T.C.
ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
YÜKSEK LİSANS TEZİ

TEKNOLOJİK YETERSİZLİKLERİN KOMUT TLB'Sİ ÜZERİNDEKİ ETKİSİNİN FİZİKSEL ADRESLERİN DİREKT ÜRETİLMESİ SURETİYLE AZALMASI

Mahir TÜRKCAN
BİLGİSAYAR MÜHENDİSİĞİ ANABİLİM DALI
Tezin Sunulduğu Tarih: **29.05.2009**

Danışman:
Yrd. Doç. Dr. İsmail KADAYIF

ÇANAKKALE

YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

MAHİR TÜRKCAN tarafından **YRD. DOÇ. DR. İSMAİL KADAYIF** yönetiminde hazırlanan **"TEKNOLOJİK YETERSİZLİKLERİN KOMUT TLB'Sİ ÜZERİNDEKİ ETKİSİNİN FİZİKSEL ADRESLERİN DİREKT ÜRETİLMESİ SURETİYLE AZALMASI "** başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

Yrd. Doç. Dr. İbrahim BULUT

Yönetici

Yrd. Doç. Dr. İsmail KADAYIF

Jüri Üyesi

Yrd. Doç. Dr. İbrahim TÜRKYILMAZ

Jüri Üyesi

Sıra No:……

Tez Savunma Tarihi: 29/05/2009

Prof. Dr. Neşet AYDIN

Müdür

Fen Bilimleri Enstitüsü

İNTİHAL (AŞIRMA) BEYAN SAYFASI

Bu tezde görsel, işitsel ve yazılı biçimde sunulan tüm bilgi ve sonuçların akademik ve etik kurallara uyularak tarafımdan elde edildiğini, tez içinde yer alan ancak bu çalışmaya özgü olmayan tüm sonuç ve bilgileri tezde kaynak göstererek belirttiğimi beyan ederim.

TEŞEKKÜR

Tezimin hazırlanması sürecinde her zaman her konudaki destek, yardım ve yönlendirmeleri için değerli hocam Yrd. Doç. Dr. İsmail KADAYIF'a, tecrübelerini ve yardımlarını esirgemeyen hocam Arş. Gör. Selçuk KOYUNCU'ya, tez çalışmalarım boyunca beni sabırla destekleyen, güvenimi kaybetmeme hiç bir zaman izin vermeyen sevgili eşim Aslıhan TÜRKCAN'a ve bugünlere gelmemde her türlü çabayı gösteren aileme tüm kalbimle teşekkürlerimi sunarım.

Mahir TÜRKCAN

SİMGELER VE KISALTMALAR LİSTESİ

TLB (Translation Lookaside Buffer) : Adres Dönüştürme Önbelleği
iTLB (Instruction Translation Lookaside Buffer) : Komut Adres Dönüştürme Önbelleği
dTLB (Data Translation Lookaside Buffer) : Veri Adres Dönüştürme Önbelleği
CFR (Context Frame Register) : İçerik Tutucu Yazmaç
SRAM (Static RAM) : Statik RAM
DRAM (Dynamic RAM) : Dinamik RAM
SSTA (Statistical Static Timing Analysis) : İstatistiksel Durağan Zamanlama Analizleri
nm : Nano metre
Cache: Önbellek
I-Cache (Instruction Cache) : Komut Önbelleği
L1 (Level 1) : Birinci Seviye
L2 (Level 2) : İkinci Seviye
Register: Yazmaç
PC (Program Counter) : Program Sayıcı
Fetch: Alıp Getirmek
VI (Virtually Indexed) : Sanal Olarak İndekslenmiş
VT (Virtually Tagged) : Sanal Olarak Etiketlenmiş
PI (Physically Indexed) : Fiziksel Olarak İndekslenmiş
PT (Physically Tagged) : Fiziksel Olarak Etiketlenmiş
P (Perfect) : Kusursuz
D (Delayed) : Ertelenmiş
O (Oracle) : Kahin
CFG (Control Flow Graph) : Kontrol Akış Diyagramı
ISA (Instruction Set Architecture) : Komut Küme Mimarisi
TR(Translation Register) : Dönüşüm Yazmacı
CFR (Context Frame Register) : Kontekst Çerçeve Yazmacı

ÖZET

TEKNOLOJİK YETERSİZLİKLERİN KOMUT TLB'Sİ ÜZERİNDEKİ ETKİSİNİN FİZİKSEL ADRESLERİN DİREKT ÜRETİLMESİ SURETİYLE AZALMASI

Mahir TÜRKCAN
Çanakkale Onsekiz Mart Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans Tezi
Danışman: Yrd. Doç. Dr. İsmail KADAYIF
29.05.2009, 39

Sürekli küçülen transistorlarla işlemci tasarımı kanal uzunluğu, eşik voltajı ve kapı oksidi kalınlığı gibi kritik transistor parametrelerinde dramatik değişmelere neden olur. Bu değişmeler erişim gecikmelerinde dalgalanmalara ve aynı tasarlanmış bileşenlerin farklı miktarlarda güç tüketimleri gibi sorunlara yol açar. İşlem değişiminin Translation Lookaside Buffer (TLB, Adres Dönüştürme Önbelleği) üzerindeki potansiyel etkisi, aynı TLB' deki farklı girişlerin, farklı erişim gecikmelerine sahip olması şeklindedir. Bu problemi çözmenin basit bir yolu, bütün TLB girişlerinin en yavaş TLB girişinin gecikmesine sahip olduğunu varsayan en kötü gecikme modelini kullanmaktır. Bu TLB tasarımındaki mantığı basitleştirirken, aynı zamanda büyük performans kayıplarına sebep olur ve bu performans kayıpları gelecekte daha gelişmiş teknolojiler için daha da artacaktır. Bu çalışmada, en kötü gecikme modeline alternatif bir yöntem sunup, bunu irdeleyeceğiz. Önerdiğimiz yaklaşım, bir sayfa için sanal-fiziksel sayfa çevirisini bir defa yapar ve bunu Context Frame Register (CFR) denilen özel bir registerda saklar ve bu çeviriyi bir daha TLB' ye gitmeden yürütme aynı komut sayfasında olduğu sürece yeniden kullanır. Bu yaklaşım iTLB ziyaretlerini minimuma indirdiği için, işlem değişiminin performansa etkisi en kötü iTLB gecikmesi yaklaşımının kullanıldığı durumda bile dramatik olarak azaltılabilir.

Anahtar Sözcükler: İşlem Değişimi, Adres Dönüşümü, Komut TLB.

ABSTRACT

MITIGATING THE EFFECT OF TECHNOLOGICAL INADEQUACY ON INSTRUCTION TLB PERFORMANCE VIA GENERATING PHYSICAL ADDRESSES DIRECTLY

Mahir TÜRKCAN

Canakkale Onsekiz Mart University

Graduate School of Science and Engineering

Thesis of Master of Science for Chair of Computer Engineering

Advisor: Ph. D. Ismail Kadayif

29.05.2009, 39

Processor design with ever-smaller transistors leads to dramatic variations in critical transistor parameters such as channel length, threshold voltage, and gate oxide thickness. These variations manifest themselves as fluctuations in access latencies and power consumptions of the identically-designed components. A potential impact of process variation on a translation lookaside buffer (TLB) is that the different entries of the same TLB can have different access latencies. A simple way of handling this problem is to adopt the worst case latency paradigm, i.e., assume that all the TLB entries have the latency of the slowest TLB entry. While this makes the design of the logic that interacts with the TLB simpler, it can also cost significant performance penalty and we can expect this penalty to increase in the future as we go to the finer process technologies. In this study, we propose and evaluate an alternate scheme to this worst case latency paradigm. Our approach performs the virtual-to-physical page translation for a page once and stores it in a special register, which is referred to the Context Frame Register (CFR), and reuses this translation (without going to the iTLB) as long as the execution remains within the same instruction page. Since this approach minimizes the number of iTLB visits, the overall impact of process variation can be reduced dramatically; even if we assume the worst case latency if/when the iTLB is visited.

Keywords : Process Variation, Address Translation, Instruction TLB (Translation Lookaside Buffer)

İÇERİK

Sayfa

BÖLÜM 1

GİRİŞ

Artan parametre değişkenliği, bugünün yarı iletken teknolojisinde karşılaşılan en ciddi sorunlardan birisidir. Bu değişkenlik, hem mikroişlemci yongası imalatında hem de çalışma zamanında kendini gösterebilir. Bu problemin, mikroişlemci teknolojisinde daha küçük boyutlara inildikçe daha da ciddileştiği tahmin edilmektedir. Genel olarak, parametre değişmesinin üç kaynağı vardır. Bunlar işlem, voltaj ve sıcaklık değişmeleridir. İşlem değişimi, planlanan veya tasarlanan devre parametrelerinin hedef değerlerinden örneğin kanal genişliği, kapı oksidi kalınlığı, eşik voltajı veya dopantların rastgele yerleşimi gibi sapmalar olarak tanımlanabilir. Voltaj değişmeleri, güç kaynağı dağıtımlarındaki dalgalanmalardan ve anahtarlama aktivitelerinin değişen frekanslarından ileri gelir. Diğer taraftan sıcaklık değişmeleri, farklı işlevsel birimlerdeki ısı yayılımı ve soğutma materyallerinin aynı özelliklere sahip olmayışından kaynaklanır. İşlem değişmesi statik ve yonga üretiminde ortaya çıkarken, voltaj ve sıcaklık değişmesi genel olarak dinamiktir ve çeşitli çalışma yükleri altında program yürütümü esnasında meydana gelir. Bu çalışmamız işlem değişimi ile sınırlı olacaktır.

İşlem değişmeleri, intra-die aygıtları arasında olabileceği gibi inter-die aygıtları arasında da olabilir (Nassif, 2001). Bu değişmeler, devrenin güç ve performans davranışını değiştirebilir ve tasarım şartnameleriyle uyuşmayan fiziksel gerçekleştirmelere yol açabilir. Yapılan bir çalışmada (Borkar ve ark., 2003) işlem değişimlerinden dolayı, aynı waferdan üretilen farklı yongalarda 20 kat farklı sızıntı enerji tüketimi olabileceğinden bahsedilmektedir. İşlemci teknolojisi daha küçük boyutlara ilerledikçe, işlem değişmesinin getirdiği sorunlar dramatik olarak büyümektedir. Örneğin 130 nm teknolojisinde bu değişimler işlevsel yonga frekansında %30 dalgalanma ve sızıntıda 5 kat artış oluşturabilirken (Datta ve ark., 2006), 90 nm silikon teknolojisinde 20 kat sızıntı artışı rapor edilmektedir (Borkar ve ark., 2003). İşlem değişimi ile mücadele için devre düzeyinde çözümler olduğu gibi (Chen ve Naffziger, 2003; Gregg ve Chen, 2004; Tschanz ve ark., 2002), mimari düzeyde de çözümlerin olabileceğini görmek önemlidir ve eğer bu çözümler varsa devre düzeyindeki çözümlerle nasıl etkileşim yaptığının araştırılması gerekir.

