TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DONANIM TABANLI DRAM OPERASYON HIZLANDIRICI TASARIMI

YÜKSEK LİSANS TEZİ

Eyüphan İPEK

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Doç. Dr. Oğuz ERGİN

Nisan 2017

Fen Bilimleri Enstitüsü Onayı

……..
Prof. Dr. Osman EROĞUL
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksininlerini sağladığını onaylarım.

……
Doç. Dr. Oğuz ERGİN
Anabilimdalı Başkanı V.

TOBB ETÜ, Fen Bilimleri Enstitüsü'nün 131111002 numaralı Yüksek Lisans Öğrencisi **Eyüphan İPEK**'in ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"DONANIM TABANLI DRAM OPERASYON HIZLANDIRICI TASARIMI"** başlıklı tezi 10.04.2017 tarihinde aşağıda imzaları olan jüri tarafından kabul edilmiştir.

Tez Danışmanı : **Doç. Dr. Oğuz ERGİN** .....
TOBB Ekonomi ve Teknoloji Üniversitesi

Jüri Üyeleri : **Doç. Dr. Özcan ÖZTÜRK (Başkan)** .....
İhsan Doğramacı Bilkent Üniversitesi

Doç. Dr. Ali BOZBEY .....
TOBB Ekonomi ve Teknoloji Üniversitesi

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

Eyüphan İPEK

ÖZET

Yüksek Lisans Tezi

DONANIM TABANLI DRAM OPERASYON HIZLANDIRICI TASARIMI

Eyüphan İPEK

TOBB Ekonomi ve Teknoloji Üniveritesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Oğuz ERGİN

Tarih: Nisan 2017

DRAM bellek işlem süreleri için kullanılan toplam süre, bellek içeren sistemlerin başarımında önemli bir etkiye sahiptir. Bu tez çalışmasında "Yedek Dize" adı verilen yeni bir yöntem sunulmaktadır. Yedek dize yöntemi, alt dizeye erişimin zamansal yerellik özelliği gösterdiği gözlemi üzerine kurulmuştur. Bu gözlemden yararlanılarak, erişilecek dizenin bir kopyasının yedek bir dizede daha tutulmasıyla DRAM işlem sürelerinin azalabileceği fikri geliştirilmiştir.
Aynı alt dizede yer alan ve aynı veriyi tutan DRAM hücrelerinin eş zamanlı aktif hale gelmesi, bellek işlemlerinin daha hızlı gerçekleşmesini sağlayacaktır. Bu yöntem, başta algılama ve geri yükleme olmak üzere erişim sürelerini azaltacaktır. Bu çalışmada, her bir alt dizeye fazladan bir (yedek) dize eklenmesi ve bu yedek dizeye zamansal yerellik açısından kullanılabilir dizelerin verilerinin saklanması önerilmektedir. Geliştirilen denetim mekanizması, yakın gelecekte erişilebilir dize verilerini yedek dizeye kopyalamaktadır. Yedek dizede tutulan veri ile hedef dize verisi aynı olduğu durumda, yedek dize mekanizması hedef ve kopyalanmış dizeyi aynı anda aktif hale getirerek DRAM erişiminin düşük gecikmelerle tamamlanmasını sağlamaktadır. Eğer saklanan dize ile hedef dize birbirlerinden farklı ise sadece hedef dize aktif hale gelmektedir ve mekanizma yakın gelecekte kullanılabilir olarak o an işlenen dizeyi yedek dizeye kopyalar. Tez çalışması süresince geniş bir test verisi ile yedek dize yöntemi geliştirilmiştir. Ortalama DRAM erişim gecikmelerinin azaldığı ve sistem başarımının daha iyi bir duruma ulaştığı gözlenmiştir.

Anahtar Kelimeler: Yedek dize, DRAM gecikmesi, Alt dize, Erişim süresi, Başarım, Hedef dize, Bellek işlem süresi.

ABSTRACT

Master of Science

HARDWARE BASED ACCELERATOR DESIGN FOR DRAM OPERATIONS

Eyüphan İPEK

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Computer Engineering Science Programme

Supervisor: Doç. Dr. Oğuz ERGİN

Date: April 2017

Overall execution time on DRAM has vital impact on memory included system's performance. In this thesis, a new method which is called "SpareRow" is proposed. Spare row method is built on the key observation that high temporal locality exists among the rows of each subarray. By this observation, the idea is developed on copying the repeatedly accessed row on spare row reduces DRAM execution time. Simultaneously enabling the DRAM cells which are on same subarray and hold same values reduces the access latency on memory operations. This method reduces operation times especially on sensing and precharging. It is purposed that to add an extra (spare) row to each subarray to selectively store a duplicate of one of the rows of the subarray according to the temporal locality. Control mechanism copies the rows which will be potentially used in short term. In case the additional row stores the data of the row to be accessed, SpareRow enables both the target row and its duplicate at the same time to complete the DRAM access with low latency. If the spare row is different from the accessed row, target row is activated and mechanism copies that row on spare row. SpareRow is evaluated on a large set of workloads. It is observed that it significantly reduces average DRAM access latency and thus improves overall system performance.

Keywords: Spare row, Dram latency, Subarray, Access latency, Target row, Execution time.

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Oğuz Ergin'e, kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine, eğitimim süresince burs sağlayan TOBB Ekonomi ve Teknoloji Üniversitesi'ne, destekleriyle her zaman yanımda olan aileme, arkadaşlarıma ve bu tezde çok büyük katkısı olan arkadaşım Hasan Hassan'a, bu süreçte gösterdiği destek ve sabırdan dolayı, her zaman yanımda olan biricik eşim Candan Tuğçe İpek'e ve tatlı kızım Efsun İpek'e çok teşekkür ederim.

İÇİNDEKİLER

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

KISALTMALAR

RAM	: Random Access Memory
ROM	: Read Only Memory
PROM	: Programmable ROM
EPROM	: Erasable Programmable ROM
EEPROM	: Electrically Erasable ROM
DRAM	: Dinamik RAM
SRAM	: Statik RAM
DDR	: Double Data Rate
MCR	: Multiple Clone Row
LSB	: Least Significant Bit
MSB	: Most Significant Bit
TL-DRAM	: Tiered-Latency DRAM
AL-DRAM	: Adaptive-Latency DRAM
SALP	: Subarray-Level Parallelism
IO	: Input/Output
ORG	: Giriş-Çıkış Pin Sayısı
ACT	: ACTIVATE
PRE	: PRECHARGE
REF	: REFRESH
FRFCFS	: First Ready, First Come First Served
FCFS	: First Come First Served
IPC	: Instruction Per Cycle

SEMBOL LİSTESİ

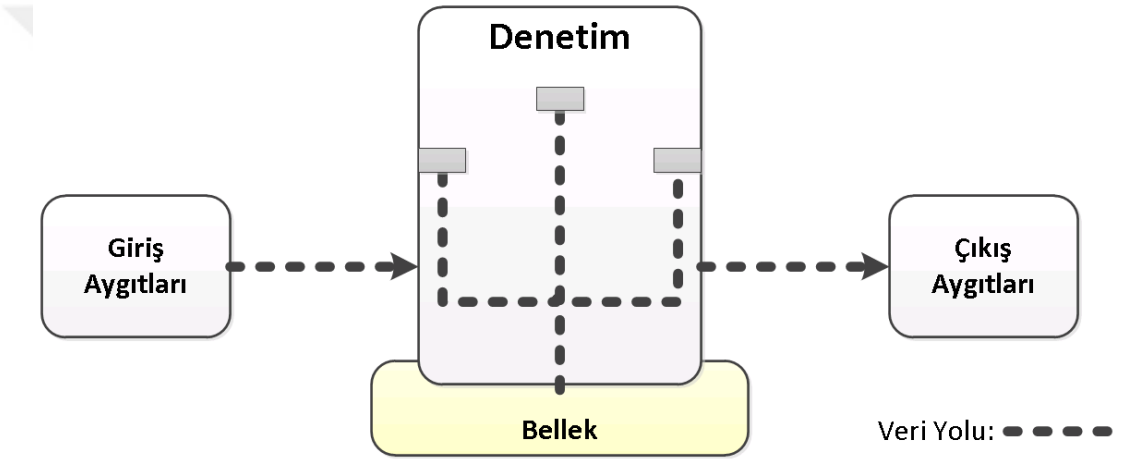
Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	Açıklama
T	Transistör
K	Kapasitör
Vdd	Hattı besleyen gerilim değeri
C_i	i'ninci indeksli kapasitör
A_i	i'ninci indeksli adres biti
C_S	Yedek hücre kapasitörü
C_C	Hedef hücre kapasitörü
C_b	Bit kattı kapasitörü
Q_i	i'ninci indeksli yük miktarı
V_i	i'ninci indeksli gerilim değeri
ms	mili saniye
Gb	Giga bit
KB	Kilo bayt
C_{bit}	Bit kattı kapasitörü
C_{cell}	Hedef hücre kapasitörü
M_S	Yedek hücre transistörü
M_C	Hedef hücre transistörü
ΔV	Birim gerilim değeri

1. GİRİŞ

1.1 Bilgisayarın Ana Bileşenleri ve Bellek

Bilgisayar donanım mimarisini oluşturan 5 ana bileşen vardır; giriş aygıtları, çıkış aygıtları, denetim, veri yolu ve bellek. Denetim ve veri yolunu işlemci olarak ortak adlandırmak da mümkündür[1]. Şekil 1-1'de bilgisayar mimarisinin 5 ana bileşeni gösterilmiştir.



Şekil 1-1: Bilgisayar mimarisinin ana bileşenleri.

Klavye ve fare gibi bilgisayar mimarisine dış dünyadan buyruk alınmasını sağlayan birimlere giriş aygıtları denir. Bilgisayar içerisinde gerçekleşen işlemlerin çıktılarını kullanıcıya aktaran aygıtlara ise (ekran, yazıcı vs.) çıkış aygıtları adı verilir.

Denetim, işlemcinin koşutlarını alan, çözen, ihtiyacı doğrultusunda veri yolu, bellek ve giriş/çıkış aygıtlarını denetim eden bilgisayar birimidir. Veri yolu aracılığı ile buyruk kümesinden alınan buyruklar denetim birimi tarafından işlenir. Saklı veriyi bellekten alarak kullanmak veya işlenen veriyi bellekte saklamak denetim biriminin temel görevlerindendir.

İşlenen verinin kalıcılığı veya saklanabilir olması, ihtiyaç durumunda tekrar tekrar kullanılabilmesi bilgisayar sistemlerinin en önemli özelliklerinden birisidir. Buyrukların sıralı işlenebilmesi için buyruk kümesinin ve sırasının tutulduğu bir

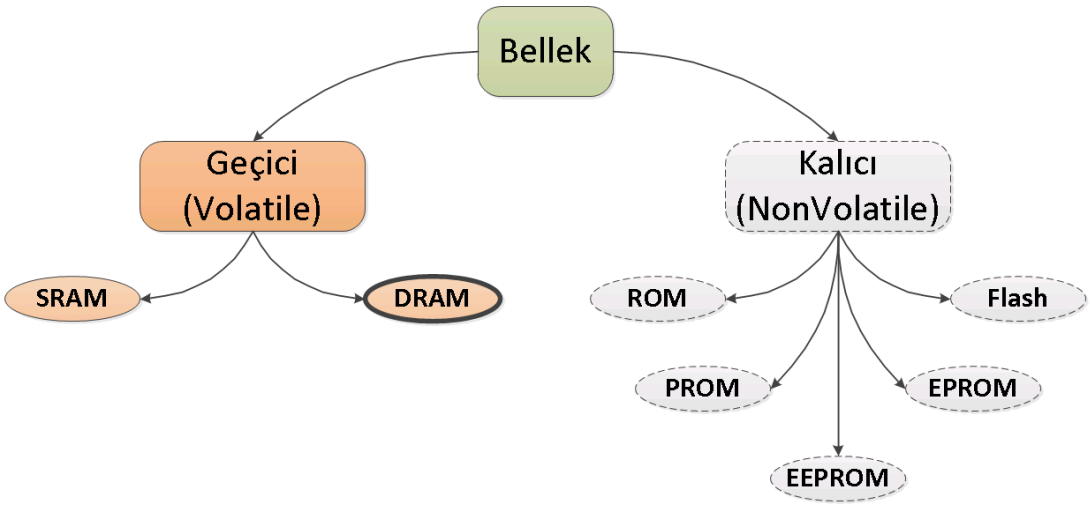
alana ihtiyaç duyulmaktadır. Aksi takdirde buyrukların sıralı işlenmesi mümkün olmayacaktır. Ayrıca buyruk işlemleri sonucunda elde edilen verinin bir sonraki saat çevrimlerinde de kullanılması ihtimaliyle saklanması mühimdir. Bellek, veri saklama ve verinin tekrar tekrar kullanılmasını sağlama özelliğine sahip olduğu için bu 5 ana bileşen içerisinde en önemli olanlardan birisidir.

