

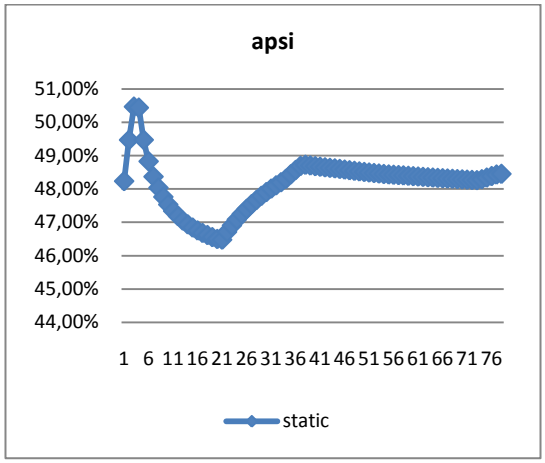

twolf
63,20%
63,10%
63,00%
62,90%
62,80%
62,70%
62,60%
62,50%
62,40%
1 6 1116212631364146515661667176818 6
dynamic

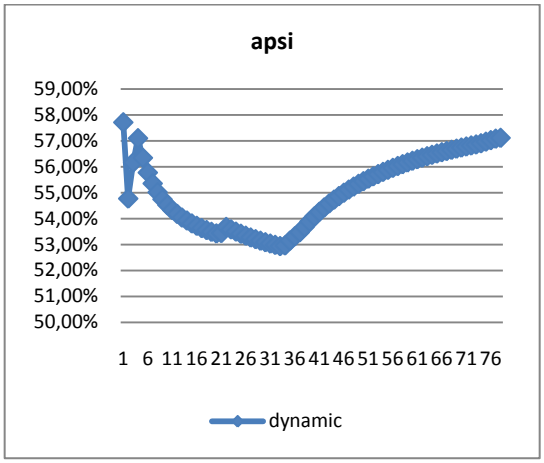

perlbmk
51,25%
51,20%
51,15%
51,10%
51,05%
51,00%
50,95%
50,90%
50,85%
1 4 7 10131619222528313437404346495 2
static

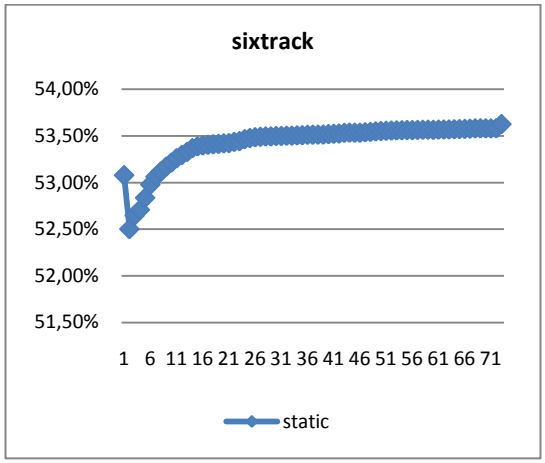

perlbmk
59,20%
59,00%
58,80%
58,60%
58,40%
58,20%
58,00%
1 4 7 10131619222528313437404346495 2
dynamic

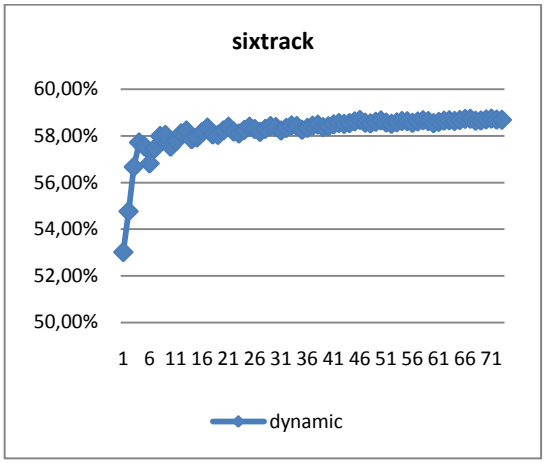

equake
55,00%
54,00%
53,00%
52,00%
51,00%
50,00%
49,00%
48,00%
47,00%
1 9 17 25 33 41 49 57 65 73 81 89 97
static

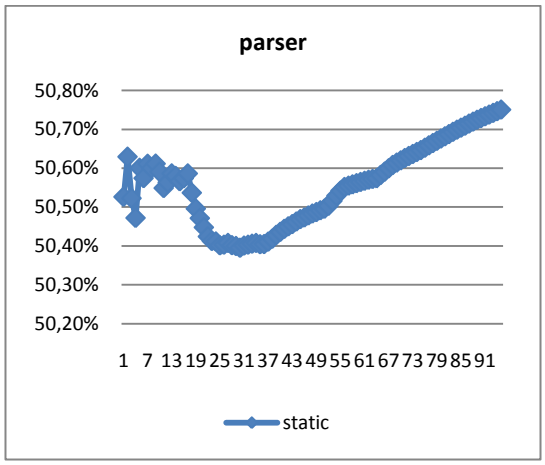

equake
66,00%
64,00%
62,00%
60,00%
58,00%
56,00%
54,00%
1 9 17 25 33 41 49 57 65 73 81 89 97
dynamic

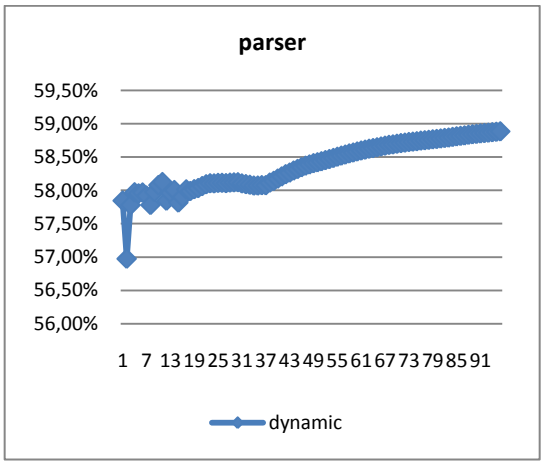

apsi
51,00%
50,00%
49,00%
48,00%
47,00%
46,00%
45,00%
44,00%
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76
static

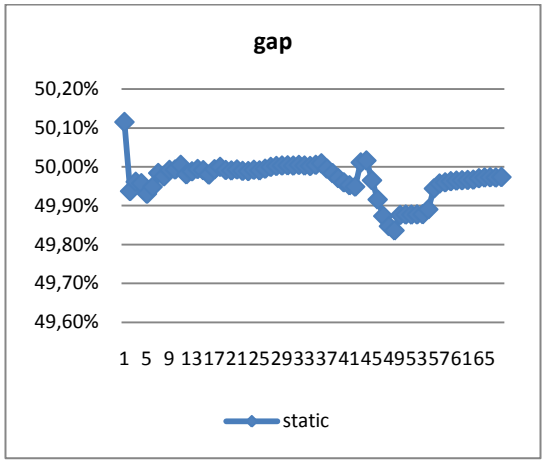

apsi
59,00%
58,00%
57,00%
56,00%
55,00%
54,00%
53,00%
52,00%
51,00%
50,00%
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76
dynamic

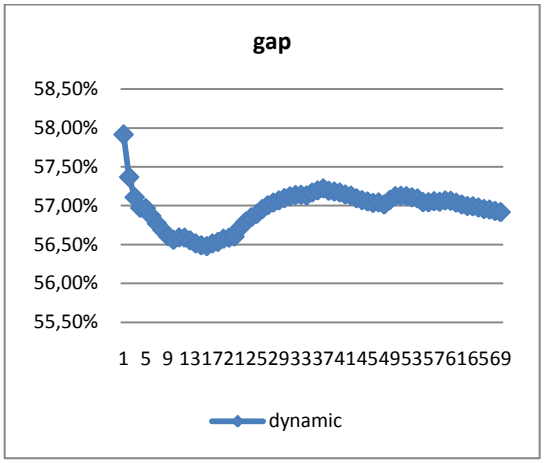

sixtrack
54,00%
53,50%
53,00%
52,50%
52,00%
51,50%
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71
static