Mikroişlemci dizaynında işlevsel frekans, en yavaş pipeline aşamasının gecikmesi tarafından belirlenir. Genellikle en yavaş pipeline aşamaları; SRAM'lerin iri doğal yapısından dolayı SRAM erişimi gerektiren aşamalar olduğu için (Humenay ve ark., 2006), L1 önbellek erişimi, TLB erişimi ve büyük tampon erişimi kritik yolu belirler. Bu yapılarda üretim işlemi sırasında ortaya çıkan herhangi bir gecikme işlevsel frekansı olumsuz olarak etkiler. Diğer taraftan, başlıca iki sebepten dolayı işlem değişimlerinden kaynaklanan kritik cihaz parametrelerindeki değişmeler SRAM hücrelerinde ciddi olabilir. Bu sebeplerden ilki, mikroişlemciler yoğunluk sebeplerinden dolayı en küçük boyuttaki transistorlarla tasarlanırlar (Papanikolau ve ark., 2005), bu da cihaz kalitesinin kontrolünde zorluklara yol açar. İkinci olarak, SRAM yapıları düşük eşik gerilimleri kullanılarak minimum gecikme öngörecek şekilde tasarlandıkları ve çok sayıda bağımsız kritik yollar içerdiklerinden SRAM'lerin farklı kısımları arasında oldukça ciddi işlem değişimleri olabilmektedir (Ozdemir ve ark., 2006). SRAM hücrelerindeki bu değişimler çeşitli güvenirlik kaygılarını açığa getirir, örneğin daha uzun erişim zamanı ve stabil olmayan okuma/yazma işlemleri gibi (Agarwal ve ark., 2005; Chen ve ark., 2005). Bu yapılardaki herhangi bir erişim gecikmesi, muhtemelen en yavaş pipeline aşamasının gecikmesini artırarak sistemin işlevsel frekansının yavaşlamasıyla sonuçlanır.

İşlem değişiminin getirdiği dezavantajları bertaraf etmek için devre seviyesinde birçok çalışmalar mevcuttur (Borkar ve ark., 2003; Humenay ve ark., 2006; Raj ve ark., 2004; Sarangi ve ark., 2007; Sinha ve ark., 2005). Bu çalışmalar genel olarak belirli modeller kullanırlar. Örneğin işlem değişiminin devre gecikmeleri ve aşırı sızıntı enerji etkilerini analiz etmek ve değişime daha duyarlı bileşenleri tespit etmek için İstatistiksel Durağan Zamanlama Analiz (SSTA) algoritmaları (Devgan ve Kashyap, 2003; Zhan ve ark., 2005) kullanılır. Daha sonra bu kolayca etkilenebilir kısımlar ya bazı teknikler kullanılarak değiştirilir; örneğin kapı boyutlandırması (gate sizing) (Sinha ve ark., 2005), uyarlamalı gövde sapmaları (adaptive body bias) (Greg ve Chen, 2004; Tschanz ve ark., 2002) ve uyarlamalı besleme voltajı (adaptive supply voltage) (Chen ve Naffziger, 2003) gibi ya da işlem değişiminin etkisini azaltmak için program yürütme esnasında devre dışı bırakılırlar. Bu tekniklerin temel karakteristik özelliği, ekstra devrelere ihtiyaç duymalarıdır (Vergos ve Nikolos, 1995). Diğer yandan bu çalışmanın amacı, ekstra donanım kullanmadan veya çalışma zamanını artırmadan mimari düzeyinde farklı TLB girişlerindeki erişim gecikmelerindeki farklara bir çözüm getirmektir.

Bu tezde ele alınan problem, komut TLB erişimlerindeki işlem değişimlerinin etkisini azaltmaktır. Performans açısından baktığımızda TLB mikroişlemciler açısından çok kritik

bir bileşendir, çünkü her adres dönüşümünde ziyaret edilir (hem komut hem de veri erişiminde). Komut getirme (fetch) pipeline aşamasında TLB erişimi ve L1 önbellek erişiminin ikisi de kod çözme (decode) aşamasına geçmeden önce L1 komut önbelleğinden komutların getirilip bir tampon bellekte tutulması için gereklidir. Kullanılmakta olan önbellek arama mekanizmasına bağlı olarak TLB erişimi program performansını etkileyen kritik bir yolda olabilir. Örneğin sanal olarak indekslenmiş fiziksel olarak etiketlenmiş (virtually-indexed, physically–tagged) önbellek erişim mekanizmasında ön bellek bloklarının etiket karşılaştırması TLB erişimi tamamlanana kadar başlayamaz. Bundan dolayı, işlem değişiminden kaynaklanan TLB erişimindeki herhangi bir gecikme TLB erişiminin önbellek indekslemesinden daha geç tamamlanmasına sebep olur; bu da seçili önbellek bloğu için etiket karşılaştırmada ve veri erişiminde gecikmelere yol açar.

Bizim çalışmamızın getirdiği temel katkı, en kötü gecikme modeli (bütün TLB girişlerinin en yavaş TLB girişinin gecikmesine sahip olduğu varsayılır) için alternatif bir yöntem önerilmesidir. Önerdiğimiz çözüm, komut TLB'si ve CPU arasına Context Frame Register (CFR) ismini verdiğimiz özel bir registerın yerleştirilmesini öngörmektedir. İlk olarak CFR kavramı daha önceki bir çalışmada TLB'lerin dinamik enerji harcamasını optimize etmek için kullanılmıştır (Kadayif ve ark., 2002). Öte yandan bu çalışmada CFR registerını, TLB erişim gecikmelerini azaltmak için kullanacağız. Bu register en son kullanılan sanal-fiziksel adres dönüşümünü tutar. Sonraki komutun ihtiyaç duyduğu adres dönüşümü önceki adres dönüşümüyle aynı ise bu dönüşüm TLB'ye gidilmeden bu registerdan elde edilir. Komut erişimleri yüksek seviyeli yerellik gösterdiği için program kontrol akışı mevcut sayfa sınırının dışına çok az sıklıkta çıkar. Dolayısıyla bu register gerekli adres dönüşümünü çoğu zaman sağlayabilir. Sonuç olarak, registerde bulunamama ve TLB'ye gitme erişimleri için en kötü erişim gecikmesinin olduğunu varsaysak bile, bu tarz erişimler çok az olacağı için (komut TLB'sine gitme) performans üzerindeki genel etkisi çok yüksek olmayacaktır.

Yukarda bahsi geçen yöntemi ve ileride açıklama getireceğimiz diğer üç yöntemi bir simülasyon platformunda simüle ettik ve SPEC2000 ile deneyler yaptık (22 adet SPEC2000 CPU benchmarkı kullandık). Elde ettiğimiz sonuçlar, önerdiğimiz yöntemin en kötü gecikme varsayımını esas alan yönteme ve erişilecek olan TLB girişinin gecikmesini önceden öngörmeye dayalı yaklaşıma oranla çok daha az bir performans kaybı ortaya koyduğunu göstermektedir. Ayrıca farklı komut sayfası büyüklükleri için deneysel sonuçlar sunduk ve CFR'ın sayfa büyüklüğü arttıkça daha etkili olduğunu gözlemledik.

Bu tezin geri kalan kısmı şu şekilde yapılandırılmıştır. Tezin 2. Bölümünde modern işlemcilerdeki önbellek ve TLB araması ile fiziksel adresin direkt olarak oluşturulması fikirlerine ve bu alanda daha önce yapılmış çalışmalara yer verilmektedir. 3. Bölümde bu tezde önerilen yöntem ve kullanılan temel konfigürasyon parametreleri. 4. Bölümde araştırma bulguları ve bunlarla ilgili değerlendirmeler ile sanal olarak indekslenmiş, sanal olarak etiketlenmiş önbellek erişim mekanizması kullanıldığında önerdiğimiz yöntemin değerlendirmesi ve yöntemimizin veri TLB'si için değiştirilerek nasıl kullanılabileceğinden bahsedilmektedir. Son olarak, 5. Bölümde sonuçlar üzerinde tartışma ve önerilerden bahsedilmektedir.

BÖLÜM 2

ÖNCEKİ ÇALIŞMALAR

2.1 Ön Bellek ve TLB Araması

Bu tezin geri kalan kısmında aksi belirtilmedikçe TLB, komut TLB'si olarak (iTLB) ve önbellek (cache) L1 komut önbelleği (iL1) olarak kullanılacaktır. TLB komut sayfaları için en son zamanlarda kullanılan sanal-fiziksel adres dönüşümlerini tutan bir önbellektir. Önbellek erişim mekanizmasına bağlı olarak TLB, her bir sanal sayfa erişiminde ilgili fiziksel sayfa numarasını elde etmek için erişilebilir. Diğer taraftan önbellekler, hızlı CPU ile yavaş DRAM arasındaki hız farkını azaltmak için sık kullanılan komut ve verileri saklarlar ve CPU'ya bunlar için hızlı erişim imkânı sağlar. Önbelleklerde arama yapılma şeklinin ön bellek performansı üstüne büyük etkisi vardır. Bu da tüm sistem performansının etkiler. TLB araması önbellek aramasına benzer. Bir setin (kümenin) indekslemesine ve daha sonra bu setteki bloklara ait etiketlerde karşılaştırma işlemine ihtiyaç duyulur. İndeksleme veya etiket karşılaştırma hem sanal adresle hem de fiziksel adresle yapılabilir ve bu da dört farklı arama mekanizması sonucunu doğurur: Virtually-Indexed, Virtually-Tagged (VI-VT); Virtually-Indexed, Physically-Tagged (VI-PT); Physically-Indexed, Physically-Tagged (PI-PT); Physically-Indexed, Virtually-Tagged (PI-VT). Bu tezde daha yaygın kullanımlarından dolayı sadece VI-PT ve VI-VT önbellekler üstünde duracağız. Fakat yaklaşımımızı diğer önbellek arama tasarımları ile çalışacak şekilde genişletmek de mümkündür.

Bu çalışmada önce VI-PT arama mekanizmasına göre çalışan önbellekler üzerinde duracağız ve daha sonra önbelleğin VI-PT arama mekanizması kullanıldığı durumda CFR'nin sunduğu hususlardan bahsedeceğiz. VI-PT önbellek arama mekanizmasının çalışması şu şekildedir. Sanal adres önbelleği indekslemek için kullanılır ve fiziksel adresi elde etmek için eş zamanlı olarak TLB'de arama yapılır. Fiziksel adresler etiket karşılaştırılmasında kullanıldığı için önbellek bloklarının etiket karşılaştırma işlemi TLB erişimi tamamlanana kadar beklemek zorundadır. Sonuç olarak fiziksel adresin etiketi seçilen set içindeki blokların etiketleri ile karşılaştırılır. Modern işlemcilerde önbellek erişiminin ve TLB erişiminin aynı cycle (çevrim) içinde tamamlanması beklenir. Bu sistemin performansı açısından çok önemlidir.