Tarihsel sürecine bakıldığında ilk olarak 1947 yılında Manchester Üniversitesi çatısı altında "Williams-Kilburn Tüpü" adıyla yüksek hızlı elektronik bileşenli bellek kullanılmıştır. Elektronik veriyi saklamak için katot ışın tüpü mevcuttur[2], elektriksel yük değişiminin anlaşılabildiği metal bir pikap levha aracılığıyla tüpte var olan verinin okunması sağlanmıştır. Elektronik cihazların veri saklamak için uygun ve önemli bir platform olduğu bu ve benzeri ürünlerle anlaşılmıştır. Teknolojinin gelişmesiyle bellek mekanizmaları bugünkü seviyelerine ulaşmıştır. Güncel uygulamalarda “Williams-Killburn Tüpü”ne oranla daha büyük verilerin saklanması sağlanmıştır. Hemen hemen her sistem içerisinde bellek birimleri bulunmaktadır. Bellek mimarileri üzerinde geliştirilen yeni donanım ve yazılım tasarımları akademik alanda güncel konular arasındadır [3].

1.2 Belleklerin Sınıflandırılması

Bellekleri farklı özelliklere göre sınıflandırmak mümkündür. Genellikle güç ile ilişkilerine ve yarı iletken özelliklerine göre sınıflandırılırlar. Belleklerin bazıları güç olduğu sürece içerisindeki veriyi muhafaza ederler, bazıları ise güçten bağımsız şekilde veriyi saklarlar. Güç kesilmesiyle veri kaybı yaşayan belleklere "Geçici" (Volatile), kaybetmeyenlere "Kalıcı" (NonVolatile) denir. Şekil 1-2’de güç durumuna ve yarı iletken yapısına göre sınıflandırılan bellek ağacı verilmiştir. Geçici bellekler RAM olarak isimlendirdiğimiz "Rastgele Erişimli Bellek"dir. Kalıcı olanlar ise ROM, PROM, EPROM, EEPROM ve Flash belleklerdir[4].

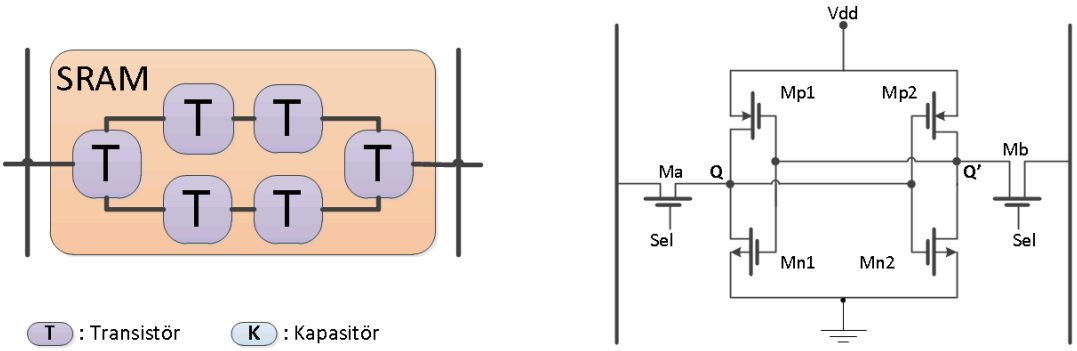
Bu tezde sunulan "Yedek Dize" yöntemi bir RAM tipi olan DRAM yongalarına özeldir. Bu kapsamda çalışma, geçici bellekler üzerinde yapılmıştır. Geçici bellekler sınıfı hem "Statik Rastgele Erişilir Bellek" (SRAM) hem de "Dinamik Rastgele Erişilir Bellek" (DRAM) tipi RAM'leri kapsamaktadır. Bu sınıflandırma RAM'lerin yarı iletken özelliğine bakılarak yapılmıştır.



Şekil 1-2: Belleklerin sınıflandırılması.

1.2.1 Sram

SRAM, yarı iletken özelliklerine göre 4 ile 6 transistör içeren hücrelere sahiptir, yapısında kapasitör yoktur. Bu nedenle veri hücreleri yük sızıntısı riski taşımamaktadır ve saklanan verinin yenilenmesine gerek yoktur. Şekil 1-3'de 6 transistörlü SRAM belleğine ait hücre gösterilmiştir[1,4].



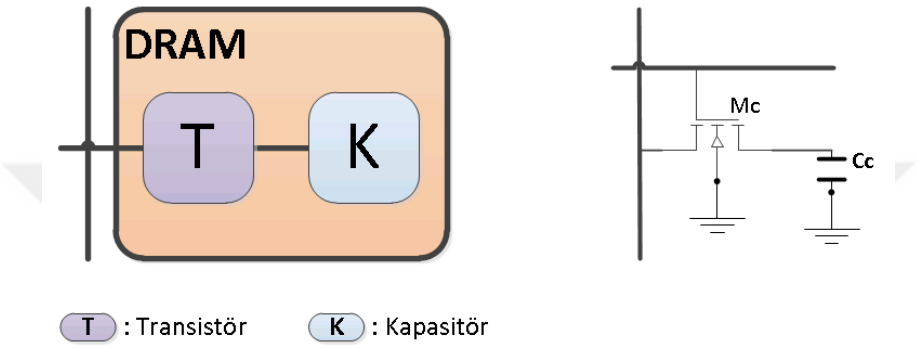
Şekil 1-3: (a) Sembolik sram hücresi ve (b) sram hücresi devresel gösterimi.

SRAM'ler DRAM'lere göre daha hızlıdır. Bu nedenle hızlı erişilmek istenen veriler işlemci mimarisinde SRAM'lerde tutulur. Genellikle seviye 1-2-3 adlarıyla ön bellek olarak kullanırlar. Şekil 1-5'de farklı seviyeli belleklere ait kapasite ve bu veriye erişim hızı sembolize edilmiştir. İşlemciye yaklaştıkca ön bellek erişim süresi (n1<n2<n3) ve doğru orantılı olarak tutabileceği veri boyutu azalmaktadır[1,4].

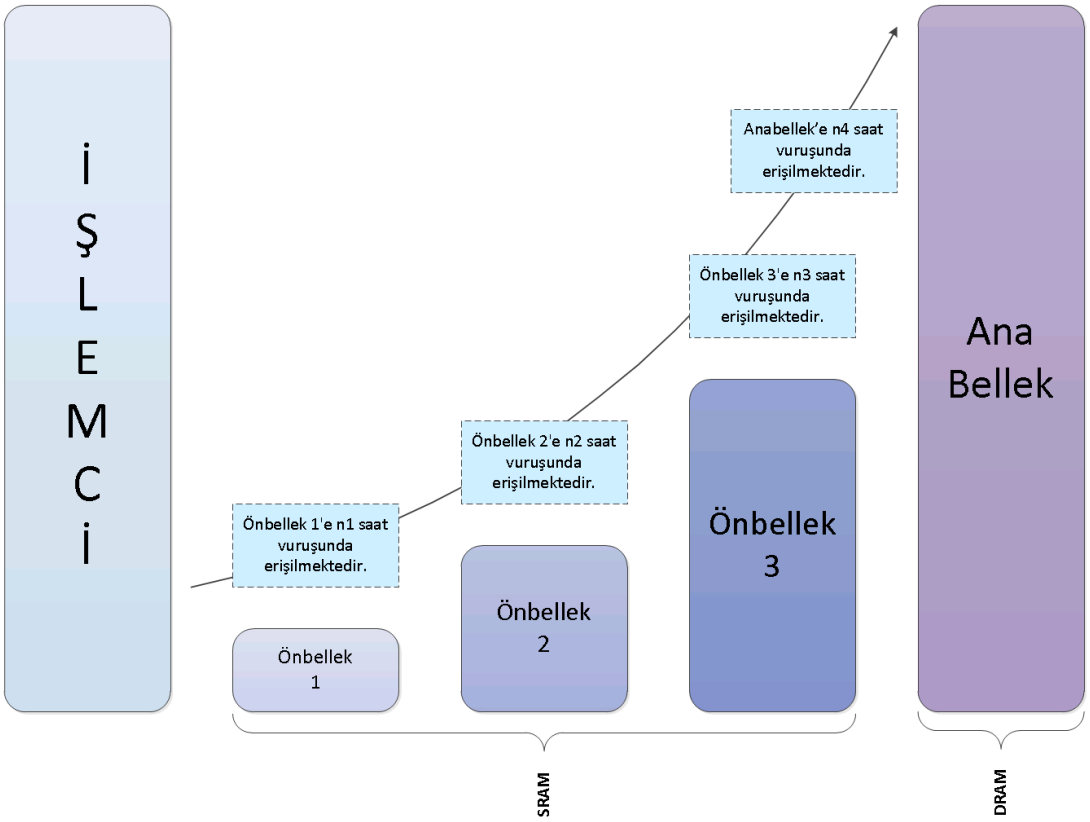
1.2.2 Dram

DRAM hücresinde bir transistör ve bir kapasitör bulunur. Veri, hücre içerisindeki kapasitörlerde tutulur. Kapasitörlerden yük sızıntısı meydana geldiği için periyodik yenilenmeye ihtiyaçları vardır.

Şekil 1-4'de görüldüğü gibi yapısında SRAM'e oranla daha az yarı iletken mevcuttur. SRAM'e kıyasla birim alana daha çok DRAM hücresi sığdırılabilir. Bu nedenle maliyeti SRAM'e göre daha azdır.



Şekil 1-4: (a) Sembolik dram hücresi ve (b) dram hücresi devresel gösterimi.



Şekil 1-5: Sram ve dram erişim süreleri.

DRAM bilgisayar mimarisinde ana bellek olarak kullanılmaktadır. Ön belleklere oranla daha büyük veri kapasitesine sahiptir. Şekil 1-5'de görüldüğü gibi veriye erişim süresi SRAM'e göre daha yavaştır (n4>n3>n2>n1)[1,4].

1.3 Tez Kapsamı

Bu tezde, bilgisayar mimarisinde önemli yere sahip bellek erişim süresi üzerine incelemeler ve erişim sürelerini iyileştirme adına yeni bir tasarım sunulmuştur. DRAM özelinde yapılan çalışmada geliştirilen "Yedek Dize" yöntemi ile bellek erişim sürelerinin azaldığı sonucuna varılmıştır.

Benzer bellek çalışmalarında kullanılan referans girdi dosyaları ile mekanizmanın başarım testleri yapılmıştır. Ayrıca mekanizmayı geliştirme aşamasında bu dosyalar üzerinde yerellik özellikleri analiz edilmiştir.

Bu tezde yer alan fikri destekleyebilmek için "Ramulator"[5] adı verilen ve akademik çalışmalar için geliştirilmiş bellek benzetim yazılımı kullanılmıştır.