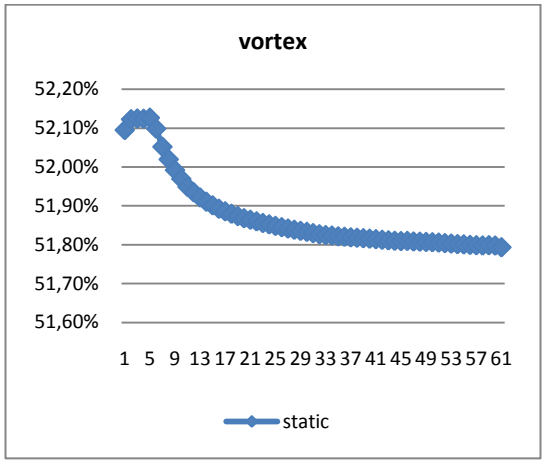

sixtrack
60,00%
58,00%
56,00%
54,00%
52,00%
50,00%
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71
dynamic

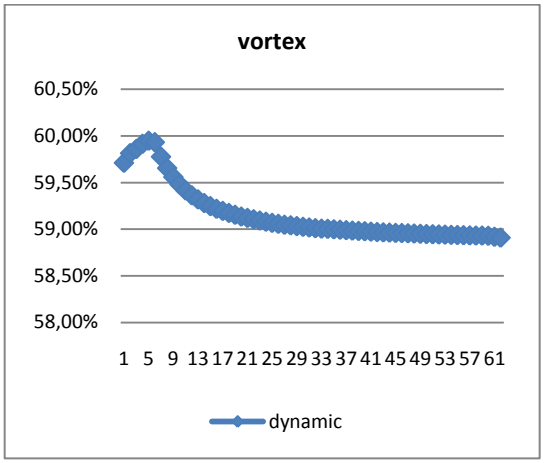

parser
50,80%
50,70%
50,60%
50,50%
50,40%
50,30%
50,20%
1 7 13 19 25 31 37 43 49 55 61 67 73 79 85 91
static

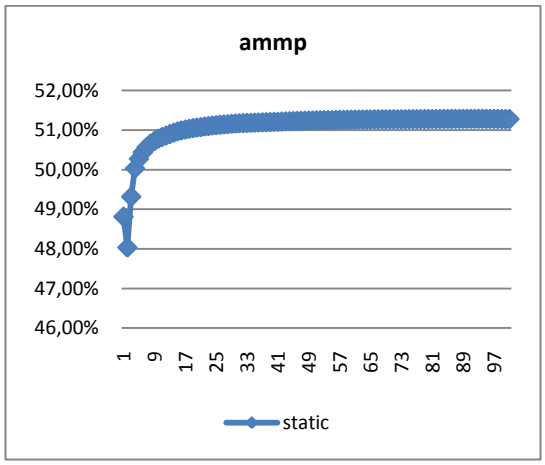

parser
59,50%
59,00%
58,50%
58,00%
57,50%
57,00%
56,50%
56,00%
1 7 13 19 25 31 37 43 49 55 61 67 73 79 85 91
dynamic

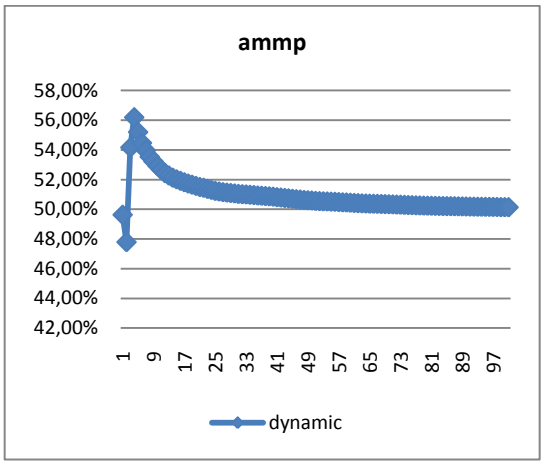

gap
50,20%
50,10%
50,00%
49,90%
49,80%
49,70%
49,60%
1 5 9 13172125293337414549535761 65
static

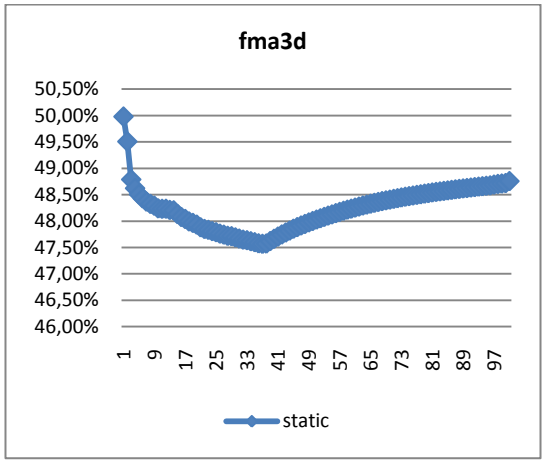

gap
58,50%
58,00%
57,50%
57,00%
56,50%
56,00%
55,50%
1 5 9 13172125293337414549535761 6569
dynamic

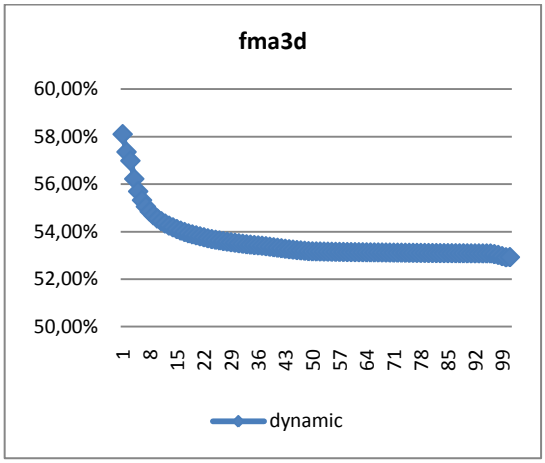

vortex
52,20%
52,10%
52,00%
51,90%
51,80%
51,70%
51,60%
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61
static

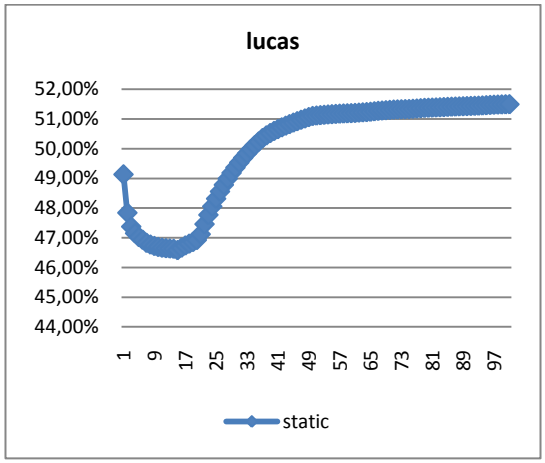

vortex
60,50%
60,00%
59,50%
59,00%
58,50%
58,00%
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61
dynamic

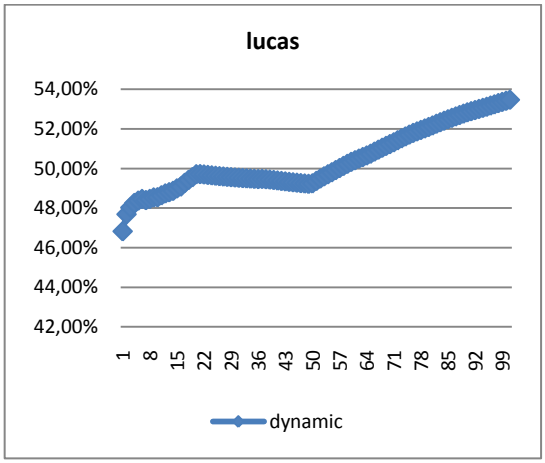

ammp
52,00%
51,00%
50,00%
49,00%
48,00%
47,00%
46,00%
1 9 17 25 33 41 49 57 65 73 81 89 97
static