Eğer bazı TLB girişleri diğerlerinden daha uzun erişim zamanı gerektirirse (işlem değişiminin sonucu olarak), TLB erişim zamanı düzensizlik gösterir. Bu çalışmanın geri

kalanında kusurlu TLB girişini, üretim sırasında işlem değişiminden etkilenen ve hedeflenen erişim gecikmesinden daha fazla gecikmeye sahip giriş anlamında kullanacağız. Kusurlu TLB girişlerinin erişim zamanı sorunlarını çözmek için farklı çözüm yolları vardır. Bu problemi çözmek için basit bir yöntem, bütün TLB girişlerinin en yavaş TLB girişinin gecikmesine sahip olduğunu varsayarak en kötü gecikme paradigmasını benimsemektir. Bunun pratikteki anlamı, TLB girişi kusurlu olsun veya olmasın bütün TLB girişleri en yavaş TLB girişinin gecikmesine sahip olduğunu varsaymaktır. Bu husus TLB ile etkileşen lojiğin tasarımını basitleştirse de en yavaş pipeline aşamasını uzatarak sistemde büyük oranda gecikmelere yol açabilir. Daha da önemlisi işlemci teknolojileri ilerledikçe bu gecikmelerin daha da ciddileşebileceğidir (Bowman ve ark., 2002; Zochowski ve ark., 2004). Bunun sonucu olarak en kötü gecikme yaklaşımının performans üzerindeki etkisi öyle şiddetli olabilir ki bu da devre tasarım özelliklerinin karşılanamamasına sebep olabilir. Bu durumda mikroişlemci çöpe atılıp verim kayıpları ortaya çıkabilir. Probleme ikinci bir çözüm yolu, adres dönüşümü kusurlu olan girişler için CPU'ya bazı senkronize edici lojik devreler ekleyerek komutların işlemesini geciktirmektir. Bu çözümde sonraki cycle'da komutların fetch işlemini geciktirmek için izleyen komutun fetch pipeline aşamasına bir geciktirici buble sokmamız gerekir. Bu çözüm, erişim gecikmesi bağlamında donanıma uniform olmayan erişim izni verir. İlk çözüm uygulama açısından basit olsa da ikinci çözüm daha iyi performansa sahiptir; çünkü ikinci çözümde sadece kusurlu girişler gecikmelere sebep olur. Sonraki bölümlerde biz, bu iki alternatif yöntemin performansını adres dönüşümlerini elde etmek için mümkün olduğunca CFR'yi kullanan bizim yöntemin performansıyla kıyaslayacağız.

CFR'nin işlem değişiminden etkilenebileceği ve zamanlama gereksinimlerini karşılayamayabileceği düşünülebilir. Burada, bu registerın işlem değişimine karşı bağışıklı olduğunu varsayıyoruz. Bu varsayımı TLB'nin bazı devre parametrelerinin ayarlanması veya adaptif gövde biası, besleme geriliminin düzenlenmesi gibi (Chen ve Naffziger, 2003) bazı devre yaklaşımlarıyla gerçekleştirebiliriz. CFR geçerli sanal-fiziksel (Virtual–Physical) dönüşümü tutar. Az sonra bahsedileceği üzere, kontrol akışı mevcut komut sayfasında kaldığı sürece, dönüşüm güvenli olarak TLB yerine CFR' dan elde edilebilir.

2.2 Fiziksel Adresin Direkt Olarak Oluşturulması

Komut akışları bağlamında CFR kavramındaki benzer felsefe VAX mimarisinde de kullanılmaktadır. VAX, TLB arama gecikmelerini azaltmak için aktif komut sayfası için adres dönüşümünü tutan bir register kullanır (Strecker, 1978). TLB'ye gitmeksizin veri

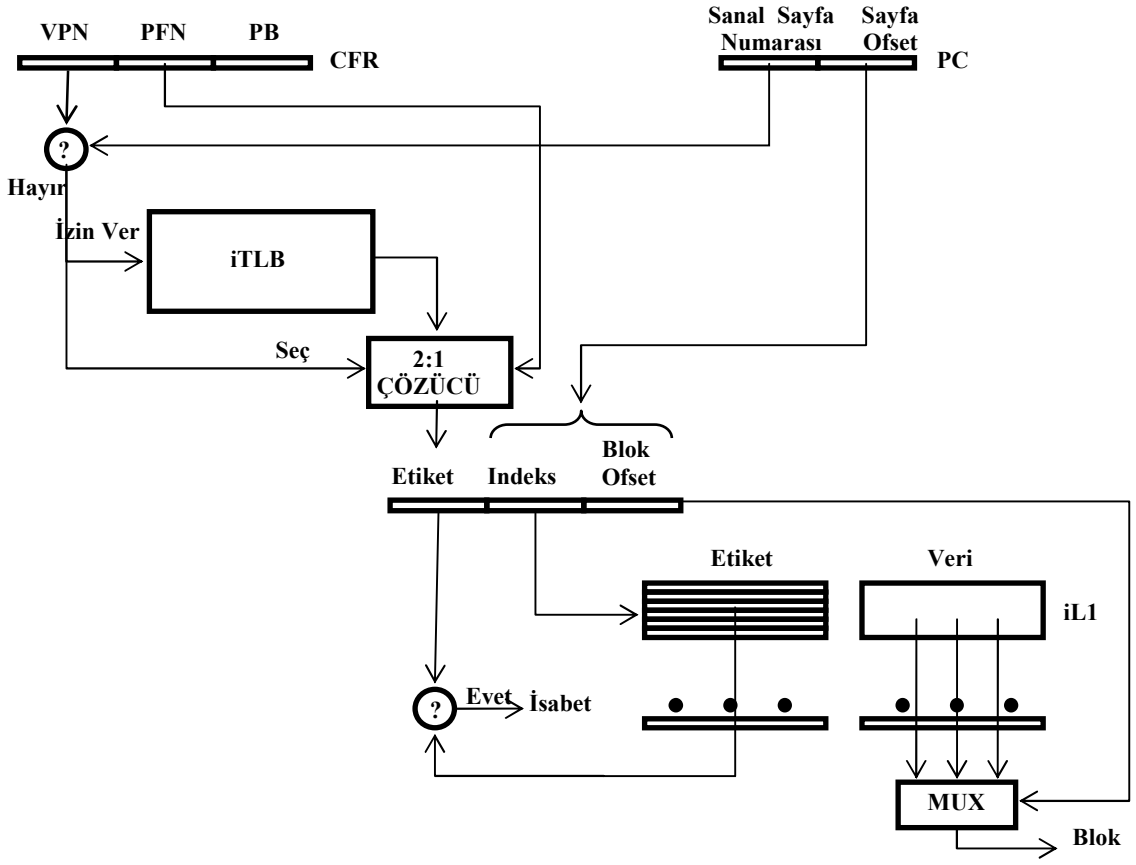
başvuruları adreslerinin elde edilmesini öneren daha önce yapılmış bir çok çalışmalar mevcuttur (Chiueh ve Katz, 1992; Knight Rosenfeld., 1984; Maddock ve ark., 1981). Daha önceki birkaç çalışmada fiziksel adreslerin direkt olarak üretilebileceği fikri kullanıldı (Kadayif ve ark., 2002; Kadayif ve ark., 2007). Bu çalışmalardan birinde (Kadayif ve ark., 2002) iTLB dinamik enerjisi minimize edilmeye çalışıldı, diğerinde ise CFR benzeri birkaç register kullanılarak dTLB'nin harcadığı dinamik enerji azaltılmaya çalışıldı (Kadayif ve ark., 2007).

Bu iki çalışmanın arkasında yatan ana fikir, iTLB/dTLB'ye erişim yerine fiziksel adresleri CFR registeri veya registerlerinden elde etmektir. Böylece TLB dinamik erişim sayısı azaltılabilmektedir. CFR registeri için erişim başına düşen dinamik enerji iTLB/dTLB'ninkinden çok daha küçük olduğu için bu çalışmalarda çok büyük dinamik enerji kazançlarından bahsedilmektedir. Bu çalışmalar ile karşılaştırdığımızda bu tezde CFR registeri, komut TLB gecikmesinde işlem değişiminin etkisini azaltmak için kullanıyoruz. Bu çalışmadaki CFR kavramı, (Kadayif ve ark., 2002) de tanımlanan kavram ile benzerdir ve formatı şu şekildedir.

[<Sanal Sayfa Numarası> <Fiziksel Frame Numarası> <Koruma Bitleri>]

Fiziksel sayfa numarasının ve buna karşılık gelen TLB girişinin koruma bitlerinin de CFR'de saklandığını varsayıyoruz. CPU sanal bir adres oluşturduğunda, oluşturulan adresin sanal sayfa numarası CFR'de saklanan ile karşılaştırılır. Eğer uyuşurlarsa fiziksel sayfa numarası CFR'den elde edilir. Uyuşmazlarsa sayfa dönüşümü TLB'den yapılır. Ancak komut yerelliği genel olarak uygulamalarda çok yüksek olduğu için yürütüm akışı büyük bir ihtimalle aynı sayfada kalır ve adres dönüşümü için bir TLB erişimine gerek kalmaz.

Sıradaki komut getirimini (fetch) VI-PT L1 arama mekanizması kullanıldığında şöyle gerçekleştirebiliriz. Sanal adres, CFR'deki sanal sayfa numarasının PC (Program Counter)'nin sayfa ofset bitleri ile birleştirilmesi suretiyle oluşturulur ve L1 önbelleği bu adresteki indeks bitleri ile indekslenir. CFR'nin fiziksel frame numarasını, Program Counter'in (PC, Program Sayacı) sayfa ofset bitlerine birleştirerek fiziksel adresi üretiriz. Bu fiziksel adresin etiket (tag) kısmı L1'de indekslenen setteki taglarla kıyaslanır. Bu arama mekanizması Şekil 1'de gösterilmektedir.



Şekil 1. VI–PT önbellek arama mekanizması kullanıldığında CFR ile L1 komut önbelleği araması.

BÖLÜM 3

MATERYAL VE YÖNTEM

3.1 Kullanılan Yöntemler

Biz, bu bölümde değerlendirilecek olan dört yöntemi açıklayacağız.

3.1.1 Kusursuz Yöntem (P)

Bu yöntem, TLB'nin işlem değişiminden kaynaklanan herhangi bir kusurlu girişe sahip olmadığı kabulüne dayanan ideal durumu temsil eder. Her bir TLB girişine erişim aynı miktar zaman zarfında tamamlanır ve böylece TLB erişimi ve L1 erişimi aynı cycle içinde tamamlanır. Geri kalan yöntemlerin performansları ideal durumun performansıyla kıyaslanacağından ideal durumla ilgili deneyler yapmayı uygun gördük.

3.1.2 Geciktirilmiş Yöntem (D)

Bu en kötü gecikme modeline karşılık gelir. Bu, TLB'nin işlem değişiminden etkilendiği yani mükemmel girişlere göre daha fazla erişim zamanı gerektiren bazı kusurlu TLB girişlerinin mevcut olduğu duruma karşı gelir. Farklı girişler arasındaki erişim gecikmesi uyumsuzluklarını çözmek için bu yöntem tasarımda en kötü erişim gecikmesini kullanır; yani bütün TLB girişleri için en yavaş TLB girişinin erişim gecikmesi kullanılır. Sonuç olarak, bu yöntemde her bir TLB erişimi kritik yol üzerindedir ve ayrıca önbellek ve TLB erişimi, TLB erişiminde karşılaşılan gecikmeden dolayı etiket karşılaştırılması gerektirildiği için tek bir cycle'da tamamlanamaz.