Bu çalışmada, DRAM donanım mimarisi, DRAM çalışma düzeneği ve gecikme sürelerini iyileştirmeye yönelik geliştirilen yedek dize yöntemi hakkında bilgilere yer verilmiştir.

Tek çekirdekli veya çok çekirdekli mimariler üzerinde test sonuçları alınmıştır. Alınan sonuçlara göre başarım kıyaslamaları yapılmıştır.

1.4 Tez Organizasyonu

Bölüm 1'de bilgisayar mimarisi ana bileşenlerinden belleğe kısaca giriş yapılmıştır. Bilgisayarın ana bileşenleri Bölüm 1.1'de; ana bileşenlerden belleklerin sınıflandırılması Bölüm 1.2'de değerlendirilmiştir. Bölüm 1.3 bu tezin genel amacı ve içeriği, Bölüm 1.4 ise organizasyon yapısı açıklanmaktadır.

Bölüm 2'de DRAM Bellek Mimarisi hakkında detaylı bilgilendirme yapılmıştır. Bu bilgilendirme kapsamında bölüm 2.1'de hedef bellek teknolojileri ve darboğaz noktalara değinilmiştir. Bölüm 2.2'de DRAM donanım hiyerarşisi verilmiştir, en önemli iki bileşen olan alt dize ve veri hücresine değinilmiştir. Bölüm 2.3'de DRAM işlem sürelerini hızlandırmaya yönelik güncel çalışmalar sunulmuştur.

Bölüm 3'de JEDEC standardında yer alan ve tez kapsamında kullanılan bilgiler paylaşılmıştır. Bölüm 3.1'de DDR3 SDRAM yongalarının nasıl adreslendiği hakkında bilgi verilmiştir. Bölüm 3.2'de DRAM için standartta yer alan durum makinası anlatılmıştır. Bölüm 3.3'de durum makinasının geçişlerini tetikleyen DRAM buyrukları verilmiştir. Bölüm 3.4'de DDR3 sınıfı bellek yongalarını daha da özele indirgemeye yönelik olan donanım paket seçimi ve tez kapsamında kullanılacak donanıma ait zamanlama değerlerinden bahsedilmiştir.

Bölüm 4'de tez çalışmasının sonuç aşamasında kullanılan bellek benzetim yazılımı "Ramulator" hakkında bilgiler verilmiştir. Bölüm 4.1'de Ramulator çalışma mekanizması, bölüm 4.2'de ramulator özellikleri anlatılmıştır.

Bölüm 5'de geliştirilen "Yedek Dize" yöntemine ait detaylı bilgi sunulmuştur. Bölüm 5.1'de yedek dize yönteminin esinlenildiği noktadan bahsedilmiştir. Bölüm 5.2'de yedek dize yöntemine ait çalışma mekanizması, mekanizmanın donanıma, buyruk işleme mekanizmasına ve ramulator benzetim yazılımına olan etkisi anlatılmıştır. Bölüm 5.3'de yedek dize yöntemi ile hedeflenen kazanımlar ve öngörülen kayıplar paylaşılmıştır.

Bölüm 6'da "Yedek Dize" yönteminin test çıktılarına yer verilmiştir. Bölüm 6.1'de test alt yapısı ve bölüm 6.2'de zamansal yerellik fikri üzerine yapılan analizler mevcuttur. Bölüm 6.3'de tek çekirdekli işlemciler üzerinde yapılan test çıktıları ve bölüm 6.4'de çok çekirdekli işlemciler üzerinde yapılan test çıktıları verilmiştir. Tezin son bölümü olan bölüm 6.5'de alınan sonuçlar üzerine yapılan değerlendirme ve yorumlar yer almaktadır.

2. DRAM BELLEK MİMARİSİ

Yarı iletken bileşenlerinden ötürü Dinamik RAM'lerin veri hücreleri SRAM'lerin veri hücrelerinden daha küçüktür. DRAM kullanımının daha yaygın olma nedenlerinden biri yonga yoğunluğunun SRAM'e göre daha fazla olmasıdır. Silikon teknolojisinin gelişmesi ile birim alana düşen hücre sayısı her geçen gün artmaktadır. Bellek alanı arttırabilme imkanı olmasına rağmen veri hücreleri üzerindeki işlemler için harcanan süreler neredeyse sabit kalmaktadır [6]. Veri başına düşen alanın küçülmesine karşı işlem sürelerinde bir gelişme sağlanamaması bellek teknolojisini sınırlayan en önemli faktör haline gelmektedir[7,8]. Tezin bu bölümünde bellek teknolojisinde hedeflenen işler, DRAM mimarisine ait yapı taşları ve bu alanda yapılan güncel çalışmalar anlatılmıştır.

2.1 DRAM Teknolojisindeki Yönelimler

Bellek işlem sürelerini iyileştirme konusunda güncel bir çok çalışma vardır. Genellemek gerekirse Şekil 2-1'de de görüldüğü gibi mevcut teknolojinin geliştirilmesinde iki yönelim vardır. Birincisi, algı yükselteçlerin kapladığı alanın büyük olmasından dolayı birim hücre başına düşen algı yükselteç sayısını azaltmaktır. Bu durum, aynı bit hattını paylaşan daha çok hücre olmasına neden olacaktır. Bit hattının kapasitansı artacaktır ve hücre içerisindeki verinin okunabilmesi daha da gecikecektir[9,10,11]. İkincisi, yonga yoğunluğunu arttırabilmek için yarı iletken entegrelerin boyutlarını küçültmektir. Bu durumda, daha küçük kapasitör ve transistörler kullanılacaktır. Kapasitörlerin küçülmesi saklanan yük miktarının azalması anlamına gelir. Bu nedenle anlamlı veri, bit hattına daha yavaş akacaktır, erişim süresi artacaktır [9,10,12,13,14]. Ayrıca saklanan yük seviyesinin düşük olması nedeniyle yük kaybı yaşanmaması için daha sık yük yenilemesi yapılmalıdır.

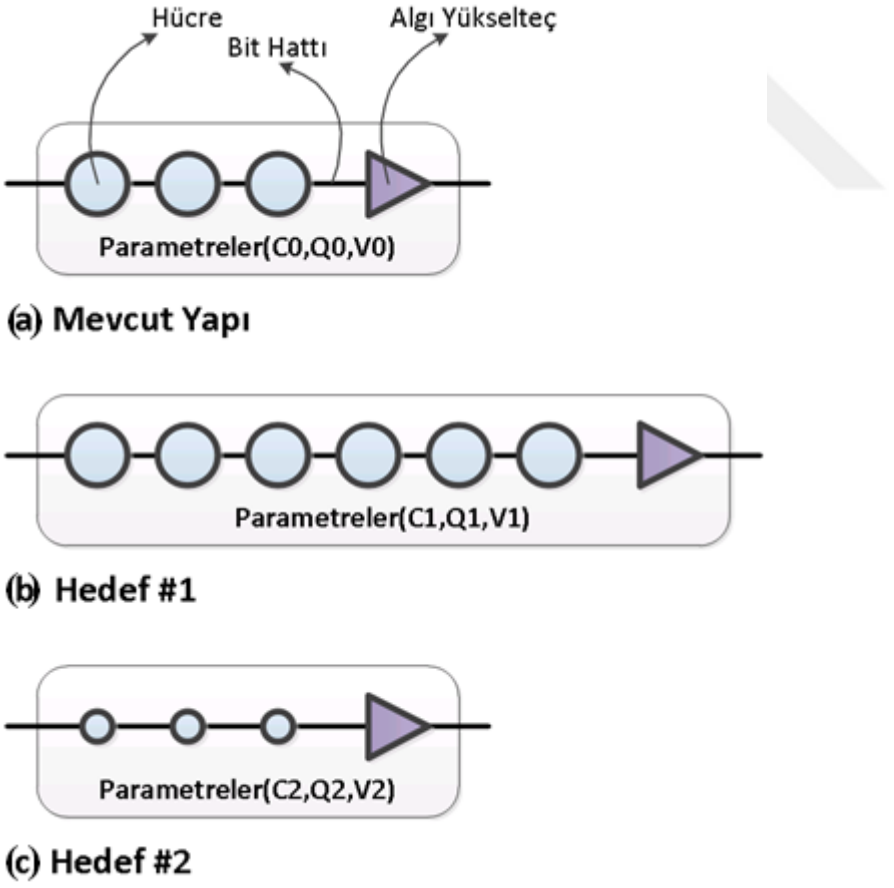
Şekil 2-1a'da mevcut teknoloji simgeselleştirilmiştir. Örnek olarak bir bellekte 3 adet hücre ve bunların bağlı olduğu 1 adet algı yükselteç vardır. Bit hattı kapasitansı

C0'dır, birim zamanda hücrelerden akacak yük Q0'dır ve bit hattı veri hızı V0'dır. Aynı şekilde bu değerler Şekil 2-1b ve Şekil 2-1c için C1, Q1, V1 ve C2, Q2, V2'dir.

Hedef 1'de gösterilen bellek tipinde mevcut sayıdan daha fazla hücre aynı algı yükseltece bağlanmıştır. Bu durumda bit hattının kapasitansı artacaktır ve bit hattı veri hızı azalacaktır (C1 > C0 => V1<V0) [9,10,11].

Hedef 2'de gösterilen bellek tipinde ise aynı sayıda bellek hücresi daha küçük alana sığdırılmıştır. Yapısındaki kapasitör ve transistörün boyutları küçültülmüştür. Bu durumda birim zamanda hücreden bit hattına akan yük miktarı azalacaktır (Q0 > Q2 => V0 > V2) [9,10,12,13,14].

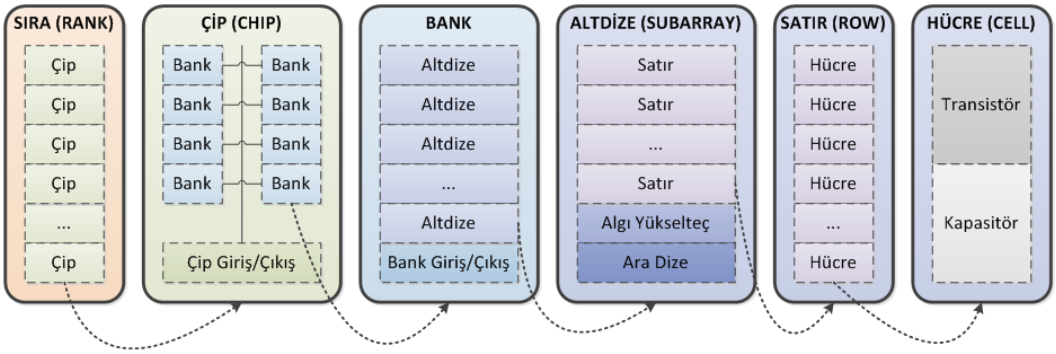
Bu iki yöneliminde olumsuz yanları mevcuttur. Temel sorun, teknolojinin değişmesiyle veri aktarım hızlarının negatif yönde etkilenmesidir. Bu nedenle bu alandaki güncel çalışmalar, teknoloji yönelimlerini destekleyecek hızlandırıcılar sunmaktadır.



Şekil 2-1: (a) Mevcut teknoloji, (b) hedef teknoloji 1 ve (c) hedef teknoloji 2.

2.2 Dram Donanım Bileşenleri

DRAM mimarisi hiyerarşik bir yapıya sahiptir. Şekil 2-2'de bu mimari sıralaması gösterilmiştir. Hiyerarşinin en üst katmanında yer alan kanallar (channel), sıralardan (rank) oluşmaktadır. Bu sıralama çipler (chip), banklar (bank) ve alt dizeler (subarray) [15] olarak devam etmektedir. DRAM bellek mimarisinin en küçük iki yapı taşı, dizeler (row) ve bellek hücreleridir[9, 10]. Bellek denetim birimi gelen talepler doğrultusunda DRAM mimarisindeki kanallara erişir. En alt katman olan veri hücrelerine kadar seçim yapılarak hedef işlem gerçekleştirilir.



