# ETİKET SADELEŞTİRME: UYANDIRMA MANTIK DEVRESİNDEKİ KARMAŞIKLIĞI DÜŞÜREREK GÜÇ KAZANIMI

**VEHBİ EŞREF BAYRAKTAR**

**YÜKSEK LİSANS TEZİ**

**BİLGİSAYAR MÜHENDİSLİĞİ**

**TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ**

**FEN BİLİMLERİ ENSTİTÜSÜ**

**EYLÜL 2011**

**ANKARA**

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Ünver KAYNAK

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU

Anabilim Dalı Başkanı

Vehbi Eşref BAYRAKTAR tarafından hazırlanan ETİKET SADELEŞTİRME: UYANDIRMA MANTIK DEVRESİNDEKİ KARMAŞIKLIĞI DÜŞÜREREK GÜÇ KAZANIMI adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Yrd. Doç. Dr. Oğuz ERGİN

Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Doç. Dr. Bülent Tavlı _____

Üye : Yrdç Doç. Dr. Çağdaş Evren GEREDE_____

Üye : Yrd. Doç. Dr. Oğuz ERGİN _____

## 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.

……..……..

Vehbi Eşref BAYRAKTAR

**Üniversitesi : TOBB Ekonomi ve Teknoloji Üniversitesi**

**Enstitüsü : Fen Bilimleri**

**Anabilim Dalı : Bilgisayar Mühendisliği**

**Tez Danışmanı : Yrd. Doç. Dr. Oğuz ERGİN**

**Tez Türü ve Tarihi : Yüksek Lisans – Temmuz 2011**

**Vehbi Eşref BAYRAKTAR**

# ETİKET SADELEŞTİRME: UYANDIRMA MANTIK DEVRESİNDEKİ KARMAŞIKLIĞI DÜŞÜREREK GÜÇ KAZANIMI

## ÖZET

Program sırasını dağıtan işlemcinin, program sırasını yeniden düzeltmeye yarayan yeniden sıralama belleği dışında, en önemli yapısı yayın kuyruğudur. İşlemci, buyrukları bellekten getirip çözdükten ve diğer bazı dönüşüm işlemlerini yaptıktan sonra çözülmüş buyrukları yayın kuyruğuna atar. Bu kuyrukta buyruklar okuyacakları kaynak verilerinin hazır olmasını bekler. Her iki kaynağı da hazır olan buyruklar yürütülmek üzere işlem birimlerine atanır, işlemlerin tamamlanmasının ardından sonuçlar yazmaç birimine yazılır. Her sonuç üreten buyruk, ürettiği sonucun yazmaç numarasını kendisinden sonra gelen ve bu değere bağımlı olan buyruklara bildirmek için yayın kuyruğuna gönderir. Yayın kuyruğundaki tüm buyruklar, bekledikleri kaynak verilerinin yazmaç numaralarını yayınlanan bu numaralarla bir karşılaştırıcı devre kullanarak karşılaştırır ve numaraların aynı olması durumunda kaynağın hazır olduğunu belirten bir biti birler. Her bir saat vuruşunda çok sayıda karşılaştırma, yazma ve okuma işleminin yapıldığı yayın kuyruğu güncel mikroişlemci çekirdeklerindeki en çok güç tüketen bileşenlerden biridir. Sayısal devrelerde iki tür güç tüketimi vardır: devingen (dinamik) güç tüketimi ve durağan (statik) güç tüketimi. Devingen güç tüketimi işlemlerin yapılması sırasında transistörlerin açılıp kapanması, ara düğümlerdeki sığaların yükle dolup boşalması nedeniyle oluşur. Durağan güç tüketimi ise, işlemcide hiçbir işlem yapılmasa da, transistörlerin, kaynak gerilimi ve toprak arasında, akım sızdırması nedeniyle oluşur. Gittikçe küçülen transistör boyutları sızdırma akımından kaynaklanan durağan güç

tüketiminin toplam güç tüketimi içindeki oranını artırmış, yüksek sıcaklıklarda bu güç tüketiminin oranı neredeyse %50 düzeyine gelmiştir.
Çok yollu işlemciler dallanma tahmini gibi teknikler kullandığında işlemci hatalı bir tahmin sonucunda olmaması gereken bir duruma düşer. Yanlışlıkla işlenilmeye başlanan buyrukların yazmaçlarının yeniden adlandırmaları bir şekilde geri alınmalı ve doğru duruma dönülmelidir.

Bu tezde işlemcinin en fazla güç tüketen yapılarından birisi olan yayın kuyruğunun devingen ve durağan güç tüketiminin azaltılması için yöntemler önerilmektedir. Önerilen yöntemler yayın kuyruğunun hem adres karşılaştıran karşılaştırma devrelerinin karmaşıklığını işlemci boru hattının daha önceki aşamalarına aktararak devingen güç tüketimini azaltmaktadır. Devingen güç tüketiminin artırılması için yayın kuyruğunda saklanan yazmaç numaraları gruplanacak ve yazmaç kuyruğu birden fazla sayıda parçaya bölünerek her bir parçada yalnızca belirli bir genişlikte yazmaç numaralarının karşılaştırılması sağlanacaktır. Bu şekilde yayın kuyruğunda önemli sayıda transistör ve bit kaldırılmakta, güç tüketimiyle birlikte gecikme de azaltılmaktadır. Burada kaldırılan karmaşıklık, güç tüketimi ve gecikme işlemcinin daha rahat işlem yapabileceği ön tarafına aktarılmaktadır.

**Anahtar Kelimeler:** Bilgisayar mimarisi, çok yollu işlemciler, Yayın Kuyruğu, Uyandırma mantık devresi, Düşük Güç Tüketimi

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

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

**Science Programme** : **Computer Engineering**

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

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

**Vehbi Eşref BAYRAKTAR**

## TAG SIMPLIFICATION: ACHIEVEING POWER EFFICIENCY THROUGH REDUCING THE COMPLEXITY OF THE WAKE UP LOGIC

## ABSTRACT

The most important structure of an out of order processor, besides the reorder buffer which reorganizes the program flow, is the issue queue. The processor places the instructions in the issue queue after fetching them from memory, decoding them and applying other transformations. In this queue, instructions wait for their source data to be ready. Instructions which have both sources ready are assigned to execution units, and their results are written to the register file after their execution completes. Each result producing instruction broadcasts the register number of its result to the issue queue to inform following instructions that depend on this value. Every instruction in the issue queue compares their unavailable source operands with these broadcasted register numbers using a comparator and sets a ready bit for the source if the comparator matches. The issue queue, which handles many comparison, write and read operations every clock cycle, is one of the most power consuming components in a contemporary microprocessor core. Digital circuits have two types of power consumption: dynamic power consumption and static power consumption. Dynamic power consumption occurs due to transistors turning on and off and capacitors in the intermediate nodes being charged and discharged. Static power consumption occurs due to current leakage between the input voltage and ground of transistors, even when no operation occurs in the processor. Gradually shrinking transistor dimensions have increased the static power consumption due to current leakage, increasing the ratio of static power consumption to almost 50% at high temperatures.

This thesis proposes techniques to reduce the dynamic and static power consumption of the issue queue, one of the most power consuming components in the processor. The proposed methods move the complexity of the adress comparison circuits of the issue queue to the earlier stages of the pipeline to reduce dynamic power consumption. Register numbers stored in the issue queue will be grouped and the register queue will be divided into more than one piece to compare only a certain width of register numbers within that piece to decrease dynamic power consumption. Using this method, a significant number of transistors and bits can be removed from the issue queue, decreasing both power consumption and latency. The complexity, power consumption and latency is moved to the front of the processor where they could be done faster.

**Key Words:** Computer Architecture, Superscalar Processors, Issue Queue, Wakeup Logic, Low Power

## TEŞEKKÜR

Yüksek lisans ve lisans çalışmalarım boyunca bana yardım eden, yönlendiren, desteğini esirgemeyen değerli hocam ve tez danışmanım Yrd. Doç. Dr. Oğuz ERGİN'e, tez konusu ile ilgili çalışmalarım sırasında büyük katkıları olan Z10 Laboratuvarında çalışan tüm lisans ve yüksek lisans öğrencilerine, çalışmalarım sırasında beni maddi açıdan destekleyen TÜBİTAK'a ve beni asla yalnız bırakmayan, en büyük destekçim olan aileme teşekkürü bir borç bilirim.

# İÇİNDEKİLER

**Sayfa**

# ÇİZELGELERİN LİSTESİ

**Çizelge** **Sayfa**

# ŞEKİLLERİN LİSTESİ

**Şekil** **Sayfa**

KISALTMALAR

| **Kısaltmalar** | **Açıklama** |
|---|---|
| **FIFO** | First In First Out |
| **ÇBB** | Çevrim Başına Buyruk |
| **RISC** | Reduced Instruction Set Computing |
| **SRAM** | Static RAM |
| **CAM** | Content Addressable Memory |
| **XOR** | Dışlayan veya |
| **GB** | Görev Birimi |
| **TLB** | Translation Lookeaside Buffer. |

## SEMBOL LİSTESİ

Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

| **Simgeler** | **Açıklamalar** |
|---|---|
| T | Periyot |
| $F_{ak}$ | Transistorün açılıp kapanma sıklığı |
| C | Kapasitans |
| Vdd | Kaynak Gerilimi |
| $\propto$ | Açılıp kapanma sıklığı |

# 1. GİRİŞ

Günümüzde son kullanıcıya yönelik üretilen mikroişlemciler büyük bir hızla gelişmektedir. Sürekli talep edilen başarım artışını sağlamak için işlemcilerin saat sıklığı ile birlikte birim zamanda işlenen buyruk sayısı da artmaktadır. Artan saat sıklığı güç tüketimini de artırdığından başarımı daha az güç tüketen bileşenlerle artırmak önemli bir sorun haline gelmiştir. Çağdaş işlemciler başarımı artırmak için buyrukların işlemci içinde program sırası dışında işlenmesi, aynı saat vuruşunda birden fazla buyruğun bellekten getirilmesi ve boru hattı yöntemi gibi teknikler kullanır. İşlemciler aynı anda çok sayıda buyruğun çalıştırılması için program içindeki buyrukların birbirinden bağımsız işlem yapabilme yeteneğine güvenmek zorundadır. Güncel programlarda programların içinde gelen buyrukların üretilen değerler açısından birbirinden bağımsız olmasından dolayı işlemci içinde program sırası dağıtılır ve program sırasında arkadan gelen bir buyruk kendisinden önce gelen ancak bağımsız işlem yapan bir buyruğun önüne geçerek önceden işlenebilir. İşlemci içinde program sırasının dağıtılmasının temel amacı, önbellekte bulamama gibi nedenlerle boru hattını tıkayan buyruklara bağımlı olmayan buyrukların, boru hattının atıl kaynaklarını kullanması ve tıkanıklık ortadan kalktığından sonuçlarını yazmaya hazır olmasıdır.

Güncel çok yollu mikroişlemciler başarımı artırmak için sıra dışı buyruk yürütme (out-of-order execution) ve dinamik zamanlama (dynamic scheduling) gibi teknikler kullanırlar. Sıra dışı buyruk yürütme tekniği donanımda buyrukları program sırasına göre değil de, bağımlılıklarının tamamlanma sırasına göre işlenebilmesine olanak sağlar. İdeal durumda hiç bir buyruk bir önceki buyruğa bağlı değilse, her buyruk kaynaklar eline ulaştığında (bellek vb. kaynaklardan) işlenebilir hale gelmektedir. Gerçekte ise buyruklar arası bağımlılıkların çözülmesi gerekmektedir. Bu bağlılıkları ise yazmaç yeniden adlandırma denen bir yöntemle ortadan kaldırırız. Bu sayede ise çok yollu bir boru hattında bulunan tüm görev birimleri birbirinden bağımsız olarak dağıtık bir şekilde çalışabilir.

Burada göze çarpan bir diğer mekanizma ise buyrukların çözme aşamasından çalışma aşamasına geçerken buyrukların geçici olarak depolandığı tampon alanıdır.

Bir buyruğunun çalışmaya başlamadan önce, çözme aşamasında yazmaç dosyasından çektiği yazmaç bileşenlerinin hazır durumda olmalıdır. Çok yollu bir işlemcide ise bu yazmaç bileşenlerinden bazıları, kendinden önce gelen herhangi bir buyruğun hedef yazmacı olması sebebiyle henüz en güncel değerini taşımıyor olabilir veya diğer bir deyişle hazır olmayabilir. Böyle bir durumda, çözme aşamasında bir bekleme yapıp başarımda herhangi bir düşüş yaşamak yerine, yayın kuyruğu denen geçici depolama tamponları ortaya çıkmıştır.