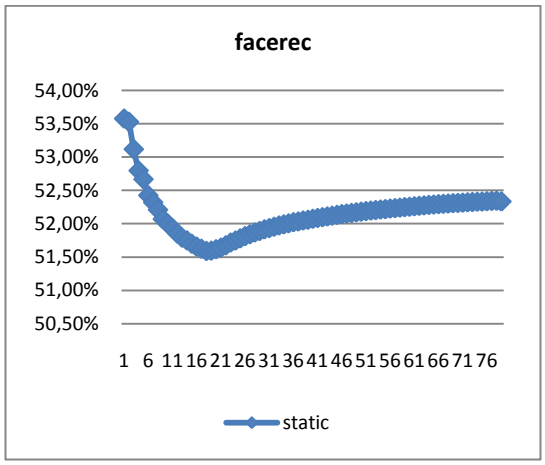

ammp
58,00%
56,00%
54,00%
52,00%
50,00%
48,00%
46,00%
44,00%
42,00%
1 9 17 25 33 41 49 57 65 73 81 89 97
dynamic

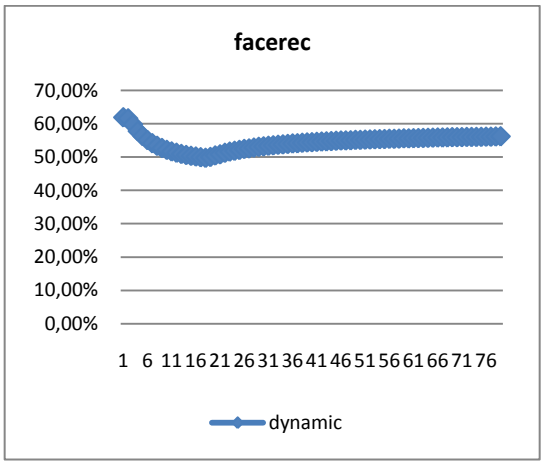






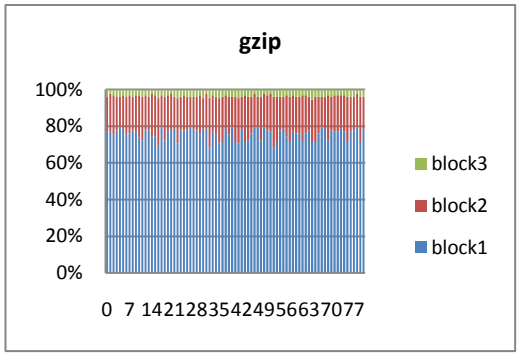


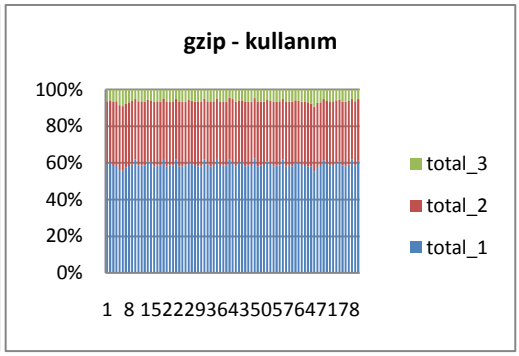


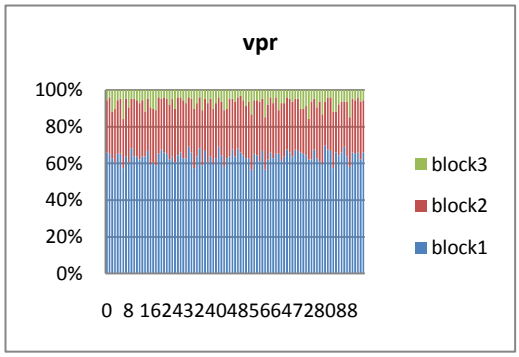


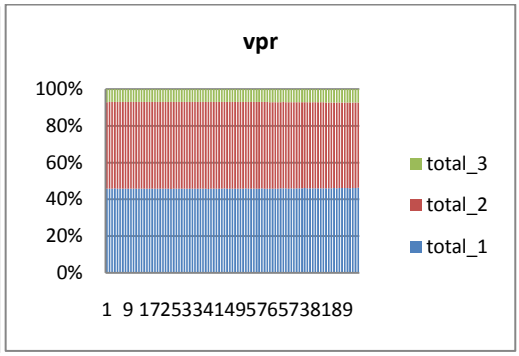
Değişken bölümleme yapmak için kullanılan satır sayısı dengeleme algoritmasının da doğruluğunun kontrol edilmesi için normal şartlarda programların dar yazmaç kullanımları ile değişken bölümlenmiş yazmaç öbeğinin dar yazmaç dağılımı karşılaştırılmıştır. Yine her bir denektaşı program için grafikler sol tarafta yazmaç

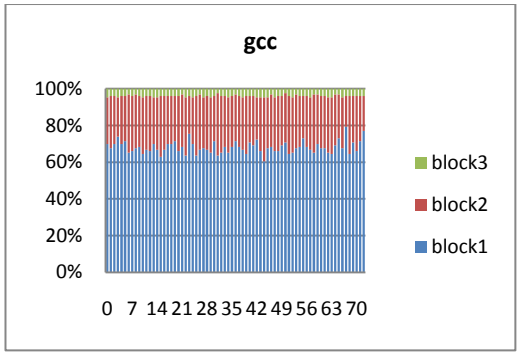
öbeği dağılımı sağ tarafta da programın darlık kullanımı olmak üzere 10 milyon çevrimlik periyotlarda listelenmiştir.