Şekil 2-2: Dram bellek mimarisi bileşenleri.

En önemli iki mimari bileşeni alt dize ve veri hücresidir. Bu tezde savunulan "Yedek Dize" ile erişim sürelerinde elde edilen kazanım, bu iki bileşen üzerinde geliştirilmiştir.

2.2.1 Alt dize (subarray)

Şekil 2-2'de de görülüğü gibi her bir mimari bileşeni, alt bileşenlere ayrılmaktadır. Mimarinin bu şekilde farklı seviyelere ayrılma nedeni bellek mimarisinin denetimini daha kolay sağlayabilmek ve işlem sürelerindeki gecikmeleri azaltabilmektir. Aynı nedenden ötürü banklar da birden çok alt dizeye ayrılmıştır. Alt dize (satır), bit hattı ve kelime hattına bağlı satırlardan, bu satırların bağlı olduğu algı yükselteç ve ara dizelerden (row buffer) meydana gelir.

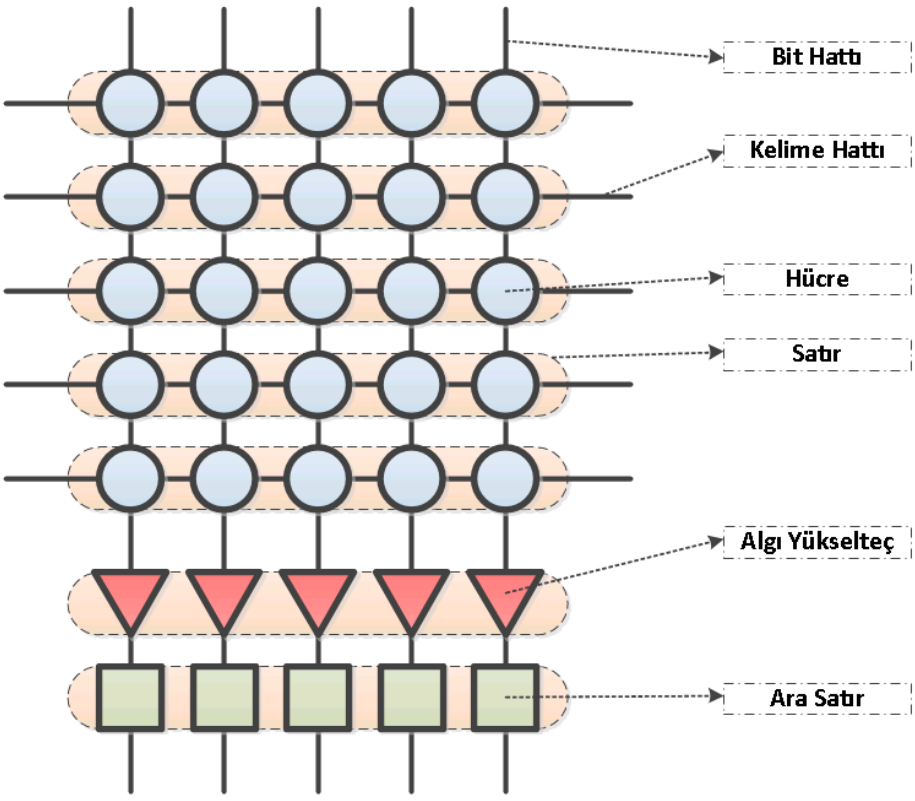
Şekil 2-3'de görüldüğü gibi hücrelerden oluşan satırlar kelime hatlarına ve bit hatlarına bağlıdırlar. İşlem yapılacak hücreler bit hatlarına anahtarlanmaktadır. Bit hattına bağlandıktan sonra hücre içerisindeki yük algı yükselteçlere akmaktadır. Algı

yükselteçler eşik değere ulaştığında okunabilir seviyeyi ara dize'ye (row buffer) kopyalamaktadır.

Hücre işlem esnasında, içerisindeki tüm yükün bit hattına boşalması nedeniyle sakladığı veriyi kaybetmektedir. Bu nedenle işlem sonunda algı yükselteçler pozitif geri besleme özelliği ile veri hücresini işlem başındaki durumuna ulaşana kadar yükler.

Şekil 2-3'de düşey olarak yer alan hücreler (aynı hizada yer alan farklı dize hücreleri) aynı bit hattını paylaşırlar ve aynı algı yükseltece bağlıdırlar.

Daha çok veriyi kaydedebilmek için daha yoğun bellek tasarımları yapmak gerekmektedir. Fakat bu yoğunluğun getirdiği olumsuz bir çok sonuç vardır. Bu olumsuz sonuçlardan en belirgini bit hattının direncinin ve kapasitansının kendine bağlı hücre sayısının artmasıyla doğru orantılı olarak değişmesidir. Bu olumsuz sonuca çözüm olarak her bir bank alt dizelere ayrılmıştır. 512 dizenin bir araya gelmesiyle alt dize bileşeni oluşturulmuştur. Her bir alt dizenin kendi yerel ara dizesi (local row buffer) mevcuttur. Yerel ara dize, evrensel ara dize'nin (global row buffer) getirdiği gecikme etkisini azaltmaktadır [16].

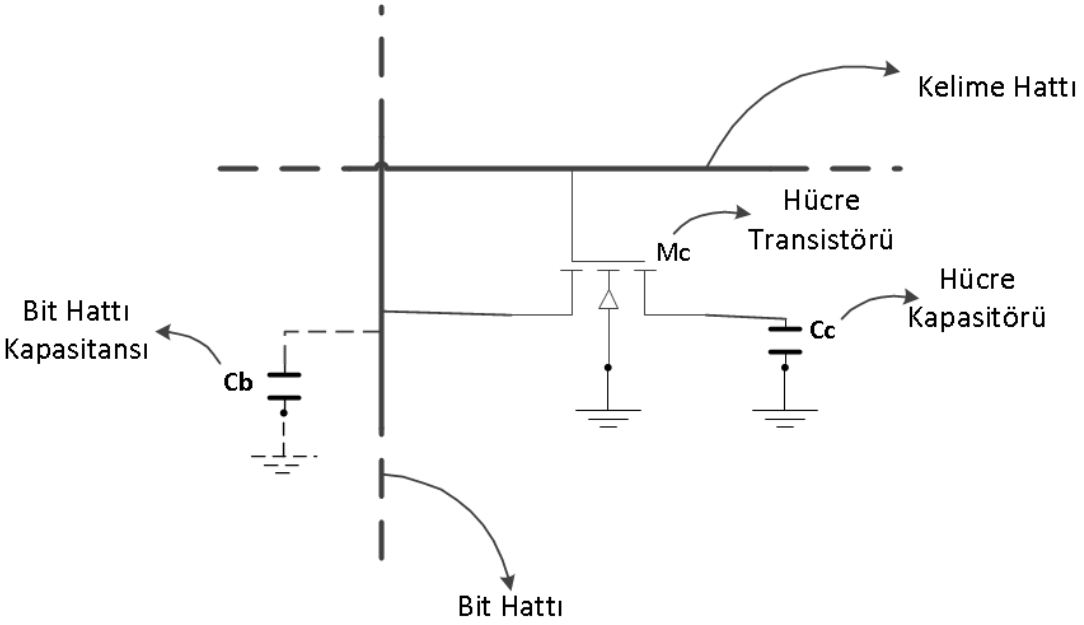


Şekil 2-3: Alt dize mimarisi.

2.2.2 Dram veri hücresi

DRAM bellek mimarisinin en küçük bileşeni veri hücresidir. DRAM hücreleri bir kapasitör ve bir transistörden oluşmaktadır. Hücresi donanımı Şekil 2-4'de verilmiştir. Veri hücresi kapasitörünü (Cc) kelime hattına anahtarlayan hücre transistörü (Mc) mevcuttur. Cc kapasitörünün görevi veri hücresinin değerini elektriksel yük formunda saklamaktır. Mc transistörünün görevi ise denetim biriminden gelen aktif buyruğunu kelime hattından alarak Cc kapasitörünü bit hattına anahtarlamaktır. Hücre bit hattına bağlandıktan sonra Cc kapasitörü içerisindeki yük bit hattı üzerinden algı yükselteçlere akmaya başlar. Algı yükselteçler, DRAM hücrelerinden büyük, kapasitörleri okuyan/ kapasitörlere yazan devre birimleridir [4].

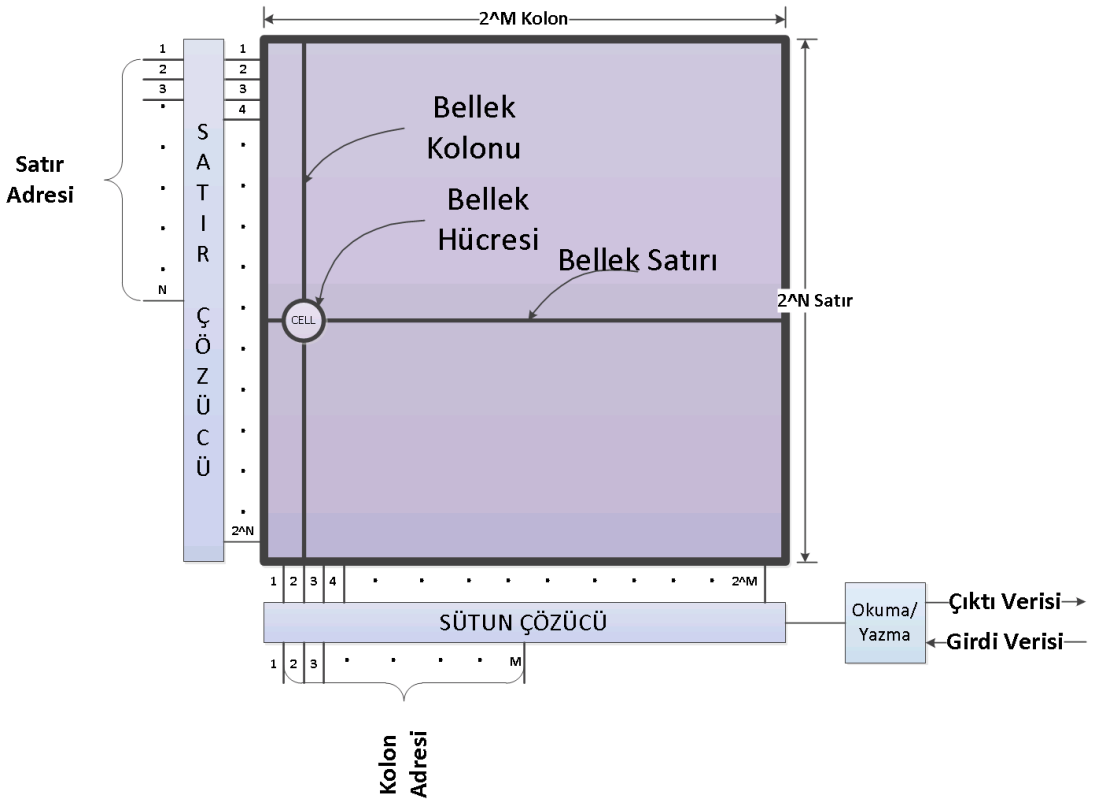
Şekil 2-4'de görülen Cb kapasitörü bit hattının kapasitansını ifade eder. Düz bir tel gibi düşünülse de üzerinde yük taşıyabilecek kapasitenin bulunmasından ötürü kapasitör şeklinde simgeleştirilebilir.



Şekil 2-4: Dram veri hücresi yapısı.