Yayın kuyruklarında bekleyen buyruklar, 2 ayrı kaynak yazmacı da hazır olduğu anda, hazır olarak etiketlenirler ve buradan yayınlanırlar. Bu 2 ayrı yazmacın hazır olup olmadığı, kuyruğa gönderilen tamamlanmış işlem sonuçlarının yazmaç numaralarının, yayın kuyruğunun satırlarında bulunan yazmaç numaraları ile karşılaştırılmasıyla yapılır.

Bu karşılaştırma işlemleri ise çok güç tüketen işlemlerin arasında gelir. Bu güç tüketimini azaltmak amacıyla daha önceden bir çok yöntem ortaya sunulmuştur. Bunların bazıları, yayın kuyruğundaki port sayısını azaltma, yayın kuyruğunu parçalara bölerek uyandırma sırasında bu parçaların bazılarını uyutarak bunlara yayın yapmama, ya da yayın kuyruğunu kaynak yazmaçlarının hazır olup olmamasına göre parçalayıp bu karşılaştırma sayısını azaltmadır.

Bizim önerdiğimiz yapı ise daha çok bu kaynak yazmaçlarının numaralarının içeriği ile ilgilidir. Burada yararlandığımız nokta ise bu karşılaştırma olaylarını tamamen iptal etmek yerine karşılaştırma devrelerinde sadeleştirme yoluna gitmektir. Örneğin, bir alpha mimarisini ele alacak olursak, bu mimaride 0'dan 127'ye kadar sıralanmış olan 128 tane fiziksel yazmaç bulunmaktadır. Bu yazmaç numaraları da yayın kuyruğuna yönlendirilen tüm yazmaç numaralarının tanım kümesini oluşturmaktadır. Burada yönlendirilen yazmaç numaraların birer bit vektörü olarak tutulduğunu hatırlarsak, buradaki sadeleşmeye uygun örüntüleri kullanarak bu yapının toplam güç tüketimi azaltılabilir. Bunun sonucunda ise bu bölümlendirme işleminin ÇBB (Çevrim Başına Buyruk) ile yani başarımla olan ilişkilerini ortaya koyduk.

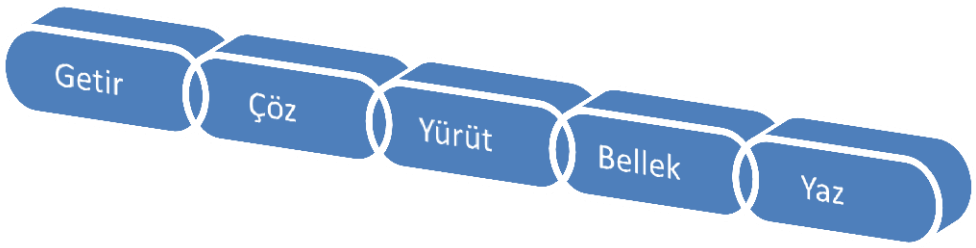
## 2. TEMEL KONULAR

Bu kısım mikro işlemcilerle ilgili bazı temel kavramları açıklayarak bu tezde anlatılan ve bahsi geçen birçok konunun anlaşılması için bir ön hazırlık evresidir.

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

### 2.1. Boru Hattı

İlk işlemciler her saat darbesinde tek bir buyruk işleyecek şekilde tasarlanmıştır. Fakat bu durum işlemcinin hızını en yavaş buyruğun hızı ile sınırlamaktadır. Buyrukların işlenmesini aşamalara bölerek birden fazla buyruğun farklı aşamalarının işlenmesi ile bu soruna çözüm getirilmiştir. Bu yöntem ise boru hattı yöntemi olarak adlandırılmıştır. İşlemcilerin hangi safhasında hangi işlemlerin ne şekilde yapılacağını belirlemek amacıyla buyruk kümeleri geliştirilmiştir. Bunlardan en popüler olanlarından biri de RISC(reduced instruction set computing)'tir. Temel bir RISC makinesinde, Şekil 2 1'de görülen beş aşamalı bir boru hattı bulunmaktadır.




Şekil 1 Temel RISC Boru Hattı

Şekil 1'de görülen boru hattı yapısında Getir safhasında buyruklar işlemciye getirilir. Çöz aşamasında ise buyruğun işlenmesi için yapılacak işlemler, kullanılacak yazmaçlar ve hangi işlem birimlerinin kullanılacağı tespit edilir. Bu aşamanın bir diğer önemli yönü ise buyruklar arası bağımlılıkların ve buyrukların yürütmeye hazır

olup olmadığının tespitinin yapılmasıdır. Yürüt aşamasında buyruk için gerekli olan hesaplama işlemleri yürütme birimlerinden uygun olanında yapılır. Bellek aşamasında ise buyruk bellek erişimi gerektiriyorsa belleğe yazma veya bellekten okuma işlemi gerçekleştirilir. Yaz aşamasında ise buyruk herhangi bir sonuç üretiyorsa üretilen sonuçlar buyruğun belirttiği sonuç yazmacına yazılır. Şekildeki boru hattı en temel haliyle gösterilmiş olup, farklı mimarilerde farklı şekillerde karşımıza çıkabilir.

Birbirinden bağımsız buyruklar geldiği sürece boru hattı kullanan bir işlemci,boru hattındaki tüm safhaları verimli bir şekilde kullanabilmektedir. Böyle bir kullanım ise boru hattının temel amacıdır. Birbirinden bağımsız buyruklar geldiğinde boru hattı Tablo 1'deki gibi görünür.

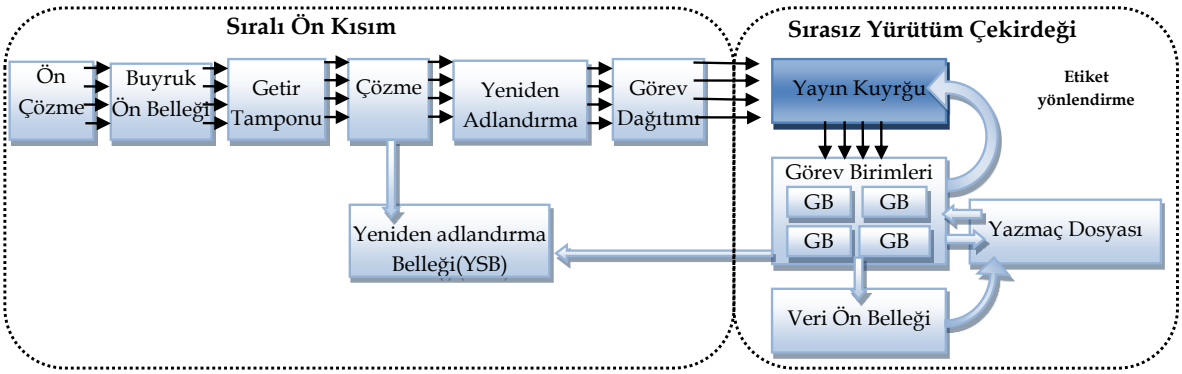
Tablo 1. Boru Hattı Çalışması

| | **Boru Hattı Aşamaları** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Buyruk No.** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **B1** | G | Ç | Yr | B | Y | | | | |
| **B2** | | G | Ç | Yr | B | Y | | | |
| **B3** | | | G | Ç | Yr | B | Y | | |
| **B4** | | | | G | Ç | Yr | B | Y | |
| **B5** | | | | | G | Ç | Yr | B | Y |

Yukarıda da görüldüğü üzere 5. saat darbesi ile birlikte, işlemcideki boru hattının tüm safhaları aynı anda kullanılmaya başlanmıştır. Ancak bu hiçbir buyruğun diğerine bağlı olmadığı, ideal durumda gözlenebilecek bir olaydır. Birçok işlemcinin tasarlanırken en büyük prensiplerinden biri de bu boru hattını sürekli bir şekilde dolu tutabilmektir.

## 2.2. Çok yollu İşlemciler

Mikro işlemciler günümüzde yüksek frekanslarda çalışmakta ve bir seferde birden fazla komut işleyebilmektedir. İşlemci frekansının artması işlemcinin güç tüketimini artırmakta, birim zamanda çok sayıda komut işlenmesi de karmaşıklığı artırarak güç tüketimine katkıda bulunmaktadır. Çok yollu (süper skalar) işlemciler birim zamanda birden fazla komut işlemek için işlemci içinde karmaşık yapılara gereksinim duyarlar. Bunun yanında başarımı artırmak için kullanılan yazmaçların yeniden adlandırılması ve sırasız yürütüm gibi teknikler, başarımı artırırken hem karmaşıklığı hem de işlemcinin içinde aynı anda bulunan komut sayısını artırır.




Şekil 2. Çok Yollu bir Mikroişlemcinin İç Yapısı

Şekil 2'de sırasız yürütüm (out of order execution) yapan çok yollu bir işlemcinin genel mimarisi gösterilmektedir.İşlemci önce yürütülecek buyrukların bellekteki adreslerini hesaplayarak buyruk önbelleğine iletir. Çok yollu bir işlemcide bellekten aynı anda getirilen buyruk sayısı birden fazladır. Buyruk belleğinden gelen buyruklar ara bir bellekte saklanır ve daha sonra buradan alınarak işlemci tarafından çözülür. İşlemcinin buyruğun hangi işlemi yapacağını ve kaynaklarının ne olduğunu anladığı aşama bu çözme aşamasıdır. Çözme aşamasında gereksinimleri anlaşılan buyruklar için işlemci yapılarında yer ayrılır. Eğer buyruğun bir sonuç yazmacı varsa, yeniden adlandırma işleminden önce boş bir yazmaç buyruğa atanır. Program sırasını dağıtan işlemcilerde, işlemlerin yapılmasının ardından program sırasının yeniden

oluşturulabilmesi için, çözme aşamasında program sırası, yeniden sıralama belleğinde saklanır. Her buyrukla ilgili bilgileri tutan ve "ilk giren ilk çıkar" (FIFO) yapısına sahip olan bu yeniden sıralama belleğindeki satır numarası buyruk için işlemci içindeki ömrü boyunca özgün ve ayırıcı bir değerdir.

İşlemci içinde işlenen ve sonuç üreten her komut, sonucunu yazmak için bir yazmaca (register) gereksinim duyar. Intel x86 komut kümesinde mimari düzeyinde derleyici tarafından görülebilen yazmaç sayısı 8'dir. Yazmaç sayısının komut kümesi düzeyinde az olması derleyicinin aynı yazmacı birden fazla komut için kullanmasına yol açar. Aynı yazmaca atanan bu iki komut genelde birbirinden bağımsızdır ve aynı anda çalıştırılabilir. Bu iki komut arasında gerçekte olmayan bağımlılık, işlemci içinde önlem alınmazsa ikinci komutun birinci komutun sonucunu yazmasını ve birinci komutun değerini okuyacak tüm komutların yazılan değeri okumasını beklemesini gerektirir. Aslında olmayan ve başarımı düşüren bu bağımlılığı ortadan kaldırmak için yazmaçların yeniden adlandırılması tekniği kullanılır. Her komutun yazmaçları işlemci içinde yeniden adlandırılır ve daha çok sayıdaki gerçek yazmaç sanal yazmaçların yerine geçer. Yazmaçların yeniden adlandırılması işlemci içinde mimari yazmacı sayısından fazla yazmaç kullanılmasını gerektirir. Örneğin Pentium 3'te 80, Pentium 4 ve Core i7'de 128 fiziksel yazmaç bulunmaktadır.

Yazmaçların yeniden adlandırılması aşamasında, buyruğun çözülmesi sırasında sonucunu yazmak üzere ayırdığı boş yazmacın numarası, daha sonra gelen bağımlı buyrukların değere erişebilmesi için, bir tabloda saklanır. Bu tabloya yeniden adlandırma tablosu denir.

Yeniden adlandırma aşamasından geçen ve sanal yazmaç numaraları gerçek yazmaç numaralarına dönüştürülmüş, çözülmüş buyruklar kaynak verileri hazır olana kadar beklemek üzere bir kuyruğa atılırlar. Buyruklar bu kuyruğa program sırasında atılsa da, bu kuyruktan program sırasının dışında çıkabilirler. Bu kuyruğa yayın kuyruğu denir.