3.1.3 Kâhin (O)

Bu yöntem, her bir sanal-fiziksel adres dönüşümünün TLB'de kusurlu bir girişte barındırılıp barındırılmadığını bir öngörüyle önceden belirlenmesi fikrene dayanır. Eğer dönüşüm kusursuz bir girişte ise TLB ve L1 erişimleri aynı çevrimde tamamlanır; aksi takdirde TLB ve L1 erişimleri tamamlanmak için ekstra bir cycle (toplam 2 cycle) alır. Önceden işaret edildiği gibi bu yöntem, CPU'ya bazı senkronize edici devrelerin ilave edilmesi ve gerektiği hallerde sonraki cycle için fetch pipeline aşamasını bekletmek suretiyle gerçekleştirilebilir.

3.1.4 CFR

Bu Şekil 1'de gösterilen önerdiğimiz yöntemdir. Bu yöntemde donanımda bir CFR registerı mevcuttur ve son komut sayfasının çevirisini tutar. Her adres dönüşümü için CFR ve TLB'ye paralel olarak erişilir. Paralel erişim, CFR'de dönüşüm yoksa TLB'ye erişimin erken başlatılmasına müsaade eder. Program akışı aynı sayfa içinde kaldığı sürece (üretilen adresin sanal sayfa numarası ile CFR'nin sanal sayfa numarası karşılaştırılarak bulunur), adres dönüşümü CFR'den sağlanır ve tamamlanmamış TLB erişimi iptal edilir. Aksi takdirde (yani yürütüm mevcut sayfa dışına çıkarsa) bir TLB erişimi oluşur ve gecikmeler arasındaki farklılıklar sebebiyle (işlem değişiminin sonucu olarak) adres dönüşümü ve komut önbelleği erişimi aynı cycle içinde tamamlanamaz (D yöntemindeki gibi). Uygulamanın kontrol akışı mevcut sayfa sınırları dışına çıktığında TLB'den sağlanan dönüşüm ile CFR'yi güncellemeye ihtiyaç duyarız. Bu güncellemenin performansa getirdiği külfet PC'nin güncellenir güncellenmez CFR'nin güncellenmesi (izleyen komut fetch cycle'dan önce) ile bertaraf edilebilir. İşletim sistemi açısından CFR registeri diğer registerlerden farklı değildir. Mevcut süreç (process) yeni bir süreç için işlemciyi terk etmeden önce CFR'nin içeriği saklanır ve yürütüm yeni sürece dönmeden önce o sürece ait önceden kaydedilen içerik CFR'ye tekrar yüklenir.

İzleyen bölümde yukarıda açıklanan 4 yöntemin deneysel değerlendirmesini sunacağız ve performans üzerindeki etkilerini sayısal verilere dökeceğiz.

3.2 Deneysel Değerlendirme

3.2.1 Kurulum

SimpleScalar simülatörün (SimpleScalar Toolset http://www.simplescalar.com) kodunu yukarıda açıklanan yöntemleri gerçekleştirecek şekilde değiştirdik. SimpleScalar, hızlı bir execution-driven simülasyonu kullanan, bir dizi işlemci ve sistemler üzerinde uygulama programlarını simüle eden bir araç setidir. Bu çalışmada bahsettiğimiz yöntemleri Alfa benzeri bir platforma entegre etmek için SimpleScalar'ın sim-outorder bileşeninin kodunu değiştirdik.

Deneylerimizde kullanılan temel simülasyon parametreleri Tablo 1'de listelenmiştir. Deneylerimizde SPEC2000 Suite'i kullandık. SPEC2000 Suite'deki herhangi bir uygulamanın tamamını simüle etmek çok uzun zaman alır. Bu yüzden her bir benchmark için Sherwood ve arkadaşlarının (Sherwood ve ark., 2001) tavsiye ettiği gibi, uygulamaya yönelik belli sayıda komutu hızlı ilerlettikten sonra izleyen 500 milyon komutu simüle ettik.

Çizelge 1. Deneylerimizdeki temel konfigürasyon parametrelerimiz ve varsayılan değerleri

İşlemci Çekirdeği (Pocessor Core)	
Fonksiyonel Birimler	8 Integer ALU 4 Integer mult./divide 8 FP add, 4 FP mult. 4 FP divide/sqrt
RUU boyu	256 komut
LSQ boyu	64 komut
Fetch/Kod Çözme/Issue/ Önerilen Genişlik	8 komut/cycle
Fetch Kuyruk Boyutu	8 komut
Önbellek ve Hafıza Hiyerarşisi (Cache and Memory Hiararchy)	
L1 Komut Önbelleği	64 KB, 4-yol (LRU), 64 byte bloklar, 1 cycle gecikme
L1 Veri Önbelleği	64 KB, 4-yol (LRU), 32 byte bloklar, 1 cycle gecikme
L2 Önbelleği	1 MB birleşik, 8-yol (LRU), 128 byte bloklar, 12 cycle gecikme
Veri/Komut TLB	128 giriş , tam birlik 30 cycle ıska cezası
DRAM	160 cycle gecikme
Sayfa Boyutu	8 K
Dallanma Tahmini (Branch Prediction)	
Dallanma Tahmincisi	Birleşik, Bimodal 2 K table, 2-Level 1 K table, 8 bit history, 4K chooser
Dallanma Hedef Tamponu (BTB)	1K-entry, 4-yol
Dönüş Adres Yığını Uzunluğu	8
Yanlış Tahmin Cezası	20 cycles

P yöntemi, Tablo 1'de listelenmiş konfigürasyon parametreleriyle kullanıldığında elde edilen bu benchmarkların önemli özellikleri Tablo 2'de verilmektedir. Bu tabloda ikinci sütun TLB erişimlerinin dinamik sayısını gösterir. Bu çalışmada önbellek bloklarını sayfa sınırlarını aşmasınlar diye hizalı olduklarını kabul ettik (mevcut derleyiciler bunu sağlamak için gerekli direktifler ve pragmalar sunarlar). Tablo 2'nin son sütunu P yöntemi kullanıldığında elde edilen yürütme cycle'larını göstermektedir.

Çizelge 2. Benchmarklarımız ve VI–PT önbellek arama mekanizması için onların önemli özellikleri

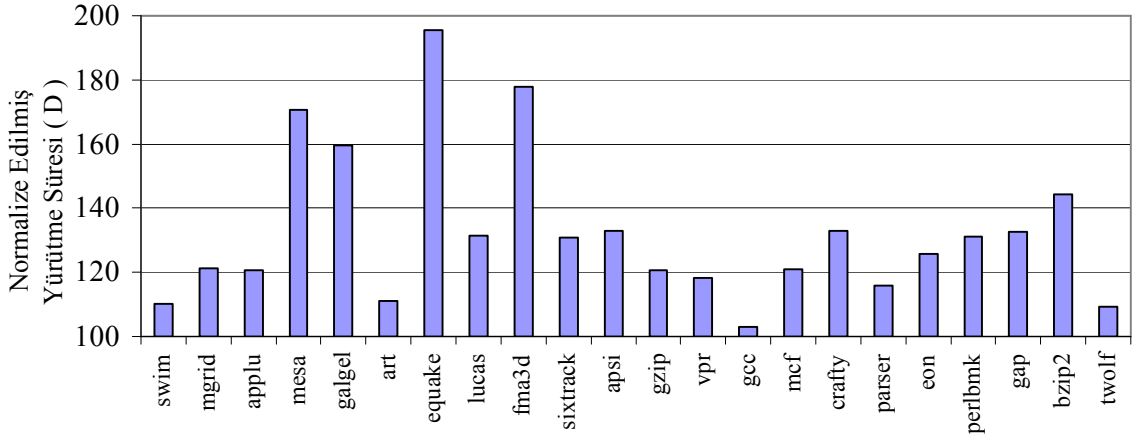
Benchmark	iTLB Erişim Sayısı	Yürütme Süresi (cycle)
swim	347110044	347982163
mgrid	258785124	260407090
applu	260821970	261959435
mesa	120163487	160312282
galgel	186777092	191221037
art	556587751	630700741
equake	109969817	130813719
lucas	236223857	278925939
fma3d	109322546	180113099
sixtrack	122194786	195349559
apsi	189621174	210554664
gzip	167916846	272520611
vpr	308024576	435778113
gcc	204429261	205505024
mcf	616617384	643506260
crafty	153696081	256467119
parser	237277415	371959380
eon	140157315	257521967
perlbmk	186549445	367955497
gap	143319970	212187011
bzip2	168465412	231748913
Twolf	404191737	569302265

BÖLÜM 4

ARASTIRMA BULGULARI VE TARTIŞMA

4.1 Sonuçlar

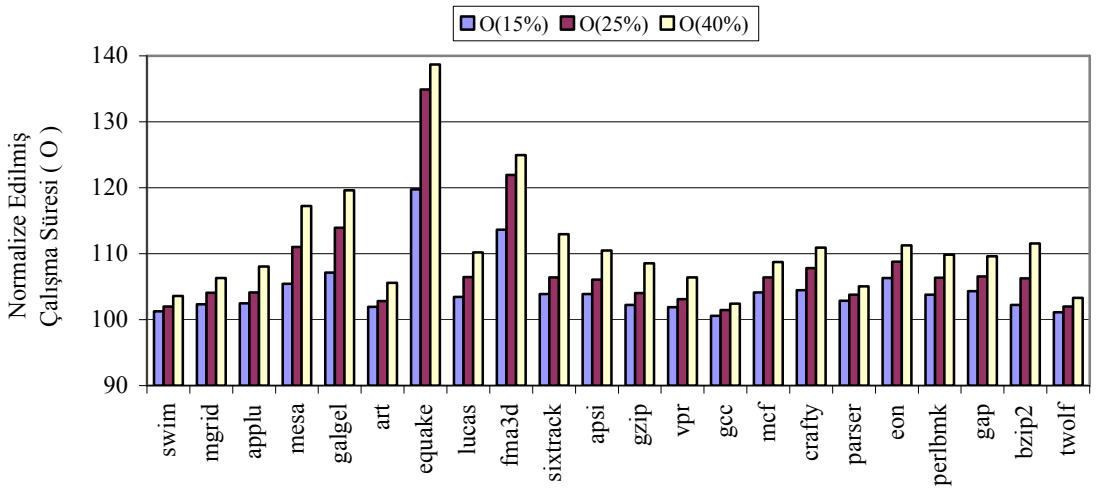
Bu bölümde verilen sonuçlar bir VI-PT önbellek arama mekanizması kullanıldığı durum içindir. D yöntemi kullanıldığında elde edilen çalışma süresi cycle cinsinden Şekil 2'de verilmektedir. Bütün değerler Tablo 2'de verilen P yöntemine ait çalışma sürelerine göre normalize edilmiştir. Şekilden açıkça görüldüğü üzere TLB erişimini geciktirmek (en kötü erişim gecikmesi temel alınarak) programların performansını önemli derecede etkiler. Bu etki özellikle mesa, equake ve fma3d gibi benchmarklarda açıkça belli olmaktadır. Örneğin, işlem değişiminden dolayı TLB erişimi gecikmesi equake ve fma3d için %95 ve %70 oranında çalışma zamanını artırmaktadır. Bu üç benchmarkta karşılaşılan önemli performans kayıpları sahip oldukları yüksek IPC (instruction per cycle) değerlerinden dolayıdır (fazla veri bağımlılıkları göstermemektedirler). Bu çoğu zaman TLB erişimlerinin kritik yola düşmesine ve önemli ölçüde de uygulama zamanının uzamasına sebep olur. İdeal duruma karşılık gelen P yöntemiyle karşılaştırıldığında D yöntemi, ortalama %33'lük bir performans kaybına sebep olmaktadır.