Bellek birimine gelen hedef hücre bilgisi, Şekil 2-5'de görüldüğü gibi hücreye ait satır adresini ve sürun adresini içerir. İlk önce, denetim birimi işlem yapılacak hücrenin bulunduğu satırı seçer. Satır seçimi sonrasında o satır üzerindeki tüm hücre transistörleri kendilerine bağlı kapasitörleri bit hatlarına bağlar. Tüm hücre yükleri bit hattının sonundaki algı yükselteçlere akmaya başlar. Eşik değere ulaşan algı yükselteçler, işlem yapılacak hücre değerlerini ara dizelere (row buffer) aktarır. Daha

sonra denetim birimi sütun seçimi yaparak tüm hücre verilerinden sadece ilgilendiklerini seçer ve işlem tamamlanır.



Şekil 2-5: Hücre koordinat seçim mekanizması.

2.3 Dram Üzerine Güncel Çalışmalar

DRAM bellek işlem sürelerinin zamansal yerelliğe bağlı geliştirilebileceği bu tezle sunulmuştur. Buna rağmen DRAM gecikmelerinin daha düşük seviyelere getirilmesi fikri güncel bir çok yayınla savunulmuştur. “Yedek Dize” fikrinin geliştirilmesi aşamasında bu yayınlardan ışık tutanlar tezin bu bölümünde özetlenmektedir.

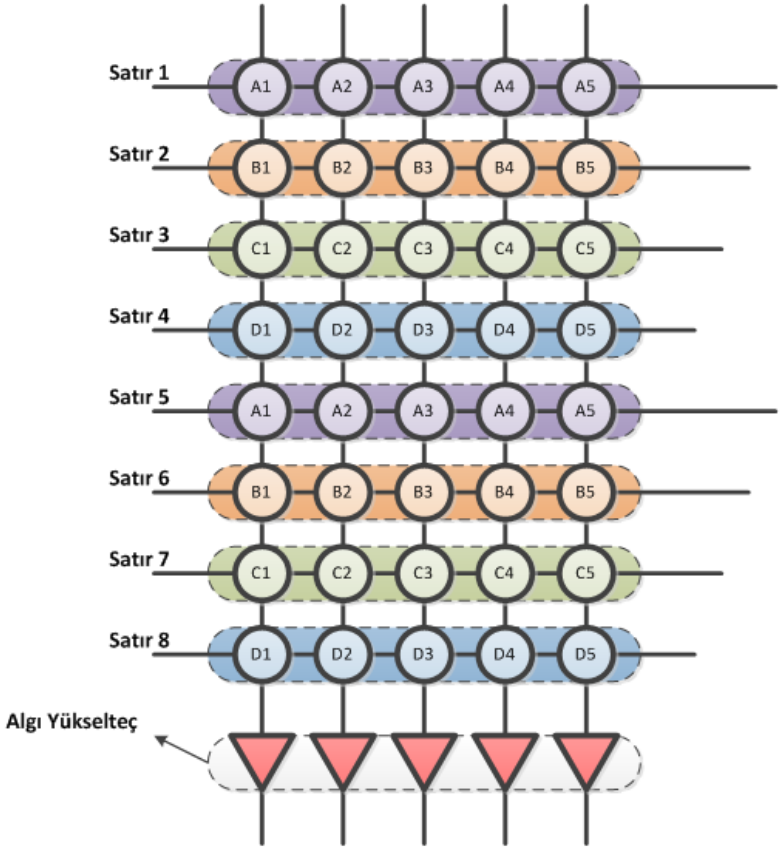
2.3.1 Çoklu kopyalanan dizeli dram (multiple clone row dram: a low latency and area optimized dram)

"Multiple Clone Row DRAM" (MCR) [14], yayınında her bir satırın en az bir yedek satırı mevcuttur. Her bir satır içerisindeki veri, o satırın yedeğinde de mevcuttur. Şekil 2-6’da 5x8'lik bir hücre dizisinden oluşan örnek bir bellek mevcuttur. Bu bellek üzerinde 2xMCR mekanizması çalıştırılacağı varsayılmıştır. 8 satırlı bir bellek olduğu için adresleme 3-bit ile gösterilecektir ("000"-"111"). 2xMCR

mekanizmasında erişim esnasında satır adresinin en önemsiz biti (MSB) kullanılmaz. Bu nedenle 1 numaralı satır ve 5 numaralı satır aynı bit dizisiyle adreslenir ("X00"), bu adreslerden herhangi birinde OKUMA/YAZMA/YENİLEME yapıldığı zaman yedeği olan satırlar da aynı işleme tabi tutulurlar. Şekil 2-6'da Satır1-Satır5, Satır2-Satır6, Satır3-Satır7 ve Satır4-Satır8 çiftleri 2xMCR mekanizmasında aynı verileri saklarlar. Genelleme yapmak gerekirse, nxMCR için log2(n) kadar en önemli bit adreslemede kullanılmaz. Her bir satır için log2(n) kadar yedek satır alanda rezerve edilir. Bu mekanizma ile DRAM gecikme sürelerinde azalma elde edilmiştir. Yayında bahsedilen üç kazanım vardır; erken erişim, erken ön dolum ve erken yenileme.

Hedef adres hücreleri ve onunla aynı veriyi tutan rezerve adres hücreleri OKUMA/ YAZMA talebi geldikten sonra aynı anda bit hattına anahtarlanmaktadır. Bu durumda bit hattına akan yük hızı artacaktır [13, 17]. Akan yükün hızlanması algı yükseltecin hatta bağlı hücrelerin içeriklerini daha hızlı algılamasını sağlayacaktır. Yayında mekanizmanın bu özelliği ile erken erişim sağlandığından bahsedilmiştir.

DRAM'deki hücreler üzerinde OKUMA/YAZMA işlemi yapılmasa bile yapısındaki kapasitörden yük sızıntısı olacağı için hücrelerde veri kaybı yaşanmaktadır. Veri kaybı riskini azaltabilmek için periyodik bir şekilde "REFRESH" buyruğu işlenir. Bu işlem esnasında hücre bit hatlarına bağlı algı yükselteçlerin pozitif geri besleme özelliği ile kapasitörlerindeki yük güncellenir. DRAM içerisindeki hücreler aynı anda yenilenemeyeceği için periyodik yenileme işlemi sıralı gerçekleşmektedir. JEDEC standardında yenileme işleminin periyodu 64ms olarak belirlenmiştir[18]. MCR mekanizmasında da yenileme işlemi standart biçimde sıralı devam etmektedir. Hedef ve ilgili hücreler aynı anda açılıp kapandığı için hedef ya da rezerve hücrelerden birinin yenilenmesi durumunda aynı veriyi taşıyan eş satırlar da yenilenmektedir. Bu nedenle 2xMCR için yenileme süresi 32ms'ye düşer. 32ms'de bir hücre yenilemesi için başlatılan işlem esnasında 64ms'de yaşanan sızıntı 32ms'de yaşanmayacağı için dolum daha hızlı gerçekleşecektir, yenileme işlemi daha hızlı tamamlanacaktır. Bu durum DRAM işlem mekanizmasında erken yenileme özelliği sağlamıştır.



Şekil 2-6: Mcr mekanizması.

OKUMA/YAZMA işlemi sonrasında algı yükselteç çıkışında veri okunduğu anda hücre kapasitörleri bit hattına boşaldığı için verileri kaybolmuş olur. Algı yükseltecin pozitif geri besleme özelliği ile tekrar dolmaya başlarlar. Bit hattına daha hızlı yük akmasıyla algı yükselteç algılama süresi azalmış olur. Daha az sürede algıladığı için bu süre zarfında kapasitörden akan yük de az olacaktır. Ön dolum işlemi daha az yük geri beslendiğinde biteceği için ön dolum daha hızlı gerçekleşir, buna erken dolum denir.

Bu çalışma sunduğu mekanizma ile "Yedek Dize" yöntemine en yakın yayındır. "Yedek Dize" yönteminde olduğu gibi aynı anda açılan birden çok satır olmasıyla DRAM gecikme sürelerinin azaldığı savunulmuştur. Tezde sunulan "Yedek Dize" yönteminin avantajı ise zamansal yerellik özelliği ile güncellenen yedek satırların, MCR yayınında rezerve tutulan satır sayısından az olmasıdır. "Yedek Dize" mekanizması donanımı daha yoğun kullanabilen bir yöntemdir.

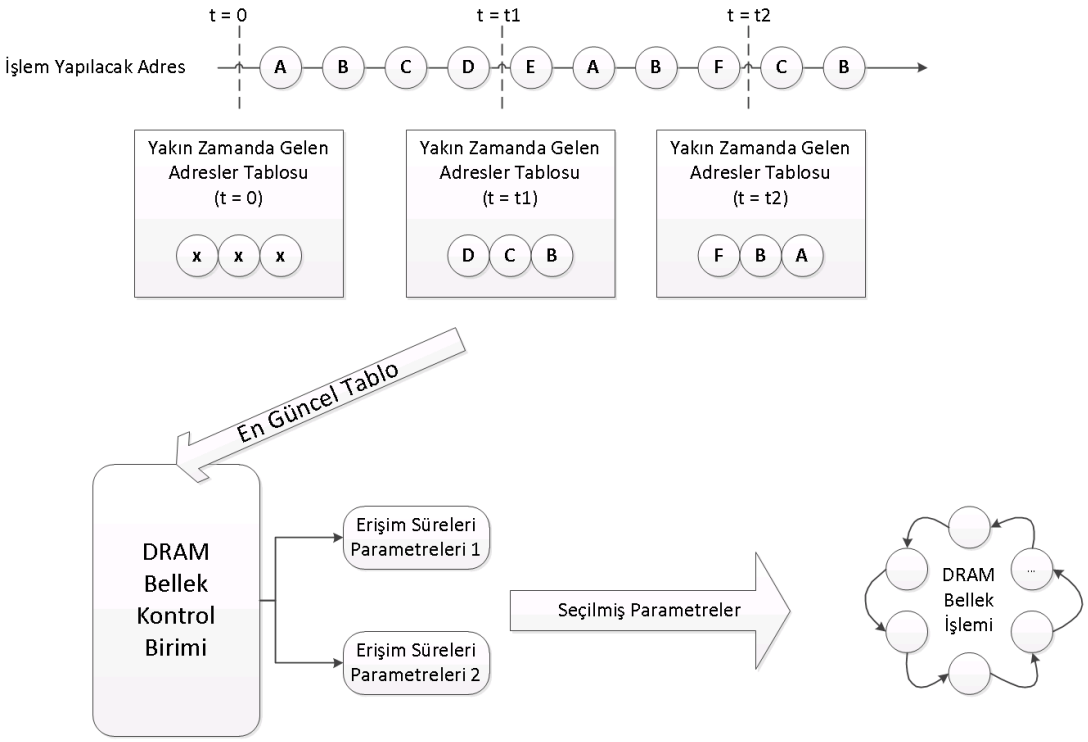
2xMCR ile her iki satırda birinin yedek olmasına karşılık yedek dize yönteminde her 512 satırdan sadece biri rezerve olarak donanımda tutulmaktadır.

MCR yayınında yük kapasitansının artması ile hücre içerisindeki yükün daha hızlı algılanabileceği ve daha sık yenilenen hücrelerde yük sızıntısı etkisinin daha az görülebileceği gösterilmiştir. Tek çekirdekli işlemci üzerinde çalıştırılan MCR yöntemi ile işlem zamanı ve okuma sürelerinin yaklaşık %8.3, %13.1 ve %14.1 seviyelerinde azaldığı gözlenmiştir. Çok çekirdekli işlemci testlerinde ise bu sonuçların %11.2, %11.4 ve %23.2 seviyelerinde seyrettiği görülmüştür.

