YAZMAÇ ÖBEĞİNİ GEÇİCİ HATALARA KARŞI KORUMAK İÇİN YERLEŞİK KARŞILAŞTIRICILI SRAM BİT HÜCRELERİNİN KULLANILMASI

MEHMET KAYAALP

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

HAZİRAN 2010

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Ünver KAYNAK

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Prof. Dr. Erdoğan DOĞDU

Anabilim Dalı Başkanı

Mehmet KAYAALP tarafından hazırlanan YAZMAÇ ÖBEĞİNİ GEÇİCİ HATALARA KARŞI KORUMAK İÇİN YERLEŞİK KARŞILAŞTIRICILI SRAM BİT HÜCRELERİNİN KULLANILMASI adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Yrd. Doç. Dr. Oğuz ERGİN

Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Yrd. Doç. Dr. Murat ÖZBAYOĞLU _____

Üye : Yrd. Doç. Dr. Oğuz ERGİN _____

Üye : Yrd. Doç. Dr. Coşku KASNAKOĞLU _____

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.

Mehmet Kayaalp

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 – Haziran 2010**

Mehmet KAYAALP

YAZMAÇ ÖBEĞİNİ GEÇİCİ HATALARA KARŞI KORUMAK İÇİN YERLEŞİK KARŞILAŞTIRICILI SRAM BİT HÜCRELERİNİN KULLANILMASI

ÖZET

Uzaydan dünyaya gelen yüklü ışınların veya alfa parçacıklarının neden oldukları geçici hatalar güvenilir mikroişlemcilerin tasarımında giderek önemli bir sorun haline gelmektedir. Transistör yoğunluğu ve işlemci alanı artışındaki eğilim önümüzdeki yıllarda geçici hataların daha da önem kazanacağını göstermektedir. Geçici hataların önlenmesinde yaygın olarak kullanılan artıklık yöntemleri, yüksek maliyetleri nedeniyle, başarım açısından hassas bir bileşen olan yazmaç öbeğinde kullanılmaya pek elverişli değildir. Pek çok çağdaş işlemcide üretilen sonuçları saklamak için büyük boyutlu bir yazmaç öbeği kullanılır ve saklanan değerler uzun süreler boyunca yazmaç öbeğinde kalabilirler. Yazmaç öbeğinin geçici hatalara karşı korunması büyük önem taşımaktadır. Bu tezde yazmaç öbeğinin gerçekleştiriminde kullanılan SRAM bit hücrelerinde devre düzeyinde değişiklik yapılarak verinin bir kopyasının daha tutulması ve yerleşik karşılaştırıcılar kullanılarak bu iki kopyanın birbiri ile karşılaştırılması ve bu sayede oluşabilecek bit hatalarının saptanabilmesi önerilmektedir. Yapılan deneysel çalışmalar ile önerilen tasarım kullanılarak düşük alan, güç ve gecikme maliyetleri ile yazmaç öbeğinin geçici hatalara karşı korunabildiği gösterilmiştir.

Anahtar Kelimeler: Geçici Hatalar, Yazmaç Öbeği, SRAM Bit Hücresi

University : **TOBB Economics and Technology University**
Institute : **Institute of Natural and Applied Sciences**
Science Programme : **Computer Engineering**
Supervisor : **Assistant Professor Dr. Oğuz ERGİN**
Degree Awarded and Date : **M.Sc. – June 2010**

Mehmet KAYAALP

PROTECTING THE REGISTER FILE AGAINST SOFT ERRORS USING SRAM BIT CELLS WITH BUILT-IN COMPARATORS

ABSTRACT

Soft errors caused by cosmic rays or alpha particles are becoming an increasingly important challenge in reliable microprocessor design. Transistor density, and die size trends show that soft errors will gain even more importance in the future. Due to their significant overheads, most common redundancy schemes used for protection against soft errors are not suitable for the register file which is a critical component for performance. Most contemporary processors employ a large physical register file to hold the produced results which may reside there for a long time. The register file is a crucial element of a microprocessor and is needed to be protected against soft errors. In this thesis, a modification to the design of the SRAM bit cells used to implement the register file is proposed to hold a redundant copy of the data and compare the two copies using built-in comparators to detect a possible bit fault. It is shown by experimental evaluations that the proposed design has low area, power and delay overheads and can protect the register file against soft errors.

Keywords: Soft Errors, Register File, SRAM Bit Cell

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 ve yüksek lisans öğrencilerine, gece gündüz demeden bu tezin hazırlanması için çalışan Fahrettin KOÇ'a, benden yardımlarını hiç esirgemeyen Yusuf Onur KOÇBERBER ve Meltem ÖZSOY'a, yoğun çalışmalarım sırasında bana sürekli destek olan aileme ve yüksek lisans çalışmalarım boyunca bursumun karşılanmasını sağlayan TÜBİTAK'a teşekkürü bir borç bilirim.

İÇİNDEKİLER

	Sayfa
ÖZET	iii
ABSTRACT	iv
TEŞEKKÜR	v
İÇİNDEKİLER	vi
ÇİZELGELERİN LİSTESİ	viii
ŞEKİLLERİN LİSTESİ	ix
KISALTMALAR	x
1. GİRİŞ	1
2. GEÇİCİ HATALAR	3
2.1 Mimari Hassaslık Katsayısı	3
2.2 Artıklık Yöntemleri	4
2.2.1 Alanda Artıklık	4
2.2.2 Zamanda Artıklık	7
2.3 İlgili Çalışmalar	7
3. SÜPERSKALAR İŞLEMCİ MİMARİSİ	8
4. YAZMAÇ ÖBEĞİ	10
5. YAZMAÇLARIN MİMARİ HASSASLIĞI	13
6. YERLEŞİK KARŞILAŞTIRICILI SRAM BİT HÜCRESİ TASARIMI	16
7. ALAN VE GÜÇ MALİYETİNİ AZALTMA	20
7.1 Kısa Ömürlü Yazmaçlar ile Alan Maliyetini Azaltma	20
7.2 3 Boyutlu Katmanlı Silikon Mimarisi ile Güç Maliyetini Azaltma	24

8. BENZETİMLİK ORTAMI VE DENEYSEL ÖLÇÜMLER 27

9. İLGİLİ ÇALIŞMALAR 31

10. SONUÇLAR 32

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

Çizelge 8.1 Benzetimlik için kullanılan mimari değişkenler 28

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil 2.1. Eşlik biti hesaplayan devrenin şeması. 6
Şekil 2.2. Hamming(7,4) kodu hesaplayan devrenin şeması. 6
Şekil 3.1. Süperskalar bir işlemci mimarisinin basitleştirilmiş gösterimi. 8
Şekil 4.1. Yazmaç öbeğinin gerçeklenmesinde kullanılan SRAM tablosu. 11
Şekil 4.2. 12 bağlantılı SRAM bit hücresi. 12
Şekil 4.3. 12 bağlantılı SRAM bit hücresinin serimi. 12
Şekil 5.1. Yazmaç ömrü. 13
Şekil 5.2. SPEC CPU2000 ölçüm programları için yazmaçların yararlı veri sakladıkları sürelere göre oranları ile beklenen yararlı veri saklama süresi. 14
Şekil 6.1. Dinamik mantık kullanan aşağı çeken bitsel karşılaştırıcı. 17
Şekil 6.2. Yerleşik karşılaştırıcılı 12 bağlantılı bit hücresi tasarımı şeması. Bir kopya daha bulundurmayı sağlayan kısımlar açık gri ile ve karşılaştırmayı sağlayan kısımlar koyu gri ile gösterilmiştir. 18
Şekil 6.3. Yerleşik karşılaştırıcılı 12 bağlantılı bit hücresi tasarımı serimi. 19
Şekil 7.1. SPEC CPU2000 ölçüm programları için aynı bellek konumundaki buyruğun iki ardışık çalışması sonucunda aynı ömre sahip olan yazmaçların oranı. 21
Şekil 7.2. SPEC CPU2000 ölçüm programları için kısa ömürlü yazmaçların yararlı veri sakladıkları sürelere göre oranları ile beklenen yararlı veri saklama süresi. 21
Şekil 7.3. SPEC CPU2000 ölçüm programları için kısa ömürlü yazmaçların oranı. 22
Şekil 7.4. SPEC CPU2000 ölçüm programları için kısa ömürlü yazmaçların MHK'ya katkıları. 22
Şekil 7.5. SPEC CPU2000 ölçüm programları için aynı bellek konumundaki buyruğun iki ardışık çalışması sonucunda da kısa ömürlü veya ikisinde de uzun ömürlü olan yazmaçların oranları. 23
Şekil 7.6. 3 boyutlu yerleşik karşılaştırıcılı bit hücresi tasarımı. Kopya ve karşılaştırma üst seviyededir. Sadelik açısından sadece iki bağlantı gösterilmiştir. 24
Şekil 7.7. 3 boyutlu korumalı yazmaç öbeği tasarımı. Alt seviyede normal yazmaç öbeği bulunmaktadır ve kopya ile karşılaştırma üst seviyededir. 25
Şekil 8.1. Farklı korumalı satır sayılarında (x-ekseni) MHK azalışı ve alan maliyeti. 29
Şekil 8.2. Farklı korumalı satır sayılarında (x-ekseni) enerji kullanımı. 30

KISALTMALAR

Kısaltmalar	Açıklama
BKM	Buyruk Kümesi Mimarisi
BSP	Buyruk Seviyesinde Parallellik
GVB	Gizli Veri Bozulması
HDK	Hata Düzeltme Kodları
HSP	Hata Saptama Kodları
İB	İşlem Birimi
MHK	Mimari Hassaslık Katsayısı
MOHY	Mimari Olarak Hatasız Yürütüm
SGH	Saptanmış Giderilemeyen Hata
SRAM	Static Random Access Memory
SPEC	Standard Performance Evaluation Corporation
YYA	Yazmaç Yeniden Adlandırma
YSA	Yeniden Sıralama Arabelleği
YK	Yürütme Kuyruğu

1. GİRİŞ

Uzaydan dünyaya yağan, atmosferi geçip yeryüzüne inebilen nötronlar veya işlemcilerin paketlenmesinde kullanılan plastik kaplama maddesinin safsızlığı nedeniyle içerisinde bulunan radyoaktif elementlerin oluşturduğu ışımalar sonucu ortaya çıkan alfa tanecikleri, çalışır durumdaki tümleşik devrelere isabet ettiğinde çarptıkları noktada elektriksel yük değişimine yol açabilirler. Bu yük değişimi yeterli seviyede olursa isabet alan devre elemanı üzerindeki bit değeri 1 iken 0'a veya 0 iken 1'e dönebilir.

Giderek küçülen CMOS üretim teknolojileri nedeniyle bir transistör üzerinde depolanan yük miktarı azalmakta ve bu da parçacık çarpmalarının bit değişimine neden olma ihtimalini artırmaktadır. Ayrıca artan işlemci alanı parçacığın çarpabileceği alanın da genişlemesine yol açmaktadır. Tek bir transistör açısından bakıldığında CMOS üretim teknolojilerinin küçülmesi hem daha düşük eşik voltajı hem de daha küçük alan sağladığından transistör başına parçacık çarpması sonucu bit değişimi CMOS teknolojisinden fazla etkilenmese de, işlemci içerisinde bulunan transistör sayısının katlanarak artması bit değişimi ihtimalini önemli seviyelere çıkarmaktadır.

Parçacık çarpması sonucu devre elemanlarında kalıcı bir hasar görülmediğinden ve üzerindeki bit değeri tekrar 1 veya 0 olarak ayarlandığında çalışmasına normal şekilde devam edebildiğinden bu tür bit değişimleri geçici hata olarak adlandırılmaktadır.

Bir devre elemanı üzerinde oluşan geçici hata her zaman gözle görülür bir hataya yol açmayabilir. Bunun nedeni çalışır durumdaki bir işlemcinin içerisinde kullanılmayan veya çıktının doğruluğunu etkilemeyen çok sayıda bit bulunmasıdır. Başarımı artırmaya yönelik yapılar bu tür bitlere bir örnek olarak verilebilir.

İşlemcilerde buyruklar tarafından üretilen sonuçlar daha sonraki buyruklar tarafından kullanılmak üzere geçici olarak yazmaç öbeği adı verilen bir tabloda saklanır. Bu

tablodaki verilerde oluşan bir hatanın hatalı bir çıktıya yol açma ihtimali yüksek olduğundan bu tablonun korunması işlemcinin güvenilirliği bakımından önem taşır.