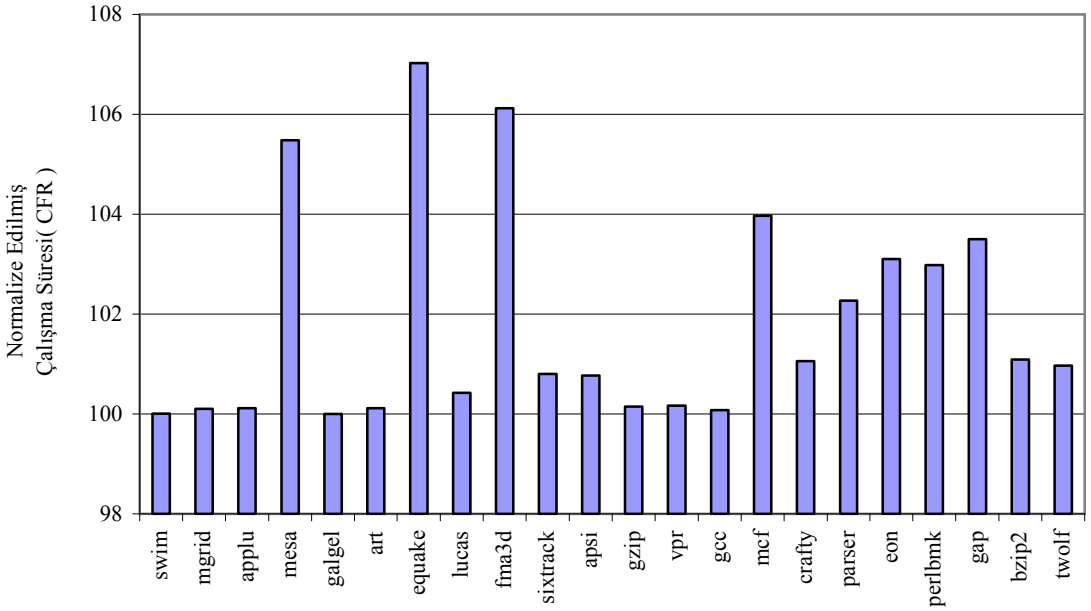
Şekil 2. D Yöntemi için normalize edilmiş yürütme süreleri. Bütün değerler Tablo 2'de verilen P Yönteminin değerlerine göre normalize edilmiştir.

Şekil 3, O yöntemi kullanıldığında elde edilen yürütme sürelerini göstermektedir. İşlem değişiminden etkilenen TLB girişlerinin yüzdesini %15'den %25'e ve %40'a kadar değiştirerek deneyler yaptık. İşlem değişimi açısından genel olarak devre aygıtlarının bölgesel yerelliğe sahip olduklarını belirtmeliyiz. Yani birbirine yakın yerleştirilen

aygıtların daha uzak yerleştirilenlere göre benzer özellikler gösterme olasılığı daha yüksektir. İşlem değişimini bilinen dağılım fonksiyonlarına uygun rasgele değişkenler olarak düşünen bazı istatistiksel yaklaşımlar olduğu gibi SRAM girişlerindeki gecikme ve güç dalgalanmalarını analiz etmede kullanılan bazı analitik yöntemler de vardır. Bu tip analizler bu çalışmanın konusuna girmediği ve bizim önerdiğimiz CFR yönteminin performansıyla ilişkili olmadığından böyle bir analizi çalışmamızda yapmadık. Bunun yerine, O yönteminin performansını değerlendirmek için işlem değişiminden dolayı kusurlu TLB girişlerinin TLB genelinde rastgele dağıldığı varsayımını yapmaktayız. Şekil 3'de her bir benchmark için 3 sütun vardır. Soldan sağa, ilk sütun TLB girişlerinin %15'inin işlem değişimi tarafından etkilendiği duruma karşılık gelirken, ikinci ve üçüncü sütunlar işlem değişiminden etkilenen TLB girişlerinin %25 ve %40 olduğu durumlara karşılık gelmektedir. %15, %25 ve %40'lık durumlar için ortalama performans kayıpları sırasıyla %4,5 , %7,7 ve %11,2'dir. Bu sonuçlar bize şunu göstermektedir; sıradaki TLB erişiminin gecikmesini önceden mükemmel bir şekilde tahmin edebilsek bile veya kusursuz TLB girişlerinin gecikmeden etkilenmesini önleyecek bir devre CPU'ya eklesek bile, işlem değişiminden dolayı ortaya çıkan performans kaybı hâla çok yüksek olacaktır.



Şekil 3. O için normalize edilmiş yürütme süreleri. Bütün değerler Tablo 2'de verilen P yöntemine ait değerlere göre normalize edilmiştir. İşlem değişiminden etkilenen TLB girişlerinin yüzdeleri %15, %25 ve %40 arasında değişmektedir.



Şekil 4. CFR yöntemi için normalize edilmiş yürütme süreleri. Bütün değerler Tablo 2'de verilen P yöntemine ait yürütme sürelerine göre normalize edilmiştir.

CFR yöntemi için performans sonuçları Şekil 4'te sunulmuştur. Şekilden kolaylıkla görülebileceği gibi CFR yöntemi bütün benchmarklar için D ve O yöntemlerinden daha iyidir. CFR yöntemiyle elde edilen en kötü performans equake benchmarkına karşı gelmektedir ve bu performans, P yönteminden elde edilen performanstan %7 daha kötüdür. Daha da önemlisi, birçok benchmark için bizim yöntemimizin performansının P yönteminin performansının %1'i dâhilinde olmasıdır. Bu üç şekli birlikte ele aldığımızda (Şekil 2, 3, 4) CFR yönteminin diğer yöntemlerden daha performanslı olduğu sonucunu çıkarabiliriz. Örnek verecek olursak, O (%15) tarafından getirilen ortalama külfet yaklaşık % 4,5 iken, buna karşılık gelen CFR durumu için ortalama performans kaybı % 1,8 civarındadır. Eğer TLB girişlerinin %25'inin kusurlu olduğunu kabul edersek, O yönteminin oluşturduğu performans kaybı %7,7'ye ulaşır. Bizim yöntemimizin test edilen diğer yöntemlerden daha iyi performans sunmasının sebebi (işlem değişimi olmayan ideal durum hariç) çoğu zaman CFR registerında istenilen sanal-fiziksel sayfa çevirisini bulabileceğimiz ve bu sayede adres çevirisini kritik yoldan kaldırabileceğimiz gerçeğidir. Bu, programın yürütümü esnasında meydana gelen CFR güncellemelerinin sayısını listeleyen Tablo 3'den daha iyi görülebilir. Her bir benchmarkta açıkça görebiliriz ki çok yüksek bir olasılıkla bizim yaklaşımımız CFR registerından gelen çeviriyi başarıyla elde edebilmektedir. Bu durum, TLB erişimi ihtiyacını ortadan kaldırır, cache ve TLB

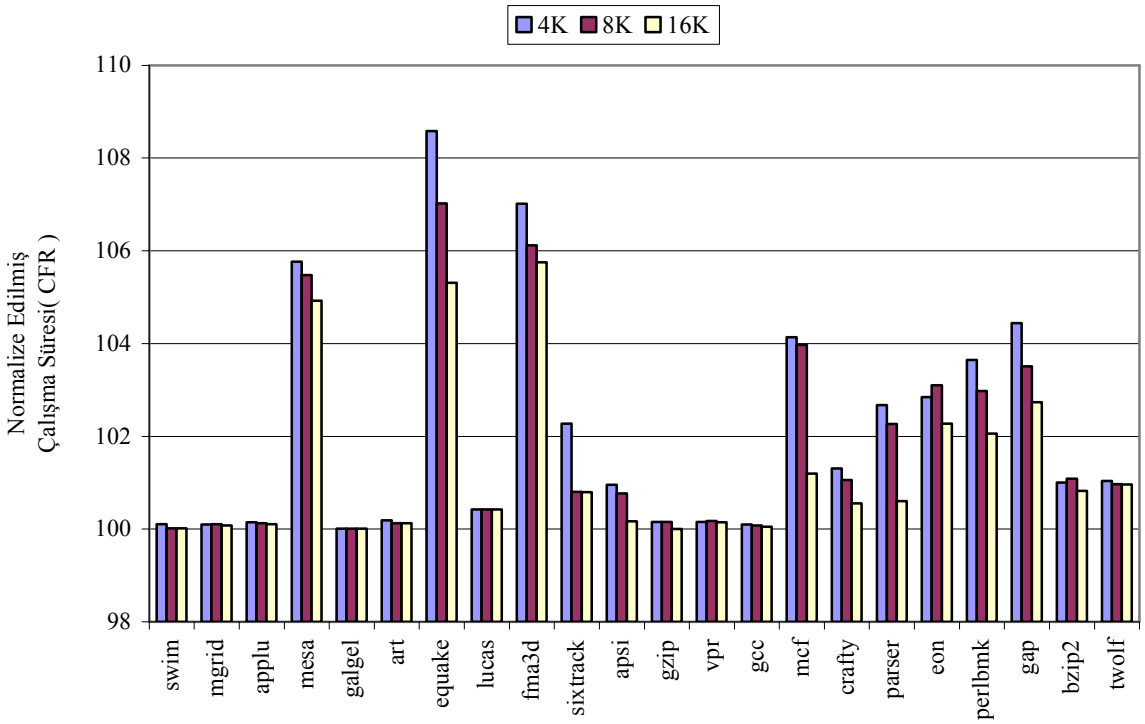
erişimlerinin aynı cycle içinde tamamlanmalarını sağlar. Burada şunu belirtmekte yarar vardır. Bir tane CFR yerine daha fazla sayıda CFR registerları kullanmak suretiyle daha fazla sayıda adres dönüşümü CFR registerlarından temin edileceğinden performansı daha fazla artırmak mümkün olabilecektir. Bu hususun daha fazla irdelenmesi bu çalışmanın konusu dışında olduğundan bunun üzerinde daha fazla durmayacağız.