2.3.2 Chargecache

ChargeCache [12], daha önce erişilmiş bir adrese daha hızlı erişebilmek için geliştirilen bir yöntemdir. ChargeCache mekanizması, yakın geçmişte erişilmiş hücrelerin, yük miktarı olarak uzun süredir erişilmemiş ya da yenilenmemiş hücrelerden fazla olduğu gözlemi üzerine kurulmuştur. Buradan yola çıkarak yeni erişilen satırların adreslerinin bir tabloda tutulması ve bu tablonun bellek denetim birimi tarafından kullanılması önerilmiştir. Denetim birimi bu tabloda yer alan herhangi bir satıra erişeceği zaman yük bakımından zengin hücreler üzerinde işlem gerçekleştireceğini bilecektir. DRAM işlem süreleri parametrelerini buna göre ayarlayarak satıra erişim sürelerini azaltabilmektedir.

Şekil 2-7'de ChargeCache mekanizmasına ait örnek bir akış verilmiştir. Burada 3 farklı zaman aralığında denetim biriminin kullanacağı yakın zamanda gelen adresler tablosu verilmiştir. t=0 anında bu tablo boştur. Tablo boşken gelebilecek adres işlem talebi için bellek denetim birimi standart işlem sürelerini kullanmaktadır (erişim süreleri parametreleri 1). Daha sonra her işlenen adresle tablo güncellenmektedir. t=t1 anında tablonun içerisinde D, C ve B adresleri bulunmaktadır. t=t1 anında gelebilecek adreslerden herhangi biri D, C veya B adresi ise hızlı işlem süreleri (erişim süreleri parametreleri 2), değilse standart işlem süreleri (erişim süreleri parametreleri 1) kullanılacaktır.Aynı durum t= t2 anı için de geçerlidir. t=t2 anından sonra gelecek F, B veya A adresleri için hızlı işlem süreleri kullanılacaktır. Bu tablo her gelen adres ile güncellenmektedir ve denetim birimi işlem yapmadan önce en güncel olan tabloyu kullanmaktadır. İlgili erişim süresi seçenekleri ile DRAM bellek işlem akışına girmektedir.



Şekil 2-7: Chargecache mekanizması.

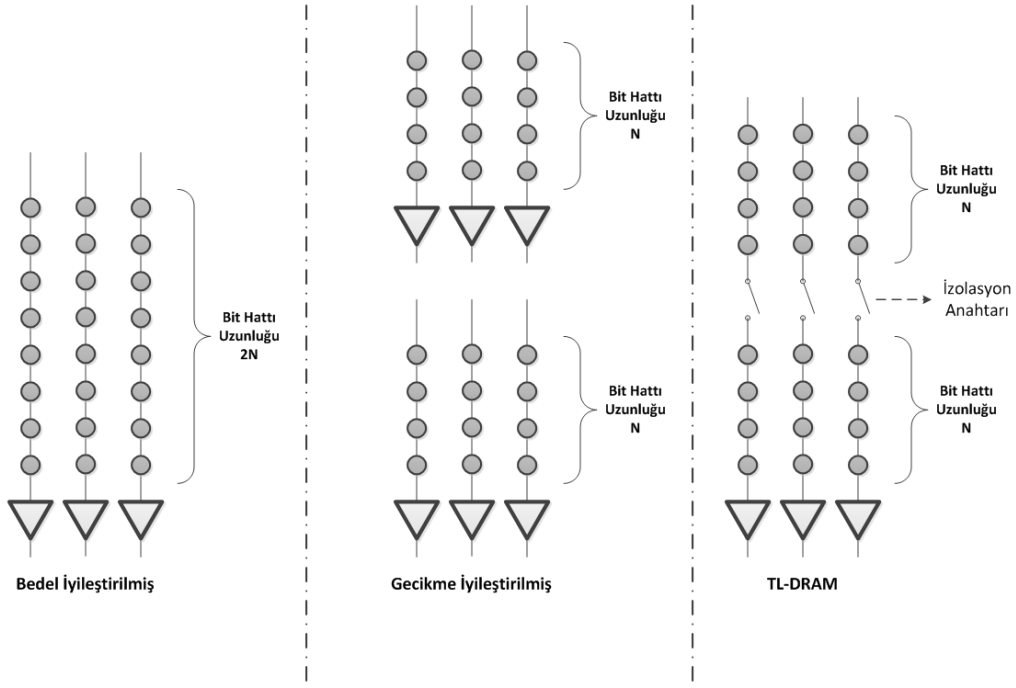
ChargeCache mekanizması ile "Yedek Dize" yönteminin benzer yanları geriye dönük satır bilgisi tutma, denetim biriminin bunu denetlemesi ve bu tabloyu erişim sürelerini iyileştirebilmek için kullanabilmesidir.

ChargeCache mekanizması ile "Yedek Dize" yöntemi arasındaki fark ise donanım tabanlıdır. ChargeCache yöntemiyle herhangi bir donanım değişikliği gerekmeden mekanizma çalışabilirken, "Yedek Dize" yönteminde her bir alt dizeye ait bir yedek dizeye ihtiyaç duyulmaktadır.

ChargeCache yöntemiyle elde edilen erişim süresi iyileşme oranları ilgili yayında tek çekirdekli ve çok çekirdekli işlemciler için verilmiştir. Tek çekirdekli işlemcilerde %2.1 ile %8.1 arasında, çok çekirdekli işlemcilerde ise %8.6 ile %11.3 oranında iyileşme gözlenmiştir.

2.3.3 Sıralı-gecikme dram (tiered-latency dram: a low latency and low cost dram architecture)

TL-DRAM [9] yayınında bit hattının kapasitansını azaltmaya yönelik bir fikir sunulmuştur. Bit hattını daha kısa parçalara bölerek, yük paylaşımının daha hızlı gerçekleşebileceği ve hücrelerin içlerindeki veriyi algı yükselteçlerin daha hızlı algılayabileceğini gösterilmiştir (Şekil 2-8).



Şekil 2-8: Tl-dram mekanizması.

Bu yayının, "Yedek Dize" yöntemi ile benzer yanı, hem denetim biriminin çalışma mekanizmasında hem de DRAM donanımında değişiklik gerekmesidir. Önerilen yöntemle donanıma izolasyon anahtarları eklenmektedir. Ayrıca bu anahtarı yönetebilecek bir denetim mekanizması geliştirilmiştir.

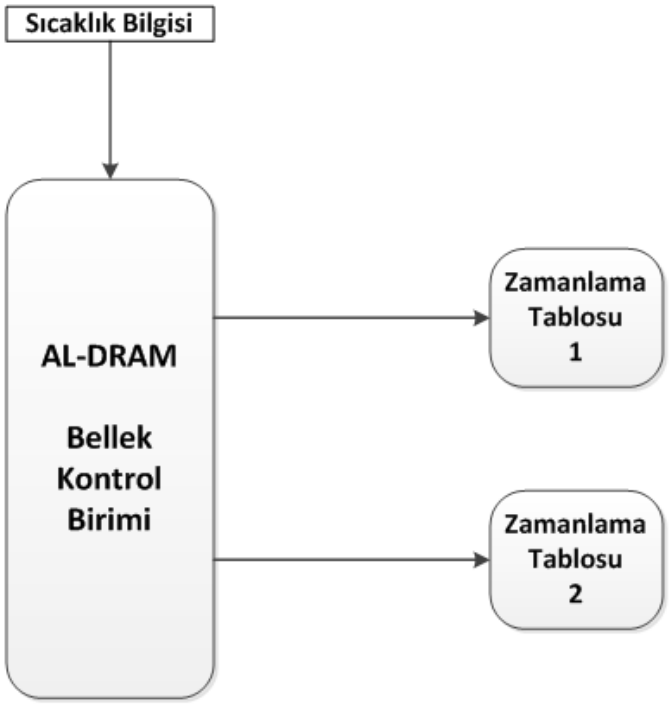
TL-DRAM mekanizmasının "Yedek Dize" yöntemi ile farkı ise bit hattı kapasitansını koşullara göre azaltabilmesidir.

2.3.4 Uyarlamalı-gecikme dram (adaptive-latency dram: optimizing dram timing for the common-case)

AL-DRAM [10] kullanıcılara gerçek zamanlı olarak çalışma koşullarının değişmesiyle zamanlama parametrelerini ayarlayabilme imkânı sağlamaktadır. Bu yayında, değişen sıcaklık koşullarına göre düşük ya da yüksek sıcaklıklarda en iyi performansla çalışacak zamanlama parametrelerini ayarlayarak gecikme sürelerinin azaltılabileceği gösterilmiştir.

AL-DRAM çalışmasında donanım üzerinde değişiklik yerine denetim birimi çalışma mekanizmasında değişiklik yapılması savunulmuştur. Erişim sürelerinin iyileştirilmesi yonga sıcaklık değerleri ile seçilebilir bir hale getirilmiştir. Şekil 2-9'da da görüldüğü gibi sıcaklık değişimine bağlı olarak denetim birimi zaman

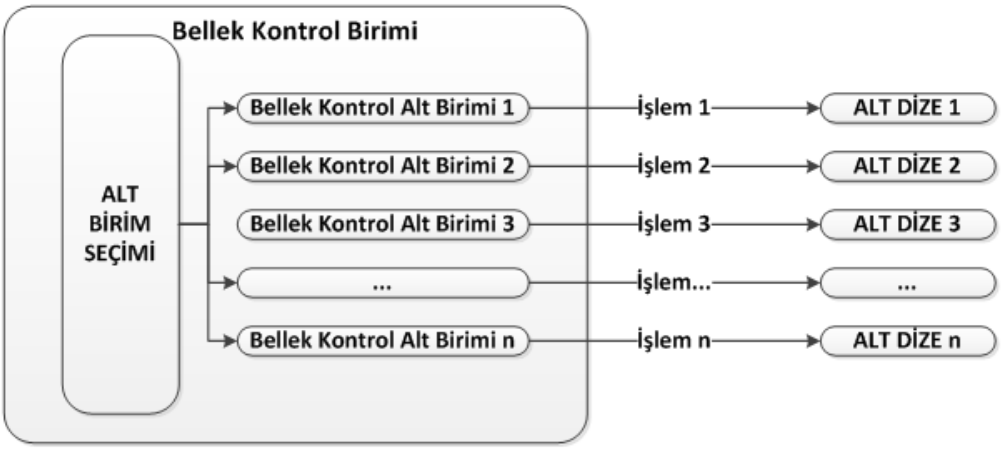
tablosu 1 ya da 2'yi seçerek çalışmaktadır. Bu değerler o anki sıcaklık için uygun erişim süreleridir.



Şekil 2-9: Al-dram mekanizması.

2.3.5 Dram mekanizmasının alt-dize seviyesinde eş zamanlı yürütülmesi (a case for exploiting subarray-level parallelism –salp- in dram)

Bu yayında, SALP [11] mekanizması ile DRAM denetim biriminin çalışma şeklini değiştirerek, farklı alt dizelerde yer alan birden çok adresin aynı anda işlenebileceği fikri sunulmuştur. Standartta ilgili adresler seri şekilde işlenmektedir. Eğer aynı anda birden çok hücre aktif hale gelirse aynı algı yükseltece bağlı hücrelerin verileri birbirlerini sönümleyebilir ve işlem sonrası yanlış yük ile yüklenebilirler. Bu riski ortadan kaldırmak adına işlenecek adreslerin aynı alt dizede olup olmadığı SALP ile denetim edilmektedir. Eğer hedef adresler aynı alt dizeye ait satırlar değil ise eş zamanlı birden çok adres işleme imkanı sağlanmaktadır. Şekil 2-10'da görüldüğü gibi her bir alt dizeye ait farklı hedef adresler aynı anda denetim birimi tarafından işlenebilir.



Şekil 2-10: Salp mekanizması.

"Yedek Dize" ile benzer yanının denetim birimi üzerinde alt dize seviyesinde değişiklik gerekmesidir.

3. JEDEC

JEDEC [21], mikroelektronik endüstrisinde ihtiyaç duyulan teknik bilgi ve gelişimi sağlayabilmek için imalatçı ve tedarikçileri bir araya getiren ve evrensel standartlar sunan bir konseydir. JEDEC çalışmaları tüm dünyaca kabul edilen, ücretsiz ve açık kaynak standartlardır.