Yazmaç öbeğinin gerçekleştirilmesinde SRAM bit hücreleri ile oluşturulan bir tablo kullanılmaktadır. Güncel süperskalar işlemciler bir çevrimde birden fazla buyruk için yazmaç öbeğinden veri okumakta ve birden fazla buyruğun sonucunu yazmaç öbeğine yazmaktadır. Bunu sağlamak için çok bağlantılı, büyük alana ve güç ihtiyacına sahip tablolar kullanılır. Ayrıca pek çok işlemcide boru hattının en uzun aşaması yazmaç öbeğine erişimi içerdiği için erişim süresi çevrim süresini etkileyebilen hassas bir parametredir. Bu yüksek gereksinimler yazmaç öbeği tasarımında yapılabilecek değişiklikleri kısıtlamaktadır.

Geçici hatalara karşı yaygın olarak kullanılan bir korunma yöntemi olan artıklık yazmaç öbeği gibi bir yapıda başarımı ciddi şekilde değiştireceği için tercih edilmemektedir. Yazmaç öbeğinin korunmasında düşük maliyetli çözümlere ihtiyaç duyulmaktadır.

Bu tezde yazmaç öbeğini gerçeklemede kullanılan SRAM bit hücrelerinin tasarımında, bitin bir kopyasını saklamayı ve iki kopyayı karşılaştırarak olası bir hatayı saptamayı sağlayabilecek değişiklikler önerilmektedir. Benzetimlikler kullanılarak alınan sonuçlar önerilen tasarımın düşük alan, güç ve gecikme maliyetine sahip olduğunu ortaya koymaktadır.

2. GEÇİCİ HATALAR

Kozmik nötron parçacıkları ve paketleme malzemesindeki alfa parçalanması sonucu ortaya çıkan alfa parçacıkları MeV seviyelerinde enerjiler taşımakta ve tümleşik devreye isabet ettiğinde enerjisi bitene kadar devrenin içine doğru yol almaktadır. İzlediği yol üzerinde elektron-delik çiftleri oluşur ve bu elektron-delik çiftleri yeteri kadar saçıldıysa eski haline dönemeyip transistörün kaynak ve savağı arasına sürüklenebilirler ve bir kanal oluşmasına veya var olan kanalın kapanmasına neden olabilirler. Bu da transistörün yanlış bir sinyal vermesine neden olur.

Geçici hatalar nedeniyle çıktının hatalı olması olasılığı pek çok etkene bağlıdır. Kozmik parçacıkların çarpma olasılığının belirlenmesi açısından çalışma ortamı önemlidir. Çünkü deniz seviyesindeki kozmik yağmur yükseklere oranla daha az etkilidir. Paketlemede kullanılan malzeme alfa parçacıklarının oranını belirler. Örneğin boron-10 izotopunun kullanımı geçici hataları artırması [2] nedeniyle bırakılmıştır. Bunlar dışında kullanılan CMOS teknolojisi, transistör türü, boyutu, yük toplama karakteristiği, mantık kapılarının bağlanış şekli, işlemcinin mimari tasarımı, buyruk kümesi mimarisi, çalıştırılan program kodu gibi pek çok etken geçici hataların oluşması ve hatalı çıktıya yol açması ihtimalini etkiler.

2.1 Mimari Hassaslık Katsayısı

Geçici hataları mimari düzeyinde inceleyebilmek ve alttaki donanımın tasarımından soyutlayabilmek için Mimari Hassaslık Katsayısı (MHK) ölçütü [1], bir donanım birimi için, o donanım biriminin herhangi bir bitinin değişiminin çalıştırılan programın çıktısında hata olarak gözlenmesi olasılığı olarak tanımlanmıştır. Mimari Olarak Hatasız Yürütüm (MOHY) bitleri de [1], değişmeleri halinde Buyruk Kümesi Mimarisi (BKM) tarafından belirlenen beklenen çıktıdan farklı hatalı bir çıktıya neden olacak olan bitler olarak tanımlanmıştır. Bu tanımdan yararlanarak MHK (2.1)'deki gibi hesaplanır.

$$MHK = \frac{Donanım\ birimi\ içindeki\ MOHY\ bitlerinin\ sayısı}{Donanım\ birimi\ içindeki\ toplam\ bit\ sayısı} \quad (2.1)$$

MOHY bitleri zaman içerisinde değiştiği ve bir çevrim için MOHY olan bir bit bir başka çevrimde MOHY olmayabildiği için aynı denklem (2.2)'deki gibi de ifade edilebilir.

$$MHK = \frac{\sum_{tüm\ çevrimler\ üzerinden} Donanım\ birimi\ içindeki\ MOHY\ bitlerinin\ sayısı}{Toplam\ çevrim\ sayısı \cdot Donanım\ birimi\ içindeki\ toplam\ bit\ sayısı} \quad (2.2)$$

2.2 Artıklık Yöntemleri

Artıklık yöntemleri geçici hatalara karşı güvenilirliğin artırılmasında yaygın olarak kullanılan yöntemlerdir. Bu yöntemler iki ana başlık altında incelenebilir: (1) alanda ve (2) zamanda artıklık.

2.2.1 Alanda Artıklık

Donanım seviyesinde, bir birimden birden fazla koyup kopyaları birbirleri ile karşılaştırarak aynı olup olmadığına bakılır ve aynı olması gereken kopyalarda bir farklılık bulunursa geçici hata olmuştur denebilir. İşlem birimleri için bir toplayıcı yerine iki toplayıcı koymak, aynı girdileri sağlayarak çıktıları karşılaştırmak ve aynılarsa geçici hata olmadığını varsaymak, alanda artıklık yoluyla toplayıcıyı oluşabilecek tek bit hataya karşı korumak anlamını taşıyacaktır.

Veri saklayan birimler içinse alanda artıklık bir verinin bir kopyasını daha saklamak ve kullanılacağı sırada iki kopyayı karşılaştırmak şeklinde olabileceği gibi, alan açısından daha verimli yöntemler de bulunmaktadır. Hata kodlama yöntemleri ile bir verinin doğrulayıcısı olabilecek bir üstveri üretilip saklanır, ve verinin kullanılacağı sırada veriden tekrar üretilen üstveri ile saklanan üstveri karşılaştırılır, sonuçlar aynı ise hata yoktur denilir.

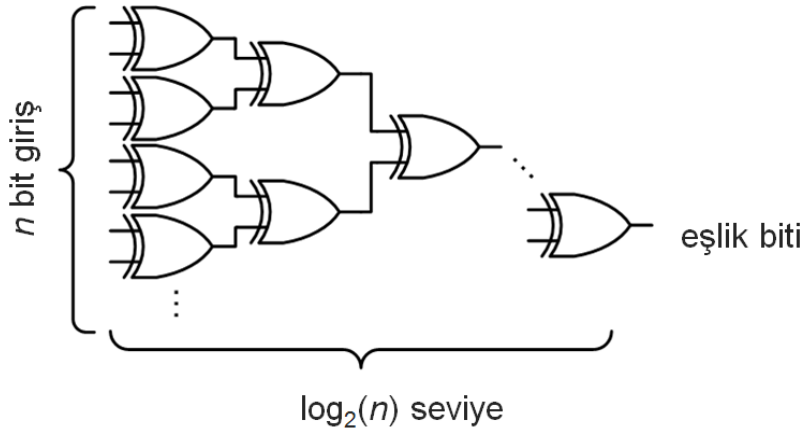
Basit bir hata kodlama yöntemi olarak eşlik biti örnek verilebilir. Korunmak istenen veri içerisindeki 1'lerin sayısının (basamak değerlerinden bağımsız olarak) toplamı tek ise eşlik biti 1, çift ise eşlik biti 0 olarak hesaplanır ve veri ile birlikte saklanır. Eğer daha sonra veriden tekrar eşlik biti hesaplanır ve saklanan eşlik biti ile farklı çıkarsa, veride veya eşlik bitinde, en az bir bit hatalıdır denir. Aynı çıkmaları durumunda hiç hata yoktur denilemez çünkü böyle bir eşlik biti kodlaması yalnızca bir bite kadar olan hataları anlamakta kullanılabilir. Birden fazla bitte hata oluşması yine veriden hesaplanan eşlik biti ile saklanan eşlik bitinin aynı olmasına yol açabilir. Bir bit dizisi için eşlik biti (2.3)'teki mantıksal işlem ile bulunur. Daha sonra bir bit hata (2.4)'te verildiği gibi hesaplanabilir.

$$x_{eşlik} = x_0 \oplus x_1 \oplus x_2 \oplus x_3 \cdots x_{n-1} \oplus x_n \tag{2.3}$$

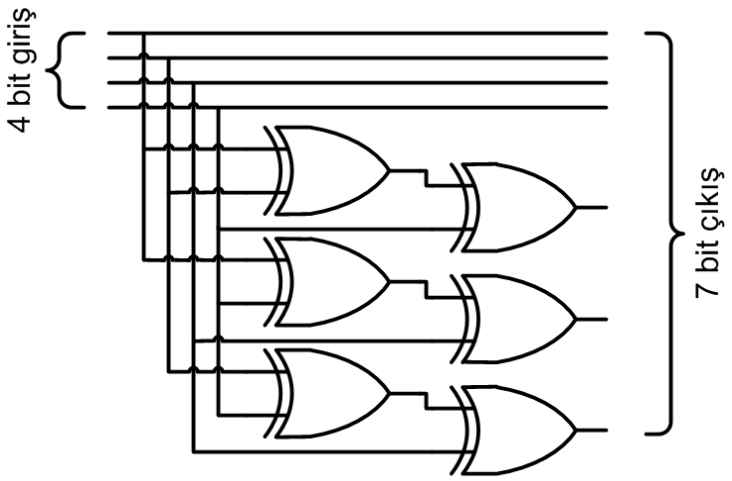
$$hata = x_0 \oplus x_1 \oplus x_2 \oplus x_3 \cdots x_{n-1} \oplus x_n \oplus x_{eşlik} \tag{2.4}$$

Hata oluşup oluşmadığını anlamaya yarayan kodlamalara Hata Saptama Kodları (HSP) denir. Verilen bir veri için ne kadar uzun üstveri üretilirse saptanabilecek azami hata sayısı da artırılabilir, hatanın yeri de saptanarak hatayı düzeltmek mümkün olabilir. Hatanın oluştuğu biti saptayarak düzeltebilmeyi sağlayan kodlamalara Hata Düzeltme Kodları (HDK) denir.

Hamming Kodu, işlemcilerde yaygın olarak kullanılan bir HDK'dir. Hamming(7,4), 4 bitlik veri için 3 adet eşlik biti kullanılarak oluşan azami bir hatayı düzeltmeye ve azami iki hatayı saptamaya izin verir.



Şekil 2.1. Eşlik biti hesaplayan devrenin şeması.



Şekil 2.2. Hamming(7,4) kodu hesaplayan devrenin şeması.

Şekil 2.1 ve Şekil 2.2'de örnek olarak eşlik biti üreten ve Hamming(7,3) kodu üreten devrelerin şeması gösterilmiştir. Veri uzunluğu arttıkça kullanılan toplam kapı sayısı, arka arkaya bağlanan kapı seviyesi ve buna bağlı olarak belirli noktalarda kapıların sürülebilmesini sağlamak için tersleyici konması gerekliliği artmaktadır. Dolayısıyla bu tür kodlama yöntemleri yüksek alan, güç, gecikme artışına ve performans kaybına sebebiyet vermektedir.

Pentium 4 [3], UltraSparc IV [4], Power4 [5], AMD K8 [6] ve Alpha EV6 [7] gibi ticari işlemcilerde ikinci düzey önbelleklerde HDK koruması ve birinci düzey önbelleklerde de HDK veya HSK koruması kullanılmıştır. Yazmaç öbeğinde bu tür bir koruma Intel'in 90nm'de üretilen Itanium adlı sunucu tipi işlemcilerinde kullanılmıştır ancak eşlik bitlerinin hesaplanması fazladan bir çevrim gerektirmiştir [8].

2.2.2 Zamanda Artıklık

Zamanda artıklık, aynı donanımı farklı zamanlarda tekrar kullanarak elde edilen çıktıların karşılaştırılması yoluyla sağlanır. Kullanılan donanımın büyüklüğüne göre farklı seviyelerde incelenebilir. Bir toplayıcı devresi de zamanda artıklık yoluyla hatalara karşı korunabilir, birden fazla işlemcide aynı kod parçası çalıştırarak da tüm kod hatalara karşı korunabilir.