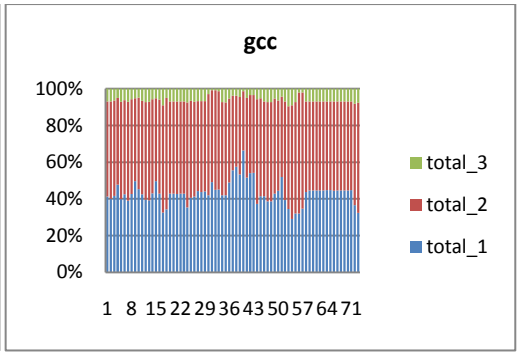


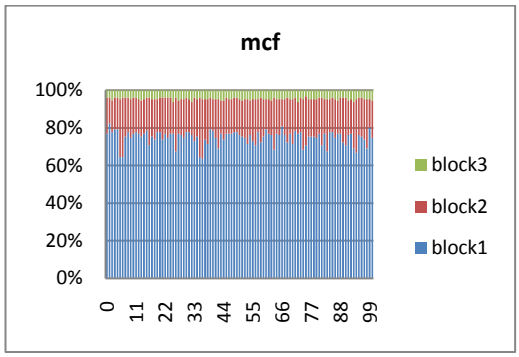


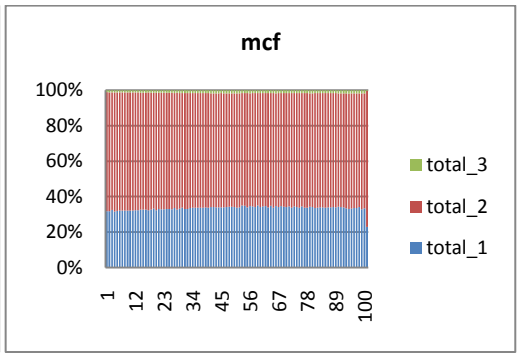


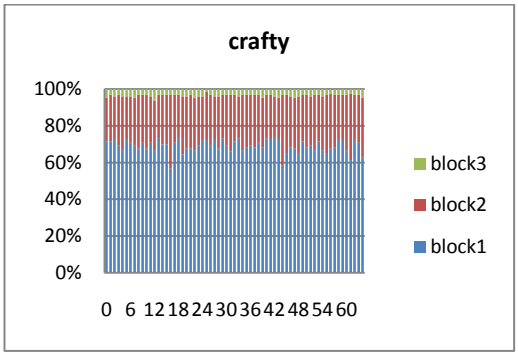


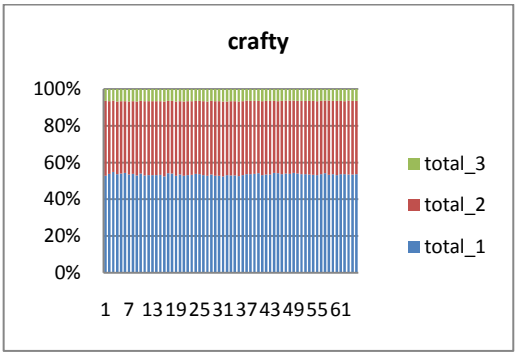


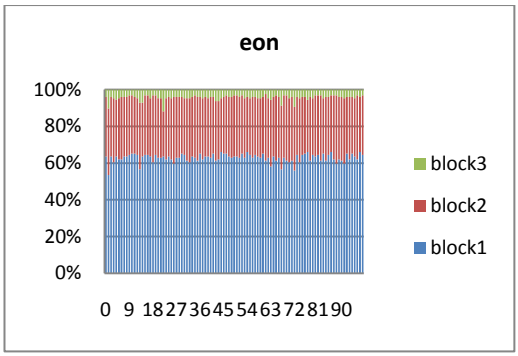


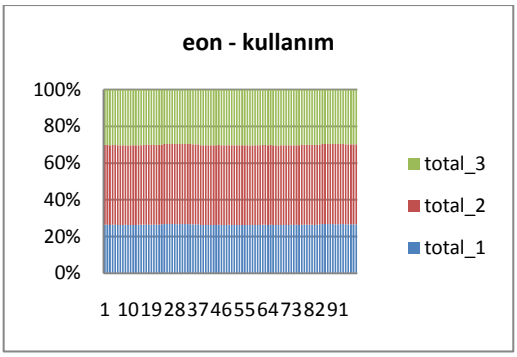


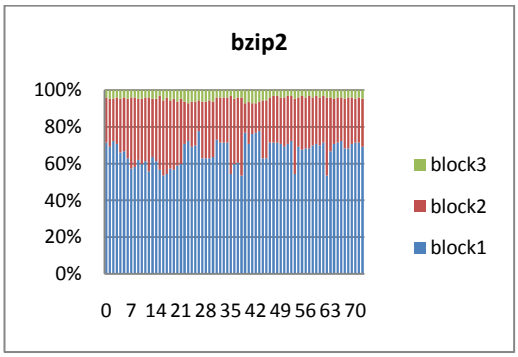


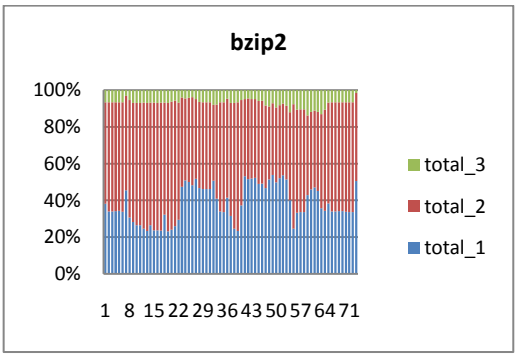

crafty
100%
80%
60%
40%
20%
0%
0 6 12 18 24 30 36 42 48 54 60
block3
block2
block1

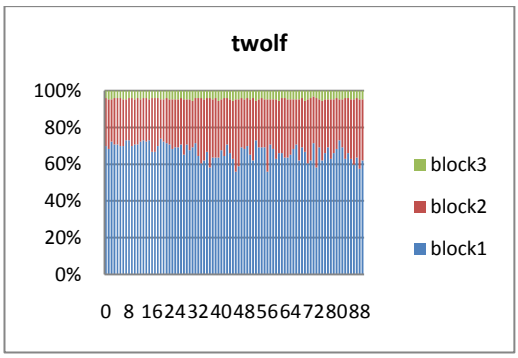

crafty
100%
80%
60%
40%
20%
0%
1 7 13 19 25 31 37 43 49 55 61
total_3
total_2
total_1

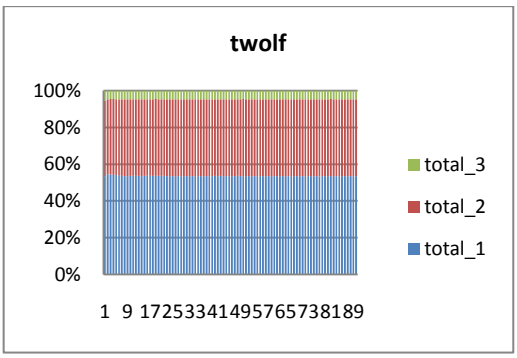

eon
100%
80%
60%
40%
20%
0%
0 9 18 27 36 45 54 63 72 81 90
block3
block2
block1

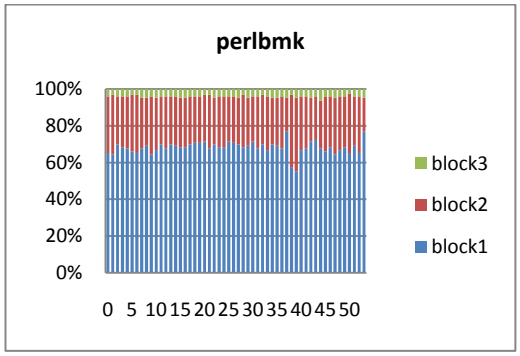

eon - kullanım
100%
80%
60%
40%
20%
0%
1 10 19 28 37 46 55 64 73 82 91
total_3
total_2
total_1

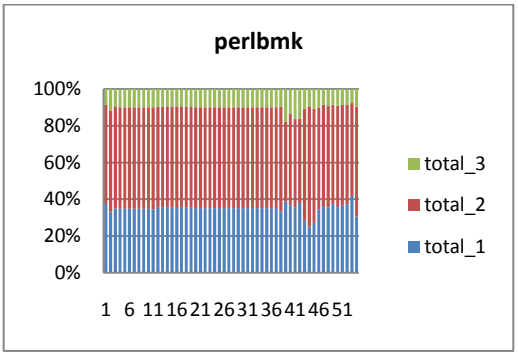

bzip2
100%
80%
60%
40%
20%
0%
0 7 14 21 28 35 42 49 56 63 70
block3
block2
block1

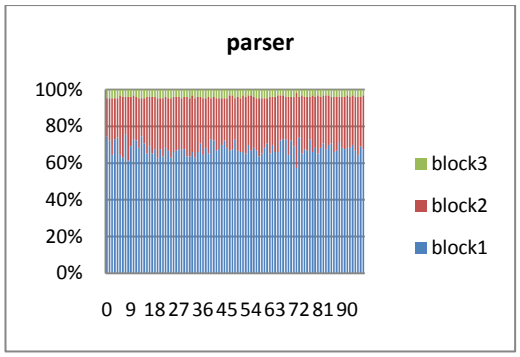

bzip2
100%
80%
60%
40%
20%
0%
1 8 15 22 29 36 43 50 57 64 71
total_3
total_2
total_1

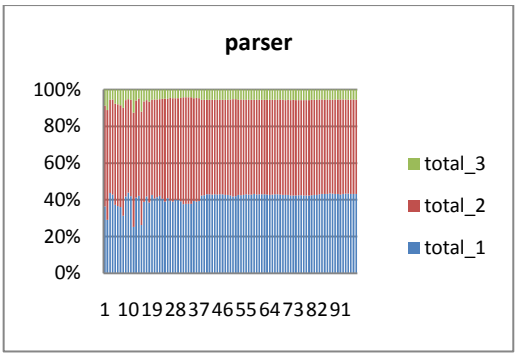

twolf
100%
80%
60%
40%
20%
0%
0 8 16 24 32 40 48 56 64 72 80 88
block3
block2
block1

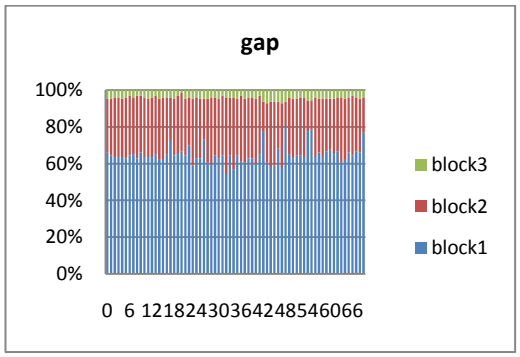

twolf
100%
80%
60%
40%
20%
0%
1 9 17 25 33 41 49 57 65 73 81 89
total_3
total_2
total_1