Yayın kuyruğundaki buyruklar, iki ayrı yazmaç kaynağının hazır olup olmadığını, kuyruğa gönderilen tamamlanmış işlem sonuçlarının yazmaç numaralarına bakarak anlarlar. Her bir kaynağın hazır olduğu anlaşıldığında ve uygun bir işlem birimi

bulunduğunda buyruk işlem birimine yayınlanır. Aynı anda yayınlanabilecek hazır buyruk sayısı, yayın kuyruğunun bir vuruşta yayınlayabileceğinden fazlaysa hazır olan buyruklardan bazıları seçilir. İşlem birimlerinde yürütülen buyruklar sonuçlarını yazmaç birimine, ortaya çıkan durum bitlerini ise yeniden sıralama belleğine yazar. Bellek işlemlerinin adres hesapları da işlem birimlerinde yapılır. Şekil 2'de gösterilmemiş olsa da bellek işlemlerinin belleğe sırayla erişmesi için ayrı bir yükleme-saklama kuyruğu (load-store queue ya da memory order buffer) ve sanal adreslerin gerçek adreslere dönüştürülmesi için bir etkin sayfalar önbelleği (translation lookeaside buffer) bulunur. Ayrıca programlardaki if-else, for ve while gibi koşula bağlı yapıların derlenmesi sonucu ortaya çıkan dallanma buyruklarının koşullarının işlemci boru hattının geç aşamalarında hesaplanması nedeniyle, boru hattını dolu tutmak için dallanmaların hangi yöne atlayacaklarını tahmin eden bir öngörü birimi de işlemcinin içinde yer alır. Dallanmanın atlayacağı noktanın yanlış öngörülmesi durumunda işlemci içine yanlış olarak alınan tüm buyrukların işlemciden atılması ve program sayacının düzeltilmesi gerekir.

Özet olarak, çok yollu işlemciler,

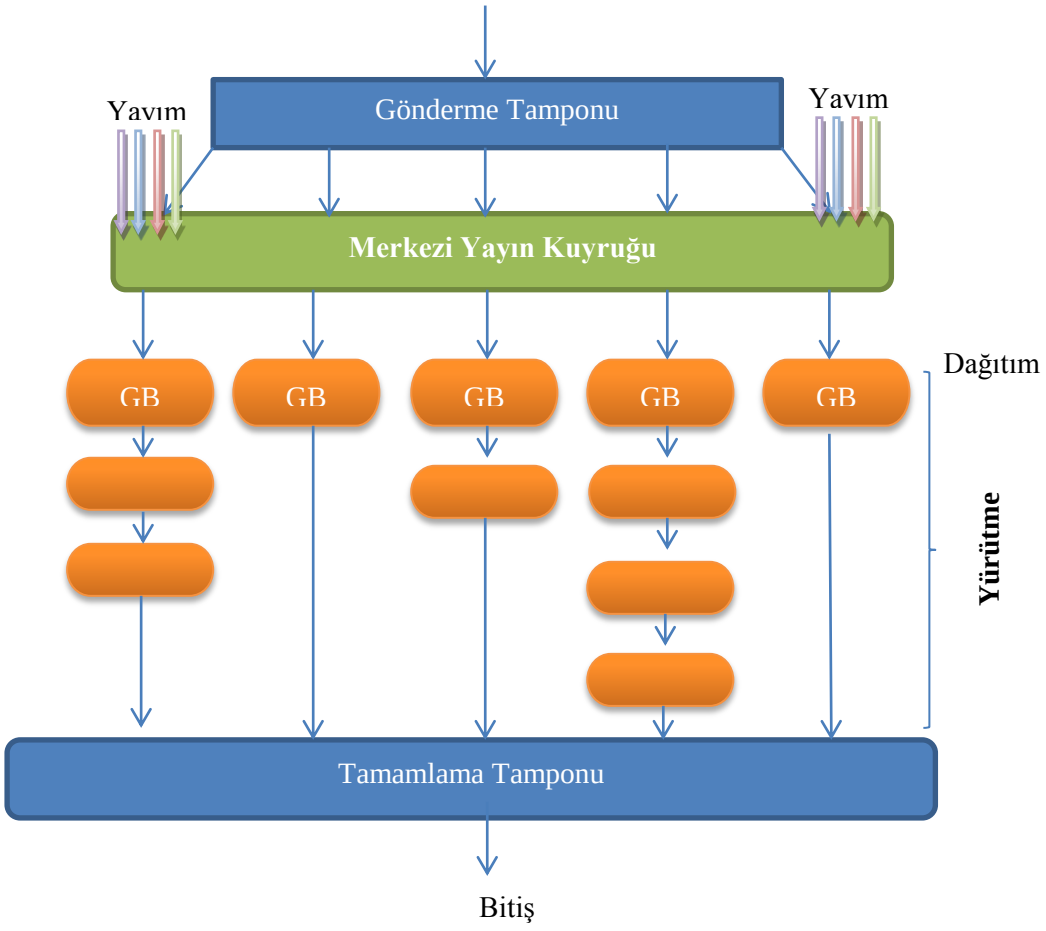
1. Eş zamanlı birden çok buyruk çekebilme, dallanmaların sonuçlarını kestirebilme ve dallanmalardan sonra gelen buyrukları çekebilme,
2. Gerçek veri bağımlılıklarını tespit edip, yürütme sırasında bu değerlere ihtiyaç duyulan yerlere haber verilmesini sağlama,
3. Birden fazla buyruğu paralel bir şekilde başlatabilme ve yayınlayabilme,
4. Birden fazla buyruğun paralel olarak işletilebilmesi için gerekli birden fazla görev birimi yapısı ve birçok bellek referansını eş zamanlı servis edebilecek bellek mekanizması,
5. Değerlerin bellek üzerinden ya da bellek işlemi buyruklarını kullanarak haberleştirmesi sağlayan yapılar ve bellek hiyerarşilerinin dinamik ve tahmin edilemez başarım tavrını yönetebilecek bir bellek arayüzü,
6. İşlem sırasını koruyacak bir şekilde buyrukların sonlandırılmasını sağlama ya da başka bir deyişle sırayla çalıştığı izlenimi verme,

gibi yöntemler ve yapılardan oluşmaktadır.

## 2.3. Yayın Kuyruğu ve Buyruk Dağıtımı

Buyrukların görev birimlerine dağıtılması çok yollu boru hatları için gereklidir. Tek yollu bir boru hattında, buyrukların tiplerine bakılmaksızın buyruklar aynı boru hattından dolayısıyla aynı görev biriminden akarlar.Şekil 3'de görüldüğü üzere, çok yollu işlemcilerde ise boru hattı heterojen ve kendi içinde boru hattı yöntemi uygulanmış olan birçok görev biriminden oluşmaktadır. Farklı tiplerdeki buyruklar farklı görev birimlerinde işlenirler. Çözme aşamasında buyruğun tipi belirlendikten sonra,buyruğun uygun olan görev birimine gönderilmesi gerekir, bu olaya buyruk dağıtımı denir.

Çok yollu bir boru hattında, buyruklar arası bağımlılık çözüldükten sonra tüm görev birimleri birbirinden bağımsız olarak çalışabilirler.



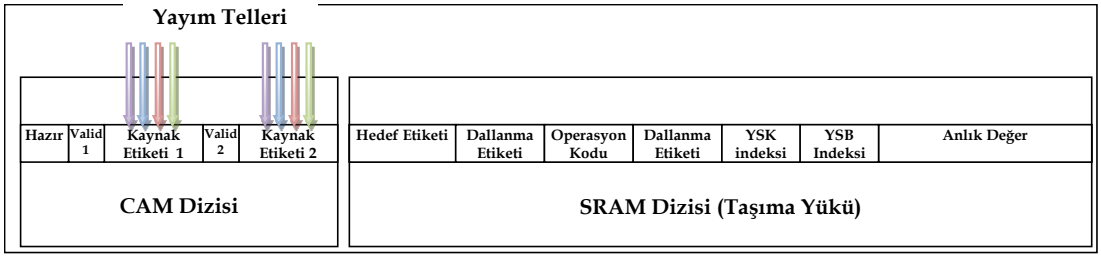

Şekil 3. Buyruk Dağıtım ve Çalıştırma Aşaması

Bu aşamada diğer önemli bir yapı ise, buyrukların çözümleme ve çalıştırılma aşamaları arasında geçici bir süre depolanması için kullanılan yayın kuyruğu yapısıdır. Yayın kuyruğundaki bir buyruğun çalıştırılmadan önce tüm kaynakları hazır olmalıdır. Çözme aşamasında buyrukların yazmaçları yazmaç dosyasından getirilir. Ancak bazen buyruğun bazı yazmaç kaynakları, kendinden önce gelen buyruklar tarafından hedef olarak kullanılabileceğinden hazır olmayabilir. Böyle bir durum oluştuğunda ise, en temel ve basit çözüm, çözümleme safhasının durdurulması ve hazır olmayan yazmaç kaynaklarının beklenmesi şeklinde olabilir. Ancak bekletme kelimesinden anlaşılacağı gibi bu çözüm, başarım üzerinde kötü bir etkiye sebep olacaktır. Bundan daha iyi bir çözüm ise, kaynak yazmaçları hazır olanları getirip, boru hattında ilerlemesine devam ettirip bir tampon alanında saklayarak, hazır olmayanları bekletmektir. Bu tampon alanlara daha önceden de belirtildiği üzere yayın kuyruğu veya diğer bir adıyla rezervasyon istasyonu denilmiştir. Bu yayın kuyruğunun kullanılması bir tasarım prensibi olarak düşünülebilir, ki bu tasarım prensibi buyruk çözümlemenin ve buyruk çalıştırmanın bağlaşımını kesmektir. Çözümleme ve yürütme safhalarında başarım farklarını dengelemekte de görev yapan bir yapı olarak da düşünülebilir. Sonuç olarak böyle bir yapı çözümleme ve yürütme aşamasının bağlaşımını kırarak, çözümleme aşamasının durdurulmasını ve yürütme aşamasının başarımının düşmesini önlemek amacıyla tasarlanmıştır.

Yayın kuyruğu bulunduğu yere bağlı olarak iki farklı isim alabilir. Eğer Şekil 3'deki gibi buyrukların dağıtımının kaynak kısmında tek bir tampon halinde kullanılmışsa,buna merkezi reservasyon istasyonu denir. Diğer taraftan dağıtımın hedef tarafınn da birden çok tampon yerleştrilmişse, buna dağıtık reservasyon istasyonu denir. Bunların çeşitli hibritleri de oluşturulabilir. Bu 2 farklı durumu ödünleşme bakımından incelersek,merkezi reservasyon istasyonu buyruk tipi gözetmeksizin tüm gelen buyruk tiplerini kabul edip , yayın kuyruğundaki tüm girdileri olası en yüksek bir oranda kullanacaktır. Diğer taraftan ise, donanım karmaşası artacaktır. Bu karmaşaya bağlı olarak da güç tüketimi artacaktır. Dağıtık rezervasyon istasyonlarında ise, bu girdilerin kullanım oranı düşük olacaktır ve olası bir şekilde içlerinden bazıları doygun duruma ulaşacaktır. Bu şekilde daha az

donanım karmaşası ve güç tüketimi olacaktır. Burada güç tasarrufunun üzerinde durmak gerekir. Nitekim tezin konusu da yayın kuyruğunda uygulanan iyileştirmeler ile güç tasarrufu sağlamaktır.