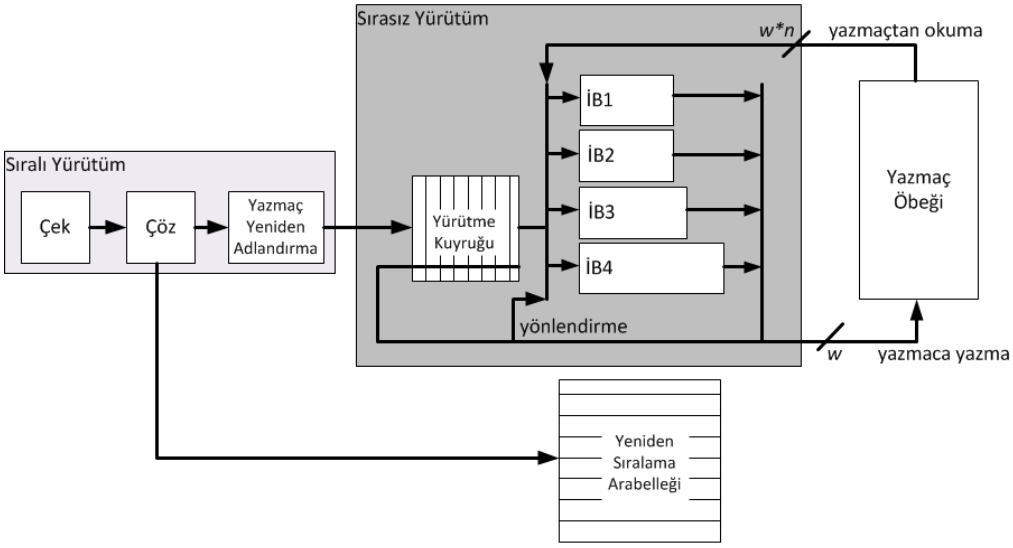
Pek çok çalışma işlemcinin kullanılmadığı zamanlarda artıklık uygulayarak başarım maliyetini düşürmeyi amaçlasa da bu tür artıklık yöntemlerinin neden olduğu güç tüketimi artışı çok fazladır.

2.3 İlgili Çalışmalar

Radyasyon kaynaklı geçici hataların saptanması 1979 yılına kadar uzanmaktadır [9][10]. Konuyla ilgili kapsamlı bir çalışma Mukherjee tarafından yapılmıştır [11]. Hem devre düzeyinde [12][13] hem de pek çok mimari seviyede artıklık uygulayarak [14][15][16][17][18] geçici hatalara karşı önlemler önerilmiştir.

3. SÜPERSKALAR İŞLEMCİ MİMARİSİ

Şekil 3.1'de sırasız yürütüm yapabilen süperskalar bir işlemci gösterilmiştir. İşlemcinin sıralı yürütüm yapan ön kısmı buyrukları bellek sisteminden çekip işlemci içerisine getirir, çözer ve Yazmaç Yeniden Adlandırma (YYA) aşamasında sonuç üretecek olan buyruklara bir sonuç yazmacı atar. Ayrıca buyruklara Yeniden Sıralama Arabelleği (YSA) içinde bir satır atanır ve işlemci içerisinde buyruğu ifade eden bilgiler burada tutulur. Buyruklar Yürütme Kuyruğunda (YK) işlenenlerinin hazır olmasını bekler ve yürütülmek üzere ilgili İşlem Birimlerine (İB) gönderilir. Sonuçların üretilmesi aşamasında hangi sonuçların hazır olduğu YK'ya yönlenderilirken hazır olan sonuçlar arkadan gelen buyrukların işlenenleri ise İB'ye girdi olarak yönlendirilir. Bu sırada daha sonra kullanılmak üzere sonuçlar yazmaç öbeğine yazılır. Eğer kullanılmak istenen işlenen yönlendirilen sonuçlar arasında değilse yazmaç öbeğinden okunur. Yürütülmeleri gereken sırada YSA'da tutulan buyruklar, bir hata (yanlış dallanma öngörüsü gibi) olmaması halinde en eski çekilenden başlanarak işlemci içerisinden atılır.



Şekil 3.1. Süperskalar bir işlemci mimarisinin basitleştirilmiş gösterimi.

Yazmaç öbeğinin aynı çevrim içerisinde işlemcinin süperskalarlık derecesi olan ve *w* ile gösterilen tasarım parametresi adedince yazma yapabilmesi gerekmektedir. BKM tarafından belirlenen, bir buyruğun sahip olabileceği azami işlenen sayısı *n* ise yazmaç öbeğinden aynı çevrim içerisinde *w***n* adet okuma yapılabilmelidir.

4. YAZMAÇ ÖBEĞİ

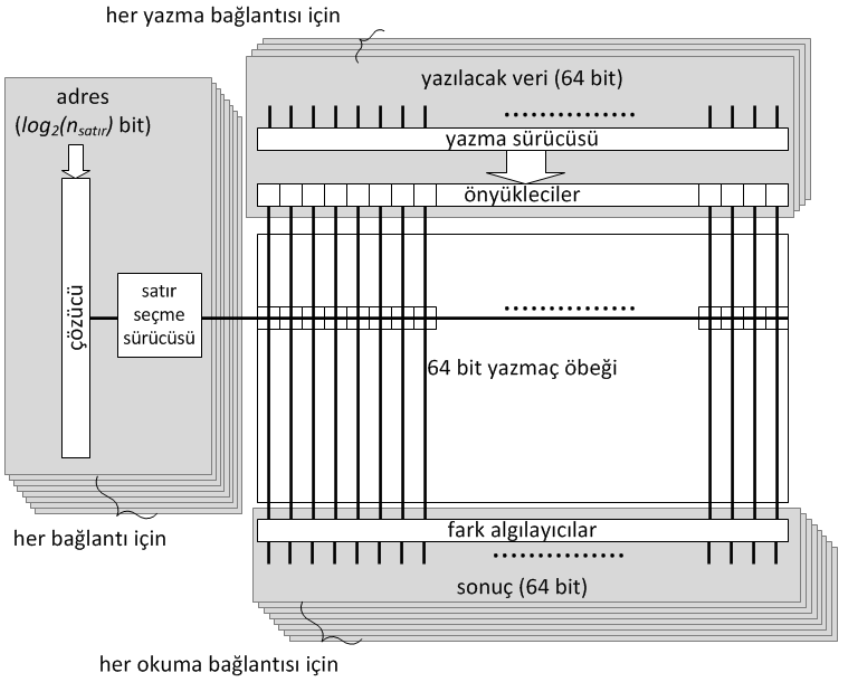
Yazmaç öbeği, işlemcilerde yürütülen buyruklar tarafından üretilen sonuçların saklandığı ve buyrukların işlenenlerinin değerlerini okumak için eriştiği yüksek öneme sahip bir birimdir. Bir çevrim içerisinde birden fazla buyruk yürütme yeteneğine sahip süperskalar işlemcilerde yazmaç öbeği aynı anda birden fazla okuma ve yazma erişimini destekleyebilmelidir. Bunu yapabilmek için yazmaç öbeğinin gerçekleştirilmesinde kullanılan SRAM bit hücresi tablosunun çok sayıda okuma ve yazma bağlantısı olmalıdır.

Bellek elemanlarının teknolojik gelişiminin işlemcilerin hızını yakalayamaması neticesinde bellek duvarı olarak adlandırılan bir tasarım engeliyle karşılaşılınca Buyruk Seviyesinde Parallellik'ten (BSP) faydalanabilmeyi amaçlayan işlemci tasarımları ortaya konmuştur. Bu tasarımda bellek erişimi gerektiren buyruklar için işlemcinin tümünü bekletmek yerine yalnızca ilgili buyruk ve o buyruğun sonucunu kullanacak olan diğer buyruklar bekletilir, bellek erişimi buyruğuna paralel olarak yürütülmesi mümkün olan buyrukların yürütülmesine devam edilir. Yüksek bellek erişimi gecikmesi ve BSP'nin kısıtlılığı başarımın bellek gecikmesi nedeniyle düşmemesi için işlemci içerisine çok sayıda buyruğun alınabilmesini gerektirmektedir. Paralel olarak yürütülebilen ve sonuç üreten buyruk sayısının yazmaç öbeği nedeniyle düşmemesi için kullanılan SRAM tablosunun yeteri kadar satıra sahip olması gerekir [19].

Yazmaç öbeği, maliyetli gereksinimleri dolayısıyla yüksek okuma gecikmesine sahiptir ve genellikle yazmaç öbeği erişimi boruhattının kritik yolu üzerinde yer alır ve dolayısıyla işlemcinin saat hızı yazmaç öbeğinin neden olduğu gecikme ile yakından ilgilidir. Yazmaç öbeğinin bu önemi, tasarımında yapılabilecek değişiklikleri sınırlandırmaktadır.

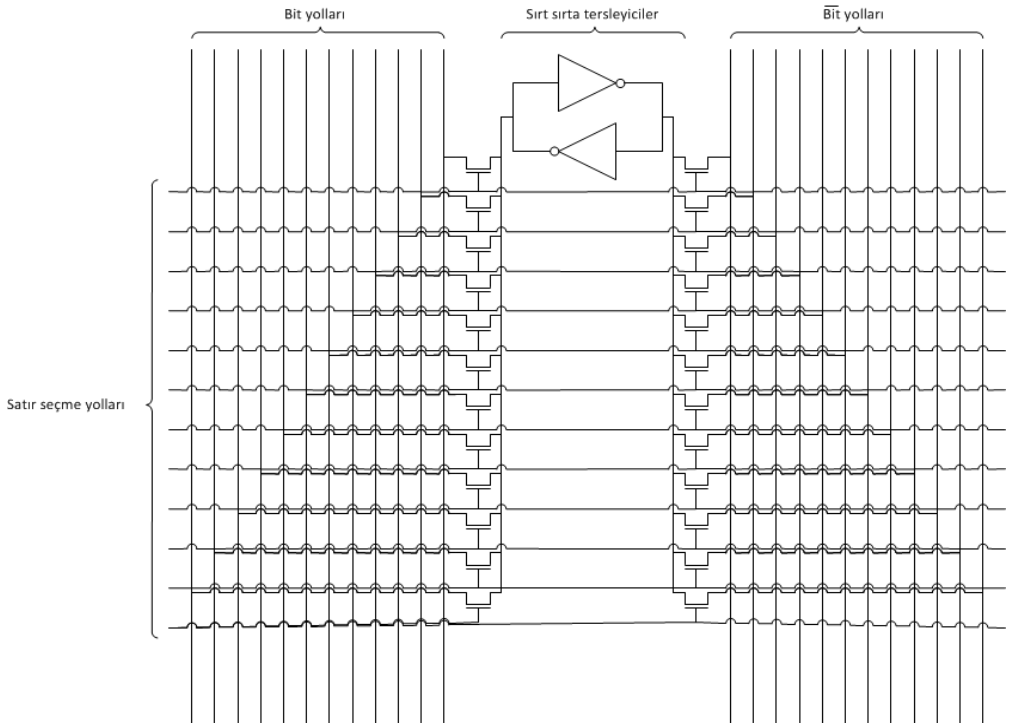
Yazmaç öbeği (Şekil 4.1), bir bit tutan sırt sırta bağlı tersleyicilere bunlara bağlı okuma ve yazma bağlantılarına sahip SRAM bit hücrelerinin (Şekil 4.2) tablosu şeklinde gerçekleştirilir. Her satırdaki her yol, o bağlantıdan yapılacak okuma/yazma

işlemi için farklı bir satırı seçen adres çözücülerin (satır seçme) çıkışlarından birine bağlanır. Her okuma/yazma işleminde her bağlantıdaki her bit yoluna yük yüklenir. Yazma yapılırken bit yolu, açılan bağlantı üzerinden sırt sırta tersleyicilere veriyi yazar. Okuma yapılırken de bit yolları önyüklenir ve sırt sırta tersleyicilerin yol üzerinde neden olduğu ufak yük değişimi fark algılayıcılar tarafından okunan bit hücresinin sakladığı bit değeri olarak yorumlanır. Her bit hücresinde yazmaç öbeğinin bağlantı sayısı kadar satır seçme yolu ve bit yolu bulunmalıdır. Bu nedenle SRAM tablosunun kapladığı alan büyük oranda bağlantı sayısına bağlıdır.