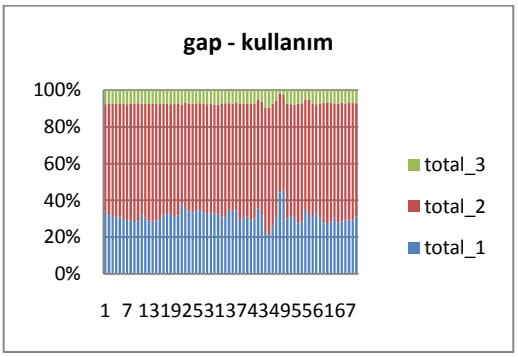

perlbmk
100%
80%
60%
40%
20%
0%
0 5 10 15 20 25 30 35 40 45 50
block3
block2
block1

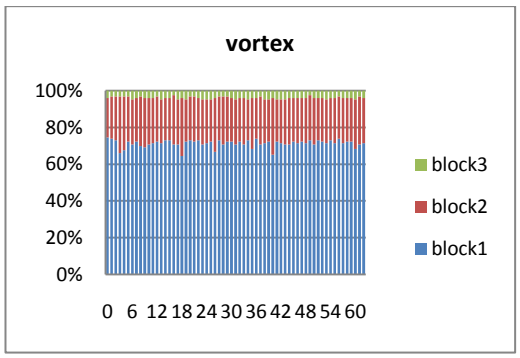

perlbmk
100%
80%
60%
40%
20%
0%
1 6 11 16 21 26 31 36 41 46 51
total_3
total_2
total_1

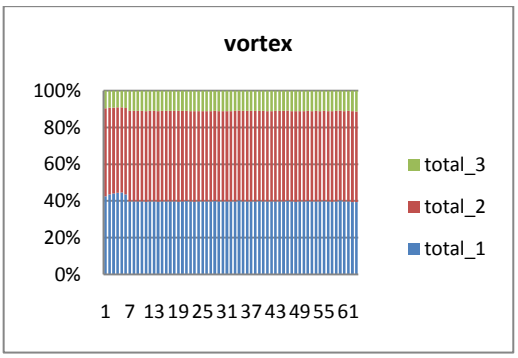

parser
100%
80%
60%
40%
20%
0%
0 9 18 27 36 45 54 63 72 81 90
block3
block2
block1

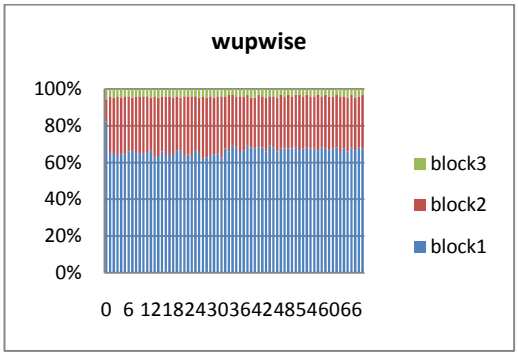

parser
100%
80%
60%
40%
20%
0%
1 10 19 28 37 46 55 64 73 82 91
total_3
total_2
total_1

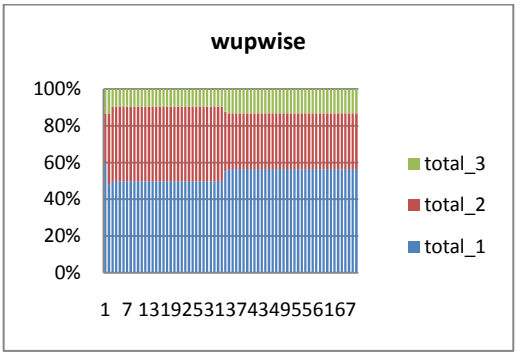

gap
100%
80%
60%
40%
20%
0%
0 6 12 18 24 30 36 42 48 54 60 66
block3
block2
block1

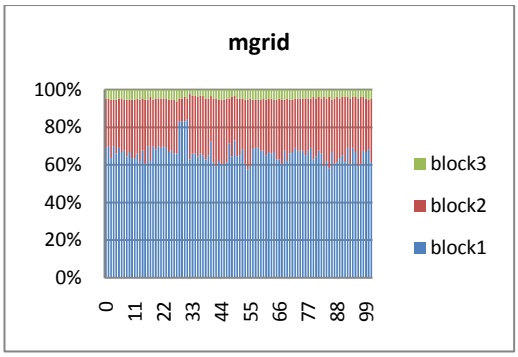

gap - kullanım
100%
80%
60%
40%
20%
0%
1 7 13 19 25 31 37 43 49 55 61 67
total_3
total_2
total_1

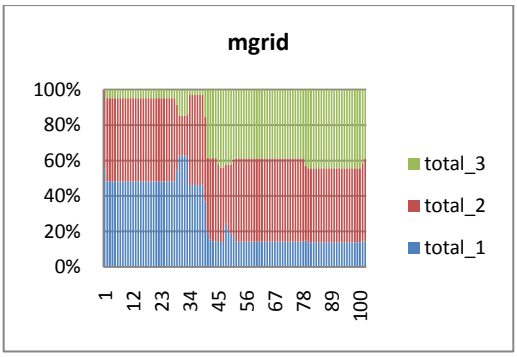

vortex
100%
80%
60%
40%
20%
0%
0 6 12 18 24 30 36 42 48 54 60
block3
block2
block1

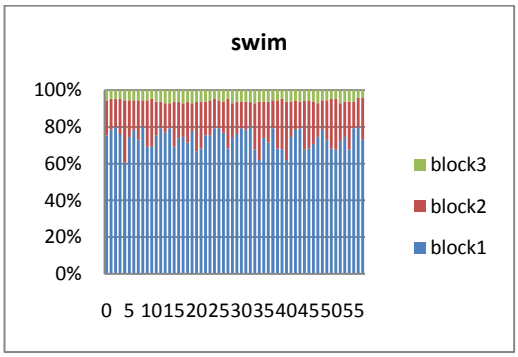

vortex
100%
80%
60%
40%
20%
0%
1 7 13 19 25 31 37 43 49 55 61
total_3
total_2
total_1

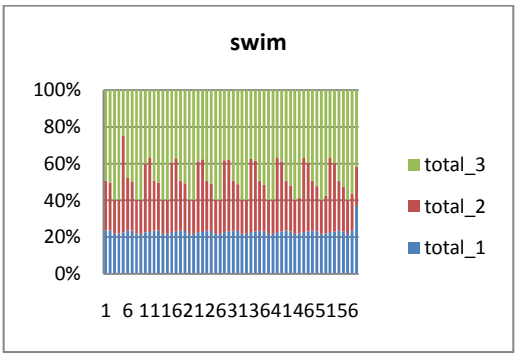

wupwise
100%
80%
60%
40%
20%
0%
0 6 12182430364248546066
block3
block2
block1

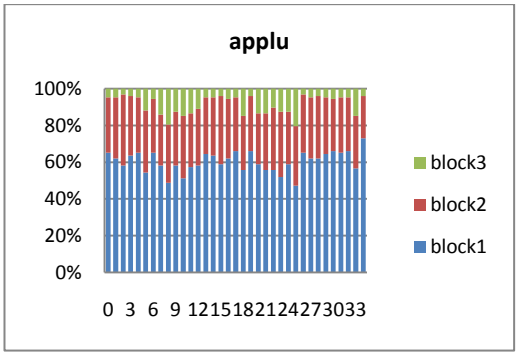

wupwise
100%
80%
60%
40%
20%
0%
1 7 13192531374349556167
total_3
total_2
total_1

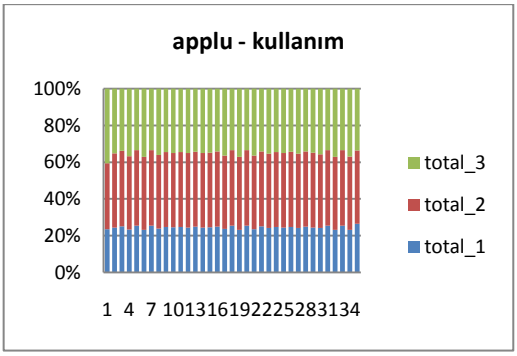

mgrid
100%
80%
60%
40%
20%
0%
0 11 22 33 44 55 66 77 88 99
block3
block2
block1

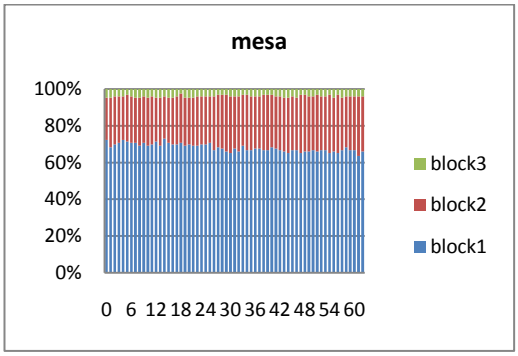

mgrid
100%
80%
60%
40%
20%
0%
1 12 23 34 45 56 67 78 89 100
total_3
total_2
total_1