Çizelge 3. CFR registerı güncellenme sayıları. Parantez içindeki sayılar TLB erişimindeki CFR registerının güncellenme yüzdelerini göstermektedir

Benchmark İsmi	**CFR Güncelleme Sayısı**
swim	10918 (<0.01%)
mgrid	199747 (0.08%)
applu	868449 (0.33%)
mesa	11213959 (9.33%)
galgel	22288 (0.01%)
art	2607318 (0.47%)
equake	8949246 (8.14%)
lucas	3945015 (1.67%)
fma3d	7724247 (7.07%)
sixtrack	6544230 (5.36%)
apsi	1989830 (1.05%)
gzip	4140030 (2.47%)
vpr	14786342 (4.80%)
gcc	278397 (0.14%)
mcf	21447366 (3.48%)
crafty	12093625 (7.87%)
parser	18294287 (7.71%)
eon	11674993 (8.33%)
perlbmk	18846428 (10.10%)
gap	12359927 (8.62%)
bzip2	5821013 (3.46%)
twolf	11311309 (2.80%)

4.2 Sayfa Boyutuna Duyarlılık

Şu ana kadar biz komut sayfası boyutunun 8K olduğunu varsaydık. CFR yönteminin yeterliliğini belirlemek için farklı komut sayfası boyutlarına duyarlılığı da ölçüldü. Bu amaç için 4K ve 16K boyutlarındaki diğer iki komut sayfası boyutlarıyla deneyler yaptık. Bu sonuçlar Şekil 5'te gösterilmektedir. Bu grafikte her bir benchmark için birinci, ikinci ve üçüncü çubuklar sırasıyla 4K, 8K ve 16K'lık sayfa boyutları kullanıldığında elde edilen yürütme sürelerinin P yöntemine göre normalize edilmiş değerlerini göstermektedir. Bu şekilden görüldüğü gibi sayfa boyutunu arttırarak performans artışı sağlamak mümkündür. Örneğin equake için sayfa boyutunu 4K'dan 8K'ya ve 16K'ya çıkararak performans kaybını %8,6'dan %7,1'e ve %5,3'e çekebiliriz. 4K'lık sayfa boyutu için CFR yönteminin öngördüğü ortalama performans kaybı %2,2 iken, sayfa boyutları 8K ve 16K olduğunda ortalama performans kayıpları %1,8 ve %1,3 olmaktadır. 16K ile kıyaslandığında, 16K'dan büyük sayfa boyutları için performansta kayda değer bir değişiklik gözlemlemediğimiz için, daha büyük boyutlu sayfalar için sonuçlar sunmayı uygun bulmamaktayız.



Şekil 5. CFR Yöntemi kullanıldığında farklı sayfa boyutları için normalize edilmiş yürütme süreleri. Sayfa boyutları 4K, 8K ve16K boyutlarındadır. Bütün değerler Tablo 2'de verilen P yöntemine ait değerlere göre normalize edilmiştir.

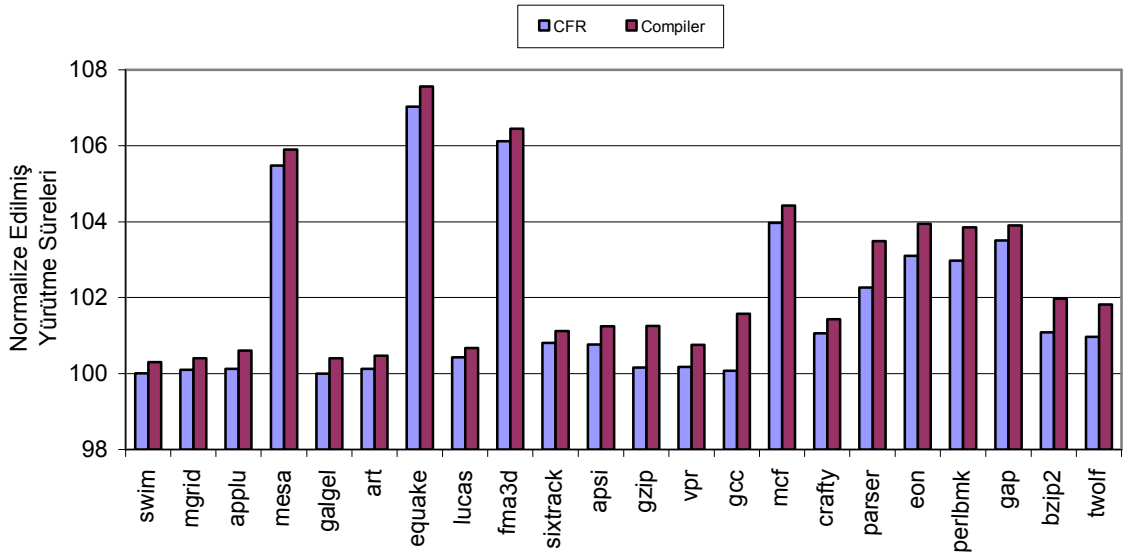
4.3 Derleyici Desteği

Önceden ifade edildiği gibi önerdiğimiz yöntemdeki temel mantık, uygulama aynı komut sayfasında kaldığından emin olduğumuz sürece adres dönüşümünü bir kereye mahsus TLB araması ile yapmak, sonucu CFR registerında saklamak ve daha sonrası için bunu TLB'ye gitmeden direk olarak buradan kullanmaya devam etmektir. Uygulama mevcut sayfa sınırlarını aştığında hedef sayfa için bir TLB araması başlatılır. Bu işlem CFR'de tutulan sanal sayfa numarası PC'de tutulan sanal sayfa numarasından farklı olunca yapılır. Aynı zamanda bir sonraki dönüşümlerde kullanılmak için TLB aramasından elde edilen dönüşümü CFR'de kaydederiz. Bu bölümde TLB aramasının derleyici tarafından başlatıldığı alternatif bir yöntemi göz önüne alıyoruz. Bu alternatif durumda derleyici ilk olarak koddan kontrol akış bilgisini çıkartır ve uygun kontrol akış grafiğini (CFG) oluşturur. Daha sonra derleyici CFG'yi inceler ve sonraki erişilecek komutun kullanılmakta olan sayfa sınırlarında olduğundan %100 emin olduğunda donanıma adres çevirisinin CFR'de var olduğunu bildirir. Aksi takdirde derleyici istenilen sayfa çevirisi için TLB araması başlatan özel bir komutu koda yerleştirir. Bu yaklaşımın olumsuz tarafı özel komut ilavelerinden dolayı kod boyutunda oluşan artıştır. Bu yolla TLB aramalarını tetiklemeye alternatif bir yaklaşım ISA (Instruction Set Architecture)'deki boş bir bit slotunu kullanarak donanıma TLB'ye erişime gerek olup olmadığı bildirilebilir. Çoğu CPU'ların 64 bitlik platformları yavaş yavaş ortaya çıkmaktadır. Bunlar komutları kodlamak için oldukça fazla bit barındırmaktadır. Bunların komut formatları genel olarak ileriye yönelik, kullanılmayan bazı bit slotları barındırır. Gelecek komutun adres çevirisi için TLB erişimine ihtiyacı olup olmadığını donanıma bildirmek için bu boş slotlardan birisini kullanabiliriz. Örneğin bu slota 0 (1)'i kodlayarak sonraki komut için donanımın adres çevirisini CFR (TLB)'den elde edebileceğini gösterebiliriz. Program akışının bir komut sayfasından diğerine geçişinin iki yolla mümkün olduğunu belirtmemiz gerekir. Birincisi, hedefi farklı sayfada olabilecek dallanma komutları; ikincisi ise, sayfa sınırlarına düşen ardışık iki komut, yani birisi sayfanın son komutu diğeri ise diğer sayfanın ilk komutu. Her iki durumda da önceki komutun ilgili bit slotuna 1 kodlanarak gerekli adres dönüşümünün TLB'den karşılanması sağlanır. Aksi durumda, ilgili bit slotuna 0 kodlanarak donanımın adres dönüşümünü CFR'den temin etmesi sağlanır. Mümkün olduğunca adres dönüşümünü CFR'den temin etmek için derleyici statik dallanmaları analiz etmelidir. Statik dallanmalar hedef adresleri derleme zamanında belirlenebilen

dallanmalar olup, bunların hedef adresleri aktif sayfanın içinde kalırsa derleyici adres dönüşümünün CFR'den temin edilmesini sağlamalıdır.

Bu derleyici tekniği iki güzel özelliğe sahiptir. Bunlardan ilki, CFR yönteminin aksine dönüşümün CFR'de bulunduğundan emin olunduğunda TLB erişimine gerek kalmayışıdır. Bu, TLB erişim sayısının azaltarak dinamik enerjide tasarruf sağlar. İkincisi ise, bu derleyici tekniğinin, adres dönüşümünün CFR'den elde edildiği sürece TLB'nin güç modunun durum koruyucu düşük güç moduna geçirilerek TLB sızıntı enerjisinde tasarruf yapılacak şekilde biçimlendirilebilmesidir. Güç tüketimindeki hususlar bu çalışmanın konusu olmadığından onlardan burada bahsetmeyeceğiz.

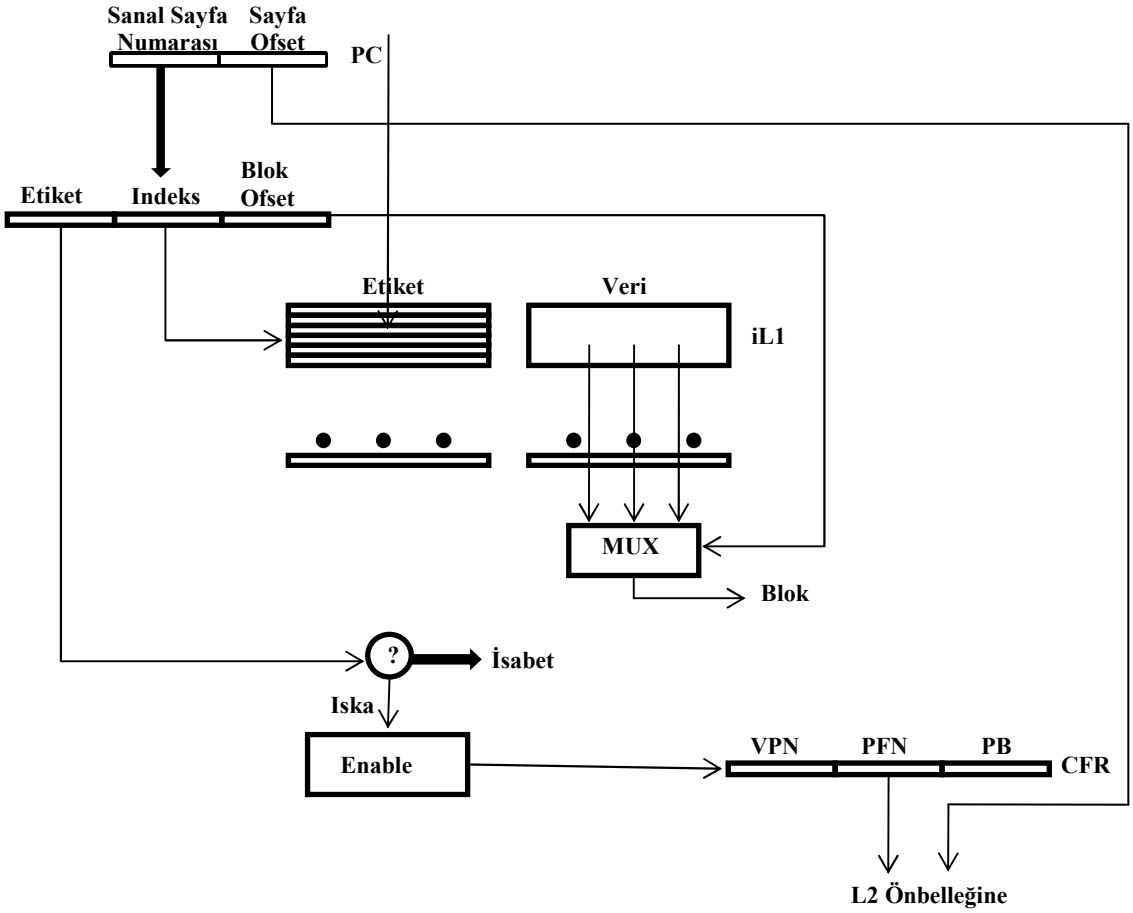
Bu alternatif yöntem ile bizim CFR yöntemimizin yürütme süreleri açısından karşılaştırılması Şekil 6'da verilmiştir. Şekilden görülebildiği gibi CFR yöntemi her bir test edilmiş benchmark için derleyici temelli yöntemden daha iyi performansa sahiptir. Hedefleri gerçekte aktif sayfa içinde kalan bazı dallanmaların hedef adresleri derleme zamanında tespit edilemediği için derleyici tabanlı yöntemin performansı daha kötüdür. Dolayısıyla, bu tip dallanmalar için derleyici adres dönüşümünü CFR yerine TLB'den temin edilmesine karar verir. Bu, bir miktar performans kayıplarına yol açar ve bu kayıplar grafikte gösterildiği gibi integer temelli uygulamalarda daha fazla belirgindir. 8K'lık sayfa boyutu için, CFR ve derleyici temelli yöntemin performansları P yönteminden sırasıyla %1,83 ve %2,43 daha kötüdür.