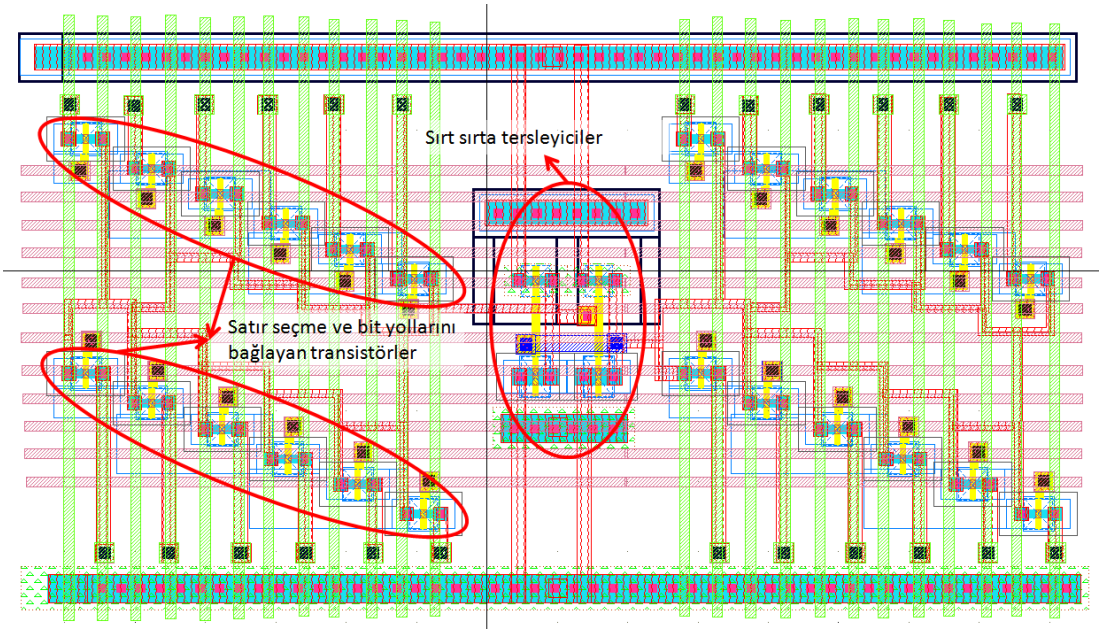


Şekil 4.1. Yazmaç öbeğinin gerçeklenmesinde kullanılan SRAM tablosu.

Aynı çevrim içerisinde işlem birimlerinde dört buyruğu yürütebilen ve dört buyruğun sonucunu yazmaç öbeğine yazabilen bir süperskalar işlemci için, eğer her buyruk en çok iki işlenene sahip olabiliyorsa, yazmaç öbeğinin toplam 12 bağlantısı bulunmalıdır. 12 bağlantılı bit hücresinin şeması Şekil 4.2'de ve serimi Şekil 4.3'de gösterilmiştir. Serim tasarımlarımıza göre asıl verinin saklandığı sırt sırta tersleyiciler bit hücresi alanının %7,5'ini kaplamaktadır.



Şekil 4.2. 12 bağlantılı SRAM bit hücresi.



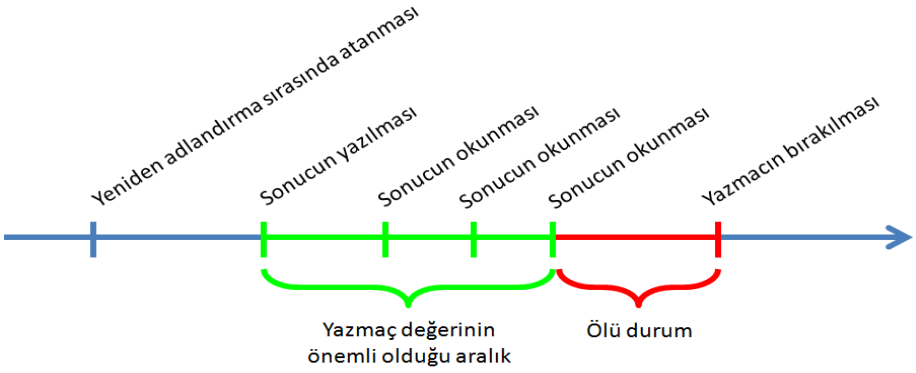
Şekil 4.3. 12 bağlantılı SRAM bit hücresinin serimi.

5. YAZMAÇLARIN MİMARİ HASSASLIĞI

Çoğu güncel işlemci, buyruklar arasında verinin aktarımından kaynaklanmayan ve buyrukların kısıtlı mimari yazmaçları tekrar kullanmasından oluşan geçersiz bağımlılıkları ortadan kaldırmak için yazmaç yeniden adlandırma yöntemini kullanır. Her sonuç üreten buyruğun hedef yazmacı mimari olarak erişilemeyen fiziksel yazmaçlardan birine yeniden adlandırılır ve aynı mimari yazmaca yapılan bir sonraki yazma işlemine dek okumalar atanan fiziksel yazmaçtan yapılır. Fiziksel yazmaçlar YYA aşamasında (boruhattının ön kısmında) ayrılır, ilgili buyruğun sonucu üretildiğinde (boruhattının son kısmında) yazılır ve ancak aynı mimari yazmacı yeniden adlandıran buyruk başarıyla işlemciden atıldığında bırakılır. Yazmacın ömrü şu kısımlara ayrılabilir:

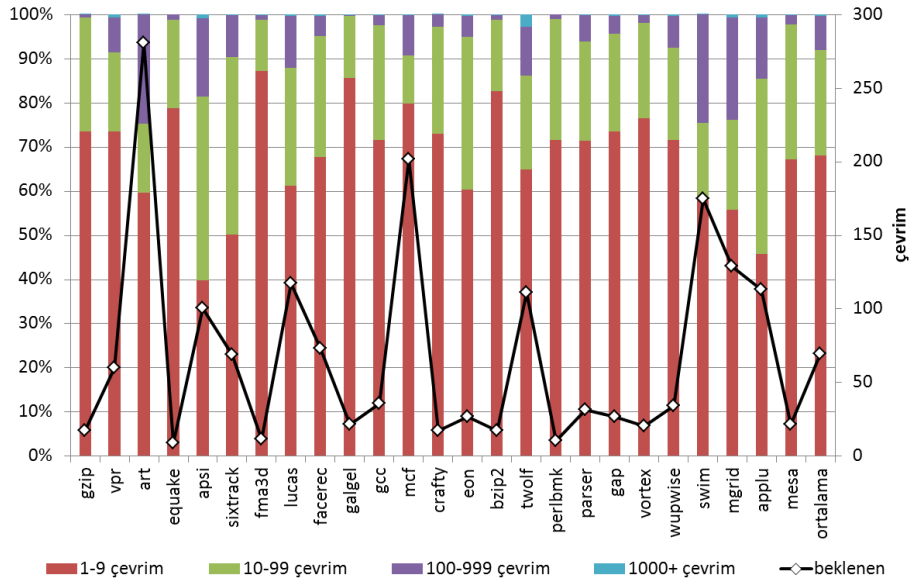
- boş: yazmacın ilk durumu,
- beklemede: yeniden adlandırılmış ve sonucun yazılması bekleniyor,
- yazılmış: sonuç yazılmış ve sonucu kullanan buyruklar okuma yapıyor,
- mimari yazmaç: sonucu yazan buyruk işlemci içinden atılmış fakat yazmaç henüz bırakılmamış.

Bir diğer gizli ama mimari hassaslık açısından önemli olan ölü durumu yazmacın son okunduğu andan bırakılmasına kadarki durum olarak tanımlanabilir (Şekil 5.1).



Şekil 5.1. Yazmaç ömrü.

Yazmaçta saklanan veri ancak sonucun yazılmasından yazmacın bırakılmasına kadar geçerlidir ve bu yazmaç ömrünün küçük bir bölümünü oluşturur. Ayrıca ölü durumdaki bir yazmacın sakladığı veri artık gerekli olmadığından veride oluşabilecek bir bit değişimi hataya neden olmayacaktır. Yazmaç verisi yazıldığı andan son kez kullanıldığı ana kadar mimari açıdan hassastır. Bir yazmacın mimari hassasiyeti düşük olsa da yazmaç öbeği yüksek kullanım oranına sahip bir yapıdır ve oluşabilecek bir hata veriyi kullanan diğer buyruklarca yayılarak gözle görülür bir hataya neden olabilir.



Şekil 5.2. SPEC CPU2000 ölçüm programları için yazmaçların yararlı veri sakladıkları sürelere göre oranları ile beklenen yararlı veri saklama süresi.

Yazmaçların yararlı bilgi sakladıkları ve geçici hatalara karşı hassas oldukları zaman diliminin çoğu zaman çok kısa olduğu gözlemlenebilir. Şekil 5.2'de SPEC CPU2000 ölçüm programlarının benzetimlikleri sonucu elde edilen yazmaçların yararlı veri sakladıkları sürelere göre yüzdeleri (sol y-ekseni) ve her program için bir yazmacın yararlı veri sakladığı sürenin beklenen değeri çevrim cinsinden (sağ y-ekseni) verilmiştir. Yazmaçların büyük bir bölümü 1 ila 9 çevrim süresince yararlı veri

saklamaktadır. Ancak beklenen süre (ağırlıklı ortalama) bazı programlar için oldukça yüksek olabilmektedir. Bunun nedeni çok uzun bir süre bırakılmayan az sayıdaki yazmacın toplam mimari hassaslığın büyük bir bölümünden sorumlu olmasıdır.

6. YERLEŞİK KARŞILAŞTIRICILI SRAM BİT HÜCRESİ TASARIMI

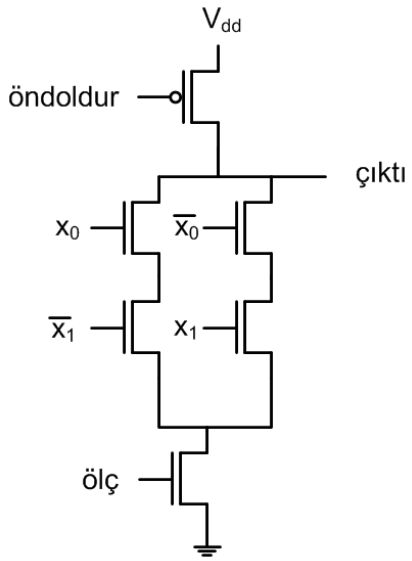
SRAM bit hücrelerine bağlantı yollarına bağlanmayan bir çift daha sırt sırta evirici ekleyerek bit hücresinin içeriğinin bir kopyasını saklamak ve gerektiğinde kopyayı asıl içerikle değiştirmek [20]'de sunulmuştu. Önerilen bit hücresi tasarımı bu fikri temel almakta ve eklenen ikinci saklama alanı ile asıl saklama alanı arasına dinamik mantık kullanan karşılaştırıcılar eklemektedir. Bu sayede yazmaç içeriğinin kopyasını almak ve istenildiğinde asıl veri ile kopyayı karşılaştırarak oluşabilecek bir bit değişimini saptamak mümkün olmaktadır. Fakat bir bit değişimi olması halinde saptansa dahi bir hatanın oluşması engellenemeyebilir ve Saptanmış Giderilemeyen Hata (SGH)'ya yol açabilir [21].

Önerilen karşılaştırıcılar kullanılarak yazmaç verisi Gizli Veri Bozulması (GVB)'na karşı korunabilir. Ancak bit değişimi olması halinde verinin düzeltilmesi ancak ilgili yazmacın henüz mimari duruma geçmemiş olması ile mümkündür. Mimari duruma henüz geçmeyen bir yazmacın değerini yazan buyruk henüz işlemci içinde olduğundan ilgili buyruğu ve sonraki tüm buyrukları iptal edip yürütmeye aynı buyruktan devam etmek hatayı ortadan kaldırır. Eğer geçici hata nedeniyle bozulan veriyi yazan buyruk işlemci içinde değilse doğru veri bulunamayacığından SGH oluşur ve gerekli müdahele diğer mekanizmalarca yapılmalıdır. Örneğin daha önceki bir kontrol noktasına dönülebilir veya çalıştırılmakta olan işlem sonlandırılabilir.

Önerilen tasarımda [20]'deki gibi bir çift sırt sırta tersleyici eklemeye ek olarak her bit hücresi içerisinde bitsel karşılaştırma yapabilmeyi sağlayan karşılaştırıcılar bulunmaktadır. Dinamik mantık kullanan aşağı çekmeli karşılaştırıcı Şekil 6.1'de gösterilmiştir. Karşılaştırıcının girişleri olan x_0 ve x_1 birbirinden farklı olduğunda sol yol veya sağ yol açılacağından, öndoldurulan çıktı, ölç anahtarı etkinleştirildiğinde aşağı çekilecektir. Eğer girişler aynı ise iki yol da kapalı olacağından çıktı yüksek kalacaktır.