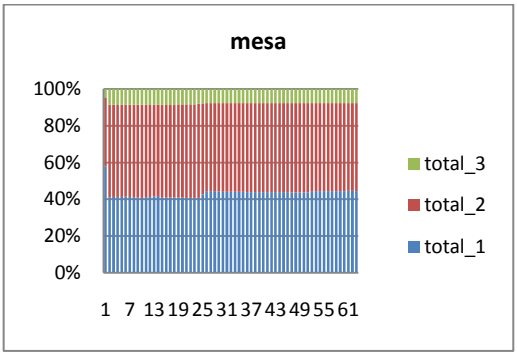

swim
100%
80%
60%
40%
20%
0%
0 5 10152025303540455055
block3
block2
block1

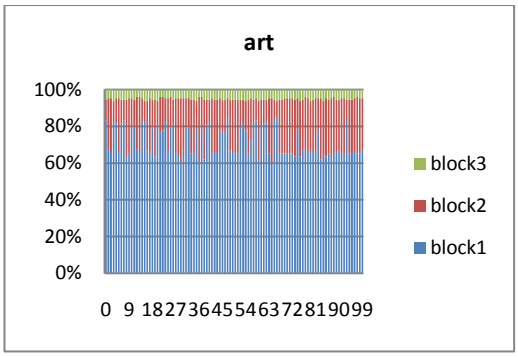

swim
100%
80%
60%
40%
20%
0%
1 6 111621263136414651 56
total_3
total_2
total_1

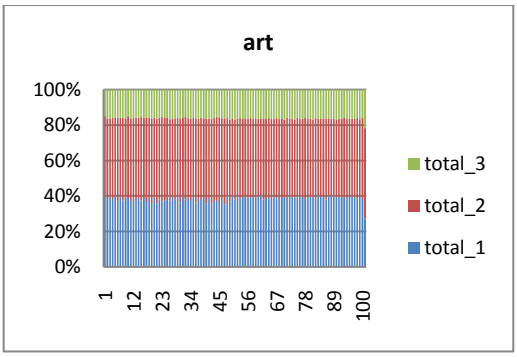

applu
100%
80%
60%
40%
20%
0%
0 3 6 9 12151821242730 33
block3
block2
block1

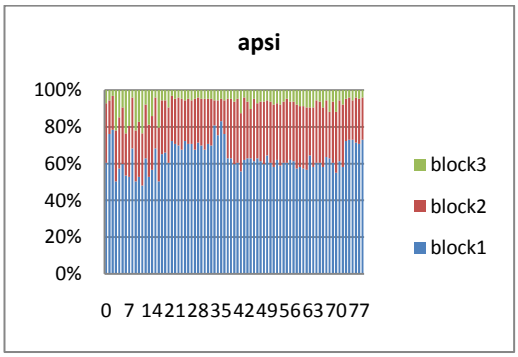

applu - kullanım
100%
80%
60%
40%
20%
0%
1 4 7 10131619222528 3134
total_3
total_2
total_1

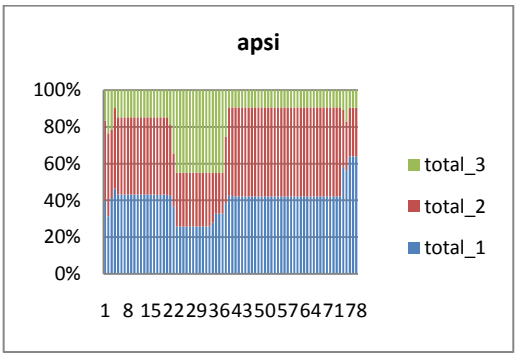

mesa
100%
80%
60%
40%
20%
0%
0 6 12 18 24 30 36 42 48 54 60
block3
block2
block1

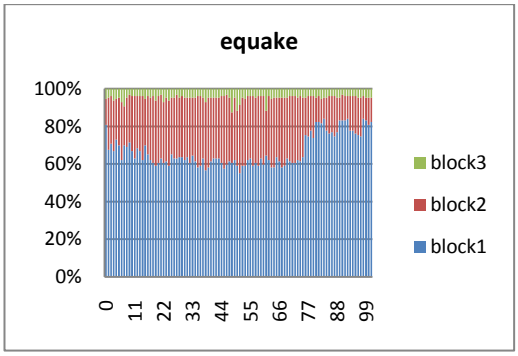

mesa
100%
80%
60%
40%
20%
0%
1 7 13 19 25 31 37 43 49 55 61
total_3
total_2
total_1

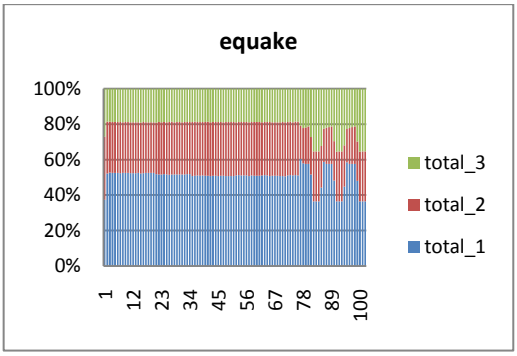

art
100%
80%
60%
40%
20%
0%
0 9 18 27 36 45 54 63 72 81 90 99
block3
block2
block1

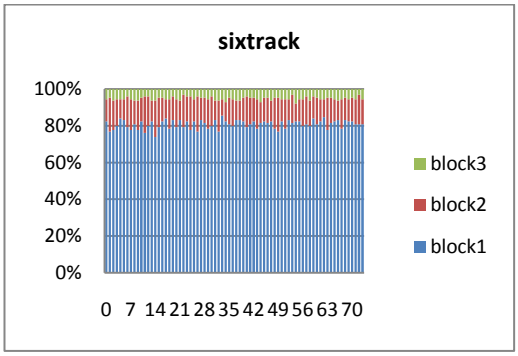

art
100%
80%
60%
40%
20%
0%
1 12 23 34 45 56 67 78 89 100
total_3
total_2
total_1

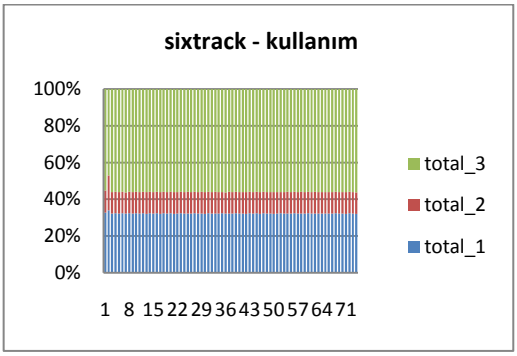

apsi
100%
80%
60%
40%
20%
0%
0 7 14 21 28 35 42 49 56 63 70 77
block3
block2
block1