JEDEC'in JESD79-3C standardında [18] DDR3 SDRAM tipi DRAM'ler için teknik özellikler, fonksiyonel özellikler, AC ve DC karakteristikler, donanım paketleri ve sinyal belirtimleri açıklanmıştır. Bu belirtim dökümanının temel amacı minimum özellik belirterek JEDEC uyumlu DDR3 SDRAM bellekleri tarifleyebilmektir. Bir diğer deyişle JEDEC uyumlu DDR3 belleklerin herhangi bir paketi için herkeste aynı bilgi kümesini oluşturabilmektir.

Benzer bir standartlaştırma daha öncesinde DDR2'ler için (JESD79-2) [19] ve DDR'lar (JESD79) [20] için de yapılmıştır. Bu standartlarda da ilgili bellekler teknik anlatımlar ile sunulmuştur.

Bu tez çalışması süresince JESD79-2C standardında yer alan pin işlev şeması, DDR3 SDRAM adresleme, fonksiyonel akış şeması, buyruklar ve elektriksel karakteristik bölümlerinden yararlanılmıştır.

3.1 DDR3 SDRAM Adresleme

Toplam veri alanı, adres genişlikleri ve sayfa boyutlarına göre DRAM'leri gruplamak mümkündür. Toplam veri alanına göre SDRAM'ler 5'e ayrılır; 512Mb, 1Gb, 2Gb, 4Gb ve 8Gb. Tüm bu gruplar için farklı sayfa boyutu, sütun adres genişliği ve satır adres genişliği değişkenleri mevcuttur [18].

Çizelge 3-1'de 8Gb boyutunda farklı IO pin sayılarına sahip SDRAM'ler için adres ve boyut bilgileri verilmiştir. Çizelge 3-2'de 4Gb boyutu için ilgili değişkenler verilmiştir. Tablolarda da görüldüğü üzere boyut sınıflandırması, satır adresi, sütun adresi ve giriş çıkış pin sayısına göre yapılmıştır. Bu değişkenlere bağlı olarak bellek konfigürasyonu ve sayfa boyutu değişmektedir. Hedef veri hücresinin aktif hale

gelmesi ve bu hücre üzerinde işlem yapılabilmesi için bellek yongası içerisinde konumu sütun ve satır adresleriyle sağlanmaktadır.

Çizelge 3-1: 8Gb sdram adres genişlikleri.

Konfigürasyon	2Gbx4	1Gbx8	512Mbx16
Bank Adresi	BA0-BA2	BA0-BA2	BA0-BA2
Satır Adresi	A0-A15	A0-A15	A0-A15
Sütun Adresi	A0-A9,11,13	A0-A9,A11	A0-A9
IO Pin Sayısı	4	8	16
Sayfa Boyu	2KB	2KB	2KB

Çizelge 3-2: 4Gb sdram adres genişlikleri.

Konfigürasyon	1Gbx4	512Mbx8	256Mbx16
Bank Adresi	BA0-BA2	BA0-BA2	BA0-BA2
Satır Adresi	A0-A15	A0-A15	A0-A14
Sütun Adresi	A0-A9,11	A0-A9	A0-A9
IO Pin Sayısı	4	8	16
Sayfa Boyu	1KB	1KB	2KB

3.1.1 Sayfa boyutu hesabı

Sayfa boyutu, aktif et buyruğu sonrasında algı yükselteçlere bağlanan dizenin bayt cinsinden boyutudur. Sayfa boyutu, sütun sayısı ve bellek I/O pin sayılarına bağlı olarak değişmektedir. Denklem (3.1) [18]'de sayfa boyutu hesabı verilmiştir.

$$Sayfa\ Boyutu = \frac{(2^N)*(ORG)}{8} \tag{3.1}$$

Denklem (3.1)'de yer alan N sayısı kolon adreslemek için kullanılan bit sayıdır. ORG değeri ise giriş çıkış pin sayısıdır.

Çizelge 3-1 ve Çizelge 3-2'de sayfa boyutları verilmiş 6 adet bellek yongası vardır. Sayfa boyutunun sütun adresi ve bellek giriş çıkış sayısına bağlı değiştiği bu çizelgelerde görülmektedir.

3.1.2 Alt dize (subarray) hesabı

Alt dize (Subarray), banklarda bulunan satırlar kümesidir. Her bir alt dize 512 adet satırdan meydana gelmektedir. Alt dize hakkında daha detaylı açıklamalar 2.2.1'de Alt Dize (Subarray) bölümünde verilmiştir. Alt dize sayısı, zamansal yerellik özelliği tespiti için bellek denetim birimi tarafından kullanılan tablonun boyutunu belirlemektedir. "Yedek Dize" yönteminde her bir alt dize için 1 adet yedek dizenin yer aldığı bir tablo mevcuttur. Bir banktaki alt dize (subarray) sayısı Denklem (3.2)'deki gibi hesaplanmaktadır.

$$AltDizeSayısı = \frac{BanktakiSatırSayısı}{512} \tag{3.2}$$

Bir bankta her 512 satır bir alt dizeyi meydana getirmektedir. Her bir alt dizede 1 adet yedek dize bulunur [9]. Bu nedenle "Yedek Dize" mekanizmasında alt dize sayısı kadar yedek dize bilgisi tutan bir tablo mevcuttur.

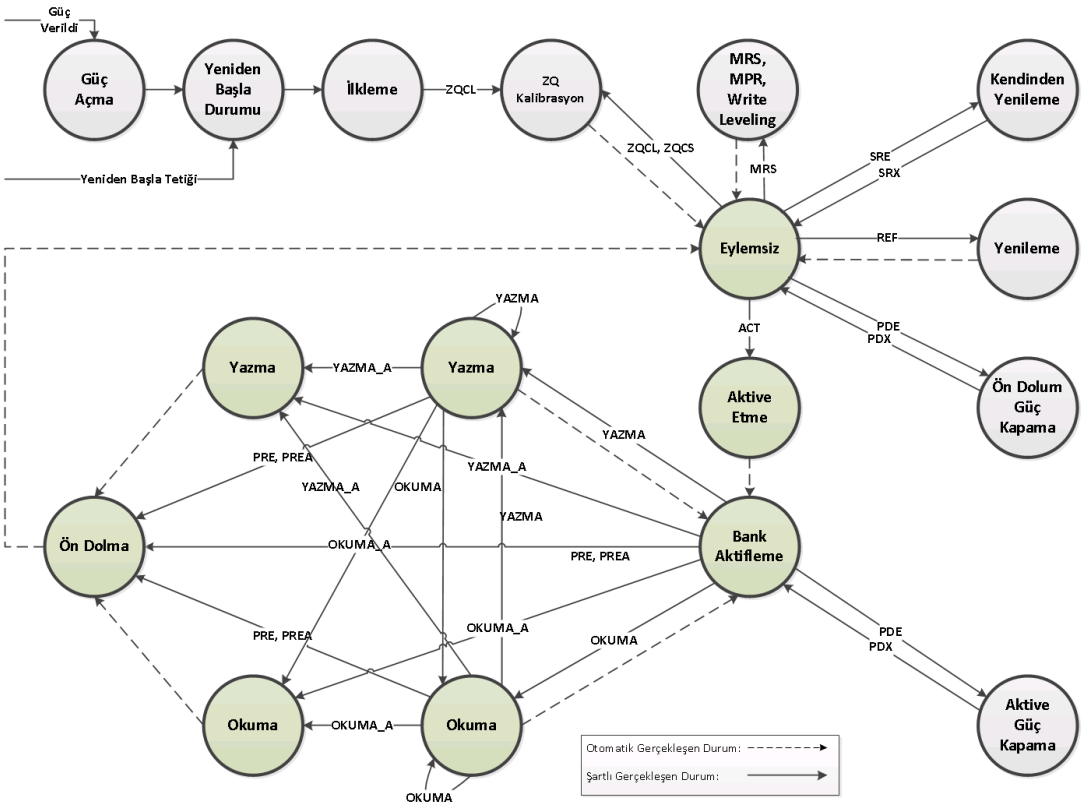
3.2 Fonksiyonel Akış Şeması

Herhangi bir bellek işlemini yapabilmek için bellek hücrelerinin standartta belirlenmiş olan durumlardan geçmesi gereklidir[18]. Şekil 3-1'de (işlem akış diyagramı) DRAM için durum geçişlerini ve buyruk işlenişini gösteren sadeleştirilmiş durum makinası verilmiştir. Verilen durum makinasında birden fazla bankta görülebilecek bazı durum geçişlerine, denetimci seçimine ve bazı diğer durumlara yer verilmemiştir. Durum makinası üzerinde görülen durum geçiş koşulları (buyruklar) bölüm 3.3'de daha detaylı olarak anlatılmıştır. Şekilde dikkat edilmesi gereken husus, sunulan tez çalışmasında geliştirilen mekanizmanın yeşil renkli durumları ilgilendirmesidir. Geliştirilen yöntemle aktive edilen bir bank için hedeflenen koşullara erişildiğinde OKUMA, YAZMA ve ÖN DOLUM aşamalarında süreden kazanım elde edilmektedir.

3.3 Dram Buyrukları

DRAM belleklerin tümü Şekil 3-1'de verilen durum makinasına benzer mekanizmalarla çalışmaktadır [18, 19, 20]. Herhangi bir buyruğun işlenebilmesi için gerekli sinyallerin entegrede sürülüyor ya da okunuyor olması gerekmektedir. Tezin

bu bölümünde DDR3 fonksiyonel akış şemasında yer alan durum makinası buyrukları hakkında bilgi verilecektir.



Şekil 3-1: Sdram işlem akış diyagramı.

Buyruklar birbirlerinden farklıdır fakat ortak olan özellikleri de mevcuttur. Tüm DRAM buyrukları, saat vuruşunun yükselen zamanında CS#, RAS#, CAS#, WE# ve CKE sinyallerinin durumuna göre anlamlandırılır. BA, RA ve CA sinyalleri ise DRAM entegresinin tipine göre farklı buyrukların işlenmesinde kullanılabilmektedir. Alt seviyede aktif RESET sinyali DRAM mekanizmasına eşzamansızdır, bu nedenle herhangi bir buyruğun işlenebilmesi için RESET sinyalinin HIGH sürülmesi gerekmektedir. "V" sembolü ilgili sinyalin HIGH ya da LOW olması gerektiğini söylemektedir. "X" sembolü ise ilgili pine verilen sinyalin herhangi bir önemi olmadığını göstermektedir [18].

“Yedek Dize” yöntemi ile etkilenen durumlar bu bölümde anlatılmıştır. Elde edilen kazanım bank aktifleme, ön dolum ve okuma buyrukları esnasında sağlanmaktadır.

3.3.1 Aktif et buyruğu sonrası elektriksel durumlar

Aktif buyruğu, işlenecek OKUMA/YAZMA buyruklarından önce ilgili bank satırını açmaya (aktif hale getirmeye) yarar. İlgili bank BA bitlerinin değerine göre seçilir.

İlgili adres ise satır adresinin o anki değeridir. Aktif etme buyruğunun uygulandığı bank adresi Ön Dolma buyruğu (PRECHARGE) gelene kadar açık kalır. Aynı bankta başka bir adres üzerinde işlem yapabilmek için açık olan adresin kapatılması gerekmektedir. Bank aktifleme buyruğu için gerekli olan koşullar Çizelge 3-3'de verilmiştir.

Çizelge 3-3: Fonksiyonlara bağlı elektriksel durumlar.