Genellikle yerleşik karşılaştırıcılı SRAM bit hücreleri saklanan veri ile girdiyi karşılaştırabilen CAM yapılarını gerçekleştirmek için kullanılır. CAM dizileri

işlemci içerisindeki pek çok yapının gerçekleştirilmesinde kullanılmaktadır. Bekleme kuyruğundaki buyrukların seçiminde, yükleme saklama kuyruğundaki sakla'dan yükle'ye veri yönlendirmede ve tam ilişkili arabelleklerde CAM dizileri bulunur. CAM dizilerinde bir veri dizinin içinde aranmaktadır ve bulunamaz ise tüm karşılaştırıcılar eşit değil çıktısı verecektir, bulunması halinde de tek bir satır eşit çıktısı verirken diğer tüm satırlar yine eşit değil çıktısı verecektir. Bu nedenle CAM tasarımında değişik karşılaştırıcılar, örneğin eşit olmadığında enerji harcayan karşılaştırıcılar [22] tercih edilebilir. Ancak eşit olması gereken ve ender olarak farklı olan iki verinin karşılaştırılmasında eşit olduğunda enerji harcayan karşılaştırıcı tasarımı uygundur. Bu tasarımda satırdaki tüm bit çiftleri eşit olduğunda neredeyse hiç enerji harcanmamaktadır ki bu geçici hata olmadığı sürece beklenen durumdur.

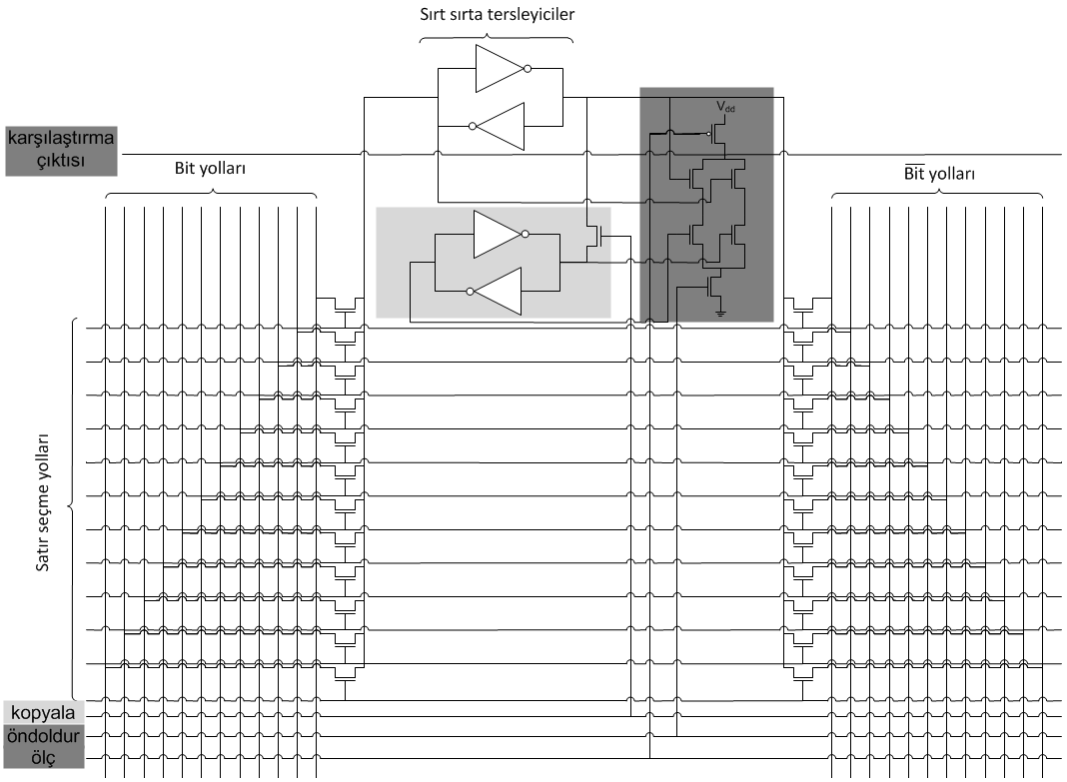


Şekil 6.1. Dinamik mantık kullanan aşağı çeken bitsel karşılaştırıcı.

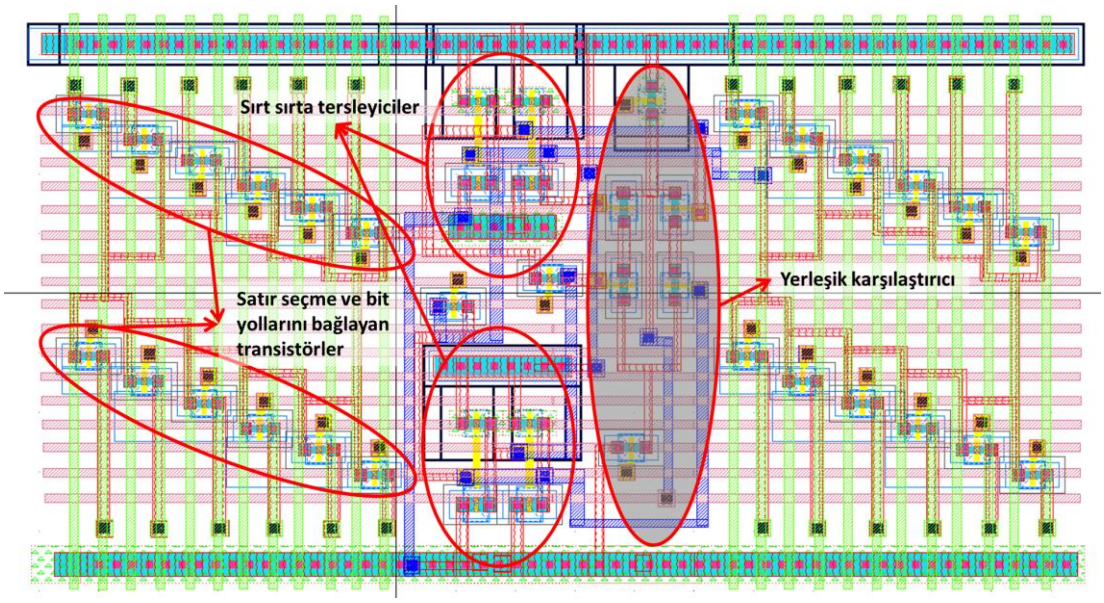
Kopyanın alınması için bit hücresine verilen kontrol noktası al işareti yapılan tasarımda ayrı bir satır seçme yolu olarak eklenmiştir ve bu sayede yazmaç öbeğinin normal çalışmasından bağımsız olarak istenilen anda istenilen yazmaç için kopya almak mümkündür. Fakat basitleştirmek adına, bu sinyalin yazma bağlantılarından

gelen sinyal ile de sürülebilir. Bu şekilde ayrı adres çözücülere ve satır seçme yollarına gerek kalmaz.

Kontrol noktası alındıktan sonra *öndoldur* sinyali sürülerek karşılaştırma çıktısı yüklenmelidir. Daha sonra, tercihen öndoldurma transistörü kapandıktan sonra, ölç sinyali verilerek eğer girdiler eşit değilse çıktının düşük gerilime çekilmesi sağlanmalıdır. Karşılaştırma işlemi, eğer bir geçici hata saptandığında sonucu üreten buyruğu ve sonrakileri iptal ederek hatadan kurtulmak isteniyorsa, sonucu üreten buyruk işlemci içerisinden atılmadan önce yapılmalıdır. Bu tasarımda bit hücresine saklanan kopyanın tekrar yazılması gerekmediğinden [20]'deki geri yazma sinyali ve bununla ilgili devre elemanlarının eklenmesine gerek yoktur.



Şekil 6.2. Yerleşik karşılaştırıcılı 12 bağlantılı bit hücresi tasarımı şeması. Bir kopya daha bulundurmayı sağlayan kısımlar açık gri ile ve karşılaştırmayı sağlayan kısımlar koyu gri ile gösterilmiştir.



Şekil 6.3. Yerleşik karşılaştırıcılı 12 bağlantılı bit hücresi tasarımı serimi.

Önerilen devre tasarımı şema ve serim düzeyinde Şekil 6.2'de ve Şekil 6.3'de gösterilmiştir. [20]'de belirtilen alan artışı %25 iken, önerilen tasarımın neden olduğu alan artışı %30 olarak ölçülmüştür. [20]'de de belirtildiği gibi yazmaç öbeğinin bağlantı sayısı arttıkça bu artış oranı düşecektir. 20 bağlantılı SRAM bit hücresi için ölçülen alan artışı %25,7'dir ancak bu çalışmada 12 bağlantılı SRAM bit hücreleri kullanılmıştır.

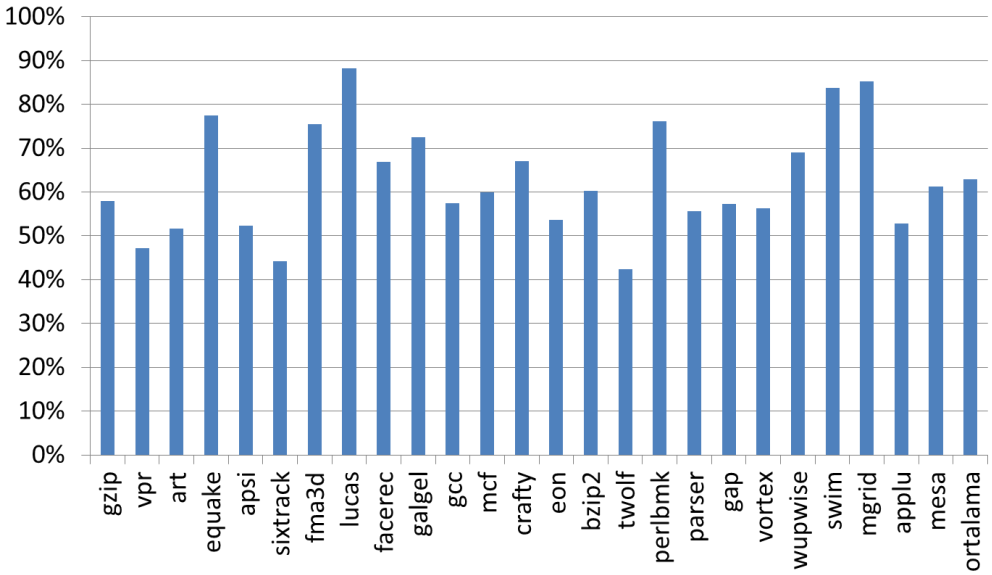
7. ALAN VE GÜÇ MALİYETİNİ AZALTMA

7.1 Kısa Ömürlü Yazmaçlar ile Alan Maliyetini Azaltma

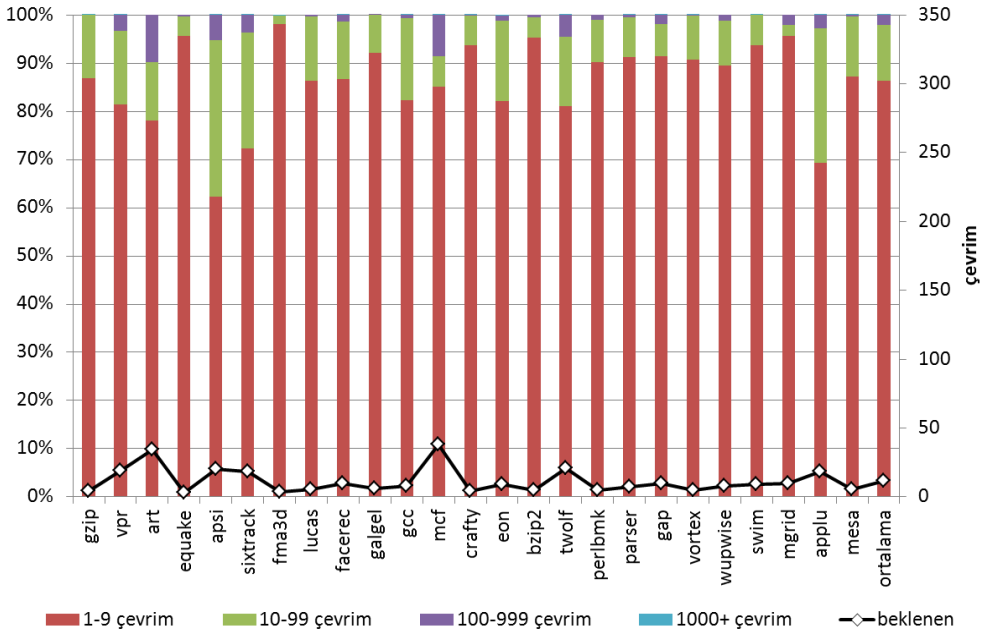
Önerilen tasarımın neden olduğu ek alan maliyetini azaltmak için yerleşik karşılaştırıcılı bit hücreleri yazmaç öbeğinin bazı satırlarında kullanılabilir. Şekil 5.2'de gösterildiği gibi birçok yazmaç kısa hassaslık sürelerine sahip olsa da beklenen süre, az sayıda ama çok uzun hassaslık süresine sahip yazmaçlar nedeniyle yüksek çıkmaktadır. Yazmaç öbeğinin korumalı ve korumasız satırlar olarak ikiye ayrılması durumunda korumalı yazmaçları çok uzun süre kullanılan yazmaçlara atamak düşük alan maliyeti ile MHK'nın oldukça azalmasını sağlar.