## 3. Yayın Kuyruğunda Güç Tüketimi




Şekil 4 Yayın Kuyruğunun İç Yapısı

Yayın kuyruğu buyrukların yürütülmeden önce saklandığı son birimdir. Yapısı Şekil 4'de gösterilmiştir. Çözülmüş buyruklara ait, daha sonra kullanılacak bilgiler SRAM (Static RAM) dizilerinde, asıl bağımlılıkların tespit edilmesi ve kaynak verilerin hazır olduğunun belirlenmesi için kullanılan yazmaç numaraları da CAM (Content Addressable Memory) dizilerinde tutulur. Her bir CAM hücresi saklanan yazmaç numarası ile tamamlanmış verilerin yayın kuyruğuna yönlendirilen yazmaç numaralarını karşılaştırır ve aynı değerin bulunması durumunda "Geçerli" biti birlenir. İki kaynağının da geçerli biti 1 olarak belirlenen buyrukların yürütmeye hazır olduğunun belirlenmesi için "Hazır" biti 1 yapılarak buyruk seçime hazır hale getirilir. Yayın kuyruğunda her vuruşta tüm satırlardaki buyruklar, kaynak etiketlerini, bütün tamamlanmış sonuç yazmacı etiketleriyle karşılaştırır. Çok yollu bir işlemcide, aynı saat vuruşunda birden fazla buyruk bellekten getirildiği ve yürütüldüğü için yayın kuyruğunda birden fazla sayıda yazmaç numarası, her saat vuruşunda yayınlanır. Her buyruğun iki (bazı işlemcilerde üç) kaynağı olması nedeniyle, yayınlanan bir yazmaç numarası için birden fazla yönlendirme yolu gereklidir. Bu durum Şekil 4'deki yayın kuyruğunda farklı renklerdeki oklarla gösterilmiştir. Eğer bir işlemci her saat vuruşunda 4 buyruğu bellekten getiriyorsa ve bu 4 buyruğu işlem birimlerinde yürütüyorsa, iki kaynağı olan buyruklar kullanıldığı durumda aynı vuruşta yayın kuyruğuna 4 farklı yazmaç numarası iki ayrı yoldan

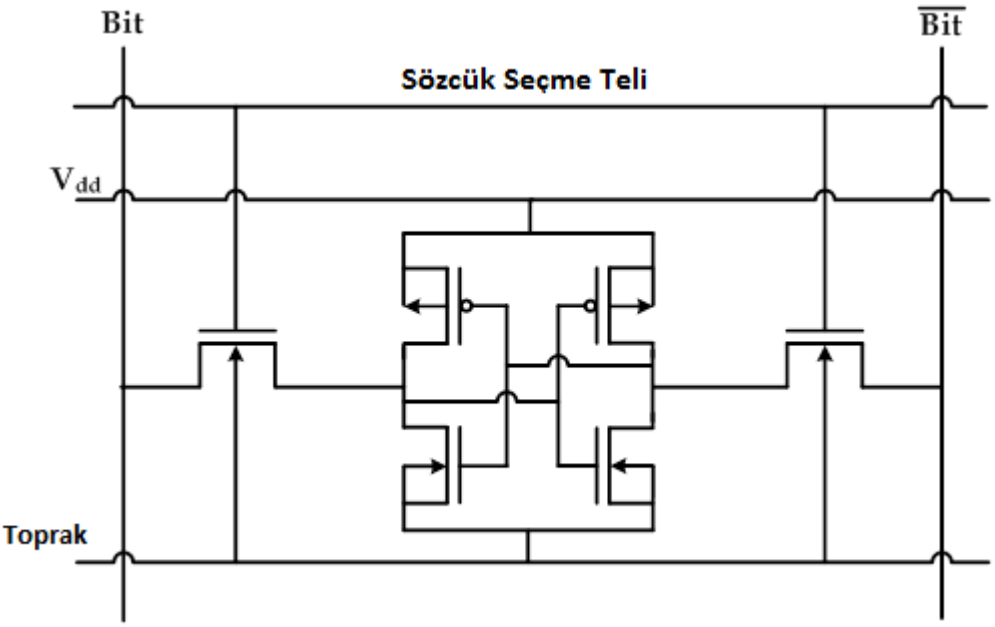
gönderilmelidir. Güncel işlemcilerdeki aynı anda yürütülebilen buyruk sayısı genellikle bellekten getirilen buyruk sayısından fazladır. Örneğin Pentium 4 işlemcisi bellekten her saat vuruşunda 3 buyruk getirmesine karşın aynı anda 6 buyruğu yürütebilmektedir[1]. Bu nedenle yayın kuyruğuna yönlendirilen yazmaç etiketi sayısı her vuruşta bellekten getirilen buyruk sayısından fazla olabilir.

Buyruk getirme oranı ve tüm aşamalarında işlenen buyruk sayısı N buyruk/vuruş olan bir işlemcide yayın kuyruğuna bir vuruşta N buyruğun bilgisi yazılıp N buyruğun bilgisi okunabilir. Bu durumda SRAM dizisindeki her bir hücrenin N yazma ve N okuma kapısı olmak zorundadır. Aynı durum CAM dizisi için de söz konusudur; ancak CAM dizisinde bu kapılara ek olarak hücrelerin her birinin gelen N ayrı etiket bitiyle karşılaştırılması için ayrıca karşılaştırma kapıları gereklidir. Her bir saat vuruşunda gönderilen çok sayıda etiket, yapılan çok sayıda karşılaştırma ve her buyruğun çok miktarda veriyi saklama gereksinimleri nedeniyle yayın kuyruğu işlemcinin en fazla güç tüketilen noktalarından biridir. Pentium 3 ve Core mimarisinde yayın kuyruğu aynı zamanda kaynak verilerini de tutmakta ve daha fazla veri saklama alanına gereksinim duymaktadır. İşlemcinin en çok güç tüketen noktalarından biri olan yayın kuyruğunun en fazla güç tüketilen noktası, çok sayıda karşılaştırma işleminin yapıldığı CAM dizisidir.

### 3.1. SRAM Bit Hücreleri

Şekil 5'de güncel mikroişlemcilerde veri saklamak için kullanılan SRAM bit hücresi gösterilmektedir. SRAM bit hücresi arka arkaya bağlanmış 2 adet eviriciden oluşur. Bu eviriciler birinin çıkışı diğerinin girişine bağlandığı için birbirlerinin çıkış değerini sabit tutacak biçimde davranış gösterirler. Güç kesilmediği sürece içerdikleri değerleri korudukları için bu hücrelere Static RAM hücresi adı verilmiştir. SRAM bit hücreleri değerlerin hem kendisini hem de değilini tutar. Bu nedenle değerde bir değişiklik yapılacağı ya da saklanan değer okunacağı zaman iki taraftan okuma ve yazma işlemi yapılır. Bunun nedeni yazarken birbirini destekleyen eviricilerin değişikliğe olan direncini kırmak, okurken ise hızlı okuma yapmak için iki tel arasındaki farklı gözlemlemektir. Bit ve bit değil telleri SRAM'de aynı

sütunda bulunan bütün bit hücrelerine ortaktır. Benzer şekilde sözcük seçme teli de aynı anda okunacak bütün bit hücreleri için ortak bir giriştir. Bir veri satırı okunurken, önce satırı gösteren sözcük seçme telinin değeri 1 yapılarak bit ve bit değil tellerinin bit hücresine erişimi sağlanır. Eğer yazma işlemi yapılacaksa bit tellerinde bu sırada değer sabit olmalıdır. Okuma işleminde ise bit telleri sözcük seçme teli birlenmeden önce Vdd'ye ya da Vdd/2'ye çekilir. Ön yükleme adı verilen bu işlemin yapılmasının nedeni bit tellerindeki yükü sürmeye küçük bit hücresinin gücünün yetmemesidir. Veri sütunun en altında bit ve bit değil tellerini izleyen analog devreler aradaki farkı anlayarak hızlıca saklanan değeri üretirler.



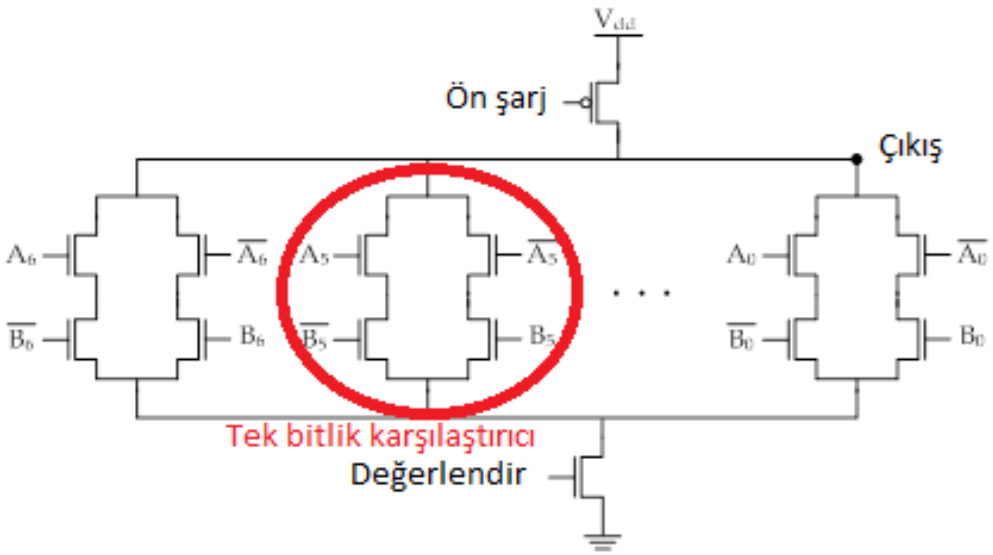

Şekil 5 Tek Kapılı SRAM Bit Hücresi

Mikroişlemcilerin yayın kuyruğunda ve yazmaç birimlerinde olduğu gibi önbelleklerinde de ağırlıklı olarak SRAM bit hücreleri kullanılır.

### 3.2. CAM Bit Hücreleri

CAM bit hücrelerinin saklama özelliği açısından SRAM bit hücrelerinden bir farkı yoktur. Bu hücrelerin tek özelliği fazladan bir karşılaştırma girişi aracılığıyla giren biti saklanan bitle karşılaştırmasıdır. Yayın kuyruğunda tek bir bit değil tüm etiketin

değerinin karşılaştırılması gerektiği için bu bit düzeyinde karşılaştırma sonuçlarının VEYA işlemine sokulması gerekir.




Şekil 6 Yayın kuyruğunda Kullanılan Etiket Karşılaştırıcı Devresi

Sıradan bir karşılaştırıcı Dışlayan VEYA (XOR) kapısı kullanılarak tasarlanabilir. Ancak bu kapılar çok yavaş olduğu için mikroişlemcilerde dinamik mantık devreleri karşılaştırma yapmak için kullanılır. Bu devreler yalnızca aynı veya değil sonucu verir ve aynı olmama durumunda güç tüketirler. Şekil 6'da bu karşılaştırıcı devrelerin çizelgesi gösterilmiştir. Bu devrenin işleyişi sırasında önce çıkış düğümü 1'e çekilir, daha sonra girişlerin değerlerine göre, eğer herhangi bir giriş bit çiftinde farklılık varsa, çıkış düğümü ilgili transistörler aracılığıyla toprağa çekilir. Tek bir bitte bile farklılık olması durumunda çıkış değerinin 0 olması böylece sağlanmış olur. Bu devredeki her bir transistör dörtlüsü ayrı bir CAM hücresinde bulunur, önyükleme ve hesaplama transistörleri ile tüm bit çiftlerinin bağlı olduğu "Aynı teli" (match line) ise veriyi (yayın kuyruğunda yazmaç etiketini) gösteren bütün CAM hücreleri için ortak kullanılır. CAM hücresindeki karşılaştırıcının yapısı gereği hücreye giren karşılaştırılacak değerin hem kendisi hem de değilinin sağlanması için iki tel bulunur.

### 3.3. Devingen Güç Tüketimi

CMOS teknolojisinde sayısal devreleri oluşturan transistörler mükemmel durumda kapılarından akım geçirmedikleri varsayılır. Olağan şartlarda açıp kapanan transistörler, ara düğümleri yükler ve bu yükler toprağa doğru boşalır. Transistörlerin açılıp kapanması sırasında oluşan bu güç tüketimine devingen güç tüketimi denir.
Devingen güç tüketimi büyük ölçüde anahtarlama gücünden oluşmaktadır. Anahtarlamada harcanan gücü modellemek için transistorun kapısının T süre içerisinde ortalama bir $f_{ak}$ frekansta açılıp kapandığını varsayarsak, buradaki yükün $Tf_{ak}$ kadar sayıda şarj ve deşarj olduğunu söyleyebiliriz. Buradan da ortalama gücü aşağıdaki förmülle hesaplayabiliriz.

$$P_{\text{aç-kapa}} = \frac{E}{T} = \frac{Tf_{ak}CV_{DD}^2}{T} = Cf_{sw}V_{DD}^2$$

Bu formülü biraz daha düzenlersek, ve açılıp kapanma sıklığını α olarak, saat sıklığı f olarak yeniden yazarsak formül aşağıdaki gibi olur.

$$P_{aç-kapa} = \propto CfV_{DD}^2$$

Yukarıdaki denklemde de görüldüğü üzere, dinamik güç tüketimini azaltmak için 4 faktörden en az birini azaltmak gerekir. Bunlar $\propto$ açılıp kapanma sıklığını, f saat vuruşu sıklığını, C toplam kapasitansı ya da Vdd yi düşürmektir.Ama saat sıklığını düşürmek performansın düşmesine sebep olacaktır.Vdd'yi yani kaynak gerilimini düşürmek ise transistörlerin hızını da azaltacağı için kaynak gerilimini de çok fazla düşürmek mümkün değildir. Dolayısıyla burada en iyi yapılabilecek şey ya açılıp kapanma sıklığı olan $\propto$ değerini azaltmak ya da toplam kapasitansı azaltmaktır. Bunu başarmanın yolu da kullanılmayan kaynakları kaldırmak ya da kapatmaktır.

### 3.4. Durağan Güç Tüketimi

Transistörler mükemmel anahtarlar değillerdir. Kaynak gerilimi ile gerilim arasına yerleştirilen ve kapalı olduklarında akım geçirmemesi gereken transistörler değişik nedenlerle akım sızdırır. Bu sızan akımın oluşturduğu güç tüketimine ise durağan güç tüketimi denir.