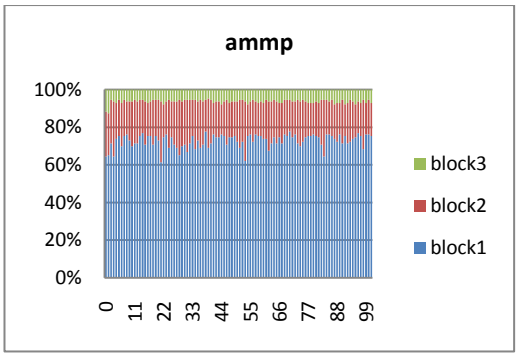

apsi
100%
80%
60%
40%
20%
0%
1 8 15 22 29 36 43 50 57 64 71 78
total_3
total_2
total_1

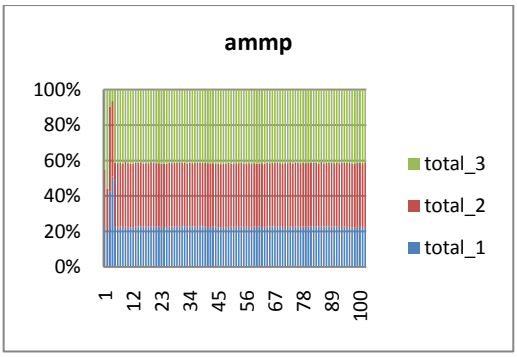

equake
100%
80%
60%
40%
20%
0%
0 11 22 33 44 55 66 77 88 99
block3
block2
block1

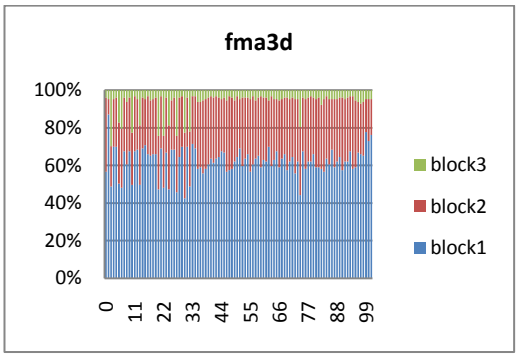

equake
100%
80%
60%
40%
20%
0%
1 12 23 34 45 56 67 78 89 100
total_3
total_2
total_1

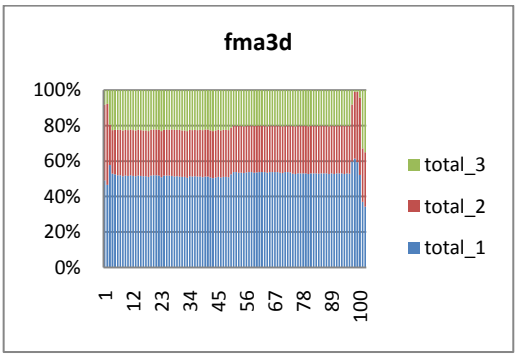

sixtrack
100%
80%
60%
40%
20%
0%
0 7 14 21 28 35 42 49 56 63 70
block3
block2
block1

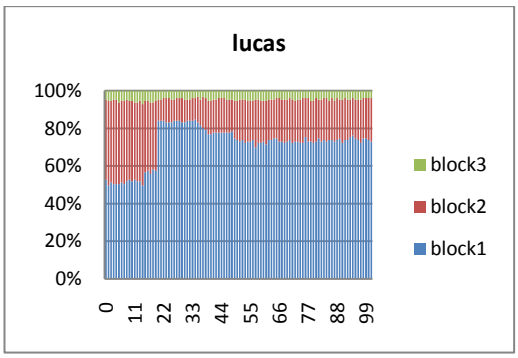

sixtrack - kullanım
100%
80%
60%
40%
20%
0%
1 8 15 22 29 36 43 50 57 64 71
total_3
total_2
total_1

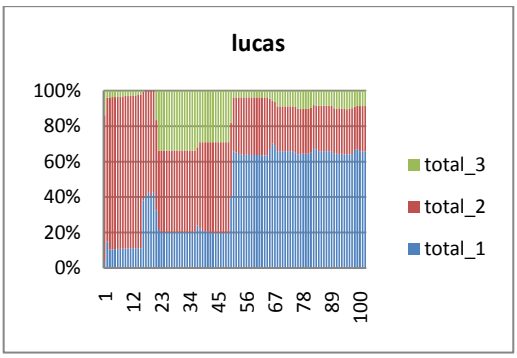

ammp
100%
80%
60%
40%
20%
0%
0 11 22 33 44 55 66 77 88 99
block3
block2
block1

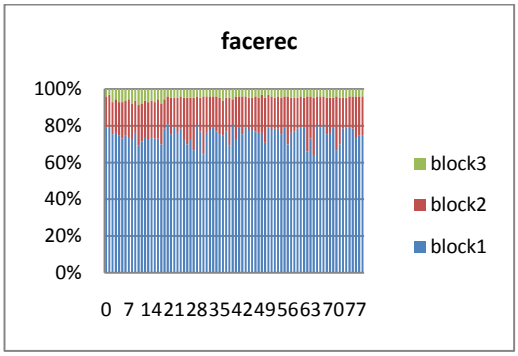

ammp
100%
80%
60%
40%
20%
0%
1 12 23 34 45 56 67 78 89 100
total_3
total_2
total_1

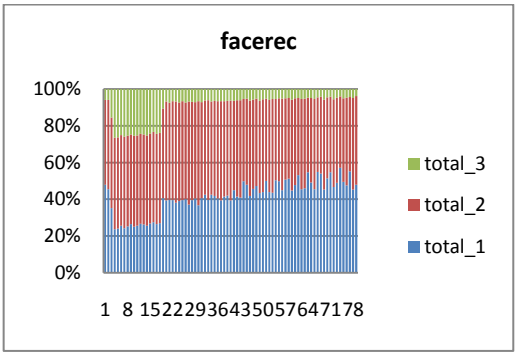

fma3d
100%
80%
60%
40%
20%
0%
0 11 22 33 44 55 66 77 88 99
block3
block2
block1


fma3d
100%
80%
60%
40%
20%
0%
1 12 23 34 45 56 67 78 89 100
total_3
total_2
total_1


lucas
100%
80%
60%
40%
20%
0%
0 11 22 33 44 55 66 77 88 99
block3
block2
block1


lucas
100%
80%
60%
40%
20%
0%
1 12 23 34 45 56 67 78 89 100
total_3
total_2
total_1


facerec
100%
80%
60%
40%
20%
0%
0 7 142128354249566370 77
block3
block2
block1


facerec
100%
80%
60%
40%
20%
0%
1 8 152229364350576471 78
total_3
total_2
total_1

BÖLÜM 6

6. SONUÇLAR

Bu tez sonucunda işlemcide güç tüketimini azaltmaya yönelik olarak yazmaç öbeğine müdahale edilmiş ve farklı iki tasarım yapılmıştır. Tasarımlardan birincisinde yazmaç öbeği farklı darlık değerlerine göre sabit parçalara ayrılmış ve gereksiz olan bitlerin yazılmaması ile enerji tasarrufu sağlanmıştır. İkinci tasarımda ise yazmaç öbeği satır sayıları program içinde gelen darlıklara göre değiştirilebilecek şekilde ayrı bloklara bölünmüştür.

Sabit bölümleme için 5 farklı bölümleme şekli ILP yardımıyla hesaplanmıştır ve bu bölümlemelerde % 50-60 arasında durağan ve devingen olarak kazanç sağlanmıştır. Değişken bölümlemede ise ilk olarak eşit bölünmüş olan yazmaç öbeği satırları program içerisinde değiştirilerek %50-60 durağan ve devingen enerjiden kazanç sağlanmıştır. Enerjiden kazançlar birbirine bu kadar yakın olmasına rağmen sabit bölümlemede başarımdan büyük bir kayıp vardır. Ortalama %7 olan kayıp kimi programlarda çok azken kimilerinde yükselmektedir. Bunun nedeni de sabit şekilde bölünmüş yazmaç öbeğinin her program için uygun olmayışıdır. Kimi programlar en geniş olan grubu sık kullanırken kimileri de en dar olan gruptan daha çok yazmaca ihtiyaç duyarlar. Değişken bölümlemede ise programın ihtiyaçlarına göre yazmaç sayısı değiştirildiği için başarım neredeyse hiç etkilenmez. Programların darlık kullanımına göre yazmaç öbeğinin darlık grubu dağılımının birbirine paralel şekilde ilerlemesi de yazmaçları açma kapama işleminin doğru şekilde yapıldığını gösterir. Kullanılmadıkları takdirde yazmaçların en dar gruba geçirilmesi de en yüksek enerji kazancının elde edilmesini sağlar.