Korumalı yazmaçların etkin kullanılabilmesi için yazmaç atama işleminin yazmaç ömrünü de göz önünde bulundurması gerekir. Bunu sağlamanın bir yolu belirli bir bellek konumundaki buyruğun son çalıştırılması sonucu yazdığı yazmacın süresini hatırlamak olabilir. Şekil 7.1'de gösterildiği gibi aynı buyruğun iki ardışık yürütümü sonucu yazdığı yazmacın ömrü büyük oranda aynı olmaktadır. Bu da gösterir ki, korumalı veya korumasız yazmaç atama kararı buyruğun önceki çalışmasıyla ilgili bilgiye dayanarak yapılabilir.

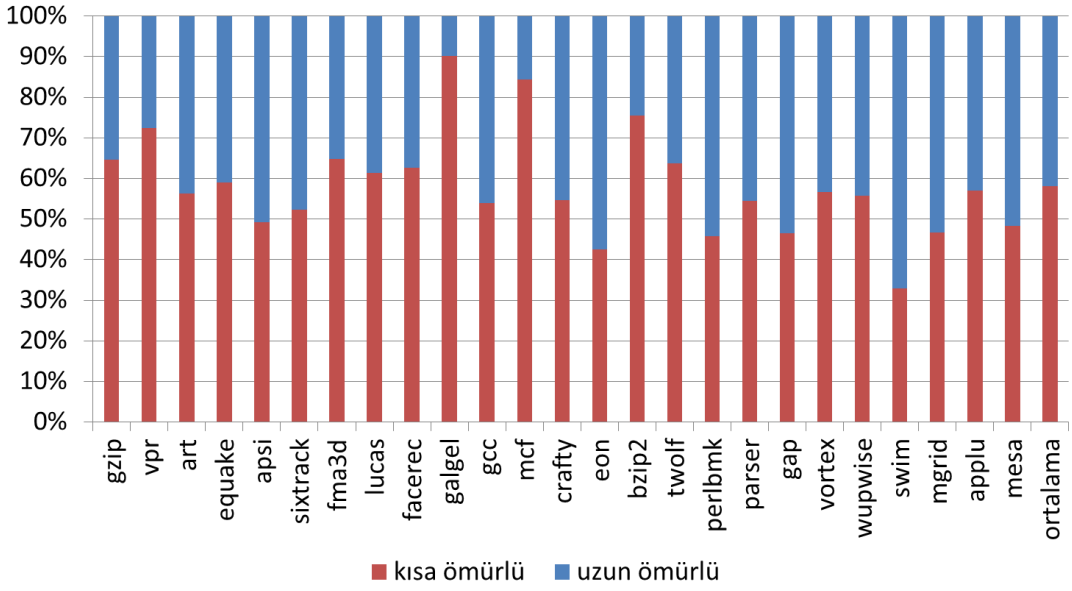
Yazmaçları ömürlerine göre sınıflandırma daha önce bazı çalışmalar tarafından kısa ömürlü yazmaçlar olarak yapılmıştır [23][24][25]. Kısa ömürlü yazmaçlar kendilerinin sonucunun yazılmasına kadar geçen süre içerisinde eşleştirildikleri mimari yazmaç tekrar başka bir fiziksel yazmaca eşleştirilmiş olan yazmaçlar olarak tanımlanmışlardır. Şekil 7.2'de yazmaç ömürlerine göre oranları ve beklenen yazmaç ömrü kısa ömürlü yazmaçlar için Şekil 5.2'e benzer şekilde gösterilmiştir. Sonuçlardan görüldüğü gibi, kısa ömürlü yazmaçlar için hassas veri sakladıkları süre çok kısadır.



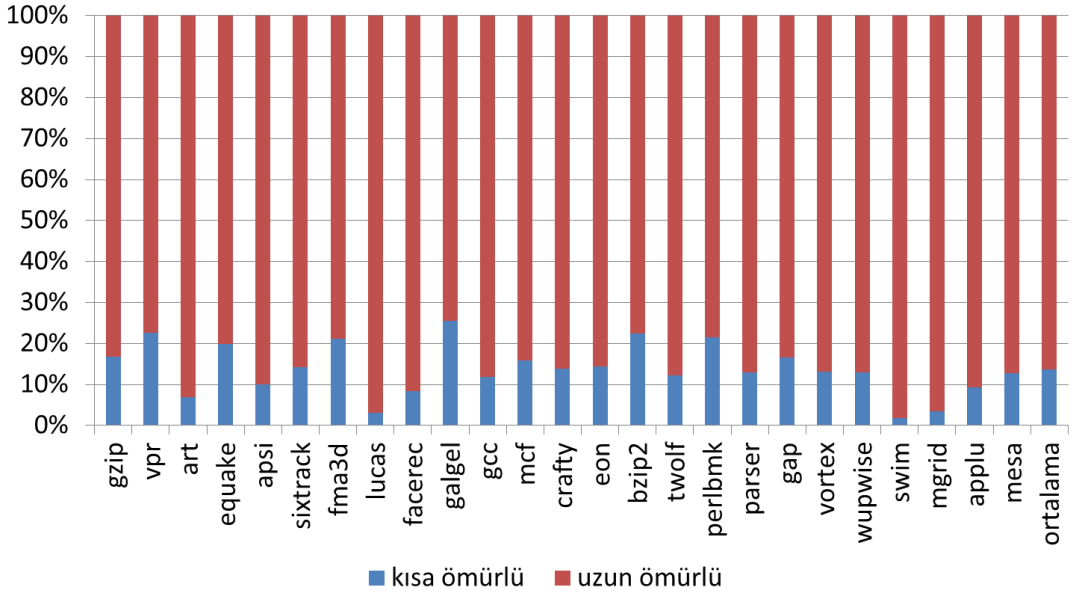
Şekil 7.1. SPEC CPU2000 ölçüm programları için aynı bellek konumundaki buyruğun iki ardışık çalışması sonucunda aynı ömre sahip olan yazmaçların oranı.



Şekil 7.2. SPEC CPU2000 ölçüm programları için kısa ömürlü yazmaçların yararlı veri sakladıkları sürelere göre oranları ile beklenen yararlı veri saklama süresi.

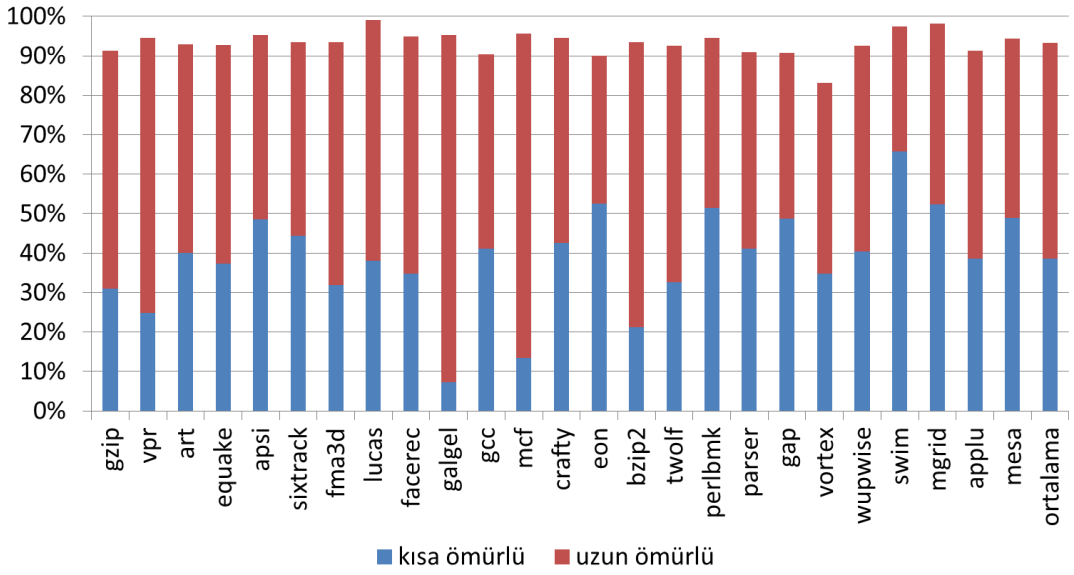


Şekil 7.3. SPEC CPU2000 ölçüm programları için kısa ömürlü yazmaçların oranı.



Şekil 7.4. SPEC CPU2000 ölçüm programları için kısa ömürlü yazmaçların MHK'ya katkıları.

Şekil 7.3 ve Şekil 7.4 gösteriyor ki, kısa ömürlü yazmaçlar yazmaçların büyük bir bölümünü oluştursalar da toplam mimari hassaslığa katkıları az olmaktadır. Şekil 7.3 ile Şekil 7.4'i karşılaştırarak tüm SPEC CPU2000 ölçüm programları için ortalamada yazmaçların %58'i kısa ömürlü iken toplam mimari hassaslığın yalnızca %12'sini oluşturduğu görülebilir. Kısa ömürlü yazmaçların oranıyla MHK'ya olan katkıları genelde orantılı görünmekle birlikte, uç bir sonucu olan *galgel*'de %90 olan kısa ömürlü yazmaç oranı, MHK'nın %25'inden sorumludur.



Şekil 7.5. SPEC CPU2000 ölçüm programları için aynı bellek konumundaki buyruğun iki ardışık çalışması sonucunda da kısa ömürlü veya ikisinde de uzun ömürlü olan yazmaçların oranları.

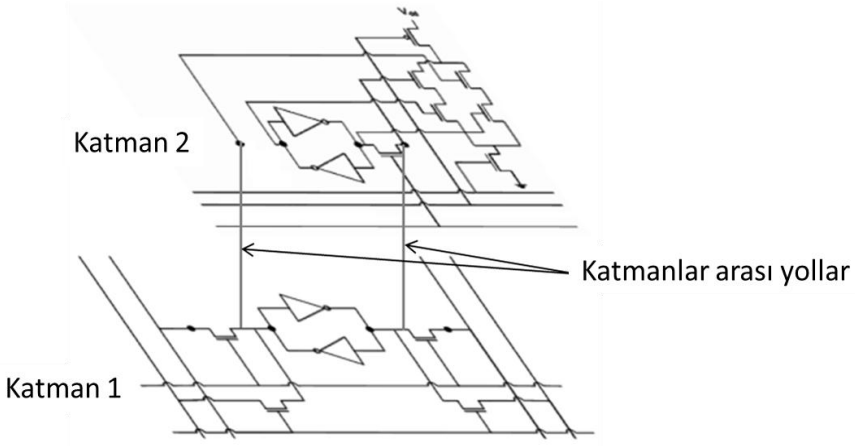
Kısa ömürlü yazmaç sınıflandırması saptaması ve hatırlaması yazmaç ömrünün kaç çevrim sürdüğünü hesaplamaktan ve hatırlamaktan çok daha basit olduğundan kullanışlıdır. Şekil 7.5'de gösterildiği gibi kısa ömürlü yazmaçlar için sadece son kısa ömürlülüğü ile aynı olacağı tahmin edilerek bile yüksek oranda (ortalamada %93) doğru tahmin edilebilir. Bununla birlikte nadiren (ortalamada %6) yanlış

tahmin olması halinde bir hata veya sorun oluşmayacak, yalnızca korumalı yazmaçlar yeterince etkin kullanılamamış olacak ve MHK'da yeterli azalma sağlanamayacaktır. Kısa ömürlü sınıflandırması ile buyruklar için tutulan geçmiş önemli ölçüde azalmaktadır. Bu çalışmada her buyruk için bir bit kullanılmıştır ve bu bit [23]'de belirtildiği gibi buyruk önbelleğinde saklanmıştır.