Durağan güç tüketimini etkileyen başlıca etkenler şunlardır:

1. Kaynak Gerilimi
2. Sıcaklık
3. Transistörlerin Eşik Gerilimi

Transistörlerin eşik gerilimi, her yeni teknolojide kaynak gerilimiyle birlikte düşürülür. Burada amaç azaltılan kaynak geriliminden kaynaklanan gecikmenin eşiğin düşürülerek dengelenmesidir. Ancak eşik gerilimi düştükçe, transistörün açılması kolaylaştığı için, sızdırma akımları da artar.

Yüksek çalışma sıcaklıklarında transistörleri oluşturan malzemedeki parçacıkların kinetik enerjisi arttığı için sızdırma akımı da artar. Günümüzde üretilen işlemciler genellikle 80 C'de çalıştıkları için sızdırma akımı oda sıcaklığına göre çok daha yüksektir. Sızdırma akımının oluşturduğu ısı sıcaklığı artırdığı ve bu sıcaklık da sızdırma akımını artırdığı için, ortaya çıkan destekleyen geri besleme (positive feedback) önlem alınmaması durumunda işlemcinin yanmasına kadar gidebilecek bir ısınma yaratabilir.

Eşik gerilimi üretim teknolojisi tarafından belirlenen bir değişkendir. Ancak transistörlerin taban gerilimine de bağlıdır. Aradaki bu bağıntı aşağıdaki denklemlerle gösterilir:

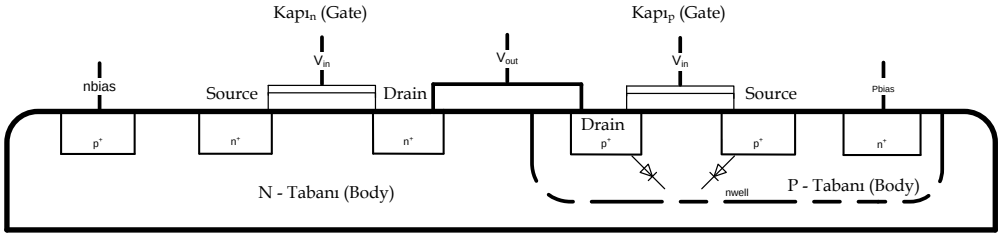
$$V_T = V_{T_0} + \gamma(\sqrt{2\theta_F + |V_{SB}|} - \sqrt{2\theta_F}) \qquad (V_{T_0} \Leftarrow (V_{SB} = 0))$$

$$[\theta_F = \frac{kT}{q}\ln(\frac{N_0}{n_i}),\ \gamma = \frac{t_{ox}}{\varepsilon_{ox}}\sqrt{2q\varepsilon_{si}N_A} \rightarrow\rightarrow 0.4 < \gamma < 1.2\ ]$$

Yukarıdaki bu denklemler karışık görünse de, pek çoğu aslında üretimle ilgilidir. Sıcaklık dışında, üretimden sonra değiştirilebilen tek değişken transitörün kaynağı ve

tabanı arasındaki gerilim farkını gösteren VSB'dir. Denklemlerden de görüldüğü gibi taban geriliminin azaltılması (ya da p türü MOS transistörler için artırılması) eşik geriliminin artmasına neden olur. Şekil 7'de transistörlerin yapısı malzemenin yandan kesiti alınarak gösterilmiştir. Transistörler aslında diyotların birleşiminden oluştuğu için transistörlerin oturduğu tabanın doğru kutuplanması çalışma biçimlerini etkiler.

Şekil 5'de SRAM tasarımında görüldüğü gibi, olağan yürütüm sırasında n türü transistörlerin tabanı toprağa, p türü transistörlerin tabanı ise kaynak gerilimine bağlanır.

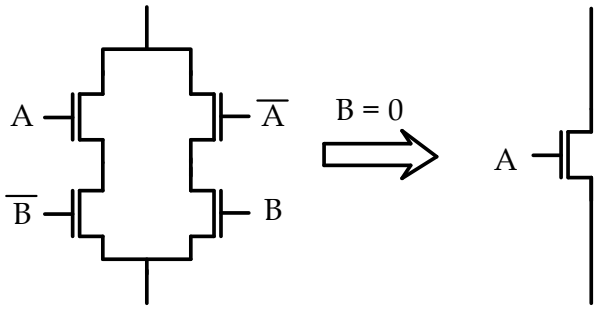



Şekil 7 Transistörlerin Yapısı

## 4.Etiket Sadeleştirme: Yayın kuyruğunda Devingen Gücün Azaltılması

Bizim önerdiğimiz ise yayın kuyruğunun devingen güç tüketiminin azaltılması sağlayan bir yöntemdir. Şekil 6 'da görülen karşılaştırıcı devrede her bir bit için aslında dışlayan veya (XOR) işlemi yapılmaktadır. Örneğin 1-bitlik A ve B değerlerinin karşılaştırılması sırasında B değerinin 0 olduğu bilinirse, C = A'B + AB' ifadesi sadece A ya eşit olacaktır.

Böylece, Şekil 6'te görülen her bir birlikte karşılaştırma öbeğindeki 4 transistörden 3'ü kaldırılabilir. Şekil 8'de görüldüğü gibi, B girişinin 0 olması, sağdaki yolu keseceğinden o yoldaki 2 transistör kaldırılabilir. Benzer biçimde B'nin değili bu durumda hep 1 olacağından solda B'nin değilinin bağlı olduğu transistör de hep geçireceği için kaldırılabilir. Böylece devre okun sağında görülen tek transistörlü hale dönüşür. Şekil 8'de görülen dönüşümün gerçekleştirilmesi A'nın değilini bütün yayın kuyruğuna gönderen sürücü devrenin üzerindeki yük ve toplam kapasitans azaltarak devingen güç tüketimini azaltacaktır.




Şekil 8 Karşılaştırıcı devrenin sadeleştirilmesi

Örneğin, Şekil 8'de B' girişinin CAM hücresinin içeriğine bağlı olduğu düşünüldüğünde, bütün yayın kuyruğundaki etiketler üzerinde bu işlem yapılabilse, A'nın değilini gönderen tel tamamen kaldırılır.

Sonuç olarak yayın kuyruğundaki CAM kısmındaki bulunan karşılaştırıcılar da duruma göre bu şekilde sadeleştirilip güç tasarrufu elde edilebilir. Bu amaçla biz yayın kuyruğunu uygun bir şekilde bölerek, bölünen kısımların bazılarında bu optimizasyonu yaparak bir çok transistörden kurtulmak yoluyla güç tasarrufu elde

etmeyi umduk. Bunun için de M-sim adı verilen bir işlemci benzetimcisi kullanarak bazı analizler ve deneyler yaptık.

### 4.1 M-Sim ve Denektaşı Programları

Önerilen yeniden adlandırma yapısının test edilmesi için M-Sim benzetimcisi ile denektaşı programlar kullanılarak deneyler yapıldı. M-Sim ise SimpleScalar benzetimcisi üzerine geliştirilmiş, Alpha işlemcilerinin buyruk kümesini çok yollu olarak çalıştırabilen bir benzetimcidir. M-Sim kullanımı kolay bir benzetimci olup, daha kısa sürede tutarlı sonuçlar verebilmektedir.

Sistemin denenmesi için Spec CPU 2006 denektaşı program grubu kullanılmıştır. Spec CPU 2006, bir endüstri standardı olup işlemciyi yoğun olarak kullanıp bir sistemin işlemcisini, bellek altsistemlerini ve derleyicisini zorlayarak test eder. Bu program grubu işlemcilerin başarımını ölçmek için çeşitli tam sayı ve kayan nokta denektaşı programları içerir. Bu denektaşı programlarının isimleri ve açıklamaları Tablo 2 ve Tablo 3'de verilmiştir.

Tablo 2 Spec 2006 Tam Sayı Denektaşı Programları

| Program | Yazıldığı Dil | Uygulama Alanı | Açıklama |
|---|---|---|---|
| perlbench | C | Programlama Dili | Perl V5.8.7 den türetilmiş bir program olup, çalışması bir e-posta indeksleyicisi ve gereksiz posta ayıklayıcısının yüklerini içerir. |
| bzip2 | C | Sıkıştırma | Bzip2 sıkıştırma algoritmasının, G/Ç dan çok bellekte çalışması sağlanmış halidir. |
| gcc | C | C derleyicisi | GCC 3.2 versiyonu temel alınarak geliştrilmiş ve Opteron için kod çıkaran bir denektaşı programıdur. |
| mcf | C | Kombinasyonel Optimizasyon | Araç zamanlama, bir ağ simpleks algoritması kullanarak toplu taşıma araçlarının zamanlamasını yapar. |

| | | | |
|---|---|---|---|
| gobmk | C | Yapay Zeka: Go Oyunu | Çok karmaşık olan Go oyununu oynar. |
| hmmer | C | Gen Sırası Bulma | Saklı Markov modellerini kullanarak protein sırası analizi yapar. |
| sjeng | C | Yapay Zeka: Santraç | Santraç oynayan yüksek seviyeli bir program. |
| libquantum | C | Fizik / Kuantum Hesaplama | Bir kuantum bilgisayarının benzetimini, Shor' un polinom zamanda çalışan faktorizasyon algoritmasını kullanarak yapar. |
| h264ref | C | Video Sıkıştırma | H264/AVC nin referans bir gerçeklemesidir. Bir videoyu 2 parametre kullanarak kodlar. |
| omnetpp | C++ | Ayrık olay simulasyonu | OMNet++ kullanarak yüksek ölçekli kampüs ağındaki ayrık olayları modeller. |
| astar | C++ | Yol bulma algoritması | 2 boyulu haritalar için yol bulma kütüphanesidir. |
| xalancbmk | C | XML işleme | XML dokümanlarını başka formattaki dokümanlara dönüştürür. |

Tablo 3 Spec 2006 Kayan Nokta Denektaşı Programları

| Program | Yazıldığı Dil | Uygulama Alanı | Açıklama |
|---|---|---|---|
| bwaves | Fortran | Sıvı Dinamiği | 3 boyutlu ses ötesi kısa süreli ince tabakalı viskozlu akım hesaplaması yapan program |
| gamess | Fortran | Kuantum Kimyası | Çok çeşitli kuantum kimyası hesaplamaları içeren bir programdır. |
| milc | C | Fizik/Kuantum kromodinamiği | Dinamik kuarklar ile ayar alanları üreten bir programdır. |
| gromacs | C,Fortran | Biyokimya / Moleküler dinamik | Yüz milyonlarca partikül için newton un hareket denklemlerini benzeştiren bir programıdır. |

| | | | |
|---|---|---|---|
| zeusmp | Fortran | Fizik | Illinois Üniversitesi,hesaplamalı astrofizik bölümünde geliştirilmiş olan sıvı dinamiği kodudur. |
| cactusADM | C,Fortran | Fizik/Genel Relativite | Çapraz-atlamalı nümerik bir yöntem kullanarak Einstein ‘ın evrim denklemlerini çözen bir programdır. |
| leslie3d | Fortran | Sıvı Dinamiği | 3 boyutta Lineer-Eddy modellerini kullanarak sıvı dinamiği hesaplamaları yapan bir programdır. |
| namd | C++ | Biyoloji/Moleküler Dinamik | Geniş ölçekli biyomoleküler sistemleri benzeştiren bir programdır. |
| dealII | C++ | Sonlu element analizi | Adaptif sonlu element analizi ve hata kestirimi yapan bir kütüphanedir. Test sırasında Helmholtz tipinde bir denklemi sabit olmayan katsayılarla çözmeye çalışır. |
| soplex | C++ | Doğrusal Programlama/ Optimizasyon | Doğrusal bir problemi simpleks algoritması ve aralıklı doğrusal cebir kullanarak çözen bir benzetim programıdır. |
| povray | C++ | Resim Işın İzleme | 1280x1024 boyutlarında olan çeşitli dokulara sahip objeler içeren bir araziyi, Perlin gürültü fonksiyonu kullanarak işleyen bir program. |
| calculix | C,Fortran | Yapısal Mekanik | Doğrusal veya doğrusal olmayan 3 boyutlu yapısal uygulamalarda kullanılan bir programdır. |
| GemsFDTD | Fortran | Hesapsal Elektromanyetizma | Maxwell denklemlerini 3 boyutta sonlu farklar zaman alanlı yöntemi kullanarak çözen bir program. |
| Tonto | Fortran | Kuantum Kimyası | Deneysel röntgen kırınım verileri daha iyi tutturabilmek için moleküler Hartree-Fock dalga fonksiyonuna kısıtlar koyarak hesaplama yapan bir program. |
| lbm | C | Sıvı Dinamiği | 3 boyutta sıkıştırılamaz sıvıları benzeştirmek için Lattice-Boltzman yöntemini kullanan bir program |

| | | | |
|---|---|---|---|
| wrf | C,Fortran | Hava durumu | Metrelerden binlerce kilometrelik alana kadar hava durumu modellemesi yapan bir program. |
| sphinx3 | C | Konuşma Tanıma | Konuşma tanımlaması yapan bir program. |

### 4.2 M-Sim Benzetim Parametreleri

M-Sim benzetimcisi ile yapılan deneylerde benzetimcinin benzettikleri makinelerin parametreleri aşağıdaki Tablo 4.4 tabloda verilmiştir. Bu tabloya baktığımızda yayın kuyruğunun 32 satırdan oluştuğunu ve 128 tane fiziksel yazmaç içerdiğini görebiliz. Şekil 3.1 de görüldüğü gibi satır başına 7 karşılaştırıcı olmak üzere toplamda 32 x 7 = 224 tane karşılaştırıcı gerektiği anlamına gelmektedir. Bizim motivasyonumuz ise burada olabildiğince sadeleştirme yaparak karşılaştırıcılardaki transistör saysını azaltıp yayın kuyruğunda güç tasarrufu elde etmektir.