Şekil 6. CFR yöntemi ve alternatif derleyici tabanlı yöntem için normalize edilmiş yürütme süreleri. Bütün değerler P yöntemine ait ilgili performans değerlerine göre normalize edilmiştir.

4.4 VI - VT Önbellek Araması

Şimdiye kadar VI-PT önbellek arama mekanizmasını kullanan bir sistemi dikkate aldık. Bu bölümde biz VI-VT arama mekanizmalı bir sistemde CFR bulunması durumundaki hususları inceleyeceğiz. Komut getirimi (fetch) safhasında TLB işlem değişimini azaltmanın etkin bir yolunun VI-VT önbellek arama mekanizması kullanımı olduğu düşünülebilir. VI-VT önbellek arama mekanizması kullanan bir sistemde, hem önbellek indeksleme işlemi hem de etiket karşılaştırma işlemi sanal adreslerle yapıldığı için önbellekte isabet olduğu sürece TLB erişimine gerek duyulmaz. Yalnızca bir isabetsizlik durumunda, L2'de arama yapmak için PC'nin sayfa ofset bitleriyle birleştirilmek üzere gereken fiziksel frame numarasını elde etmek amacıyla CFR'ye ulaşmaya ihtiyacımız vardır. Bu arama mekanizması Şekil 7'de gösterilmektedir. VI-VT önbellek arama mekanizması basit olarak görünse ve birçok adres dönüşümlerinin elimine etse bile (buna bağlı olarak TLB erişimlerini azaltsa) bazı dezavantajları vardır. İlk olarak bu strateji aliasing problemlerine yol açar. Bu problem adres uzaylarını ayırt etmek için yüksek öncelikli birkaç bitin eklenmesiyle çözülebilir. İkinci ve en ciddi problem ise adres çevirisi ve L2 cache erişimi seri şekilde yapıldığı için yüksek ıska oranlarının görüldüğü durumlarda performans kayıplarına yol açar. Yani adres çevirisi için TLB erişimi cache araması tamamlandıktan sonra başlar (önbellekte bir hata yakaladıktan sonra).

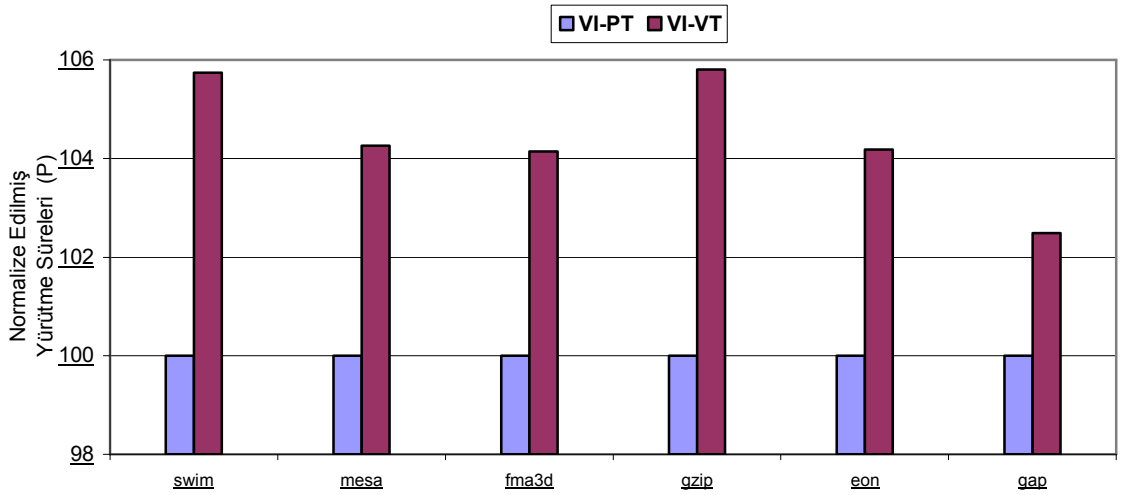


Şekil 7. VI-VT önbellek arama mekanizması altında CFR kullanarak L1 komut önbelleği araması.

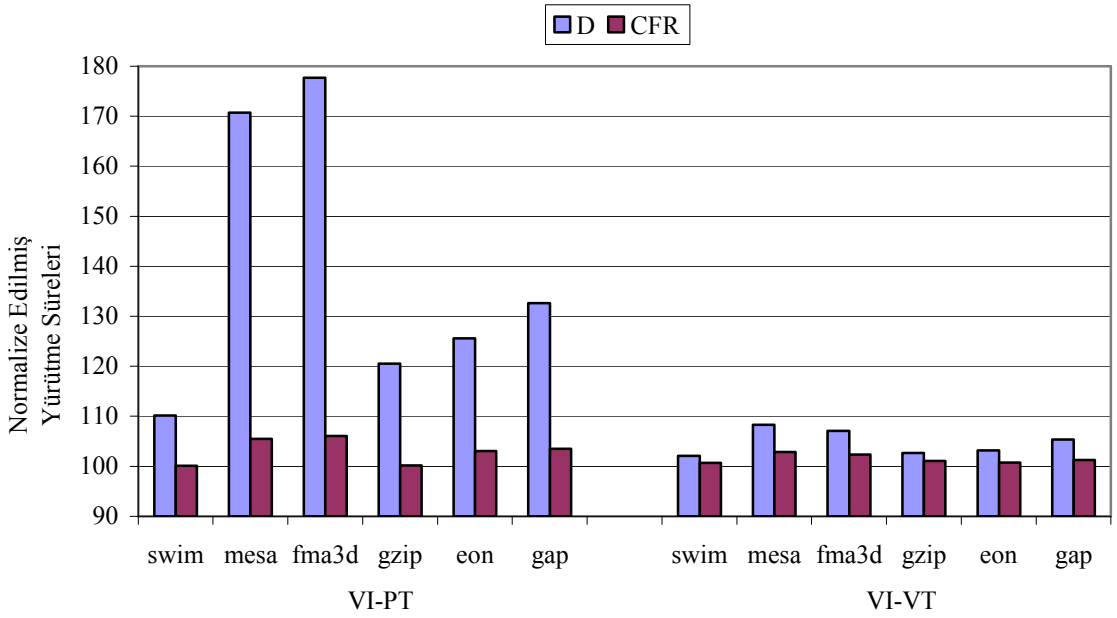
VI-PT ve VI-VT önbellek arama mekanizmaları kullanıldığında bazı benchmarklar için P yöntemine ait normalize edilmiş çalışma zamanları Şekil 8'de gösterilmektedir. Şekilden görüldüğü gibi VI-PT'ye nazaran VI-VT araması genelde oldukça fazla performans kayıplarına sebep olur. Bu sebepten dolayı işlemci tasarımında özellikle yüksek performanslı sistemlerde VI-VT araması VI-PT kadar yaygın kullanılmaz.

VI-VT önbellek arama mekanizması ile TLB'ye sadece önbellek ıskalarında başvurduğumuz için TLB işlem değişiminin performans üzerindeki etkisi Şekil 9'da gösterildiği gibi VI-PT aramasıyla karşılaştırıldığında izafi olarak sınırlıdır. Şeklin sol tarafı VI-PT için normalize edilmiş performans değerlerini gösterirken, şeklin sağ tarafı VI-VT için performans değerlerini göstermektedir. Bu şekilden gözlemlediğimiz şudur. Her ne kadar VI-PT önbellek aramasıyla kıyaslandığında bizim CFR yöntemimiz VI-VT önbellek aramasında işlem değişimini azaltmada daha az etkili ise de CFR, VI-VT yöntemi

için makul bir düzeyde performans kaybını azaltır. Örneğin mesa ele alınacak olursa, CFR yöntemi VI-PT için TLB işlem değişiminin sebep olduğu performans kaybını %71'den %6'ya düşürürken, VI-VT için performans kaybını %8'den %3'e indirir.



Şekil 8. VI-PT ve VI-VT önbellek arama mekanizmaları için P yöntemine ait normalize edilmiş yürütme süreleri.



Şekil 9. TLB girişlerinin % 25'inin işlem değişiminden etkilendiği varsayımı altında, D ve CFR yöntemlerinin VI–PT ve VI–VT önbellek arama mekanizmaları için normalize edilmiş yürütme süreleri. Bütün değerler ilgili önbellek arama mekanizması kullanıldığında P yöntemiyle elde edilen değerlere göre normalize edilmiştir.

Burada özet olarak şunu söyleyebiliriz. VI-PT önbellek arama mekanizmasına göre VI-VT önbellek arama mekanizması performans kaybına yol açsa bile, VI-VT için işlem değişiminin performansa etkisi daha sınırlıdır. Dahası, CFR yöntemi, işlem değişiminin gecikme etkisini azaltmada VI-PT için daha önemlidir.

4.5 Fiziksel Veri Adreslerinin Direkt Olarak Oluşturulması

Veri TLB işlem değişiminin performans üzerine olan etkisini azaltmak için, fiziksel adresi çoğu zaman veri TLB'ye gitmeden direk olarak oluşturmak mümkündür. Bunu başarmak için, tek bir CFR registeri yerine donanımın, seçili sayfalar için sayfa dönüşümlerini sağlayacak birkaç sanal-fiziksel Translation (dönüşüm) Registeri (TR) sağlaması gerekir. Derleyici direk olarak bu registerları fiziksel adresleri oluşturmak için kullanır. Bu yaklaşımda derleyici, hangi Translation Registerının hangi veri referansları için kullanılacağını belirler ve bunları sadece dönüşümün mevcut olduğu durumda kullanır. Aksi takdirde dönüşümler veri TLB'den sağlanır. Veri TLB'nin dinamik enerjisini azaltmak için, bu TR'leri daha önceki bir çalışmada kullanıldı (Kadayıf ve ark., 2007). TLB erişimi ile karşılaştırıldığında bir TR registerı göreceli olarak çok küçük miktarda dinamik güç tüketmektedir. Bahsi geçen çalışmada veri referansları bazı kategorilerde sınıflandırıldı; bunlar global scalar değişkenler, global dizi değişkenler, heap veri referansları ve stack veri referansları. Yine aynı çalışmada mümkün olduğunca çok adres dönüşümlerini TR'lerden elde edebilmek için TR'lerin farklı kategorilerdeki veri referanslarına nasıl tahsis edilmesi gerektiği ve bu registerlerin daha verimli kullanılması için bazı yüksek seviyeli derleyici optimizasyonları (strip-mining, loop stripping ve loop distribution gibi) üstünde duruldu.