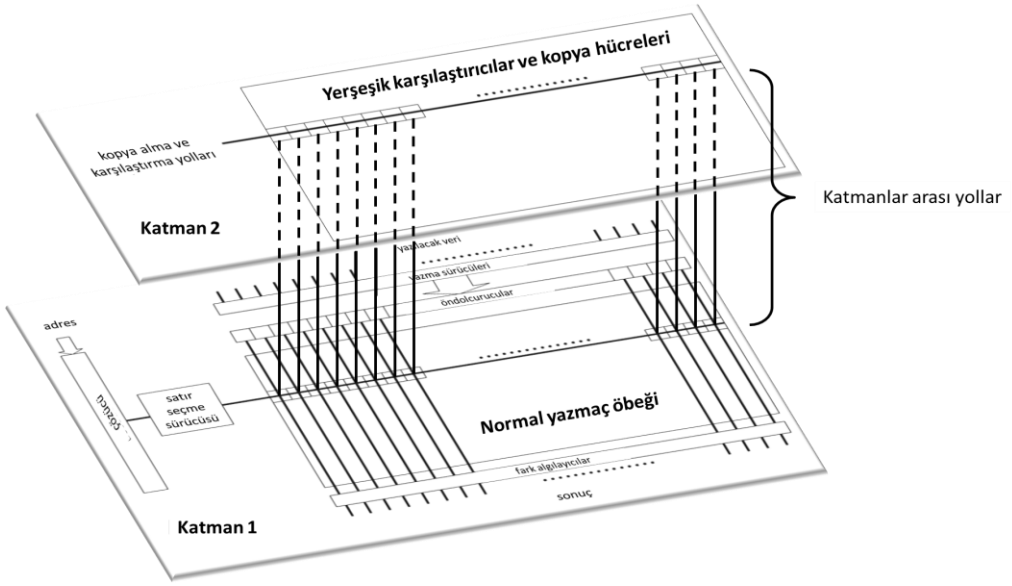
7.2 3 Boyutlu Katmanlı Silikon Mimarisi ile Güç Maliyetini Azaltma

Önerilen tasarımın neden olduğu güç artışı temel olarak uzunluğu ve sığası artan tellerden kaynaklanmaktadır. 3 boyutlu katmanlı silikon mimarisi, tellerin neden olduğu gecikme, alan ve güç maliyetini önemli ölçüde azaltan yeni bir üretim yöntemidir. Üst üste yerleştirilen katmanları delerek açılan katmanlar arası yollar düşük gecikmeli, yüksek bant genişlikli ve düşük güç tüketimine sahip iletişim sağlamaktadır [26].

Önerilen SRAM bit hücresi tasarımı 3 boyutlu katmanlı silikon mimarisi kullanılarak daha etkin tasarlanabilir. [26]'da verilen bağlantı-ayrımlı 3 boyutlu yazmaç öbeği tasarımına benzer şekilde önerilen tasarım Şekil 7.6'da verilmiştir. Bu şekilde oluşturulan yazmaç öbeği şeması da Şekil 7.7'de görüldüğü gibidir.



Şekil 7.6. 3 boyutlu yerleşik karşılaştırıcılı bit hücresi tasarımı. Kopya ve karşılaştırma üst seviyededir. Sadelik açısından sadece iki bağlantı gösterilmiştir.



Şekil 7.7. 3 boyutlu korumalı yazmaç öbeği tasarımı. Alt seviyede normal yazmaç öbeği bulunmaktadır ve kopya ile karşılaştırma üst seviyededir.

3 boyutlu tasarımın bir başka faydası olan geçici hatalara karşı sağladığı fazladan koruma [27]'de ortaya konmuştur. Geçici hata oluşturabilecek yüksek enerjili bir parçacığın iç tarafta bulunan bir katmanın aktif yüzeyine erişebilmesi için dıştaki katmanın içinden geçmesi gerekmektedir. Parçacık, aldığı yol boyunca enerji kaybettiğinden belirli bir mesafe kat ettikten sonra geçici hata oluşturamayacak kadar düşük enerjili hale gelecektir. Bu nedenle çarpan parçacıkların çoğu en dış katman tarafından durdurulurken iç tarafta bulunan katmanlarda geçici hata oluşma oranı azalmaktadır. Bu dış yüzeyin sağladığı kalkan etkisi %90 olarak gösterilmiştir. Bu katmanlar arası farklılıktan yararlanmak amacıyla bit hücrelerinde kopyaları tutan kısımları daha yüksek geçici hata oranına sahip katmana yerleştirilebilir ve karşılaştırıcıların eşit değil çıktısı vermesi durumunda hatayı ortadan kaldırmak yerine düşük geçici hata oranına sahip katmandaki kopya hatasız varsayılabilir. Bu yerleşim aynı zamanda ısı yayılımı bakımından da tercih edilir. Okuma yazma bağlantıları olan sırt sırta tersleyiciler daha çok kullanılacağından kopyayı tutan sırt sırta tersleyicilerden ve karşılaştırıcıdan daha önce ısınacaktır. Isı kuyusuna ve eğer

varsa pervaneye daha yakın olan kısım daha kolay soğuyacağından bu yerleştirme ile hem geçici hatalara karşı fazladan koruma sağlanır, hem de ısı açısından tercih edilen tasarım sağlanmış olur.

8. BENZETİMLİK ORTAMI VE DENEYSEL ÖLÇÜMLER

Önerilen tasarımlar mikroişlemci benzetimleri ve alt seviye devre benzetim araçları ile değerlendirilmiştir. Kullanılan M-Sim mikroişlemci benzetimi [28] için kullanılan mimari değişkenler Çizelge 8.1'de listelenmiştir. İşlemci benzetimliklerinden alınan sonuçlar ile Cadence devre tasarım araçlarıyla [29] tasarlanan ve ölçülen yazmaç öbeği tasarımı sonuçları birleştirilmiştir. İşlemcinin SPEC CPU2000 ölçüm programları çalıştırılarak elde edilen istatistikleri ile devre düzeyinde ölçülen güç tüketim istatistikleri birleştirilmiştir. Güç tüketimi 90 nanometrede CMOS (UMC) teknolojisinde 1 Volt V_{DD} ile 80ºC sıcaklıkta ölçülmüştür.

Yazmaç öbeğini korumalı ve korumasız satırlar olarak iki parçaya ayırıp alınan sonuçlar gösterdi ki, daha az alan maliyeti ile MHK'yı oldukça azaltmak mümkün. Şekil 8.1'de ölçüm programları için ulaşılan MHK azalışı ile alan maliyetinin ilişkisini farklı korumalı satır sayıları için göstermektedir. Alt kısımda görülen gri kısım korumalı satır sayısı arttıkça artan alan maliyetini göstermektedir. Sol y-ekseni MHK'daki azalmanın oranını verirken sağ y-ekseni yazmaç öbeğinin alan artış oranının vermektedir. Bu sonuçlar göstermektedir ki, 128 yazmacın 68'ini korumalı yazmaç olarak gerçekleyerek ortalamada %80 MHK azalışı sağlanabilmektedir.

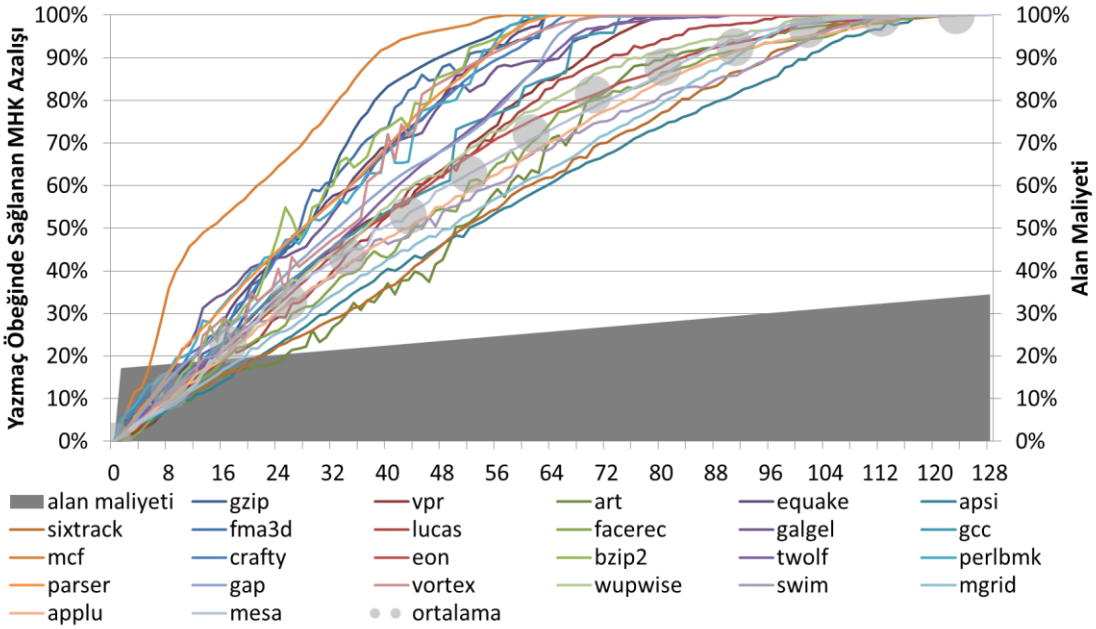
Korumalı yazmaç sayısı sıfır iken (normal yazmaç öbeği), alan maliyeti de sıfırdır ancak yalnız bir yazmaç korumalı iken bile alan maliyeti %17 olarak gösterilmiştir. Bu ani artışın nedeni yazmaç öbeği alanının yazmaç öbeği etrafında çizilen bir dörtgen ile hesaplanması ve aslında boş olan girinti ve çıkıntıların sayılmamasıdır. Tek bir yazmaç için bile koruma uygulandığında bu satır genişliğinin ve dolayısıyla toplam yazmaç öbeği alanının %17 artmasına sebep olmaktadır. Birden fazla yazmacı korumalı yapmanın neden olduğu maliyet ise düşük oranda artan satır yüksekliğinden kaynaklanmaktadır. Eğer yazmaç öbeği tarafından kullanılan gerçek alan hesaplanırsa bu şekilde %0'dan %34'e düz bir çizgi olarak ifade edilecektir. Yapılan tasarımın yazmaç erişim zamanına etkisi satırların tümü korumalı olduğunda bile %2 artış seviyesindedir.

Çizelge 8.1 Benzetimlik için kullanılan mimari değişkenler

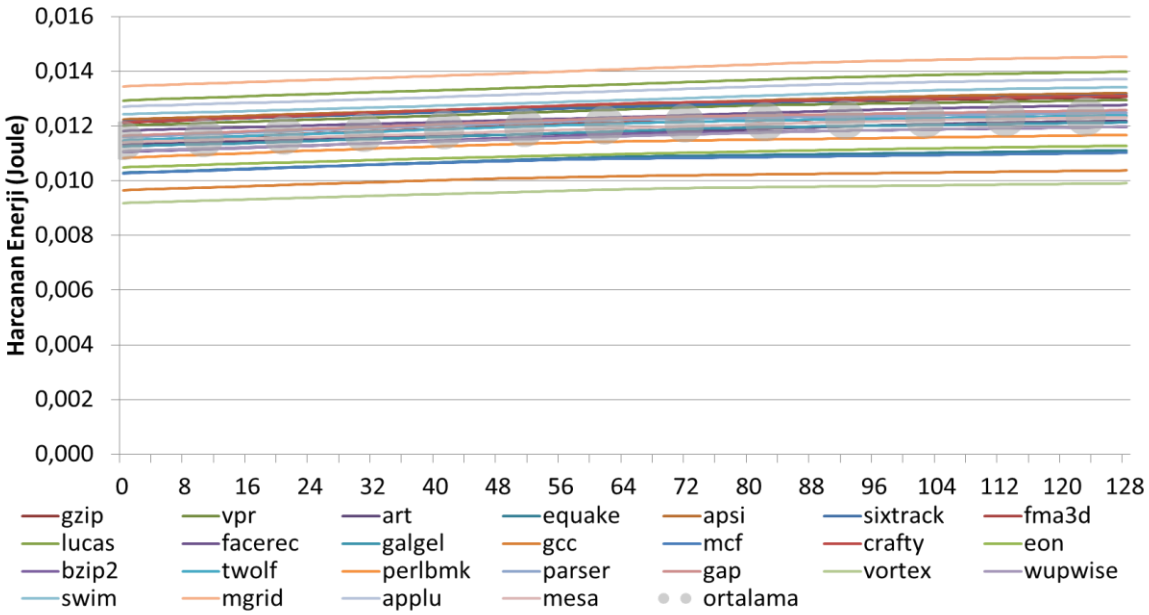
Değişken	Değer
Boru Hattı Genişliği	4'er buyruk çek, çöz, yürüt, yaz
Pencere Boyu	32 satır YK, 48 satır Yükleme/Saklama kuyruğu, 128 satır YSA, 128 satır Yazmaç Öbeği
İşlem Birimleri ve Gecikmeleri	Tamsayı Aritmetik Mantık (4/1), Tamsayı Çarpma/Bölme (1-3/20), Yükleme/Saklama (2-1/1), Kayan Nokta Toplama (4/2), Kayan Nokta Çarpma/Bölme (1-4/12)
L1 Buyruk Önbelleği	64 KB, 2 yollu kümeli ilişkili, 64 bayt satır, 1 çevrim isabet gecikmesi, En Uzun Zamandır Kullanılmayan Tahliyesi
L1 Veri Önbelleği	64 KB, 4 yollu kümeli ilişkili, 64 bayt satır, 1 çevrim isabet gecikmesi, En Uzun Zamandır Kullanılmayan Tahliyesi
L2 Birleşik Önbellek	512 KB, 16 yollu kümeli ilişkili, 64 bayt satır, 10 çevrim isabet gecikmesi, En Uzun Zamandır Kullanılmayan Tahliyesi
Dallanma Hedef Arabelleği	2048 satır, 2 yollu kümeli ilişkili
Dallanma Öngörü Birimi	2048 satır, çift kutuplu öngörücü
Buyruk Adres Dönüştürme Önbelleği	64 satır, 4 yollu kümeli ilişkili

Şekil 8.2'de ölçüm programlarının çalıştırılmaları sonucu yazmaç öbeği tarafından harcanan enerjinin farklı korumalı satır sayıları için değişimi gösterilmiştir. Korumalı satır sayısı arttıkça enerji doğrusal olarak az miktarda artmaktadır. Artışın nedeni kopya alma ve karşılaştırma işlemlerinde kullanılmayan öndoldurucuların ve yazma sürücülerinin enerjisindeki %5'lik artıştır. Kopya alırken ve karşılaştırırken de

kullanılan satır seçme yolu sürücülerinin enerjisindeki artış %8 olduysa da satır seçme yolu sürücüsünde harcanan enerji, okuma işleminde harcanan enerjinin %4'ünü oluşturmaktadır.