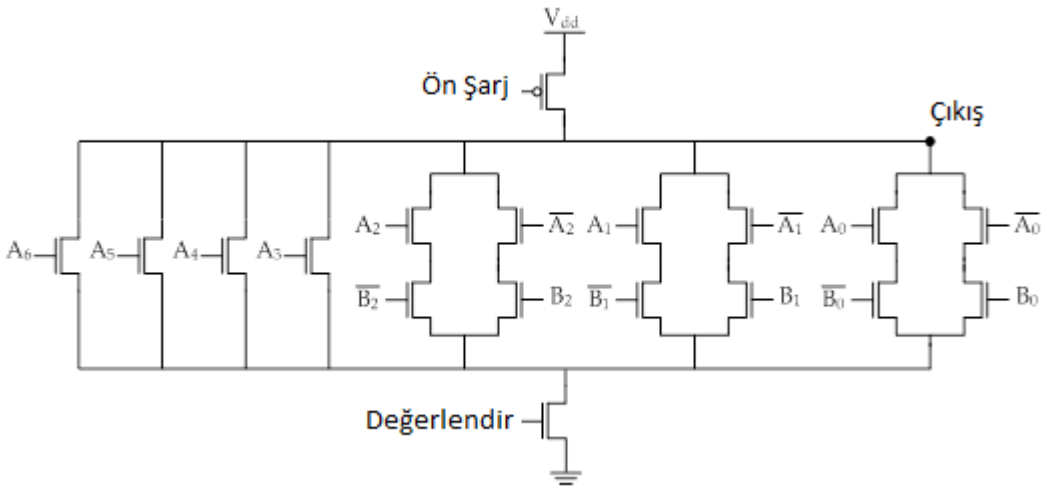
Tablo 4 M-Sim Benzetim Parametreleri

| **Parametre** | **Yapılandırma** |
|---|---|
| Makine Genişliği | 4 buyruk getir,4 buyruk yayınla,4 buyruk bitir |
| Pencere Boyu | 32 satır yayın kuyruğu<br>48 satır yükle kuyruğu<br>32 satır sakla kuyruğu<br>128 satır YSB |
| İşlem Birimleri (Sayılar) | Tamsayı AMB (4)<br>Tamsayı Çarpma/Bölme (1)<br>Bellek Kapıları (2)<br>Kayan Nokta AMB (4)<br>Kayan Nokta Çarpma/Bölme (1) |
| Fiziksel Yazmaçlar | 128 tamsayı yazmaç öbeği,128 kayan nokta yazmaç öbeği |
| L1 Veri Önbelleği | 256 kümeli,64 byte blok genişliği,4 yollu, 1 çevrim isabet zamanı |

| | |
|---|---|
| L1 Buyruk Önbelleği | 512 kümeli,64 byte blok genişliği,2 yollu,1 çevrim isabet zamanı |
| L2 Tümleşik Önbellek | 512 kümeli, 64 byteblok genişliği,16 yollu, 10 cycle çevrim isabet zamanı |
| Dallanma Öngörücüsü | Çift durumlu Tahmin Edici,2048 tablo girdisi |
| Dallanma Hedef Tamponu | 512 küme, 4 yollu |
| Buyruk etkin sayfalar önbelleği | 16 kümeli,4KB blok genişliği,4 yollu, 30 çevrim kaçırma gecikme süresi |
| Veri etkin sayfalar Önbelleği | 32 kümeli,4KB blok genişliği,4yollu, 30 çevrim kaçırma gecikme süresi |

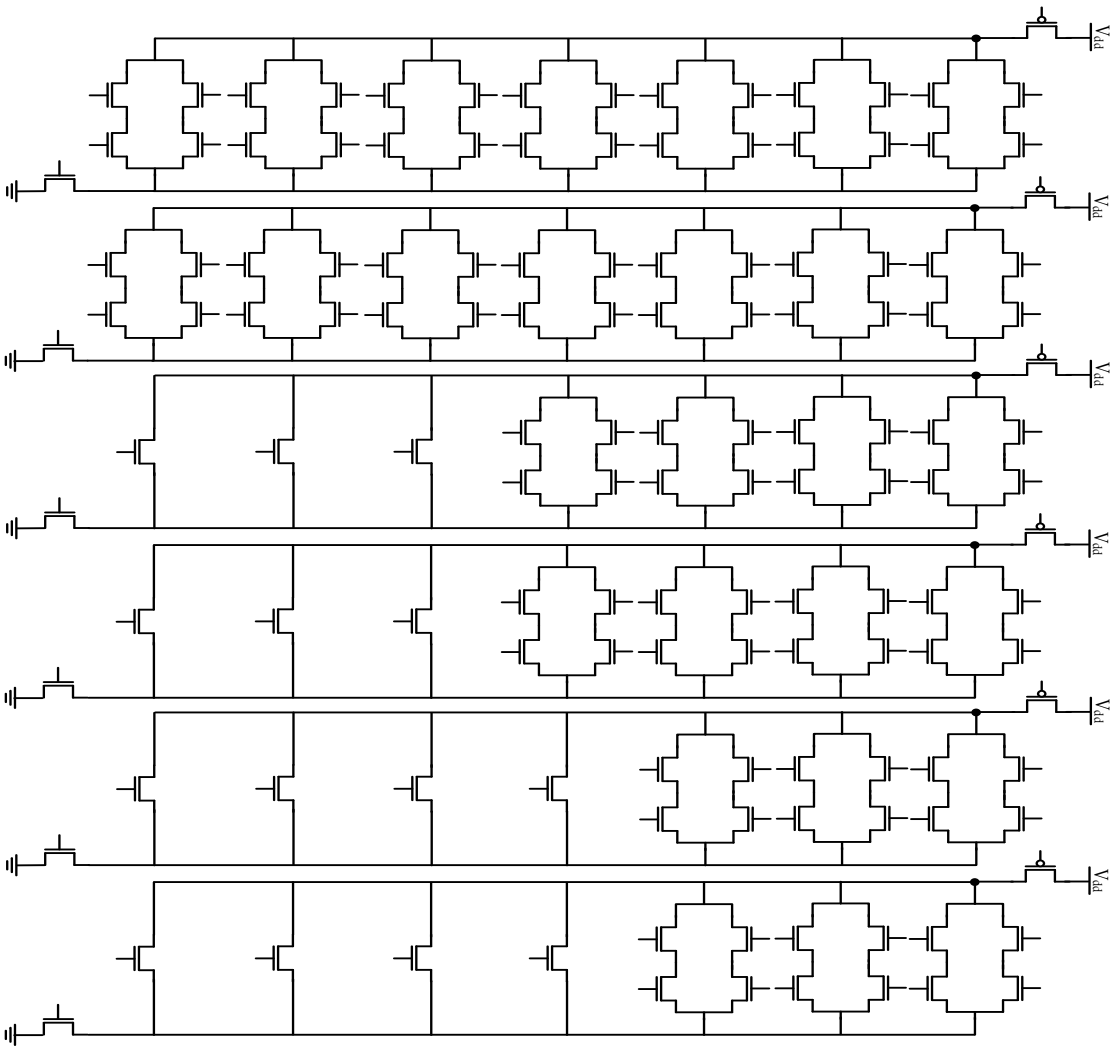
### 4.3 Yayın Kuyruğunun Bölünmesi

Yayın kuyruğunda saklanan etiket değerleri büyük oranda değişkenlik gösterir. Yazmaçların yeniden adlandırılması yöntemi nedeniyle mimari düzeyindeki yazmaçlar tamamen rastgele yazmaçlarla eşleştirilebilir. Bu atama işlemi rastgele olduğu için yayın kuyruğuna giren buyrukların kaynak etiketlerinin üst bitlerinde belli oranda 0 taşıyacakları açıktır.




Şekil 9 Bir satırda 3 tam, 4 sadeleştirilmiş karşılaştırıcı

Eğer her iki kaynak etiketi de saklama alanından daha az bitle ifade edilebilen bir buyruk yayın kuyruğuna gelirse bu buyruğun etiket karşılaştırıcılarında Şekil'de gösterilen dönüşümden yararlanılabilir. Örneğin, 128 fiziksel yazmacı olan bir işlemciye dışarıdan gelen (Alpha gösteriminde) add r1, r2, r3 buyruğu yeniden adlandırmadan sonra add p2, p5, p7 buyruğuna dönüşürse, kaynak yazmaçlarının (p5 ve p7) 7 bitlik yazmaç etiketlerinin üstteki 4 biti tamamen sıfır olacaktır (0000101 ve 0000111). Böyle bir durumda, Şekil 9'deki gibi 4 bitlik sadeleştirilmiş ve 3 tam bitlik karşılaştırıcı kullanarak yapılan karşılaştırma bizim için yeterli ve doğru olacaktır.




Şekil 10 Etiket Sadeleştirmenin Uygulaması

Böylece, görüldüğü üzere gelen buyruklar, yazmaç etiketlerine göre gruplandırılıp yayın kuyruğundaki bazı karşılaştırıcı devreler sadeleştirilecektir. Ortaya çıkacak buyruk gruplarının 3 bitlik, 4 bitlik ve 7 bitlik etiketlere sahip olacağı varsayılırsa yayın kuyruğu kabaca Şekil 10'da görüldüğü hale dönüşür. Böylece yönlendirme yollarındaki bazı bitlerin üzerindeki yükün azalacağı ve devingen güç tüketiminde önemli oranda azalış sağlanabileceği öngörülmektedir.

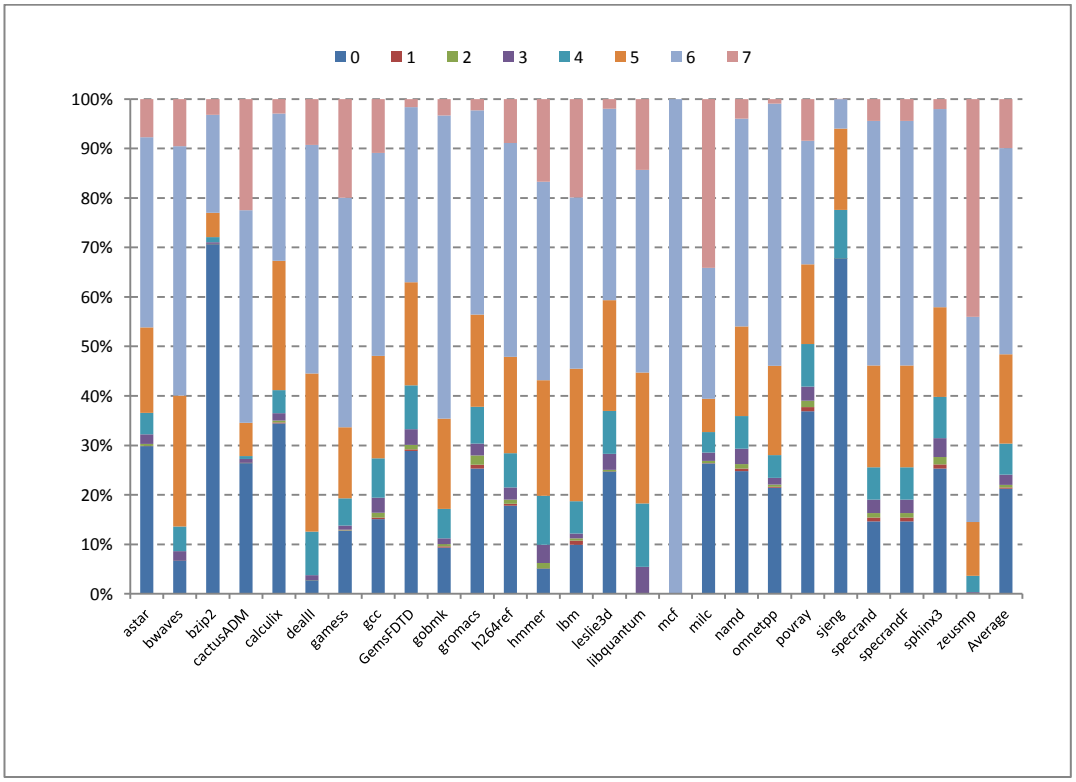
Böyle bir yöntemin başarıya ulaşması için işlemcide ortaya çıkacak yazmaç etiketlerinin ne şekilde gruplanabileceği önemlidir. Tamamen rastgele bir dağılım varsayılsa bile, yayın kuyruğundaki tellerdeki yükün azaltılmasının sağlanacağı kesindir. Burada işlemcinin ön kısmında gruplama yapılırken harcanacak enerjinin az olması önerilen yöntemin etkin olamasını sağlamaktadır. İşlemcinin ön kısmında yazmaç atamasını işlemci kendisi yaptığı için buyruğun hangi yayın kuyruğu bölmesine gideceği de buyruğa bilgi olarak eklenebilir. Diğer bir seçenek ise yayın kuyruğuna atma aşamasına karşılaştırıcı devreler eklemektir. Her ne kadar yayın kuyruğundan kaldırılan karşılaştırıcı devrelerin ön kısımda yeniden kullanıldığı ve bunun enerji tasarrufunu ortadan kaldırdığı akla gelse de önerdiğimiz yöntemin sağladığı başlıca 2 avantaj vardır:

1. Ön tarafa yalnızca karşılaştırıcı eklenmektedir. Yayın kuyruğunda ise bütün yapı boyunca uzanan tellerin üzerindeki yük kaldırılmaktadır. Karşılaştırma için harcanacak güç tüketimi buyruk başına 1 keredir, ancak yayın kuyruğunda yapılacak tasarruf süreklidir.

2. İşlemcide güç tüketiminin yarattığı asıl sorun tüketimin yoğunlaşmasıdır. Soğutma yetenekleri belirli bir dereceye kadar soğutmak üzere kurulur, harcanan toplam gücün burada önemi dolaylıdır. Eğer sıcaklık tek bir noktaya yoğunlaşırsa, genel işlemci sıcaklığı belirlenen dereceyi geçmese de noktanın sıcaklığı geçebilir. Bu nedenle ısıyı yaratan etkinliğin işlemcinin farklı birimlerine dağıtılması yararlıdır.

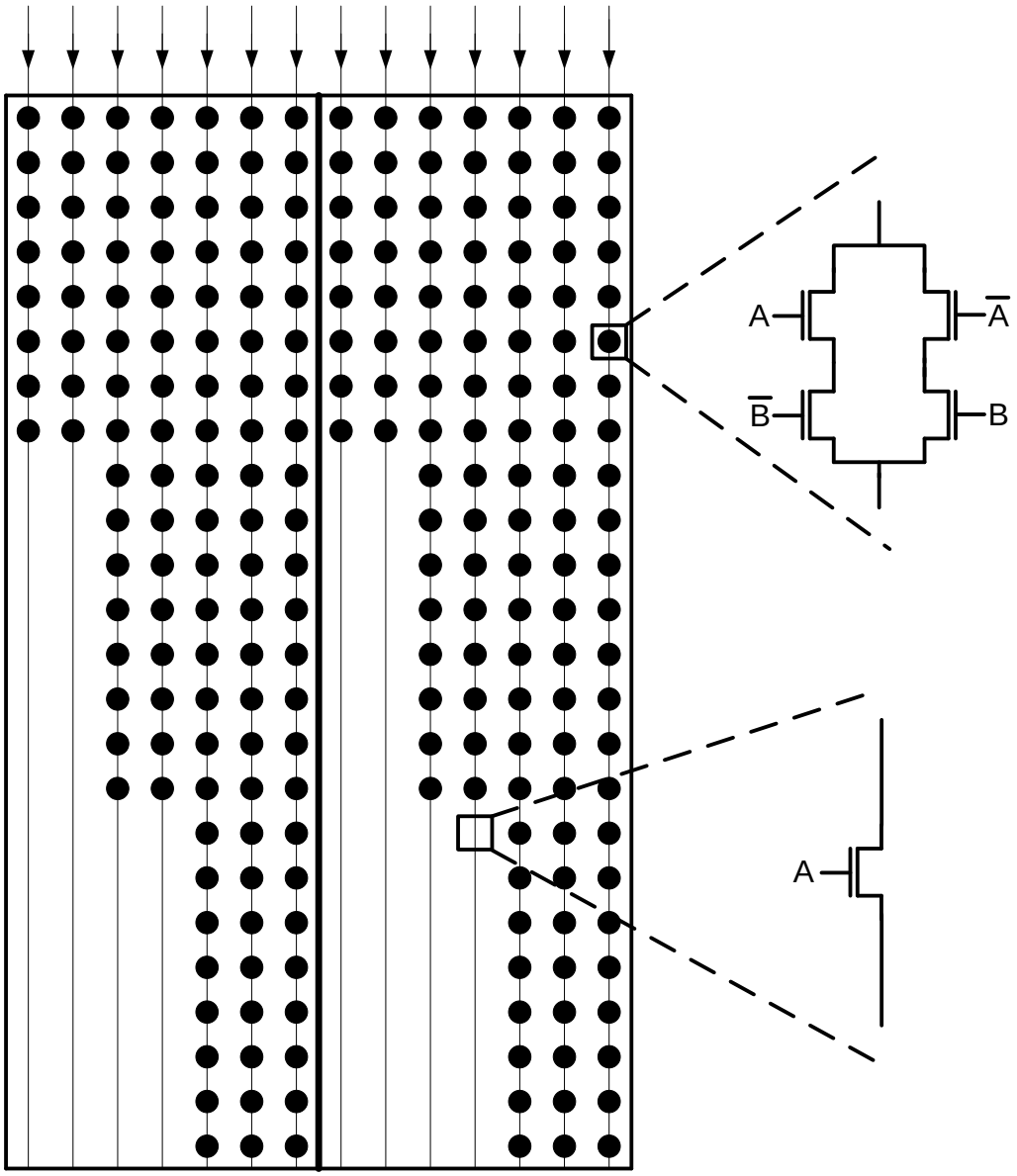
Yayın kuyruğu birden fazla bileşene ayrılacaktır. Kuyruğun kendisi fiziksel olarak Şekil 10'da gösterildiği gibi tek parça halinde kalsa da, etiketi dar olan satırlara ancak etiketleri o satırın gösterebildiği boyutta olanlar yada daha küçük boyutta olanlar

girebilecektir. Bu mantıksal olarak yayın kuyruğunu çok parçaya ayırmak demektir. Ortaya çıkabilecek parçaların sayısı en fazla 8 olacaktır (tek bitle gösterilebilen 0 numaralı yazmaç için tüm bitler sadeleşeceği için). Tez kapsamında kaç adet kuyruk kullanılacağı ve bu kuyrukların boyunun ne olması gerektiği araştırılacaktır. Etiketleri dar olan bir buyruk, etiketleri geniş olan bölmelere girebileceği ancak bunun tersi mümkün olmayacağı için geniş etiketli buyrukların çokça geldiği anlarda başarım düşüşü olabilir. Bu başarım düşüşü yayın kuyruğunda uygun yer olmaması ve işlemcinin uygun yer olana kadar beklemesinden kaynaklanır. Bu başarım düşüşünü en azda tutarken güç tasarrufunun en yüksek olduğu tasarım noktasını bulmak için bazı deneyler yaptık. Bu deneylere başlamadan önce hangi etiketlerin hangi sıklıkta geldiğini hesaplamamız gerekecektir. Bu istatiksel bilgi ise Şekil 11'de gösterilmiştir.




Şekil 11 Etiket genişliklerinin oranları

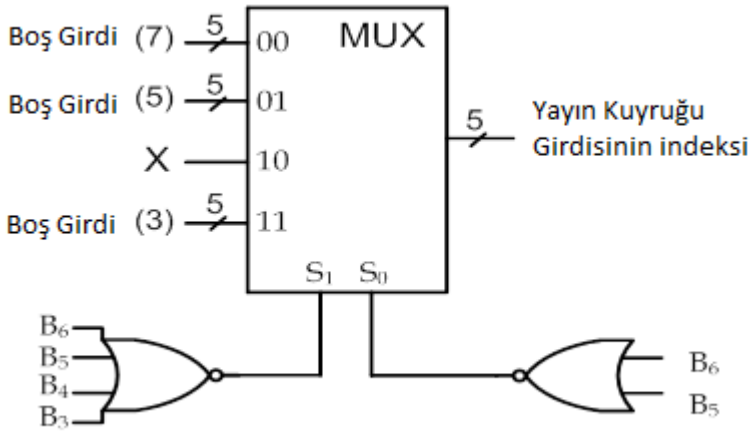
Şekil 11'da yayın kuyruğundaki buyrukların kaynak yazmaçlarının etiketlerinin genişliklerinin dağılımı gösterilmektedir. Burada da görüldüğü üzere etiket genişliklerinin ortalama değerlerine bakıldığında yayın kuyruğunu 3,5,7 olarak bölmeye karar verdik. Yayın kuyruğundaki bölme işlemi yapıldıktan sonra, yayın kuyruğu Şekil 12'de olduğu gibi gözükecektir. Burada not edilmesi gereken bir durum ise, bir buyruğun 2 tane kaynak yazmacı olduğunda, bölünmüş olan kuyruklardan birine girerken uygulanacak taksim yöntemi, kaynak yazmaçlarından etiketi büyük olanın uzunluğu ile belirlenecektir




Şekil 12 Bölünmüş Yayın Kuyruğı

### 4.4 Karmaşanın İşlemcinin Ön Tarafına Aktarılması

Yayın kuyruğu birkaç parçaya bölündükten sonra, her bir parça farklı bir karşılaştırıcı yapısına sahip olacaktır. Burada görüldüğü üzere bir yayın kuyruğu yapılandırmasını oluşturan 3 faktör vardır. Bunlar, yayın kuyruğunun kaç farklı parçaya bölündüğü,bu parçaların herbirindeki sadeleştirilmiş karşılaştırıcı sayısı veya diğer bir deyişle başında sıfır olan bitlerin sayısı ve bu parçaların boyutlarıdır. Nitekim bizim yapılandırmamız, yayın kuyruğunu 3 parçaya bölüp, bu parçalardaki başında sıfır olan bitlerin sayısını da 3,5 ve 7 olarak belirlemiştir. Burada hangi buyruğun hangi kuyruğa gireceğini belirlemek için, ekstradan bir çözme devresi gerekecektir. Bu devre bir 1 karşılaştırıcı ve bir tane çözücüden oluşacaktır. Karşılaştırıcı en fazla 2 tane olabilecek olan kaynak yazmaçlarının numaralarından en büyüğünü seçmek için kullanılır. Çözücü ise bu en büyük yazmaç numarasının hangi kuyruğa gideceğini belirlemek için kullanılır. Genel olarak çözücünün yapısı Şekil 13'de görülebilir.




Şekil 13 Çözücü devresinin içeriği

## 5. Deneyler ve Sonuçlar

Daha önce de bahsettiğimiz gibi herhangi bir yapılandırmayı tanımlayabilmek için 3 tane kriteri belirlememiz gerekiyor idi. Bunlar, yayın kuyruğunun kaça bölüneceği, her bir kuyruktaki başında sıfır olan bit sayısı ve son olarak da bu kuyrukların derinlikleri idi. Şu ana kadar kaça bölüneceğine ve bu kuyrukların kaç bitlik kuyruklar olacağına karar verip bu durum için gereken tüm hazırlıkları yapmış bulunmaktayız. Şimdi ise deney yapılandırması tamamlamak amacıyla bu parçalanmış kuyrukların derinliklerini değiştirerek ÇBB( Çevrim Başına Buyruk) üzerindeki etkilerini ölçüp elde edilen güç tasaarufu ile karşılaştırmasını yapacağız. M-sim de benzetilen işlemci önceden de belirtildiği üzere 32 adet yayın kuyruğu girdisine sahip idi. Bu durumda oluşan 3 kuyruğun derinlikleri toplamı 32 olmak zorundadır. Tabiki bunu yapmadan önce bir referans çalışması da yaptık. Kuyruğu bölmeden bir benzetim sonucunda elde ettiğimiz verileri de dikkate aldık. Tablo 5'de gösterilen yapılandırmaları teker teker denedik.