BÖLÜM 5

SONUÇLAR VE ÖNERİLER

İşlem değişiminin devre performansı üzerine artan etkisi, tasarımcıları bu değişimlerin sonuçları ile mücadele etmek için mimari seviyede çözümler bulmaya zorlamaktadır (devre seviyesi çözümlerine ilave olarak). Bu çalışmanın ana katkısı, komut TLB performansı üzerine olan işlem değişiminin potansiyel etkisini azaltacak mimari bir çözüm sunmasıdır.

Amaçlanan çözümümüz CPU ve komut TLB'si arasına bir register ekleyerek son sanal-fiziksel adres dönüşümünü bu registerde tutmaya çalışmaktadır. Her adres dönüşümü için TLB erişimi ve CFR erişimi paralel olarak başlar. Ancak, eğer sıradaki dönüşüm bu registerde yakalanırsa TLB erişimi iptal edilir. Böylece işlem değişiminin potansiyel etkisinden de sakınılmış olunur. Bu yaklaşımın faydalarını nicelendirmek için SPEC2000 benchmark suite kullanılarak deneyler yapıldı.

Sonuçlarımız gösteriyor ki önerilen yaklaşım, en kötü durum gecikme savına (yani tüm TLB girişlerinin en yavaş giriş gecikmesine sahip olan girişin gecikmesine eşit olduğunun varsayıldığı duruma) ve TLB'de erişilecek olan girişin gecikmesini önceden doğru olarak tahmin edebilen ve buna göre TLB gecikmelerini ayarlayabilen yönteme göre, işlem değişiminin performans üzerindeki etkisini azaltmada çok daha başarılıdır. Yaklaşımımızı çeşitli sayfa boyutları için test ettik ve sayfa boyu arttıkça performans artışının daha büyük olduğunu gözlemledik.

VI-PT önbellek arama sistemlerine ek olarak, CFR registerının VI-VT önbellek arama mekanizmalı bir sistemde kullanılarak TLB işlem değişiminin yol açtığı performans kaybını azaltmada kullanılabileceğini gösterdik. Ayrıca, CFR registeri kullanımının işlem değişiminin performans üzerindeki etkisini azaltmada VI-PT önbellek erişim mekanizmalı sistemlerde daha etkili olduğunu belirledik. Bunun temel sebebi, VI-VT önbellek erişim mekanizmasına sahip olan bir sistemin sadece birincil önbellekte ıska olması durumunda adres dönüşüme ihtiyaç duyması, TLB üzerinde oluşabilecek işlem değişiminin performansa olan eksinini oldukça sınırlı hale getirmesinden kaynaklanmaktadır. Ancak deneylerimiz VI-VT erişim mekanizmasına sahip sistemler de bile CFR yönteminin makul bir düzeyde performans artışı getirebileceğini göstermektedir.

Derleyici de bu registerı kullanabilir ve sonraki komut için adres dönüşümünün bu registerda olduğunu donanıma bildirmek için programın CFG'sini oluşturarak komutu uygun bir şekilde işaretleyebilir. Bu derleyici tabanlı yöntem enerji tasarrufu için daha uygun olmasına rağmen performansı CFR yönteminin biraz gerisinde kalmaktadır.

KAYNAKLAR

Agarwal A., Paul B.C. , Mukhopadhyay S. ve Roy K., 2005. Procecess Variation In Embedded Memories: Failure Analysis and Variation Aware Architecture. *IEEE Journal Of Solid-State Circuits ,* 40 (9) : 1804-1814.

Borkar S., Karnik T., Narendra S., Tschanz J. ve Keshavarzi A., De V., 2003. Parameter Variations and Impact On Circuits And Microarchitecture. *ACM/IEEE Design Automation Conference* : 338-342.

Bowman K., Duvall S. G. ve Meindl J. D., 2002. Impact Of Die-To-Die and Within-Die Parameter Fluctuations On The Maximum Clock Frequency Distribution For Gigascale Integration. *IEEE Journal Of Solid-State Circuits*, 37(2) 183-190 p.

Chen Q., Mahmoodi H., Bhunia S. ve Roy K., 2005. Modeling and Testing Of Sram For New Failure Mechanisms Due To Process Variations.Nanoscale Cmos. *IEEE VLSI Testing Symposium :* 292-297.

Chen T. ve Naffziger S., 2003. Comparison Of Adaptive Body Bias (Abb) and Adaptive Supply Voltage (Asv) For Improving Delay and Leakage Under The Presence Of Process Variation. *IEEE Transactions on VLSI Systems* :11(5) 888-899.

Chiueh T. C. ve Katz R. H.,1992. Eliminating The Address Translation Bottleneck For Physical Address Cache, *Internatinal Conference on Architectural Support For Programming Languages And Operating Systems*, 137-148.

Datta A., Bhunia S., Choi J. H., Mukhopadhyay S. ve Roy K., 2006. Speed Binning Aware Design Methodology to Improve Profit Under Parameter Variations. *Asia South Pasific Design Automation Conference :* 712-717.

Devgan A. ve Kashyap C., 2003. Block-Based Static Timing Analysis With Uncertainty. *International Conference On Computer-Aided Design*, 607-614.

Gregg J. ve Chen T. W., 2004. Post Silicon Power/Performance Optimization In The Presence Of Process Variation Using İndividual Well Adaptive Body Biasing(Iwabb). *International Symposium On Quality Electronic Design,* 453-458.

Humenay E., Tarjan D. ve Skadron K., 2006. Impact Of Parameter Variations On Multicore Chips. *The 2006 Workshop On Architectural Support For Gigascale Integration, Held In Conjunction With The 33rd International Symposium On Computer Architecture.*

Kadayif I., Sivasubramaniam A., Kandemir M., Kandiraju G. ve Chen G.,2002.Generating Physical Adresses Directly For Saving Instruction TLB Energy. *International Symposium On Microarchitecture,* 185-196.

Kadayif I., Nath P., Kandemir M. ve Sivasubramaniam A., 2007. Reducing Data TLB Power Via Compiler-Directed Address Generation. *IEEE Transactions On Computer Aided Design,:* 26(2) 312-324.

Knight J. ve Rosenfeld P., 1984. Segmented Virtual To Real Translation Assist, *IBM Technical Disclosure Bulletin* : 27(2) 1077-1078.

Maddock R., Marks B., Minshull J. ve Pinnel M., 1981. Hardware Address Resolution For Variable Length Segments. *IBM Technical Disclosure Bulletin* : 23(11) 5186-5187.

Nassif S. R., 2001. Modeling And Analysis Of Manufacturing Variations. *Custom Integrated Circuits Conference,* 223-228.

Ozdemir S., Sinha D., Memik G., Adams J. ve Zhou H., 2006. Yield-Aware Cache Architectures. *IEEE/ACM International Symposium on Microarchitecture*, 15-25.

Papanikolaou A., Lobmaier F., Wang H., Miranda M. ve Catthoor F., 2005. A System-Level Methodology For Fully Compensating Process Variability Impact Of Memory Organizations In Periodic Applicatios. *International Conference On Hardware/Software Codesign And System Synthesis*, 117-122.

Raj S., Vrudhula S. B. K. ve Wang J., 2004. A Methodology to Improve Timing Yield in The Presence Of Process Variations. *Design Automation Conference,* 448-453.

Sarangi S. R., Greskamp B. ve Torrellas J., 2007. A Model For Timing Errors in Processors With Parameter Variation. *International Symposium On Quality Electronic Design,* 647-654.

Sherwood T., Perelman E. ve Calder B., 2001. Basic Block Distribution Analysis to Find Periodic Behavior and Simulation Points in Applications. *International Conference on Parallel Architectures and Compilation Techniques,* 3-14.

SimpleScalar LLC Toolset. http://www.simplescalar.com

Sinha D., Shenoy N. V. ve Zhou H., 2005. Statistical Gate Sizing For Timing Yield Optimization. *International Conference on Computer-Aided Design,* 1037-1041.

Standard Performance Evaluation Corporation, SPEC2000 Benchmark. http://www.spec.org

Strecker W. D., 1978. Vax-11/780: A Virtual Address Extensition To Thr DEC PDP-11 Family, Afips Ncc 47 967-980.

Tschanz J., Kao J. T., Narendra S. G., Nair R., Antoniadis D. A. , Chandrakasan A. P. ve De V., 2002. Adaptive Body Bias For Reducing Impacts of Die-To-Die and Within-Die Parameter Variations on Microprocessor Frequency and Leakage. *IEEE Journal of Solid-State Circuits* : 37(11) 1396-1402.

Vergos H. T ve Nikolos D., 1995. Performance Recovery in Direct-Mapped Faulty Cache Via The Use Of Very Small Fully Associative Spare Cache. *International Computer Performance and Dependability Symposium,* 326-332.

Zhan Y., Strojwas A. J., Li X., Pileggi L. T., Newmark D. ve Sharma M., 2005. Correlation-Aware Statistical Timing Analysis with Non-Gaussian Delay Distributions. *Design Automation Conference,* 77–82.

Zuchowski P. S., Habitz P. A., Hayes J. D. ve Oppold J. H., 2004. Process And Environmental Variation Impacts on ASIC Timing. *International Conference On Computer-Aided Design,* 336-342.

ÇİZELGELER LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı: Mahir TÜRKCAN

Doğum Yeri: İstanbul

Doğum Tarihi: 12.02.1982

EĞİTİM DURUMU

Lisans Öğrenimi: Marmara Üniversitesi Teknik Eğitim Fakültesi Elektronik-Bilgisayar Bölümü Bilgisayar ve Kontrol Öğretmenliği (2001-2005)

Yüksek Lisans Öğrenimi: Çanakkale Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Bölümü (2006-….)

Bildiği Yabancı Diller : İngilizce

İŞ DENEYİMİ

Akçakoyun Yatılı İlköğretim Bölge Okulu Bilgisayar Öğretmeni Yenice / Çanakkale 2006-

Koç İlköğretim Okulu Bilgisayar Öğretmeni Gebze / Kocaeli 2005-2006

The Marmara Residences, İstanbul, Gece Otel Sorumlusu 2002-2005

Alarko Carrier Sanayi ve Ticaret A.Ş, Gebze / Kocaeli, Bina Otomasyon Sistemleri Bölümü 2004 Yaz

TUBİTAK MAM, Gebze / Kocaeli, Bilgi İşlem Bölümü 2003 Yaz

İLETİŞİM

E-Posta Adresi**:** mahirturkcan@gmail.com