Şekil 8.1. Farklı korumalı satır sayılarında (x-ekseni) MHK azalışı ve alan maliyeti.



Şekil 8.2. Farklı korumalı satır sayılarında (x-ekseni) enerji kullanımı.

9. İLGİLİ ÇALIŞMALAR

Yazmaç öbeğinin geçici hatalara karşı güvenilirliğini artırmak için çeşitli yöntemler sunulmuştur. Dar değerlerin işaretlenerek [30] veya yazmaç içerisinde kopyalanarak [31] korunması önerilmiştir. [32]'de yazmaç değerlerinin kullanılmayan yazmaçlara kopyalanması önerilmiştir. [33]'de tanımladıkları Yazmaç Hassaslık Katsayısı'nı azaltmak için derleyici seviyesinde yöntemler sunulmuştur. [34]'de güvenilirliği arttırmak için kısa ömürlü yazmaçlardan faydalanmak önerilmiştir. Yazmaç öbeğine paralel olarak erişilen bir HDK öbeğine uzun ömürlü yazmaçların HDK değerlerini hesaplayarak yazmayı ve daha sonra yazmaç verisini doğrulamayı önermişlerdir. Bu çalışmada önerilen yöntemler ile MHK'yı azaltmak için kısa ömürlü yazmaçlardan yararlanma açısından bazı ortak yönleri bulunsa da bu iki yöntem daha fazla yazmaç öbeği güvenilirliği sağlamak için birlikte kullanılabilir.

10. SONUÇLAR

Bu çalışmada geçici hataları saptamada kullanılabilecek yerleşik karşılaştırıcılı SRAM bit hücresi tasarımı sunulmuştur. Normal okuma/yazma bağlantıları ile erişilemeyen fazladan bir çift sırt sırta tersleyici eklenerek düşük alan maliyeti ile bit hücresinde saklanan değerin bir kopyasının daha saklanabileceği ve ufak yerleşik karşılaştırıcının bit hücresi içerisine eklenerek saklanan değerin geçici hatalara karşı korunabileceği gösterilmiştir. Gerçeklemeye bağlı olmakla birlikte GVB'nin engellenmesi ve en kötü durumda SGH'ye çevrilmesi mümkündür. Önerilen bit hücresi tasarımı ile gerçeklenen bir yazmaç öbeği için alan maliyetinin %34, güç maliyetinin %7,9 ve okuma gecikmesinin %2 arttığı gösterilmiştir.

Maliyetlerin azaltılması amacıyla yazmaç öbeğinde bazı satırların korunarak uzun ömürlü yazmaçların bu korumalı satırlara atanması önerilmiştir. Bu yöntem ile tüm yazmaç değerlerinin korunması mümkün olmasa da düşük alan maliyeti ile MHK'nın önemli bir miktarda azaltılabildiği gösterilmiştir. Ayrıca yeni bir üretim yöntemi olan 3 boyutlu katmanlı silikon mimarisi ile maliyetin nasıl azaltılabileceği bu çalışmada sunulmuştur.

KAYNAKLAR

[1] Mukherjee, S.S., Weaver, C., Emer, J., Reinhardt, S.K., Austin, T., A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor, 36th Annual IEEE/ACM International Symposium on Microarchitecture, 29-40, California, U.S.A, Aralık 2003.

[2] Baumann, R., Hossain, T., Smith, E., Murata, S., Kitagawa, H., Boron as a Primary Source of Radiation in High Density DRAMs, IEEE Symposium on VLSI, 81-82, Haziran 1995.

[3] Intel. "Intel Pentium 4 Processor on 90 nm Process Datasheet". Intel Corporation, Nisan 2004.

[4] Sun Microsystems. UltraSPARC IV Processor Architecture Overview. Sun Microsystems Technical Whitepaper, Şubat 2004.

[5] Bossen, D.C., Tendler, J.M., Reick, K.. Power4 System Design for High Reliability, IEEE Micro, 22(2), 16-24, Mart/Nisan 2002.

[6] Advanced Micro Devices. "AMD Eighth-Generation Processor Architecture," Advanced Micro Devices Whitepaper, Ekim 2001.

[7] R.E. Kessler. "The Alpha 21264 Microprocessor," IEEE Micro, 19(2), 24-36, Mart/Nisan 1999.

[8] E.S. Fetzer, D. Dahle, C. Little, K. Safford, The Parity Protected, Multithreaded Register Files on the 90-nm Itanium Microprocessor, IEEE Journal of Solid-State Circuits, 41(1), Ocak 2006.

[9] May, T.C., Woods, M.H., Alpha-Particle-Induced Soft Errors in Dynamic Memories, IEEE Transactions on Electronic Devices, 26(1), 2-9, Ocak 1979.

[10] Ziegler, J.F., Lanford, W. A., The Effect of Cosmic Rays on Computer Memories, Science, 206(776), 1979.

[11] Mukherjee, S., Architecture Design for Soft Errors, *Morgan Kaufmann*, Burlington, Massachusetts, 2008.

[12] Zhou, Q., Mohanram, K., Cost-effective radiation hardening technique for combinational logic, IEEE/ACM International Conference on Computer-aided design, 100-106, 2004.

[13] Zhang, M., Mitra, S., Mak, T.M., Seifert, N., Wang, N.J., Shi, Q., Kim, K.S., Shanbhag, N.R., Patel, S.J., Sequential element design with built-in soft error resilience, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 14(12), 1368-1378, 2006.

[14] Rotenberg, E., AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors, 29th Annual International Symposium on Fault-Tolerant Computing, 84-91, 1999.

[15] Spainhower, L., Gregg, T.A., G4: A Fault-Tolerant CMOS Mainframe, 28th Annual International Symposium on Fault-Tolerant Computing, 432-440, 1998.

[16] Gaisler, J., Evaluation of a 32-bit Microprocessor with Built-In Concurrent Error-Detection, In Proceedings of International Symposium on Fault-Tolerant Computing, 42-46, 1997.

[17] Zhang, W., Gurumurthi, S., Kandemir, M., Sivasubramaniam, A., ICR: In-Cache Replication for Enhancing Data Cache Reliability, International Conference on Dependable Systems and Networks (DSN'03), 291-300, 2003.

[18] Mitra, S., Zhang, M., Seifert, N., Mak, T.M., Kim, K.S, Soft Error Resilient System Design through Error Correction, Fourteenth International Conference on Very Large Scale Integration of System on Chip, 143-156, 2006.
[19] Farkas, K.I., Chow, P., Jouppi, N.P., Register File Design Considerations in Dynamically Scheduled Processors, IEEE Symposium on High-Performance Computer Architecture, 40-51, 1996.
[20] Ergin, O., Balkan, D., Ponomarev, D., Ghose, K., Early Register Deallocation Mechanisms Using Checkpointed Register Files, IEEE Transactions on Computers, 55(9), 1153-1166, Eylül 2006.
[21] Mukherjee, S.S., Emer, J., Reinhardt, S.K., The soft error problem: An architectural perspective, In Proceedings of International Symposium on High-Performance Computer Architecture, 243-247, 2005.
[22] Ponomarev, D., Küçük, G., Ergin, O., Ghose, K., Energy-Efficient Comparators for Superscalar Datapaths, IEEE Transactions on Computers, 53(7), 892-904 , Temmuz 2004.
[23] Ponomarev, D., Kucuk, G., Ergin, O., Ghose, K., Reducing datapath energy through the isolation of short-lived operands, International Conference on Parallel Architectures and Compilation Techniques, 258-268, 2003.
[24] Balkan, D., Sharkey, J., Ponomarev, D., Ghose, K., SPARTAN: speculative avoidance of register allocations to transient values for performance and energy efficiency, International Conference on Parallel Architectures and Compilation Techniques, 265-274, 2006.
[25] Lozano, L.A, Gao, G.R., Exploiting short-lived variables in superscalar processors, International Symposium on Microarchitecture, 292-302, 1995.
[26] Puttaswamy, K., Loh, G.H., 3D-Integrated SRAM Components for High-Performance Microprocessors, IEEE Transactions on Computers, 58(10), 1369-1381, Ekim 2009.
[27] Zhang, W., Li, T., Microarchitecture Soft Error Vulnerability Characterization and Mitigation under 3D Integration Technology, International Symposium on Microarchitecture, 435-446, 2008.
[28] M-Sim: A Flexible, Multithreaded Architectural Simulation Environment, Technical Report CS-TR-05-DP01, Department of Computer Science, State University of New York at Binghamton, Ekim 2005.
[29] "Virtuoso Spectre Circuit Simulator", erişim adresi: http://www.cadence.com/products/cic/spectre_circuit/ , erişim tarihi: 13 Temmuz 2009.
[30] Ergin, O., Unsal, O., Vera, X., Gonzalez, A., Reducing Soft Errors through Operand Width Aware Policies, IEEE Transactions on Dependable and Secure Computing, 6(3), 217-230, Temmuz/Eylül 2009.
[31] Hu, J., Wang, S., Ziavras, S.G., In-Register Duplication: Exploiting Narrow-Width Value for Improving Register File Reliability, International Conference on Dependable Systems and Networks, 281-290, 2006.
[32] Memik, G., Kandemir, M.T., Ozturk, O., Increasing Register File Immunity to Transient Errors, Conference on Design, Automation and Test in Europe - Volume 1, 586-591, 2005.
[33] Yan, J., Zhang, W., Compiler-guided register reliability improvement against soft errors," International Conference on Embedded Software, 203-209, 2005.

[34] Montesinos, P., Liu, W., Torrellas, J., Using Register Lifetime Predictions to Protect Register Files Against Soft Errors, International Conference on Dependable Systems and Networks, Haziran 2007.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: KAYAALP, Mehmet
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 01.08.1987 Ankara
Medeni hali	: Bekar
Telefon	: 0 (312) 292 42 90
Faks	: 0 (312) 292 42 90
e-mail	: mkayaalp@etu.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	TOBB ETÜ/Bilgisayar Mühendisliği	2008

İş Deneyimi

Yıl	Yer	Görev
Ocak – Nisan 2006	Baştaş Çimento	Ortak Eğitim
Mayıs – Ağustos 2007	Mikes Elektronik	Ortak Eğitim
Haziran – Ağustos 2008	UPA Makine	Ortak Eğitim

Yabancı Dil

İngilizce, İspanyolca

Yayınlar

Kayaalp, M., Ergin, O., Ünsal, O., Valero, M.. Exploiting Inactive Rename Slots for Detecting Soft Errors, International Conference on Architecture of Computing Systems, 126-137, 2010.

Kayaalp, M., Özyer, T., Özyer, S.T., A Collaborative and Content Based Event Recommendation System Integrated with Data Collection Scrapers and Services at a Social Networking Site, International Conference on Advances in Social Network Analysis and Mining , 113-118, 2009.

Tavlı, B., Kayaalp, M., Ceylan, O., Bağcı, İ.E., Data processing and communication strategies for lifetime optimization in wireless sensor networks, AEU - International Journal of Electronics and Communications (kabul edildi), doi:10.1016/j.aeue.2009.08.004, 2008.