Tablo 5 Etiket sadeleştirme için çeşitli yapılandırmalar

| **Yapılandırmalar** | ***Kuyruk Derinlikleri*** | | |
|---|---|---|---|
| | ***Kuyruk - 7*** | ***Kuyruk - 5*** | ***Kuyruk - 3*** |
| 1 | 20 | 9 | 3 |
| 2 | 16 | 10 | 6 |
| 3 | 13 | 13 | 6 |
| 4 | 13 | 10 | 9 |
| 5 | 13 | 6 | 13 |
| 6 | 12 | 11 | 9 |
| 7 | 9 | 10 | 13 |
| 8 | 7 | 13 | 12 |
| 9 | 6 | 16 | 10 |
| 10 | 5 | 10 | 17 |

## 5.1. Etiket Sadeleştirme Yönteminin Sonuçları

Burada görüldüğü üzere, 10 farklı yapılandırma denedik, ve sonuç olarak da Şekil 14'de olan grafiği elde ettik.


Şekil 14 Enerji tasarrufuna karşı çbb grafiği

Yayın kuyruğunu 3 e bölmek işlemcinin ön tarafındaki karmaşayı çok az bir şekilde artırmıştır. İşlemcinin ön tarafına eklenecek olan çözücü, 1 MUX ve 2 tane extra NOR kapısından meydana gelmiştir. Yapılan öçümlerde bu donanımın 43.1fj gibi bir ekstra enerji tüketimi meydana getirdiği görülmüştür. Şekil 14'de görüldüğü üzere ÇBB ile Güç Tasarrufu arasındaki ilişkiyi dikkate alarak, deneylerimizin içerisinde yayın kuyruğunun CAM kısmında %15 e varan bir güç tasarrufunu fazla bir performans kaybı olmadan elde edebileceğimizi gördük. Tabi diğer yandan da ÇBB kaybını göze alarak daha fazla güç tasarrufu elde etmek de mümkündür.

Sonuç olarak, bu tezde amacımız yayın kuyruğunda etiket sadeleştirmesi yaparak ÇBB yi düşürmeden güç kaybını önlemek idi. Nitekim de yayın kuyruğunu değişik sayıda parçalara bölüp, bu parçalardaki karşılaştırıcı bit uzunlukları ile ayarladıktan

sonra, karmaşanın bir kısmını işlemcinin ön tarafına atarak, yayın kuyruğunun CAM kısmında %20'lik bir kazanım sağlayabileceğimizi gösterdik. Ancak böyle bir kazanım ÇBB nin biraz düşmesi koşuluyla gerçekleşecektir. Tabiki %15 lik bir güç tasarrufunun nerdeyse performansta hiç bir değişiklik olmadan mümkün olduğunu da tespit ettik.

Yayın kuyruğu bir işlemci içerisindeki en sıcak noktalardan bir tanesidir, burada elde edilecek aktivite düşüşü , bir işlemcinin termal olarak da rahatlamasına sebep olacaktır. Nitekim işlemcinin çalışması ile oluşan ısının işlemcinin farklı taraflarına eşit olarak dağıtılması ile ısının atılması daha kolay olacaktır.

### 5.2. Etiket Sadeleştirmenin Geliştrilmesi

Etiket sadeleştirmede buyrukların gruplara ayrılmasının yaratabileceği başarım düşüşünü azaltmak için yayın kuyruğuna giren kaynak yazmaçlarının hazır olup olmadığına bakılabilir. Etiketleri 7 bitlik olsa bile, yayın kuyruğuna girerken değerlerin yazmaç biriminde hazır olması durumunda, buyruklar en dar parçaya atanabilir. Bu geliştirme sayesinde başarım düşüşünün sınırlı tutulacağı ve sadeleşmiş karşılaştırıcılara sahip yayın kuyruğu satırlarının daha fazla kullanılarak elde edilecek güç tasarrufunun artırılacağı öngörülmektedir. Benzer biçimde hiç kaynak etiketi kullanmayan ya da yalnızca tek kaynak etiketi kullanan buyrukların bu özellikleri kullanarak getirileri artırılabilir.Yazmaç ataması işleminde bu kuyrukların daha verimli parçalara gidebilmelerini sağlayacak yazmaç atama yöntemleri araştırılabilir.

### 5.3. İlişkili Çalışmalar

Literatürde yayın kuyruğunun CAM kısmının güç tüketimi düşürmek için birçok araştırma yapılmıştır. Araştırmacılar güç tüketimini düşürmek için bir çok öneride bulunmuştur. Kimi araştırmacılar normal karşılaştırıcılar yerine eşit olduğunda güç tüketen karşılaştırıcı kullanımı öne sürmüştür[12]. Yayın kuyruğundaki port sayılarının azaltılması ile güç tüketiminin düşürülmesi [13]'de öne sürülmüştür.[16] da ise birden fazla buyruğa tek bir buyruk gibi davranıp uyandırma mantık

devresinde daha az güç harcama öne sürülmüştür. [17]'deki çalışmada ise, bir program tarafından oluşturulan yazmaç değerlerinin en fazla 1 defa okunduğu ortaya konmuştur, buna binaen yazmaç numarasına dayalı bir tablo vasıtasıyla okuma işleminin yapılmasının ve uyandırma devresindeki aşırı karşılaştırma işleminin önüne geçilmesi hedeflenmiştir.[18]'de ise kaynak yazmacı olmayan veya zaten kaynak yazmaçları hazır olan buyruklar için uyandırma devresini devre dışı bırakmak önerilmiştir. [11]'de ise yayın kuyruğuna giren buyrukların en az bir tane kaynak yazmacının hazır olduğu görülmüştür, bu da yayın kuyruğunda kaynak yazmacı numaralarının olduğu kısımların en az bir tanesinin boşu boşuna enerji harcadığı manasına gelir ki bunları iptal etmek için yayın kuyruğu kaynak yazmaçlarının etiketlerine bağlı olarak 0 etiketi olanlar, 1 etiketi olanlar ve 2 etiketi olanlara bölünmüştür. Nitekim etkili bir şekilde güç tasarrufu sağlanmıştır.[21] de ise sonuç üreten buyrukların sonuçlarının en fazla 1 tane buyruk tarafından kullanıldığı görülmüş, üretici buyrukların kendi sonuçlarını kullanan harcayıcı buyrukların yayın kuyruğundaki yerini hatırlamalarını öne sürülerek güç tasarrufu elde edilmiştir. Kimi araştırmacılar ise yayın kuyruğu içindeki aktiviteyi ölçerek, boyutunu kontrol altında tutmaya çalışmışlardır[4][23]. Diğerlerine göre daha yeni bir çalışmada ise yayın kuyruğunu anlık değerlerin genişliğine göre bölüp, anlık değerleri başka bir tabloda tutulmak süretiyle güç kazanımı sağlanmaya çalışılmıştır[19].

## KAYNAKLAR


[1] D. Levitan, T. Thomas, P. Tu, "The PowerPC 620 microprocessor: a high performance superscalar RISC microprocessor," compcon, pp.285, 40th IEEE Computer Society International Conference (COMPCON'95), 1995

[2] G. Hinton et al., "A 0.18-um CMOS IA-32 processor with a 4-GHz integer execution unit" IEEE Journal of Solid-State Circuits, vol. 36, no. 11, November 2001

[3] R. Kalla,B. Sinheroy, J. M. Tendler, "IBM Power5 chip: a dual-core multithreaded processor," IEEE Micro, vol. 24, issue 2, March/April 2004

[4] A. Buyuktosunoglu et al., "Adaptive issue queue for reduced power at high Performance," IBM Research Report RC 21874, Yorktown Heights, New York, Nov. 2000.

[5] D. V. Ponomarev, G. Kucuk, O. Ergin, K. Ghose, P.M. Kogge "Energy-efficient issue queue design," IEEE Transactions on VLSI systems, vol. 11, issue 5, November 2003

[6] J.J. Sharkey, D. V. Ponomarev, K. Ghose, "M-SIM: A flexible, multithreaded architectural simulation environment," Tech Report CS-TR-05-DP01, Dept. of C.S., State Univ of New York at Binghamton, Oct 2005. http://www.cs.binghamton.edu/~jsharke/m-sim/

[7] S. Palacharla, N. P. Jouppi, J. E. Smith, "Complexity-Effective supersclarar processors," ISCA 97 Proceedings of the 24th annual international symposium on computer architecture, vol. 25, issue 2, May 1997

[8] M.K. Gowan, L.L. Biro, D.B. Jackson, "Power considerations in the design of the Alpha 21264 microprocessor," Design Automation Conference, pp. 726 – 731, June 1998

[9] K. Wilcox, S. Manne "Alpha processors: A history of power issues and a look to the future", Proceedings of the CooIChips tutorial. An Industrial Perspective on Low Power Processor Design in conjunction MICRO-33, 1999

[10] J. Stark, M.D. Brown, Y.P. Patt, "On pipelining dynamic scheduling logic," MICRO 33 proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture, ACM New York , NY, USA 2000

[11] D. Ernst, T. Austin, "Efficient dynamic scheduling through tag elimination," In Proc. of the 29th Int'l Symp. on Computer Architecture, pp. 37–46, May 2002.

[12] O. Ergin, K. Ghose, G. Kucuk, and D. Ponomarev, "A circuit-level implementation of fast, energy-efficient comparators for high-performance microprocessors," in Proc. Int. Conf. Computer Design (ICCD), 2002, pp. 118–121.

[13] I. Kim and M. H. Lipasti, "Half-price architecture", in Proc. Of 30th International Symposium on Computer Architecture, 2003.

[14] J.J. Sharkey, D.V. Ponomarev, K. Ghose, and O. Ergin, "Instruction packing: reducing power and delay of the dynamic scheduling logic," in ISLPED, pp. 30-35, 2005

[15] J.J. Sharkey, D. Ponomarev, K. Ghose, O. Ergin, "Reducing delay and power consumption of the wakeup logic through instruction packing and tag memoization," in Proc. of the 4th Workshop on Power-Aware Computer Systems, 2004.

[16] H. Sasaki, M. Kondo, H. Nakamura, "Energy-efficient dynamic instruction scheduling logic through instruction grouping," In Proc. ISLPED, pp. 43–48, Oct. 2006.

[17] R. Canal, A. Gonzalez, "Reducing the complexity of the issue logic," proceedings of the 14th international conference on supercomputing, pp. 327 – 335, May 2000.

[18] D. Folegnani, A. Gonzalez, "Energy-effective issue logic," In Proceedings of the 28th Annual International Symposium on Computer Architecture, pp. 230 - 239, July 2001.

[19]I.C. Kaynak, Y.O. Kocberber, O. Ergin, "Reducing the energy dissipation of the issue queue by exploiting narrow immediate operands," Journal of Circuits, Systems and Computers (JCSC), vol. 19, issue 8, pp. 1689 – 1709, December 2010.

[20]J. Leenstraet. al., “A 1.8 GHz instruction window buffer for an out-of-order microprocessor core,” IEEE Journal of Solid-State Circuits, vol. 36, issue 11, pp. 1628-1635, August 2002.

[21]M. Huang, J. Renau and J. Torrellas, “Energy-Efficient hybrid wakeup logic,” in proceedings of Intl. Symposium on Low-Power Electronics Design, 2002.

[22]D. Brooks, V. Tiwari, M. Martonosi, “Wattch: a framework for architectural-level power analysis and optimizations”, Proceedings of the 27th annual international symposium on Computer architecture, p.83-94, June 2000, Vancouver, British Columbia

[23]D. V. Ponomarev, G. Kucuk, K. Ghose, “Dynamic Resizing of Superscalar Datapath Components for Energy Efficiency”, IEEE Trans. Computers 55(2): 199-213 (2006)

# ÖZGEÇMİŞ

**Kişisel Bilgiler**


| | |
|---|---|
| Soyadı, Adı | : BAYRAKTAR, Vehbi Eşref |
| Uyruğu | : T.C. |
| Doğum Tarihi ve Yeri | : 29.03.1985 |
| Medeni Hali | : Bekâr |
| Telefon | : 0 (505) 7588542 |
| Faks | : 0 (312) 292 4290 |
| E-Posta | : vebayraktar@etu.edu.tr |


**Eğitim**

| Derece | Eğitim Birimi | Mezuniyet Tarihi |
|---|---|---|
| Y. Lisans | TOBB ETÜ Bilgisayar Mühendisliği | 2011 (beklenen) |
| Lisans | Marmara Bilgisayar Mühendisliği | 2008 |

**İş Deneyimi**

| Yıl | Yer | Görev |
|---|---|---|
| 2009 – 2011 | TOBB ETÜ | Araştırma Asistanlığı |

**Yabancı Dil**

İngilizce (ileri seviye)

**Yayınlar**