Bu çalışmanın devamında farklı kontrol periyotları ve algoritmaları ile devingen yazmaç açma kapama ile daha iyi verim elde edilebilecek yazmaç öbeği tasarlanabilir. Uzun süre kullanılmayan yazmaçların en dar gruba alınması yerine tamamen kapatılması da enerjiden daha iyi bir kazanç sağlayacaktır. Ayrıca yazmaç öbeğine benzer yapıları bulunan yayın kuyruğu, yeniden sıralama belleği ve

yükle/sakla kuyruğuna da aynı yöntem uygulanarak işlemcide daha verimli enerji kullanımı sağlanabilir.

BÖLÜM 7

7. KAYNAKLAR

[1] Brooks, D., Tiwari, V., Martonosi, M., Wattch: a framework for architectural-level power analysis and optimizations, Computer Architecture, 2000. Proceedings of the 27th International Symposium, 83-94, 2000

[2] Brooks, D., Martonosi, M., Dynamically Exploiting Narrow Width Operands to Improve Processor Power and Performance, Proceedings of the 5th International Symposium on High Performance Computer Architecture, 13, Ocak 1999

[3] Smith, J.E., Sohi, G.S., The microarchitecture of superscalar processors, *Proceedings of the IEEE* , 83(12), 1609-1624, Aralık 1995

[4] Yourst, M.T., PTLsim: A Cycle Accurate Full System x86-64 Microarchitectural Simulator, Performance Analysis of Systems & Software, 2007. ISPASS 2007. IEEE International Symposium, 23-34, Nisan 2007

[5] Zyuban, V., Kogge, P., Split Register File Architectures for Inherently Lower Power Microprocessors, Power-Driven Microarchitecture Workshop, in conjunction with ISCA'98, Haziran 1998

[6] Ergin, O.; Balkan, D.; Ghose, K.; Ponomarev, D., Register Packing: Exploiting Narrow-Width Operands for Reducing Register File Pressure, Microarchitecture, 2004. MICRO-37 2004. 37th International Symposium, 304-315, Aralık 2004

[7] Tomasulo, R.M., An Efficient Algorithm for Exploiting Multiple Arithmetic Units, IBM Journal of Research and Development, 25-33, Ocak 1967

[8] Kondo, M., Nakamura, H., A small, fast and low-power register file by bit-partitioning, Proceedings of the 11th International Symposium on High-Performance Computer Architecture, 2005

[9] Lipasti, M.H., Mestan, B.R., Gunadi, E., Physical register inlining, Proceedings of the 31st Annual International Symposium on Computer Architecture, 2004

[10] Aggarwal, A., Franklin, M., Energy efficient asymmetrically ported register files, Proceedings of ICCD'03, 2003

[11] Loh, G.H., Exploiting data-width locality to increase superscalar execution bandwidth, Proceedings of the 35th International Symposium on Microarchitecture (MICRO), 2002

[12] Park, I., Powell, M., Vijaykumar, T., Reducing register ports for higher speed and lower energy, Proceedings of the International Symposium on Microarchitecture, 2002.

[13] Ergin, O., Exploiting narrow values for energy efficiency in the register files of superscalar microprocessors, 16th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS'06), Lecture Notes in Computer Science (LNCS), Springer-Verlag, 2006

[14] Hu, J., Wang, S., Ziavras, S.G., In-register duplication: Exploiting narrow-width value for improving register file reliability, Proceedings of the International Conference on Dependable Systems and Networks (DSN-2006), Philadelphia, PA, 2006

[15] Cruz, J.L., Gonzalez, A., Valero, M., N.P. Topham, Multiple-bankedregister file architectures, Proceedings of the 27th Annual International Symposium on Computer Architecture, 2000

[16] Tseng, J., Asanovic, K., Banked multiported register files for highfrequency superscalar microprocessors, 30th International Symposium on Computer Architecture (ISCA-30), San Diego, CA, 2003

[17] Wang, S., Yang, H., Hu J., Ziavras S.G., Asymmetrically Banked Value-Aware Register Files, VLSI, 2007. ISVLSI '07. IEEE Computer Society Annual Symposium, 363-368, Mart 2007

[18] Manne, S., Klauser, A., Grunwald, D., Pipeline gating: speculation control for energy reduction, Computer Architecture, 1998. Proceedings. The 25th Annual International Symposium on , 132-141, Haziran 1998

[19] Ponomarev, D., Kucuk, G., Ghose, K., Reducing power requirements of instruction scheduling through dynamic allocation of multiple datapath resources, Microarchitecture, 2001. MICRO-34. Proceedings. 34th ACM/IEEE International Symposium, 90-101, Aralık 2001

[20] Nalluri, R., Garg, R., Panda, P.R., Customization of Register File Banking Architecture for Low Power. Proceedings of the 20th international Conference on VLSI Design Held Jointly with 6th international Conference: Embedded

Systems. VLSID. IEEE Computer Society, 239-244, Washington, DC, Ocak 2007

[21] Saito, T., Maeda, M., Hironaka, T., Tanigawa, K., Sueyoshi, T., Aoyama, K., Koide, T., Mattausch, H.J., Design of superscalar processor with multi-bank register file, Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium, 3507-3510, Mayıs 2005

[22] Johnson, W. M. 1989 Super-Scalar Processor Design. Technical Report. UMI Order Number: CSL-TR-89-383., Stanford University.

[23] McFarling, S. 1993 Combining Branch Predictors. Technical Note. UMI Order Number: WRL-TN-36, Western Research Laboratory.

[24] Gwennap, L., Intel's P6 Uses Decoupled Superscalar Design, Microprocessor Report, Vol. 9, No. 2, pp. 9-15

[25] Villa, L., Zhang, M., and Asanović, K., Dynamic zero compression for cache energy reduction. In Proceedings of the 33rd Annual ACM/IEEE international Symposium on Microarchitecture (Monterey, California, United States). MICRO 33. ACM, 214-220, New York, NY, 2000

[26] Khasawneh, S.T., Ghose, K., An Adaptive Technique for Reducing Leakage and Dynamic Power in Register Files and Reorder Buffers, Lecture Notes in Computer Science, Volume 3728, 498 – 507 (PATMOS 2005), Ağustos 2005

[27] "List of NP complete problems" http://en.wikipedia.org/wiki/List_of_NP-complete_problems erişim tarihi: 13.07.2009

[28] "Virtuoso Spectre Circuit Simulator" http://www.cadence.com/products/cic/spectre_circuit/pages/default.aspx erişim tarihi: 13.07.2009

[29] Patterson, D.A., Hennessy, J.L., Computer Architecture: a Quantitative Approach, Morgan Kaufmann Publishers Inc. 1990

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ÖZSOY, Meltem

Uyruğu : T.C.

Doğum tarihi ve yeri : 26.07.1986 Ankara

Medeni hali : Bekar

Telefon : 0 (312) 425 37 60

e-mail : mozsoy@etu.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	TOBB Ekonomi ve Teknoloji Üniversitesi/Bilgisayar Müh.	2008

İş Deneyimi

Yıl	Yer	Görev
Ocak – Nisan 2006	Nabay Tekstil	Ortak Eğitim
Ocak – Nisan 2007	Havelsan	Ortak Eğitim
Mayıs – Ağustos 2008	UPA Makine	Ortak Eğitim

Yabancı Dil

İngilizce

Yayınlar

Telsiz Bilgisayar Mimarisi, Oğuz Ergin, Yusuf Onur Koçberber, Meltem Özsoy GömSis 2008, İstanbul, Kasım 2008

Bilgisayar Mimarisi Temelleri: Kitap, Meltem Özsoy, Oğuz Ergin (Hazırlanma aşamasında)