Fonksiyon	Kısaltması	CKE		CS#	RAS#	CAS#	WE#	Bank Adresi	A13-A15	A12-BC#	A10-AP	A0-A9, A11
		Önceki Period	O Anki Period									
Bank Aktifleme	ACT	H	H	L	L	H	H	BA	Satır Adresi (RA)			
Ön Dolum	PRE	H	H	L	L	H	L	BA	V	V	L	V
Tüm Ön Dolum	PREA	H	H	L	L	H	L	V	V	V	H	V
Okuma	RD	H	H	L	H	L	H	BA	RFU	V	L	CA
Yazma	WR	H	H	L	H	L	L	BA	RFU	V	L	CA

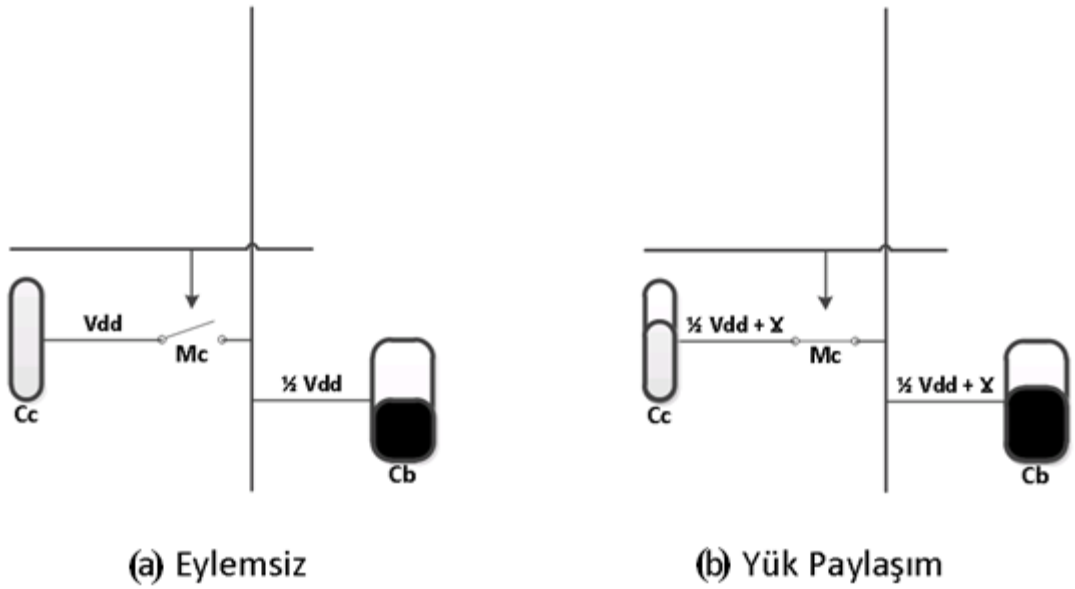
Aktif etme buyruğu geldikten sonra veri hücresi okunabilir ya da yazılabilir duruma gelirken ilgili hücre ve bağlı olduğu bit hattı farklı elektriksel durumlardan geçmektedir.

Bir hücreye aktif et buyruğu gelmeden önce hücrenin eylemsiz durumda olması gerekmektedir. Şekil 3-2a'da görüldüğü gibi hücre eylemsiz durumdayken tamamen yüklü durumdadır. Hücreyi bit hattına bağlayan anahtar açık durumdadır. Bit hattı üzerinde besleme geriliminin yarısı kadar yük vardır.

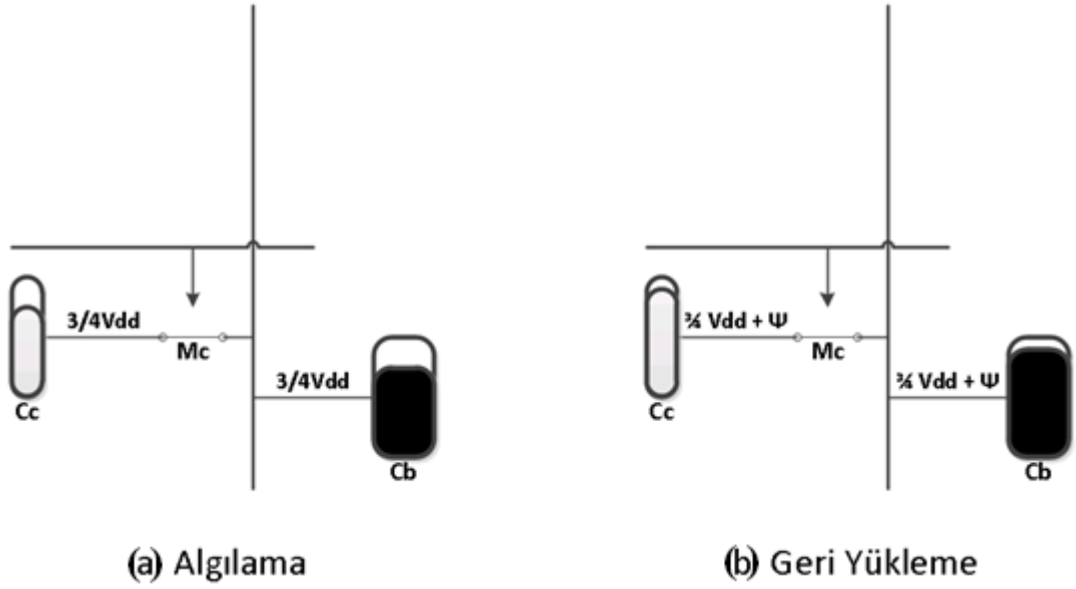
Aktif et buyruğu geldikten sonra eylemsiz durumda olan hücrenin transistörü, hücre kapasitörünü bit hattına bağlar. Kapasitör içerisindeki yük, bit hattına bağlı algı yükselteçlere doğru boşalmaya başlar. Bu durum yük paylaşımı olarak adlandırılmaktadır (Şekil 3-2b). Yük paylaşımı esnasında bit hattı üzerindeki yük artarken, hücrenin yükü azalmaktadır.

Bit hattına akan yük, bir süre sonra algı yükselteç eşiğine ulaşır (Şekil 3-4'de t = t1 anında eşik değere ulaşmaktadır). Bu durumda algı yükselteç, hücre içerisindeki yükün '1' veya '0' olduğunu algılar. Bu duruma algılama adı verilir (Şekil 3-3a). Bu aşamada algı yükselteç, ara dizelere (row buffer) ilgili veriyi aktarır. Kullanıcı hücre içerisinde hangi veri olduğunu bu aşamadan sonra okuyabilir duruma gelir.

Algılama durumu sonrasında algı yükselteçlerin pozitif geri besleme özelliği ile hücre eylemsiz durumdaki yükle tekrar yüklenmeye başlar. Şekil 3-3b'de yer alan bu duruma geri yükleme adı verilir. Hücre eylemsiz durumdaki yük miktarına ulaşana kadar geri yükleme durumunda sayılır.

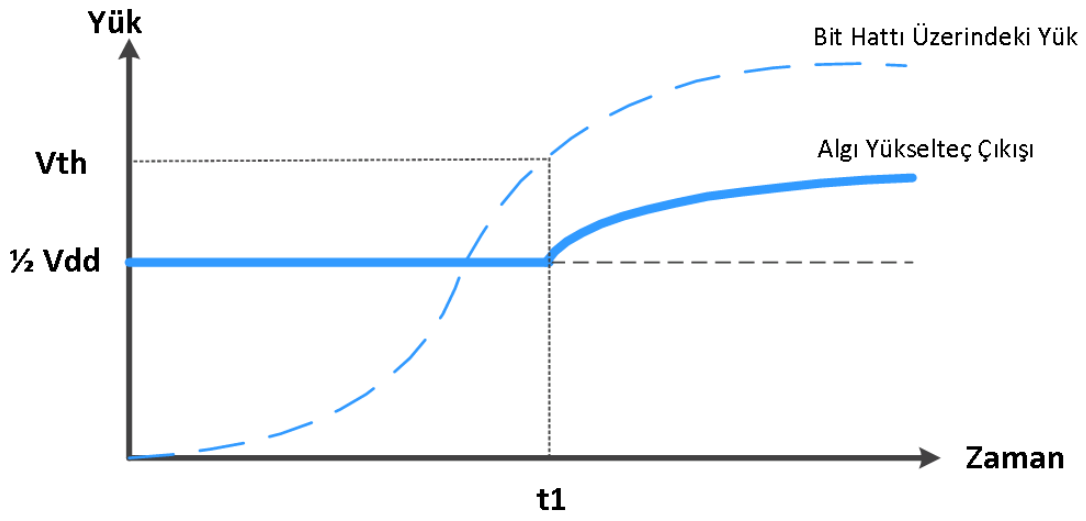


Şekil 3-2: (a) Eylemsiz ve (b) yük paylaşım durumları.

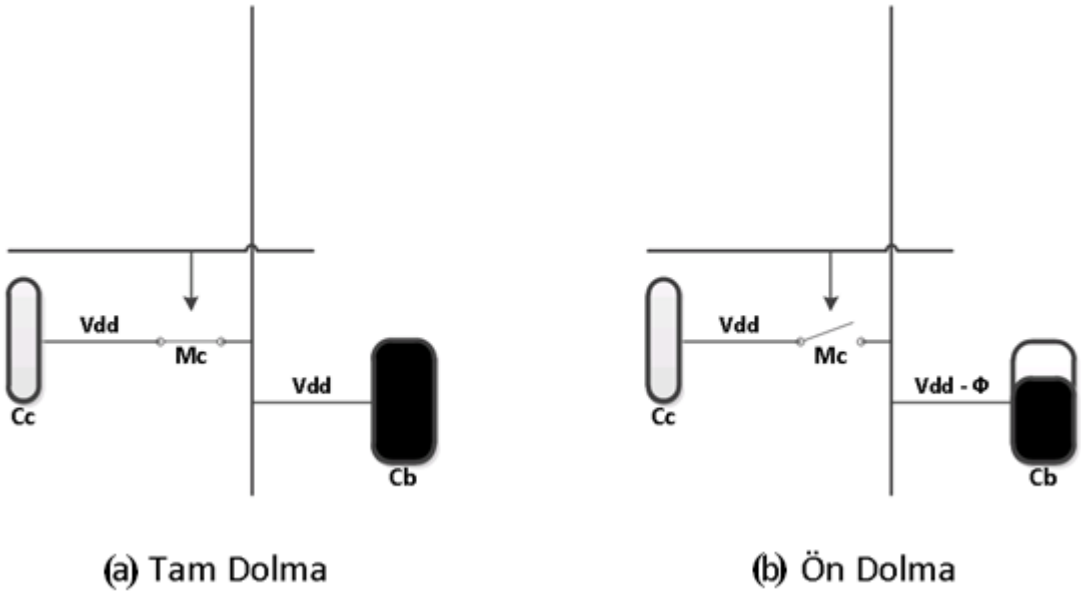


Şekil 3-3: (a) Algılama ve (b) geri yükleme durumları.

Eğer hücre bu değere ulaşırsa hücre tam dolmuş duruma gelir (Şekil 3-5a). Bu anda bit hattı da tamamen yüklü durumdadır. Hücre anahtarı, kapasitör ile bit hattı arasındaki bağı koparır. Daha sonra bit hattı, besleme geriliminin yarısına ulaşana kadar yük kaybeder. Bu duruma ön dolma adı verilir (Şekil 3-5b).



Şekil 3-4: Zamana bağlı algılama seviyesi grafiği.



Şekil 3-5: (a) Tam dolma ve (b) ön dolma durumları.

Ön dolum (PRECHARGE) buyruğu açık olan bank satırlarını kapatmak için kullanılır. Bir sonraki ACT buyruğunun işlenebilmesi için PRE buyruğundan sonra belirli bir süre beklemek gerekmektedir. PRE buyruğu uygulanan bank "Eylemsiz" durumuna geçiş yapmaktadır ve OKUMA/YAZMA işlemlerinden önce kesinlikle ACT buyruğunu işlemesi gerekmektedir. Ön dolum buyruğu için gerekli olan koşullar Çizelge 3-3'de verilmiştir.

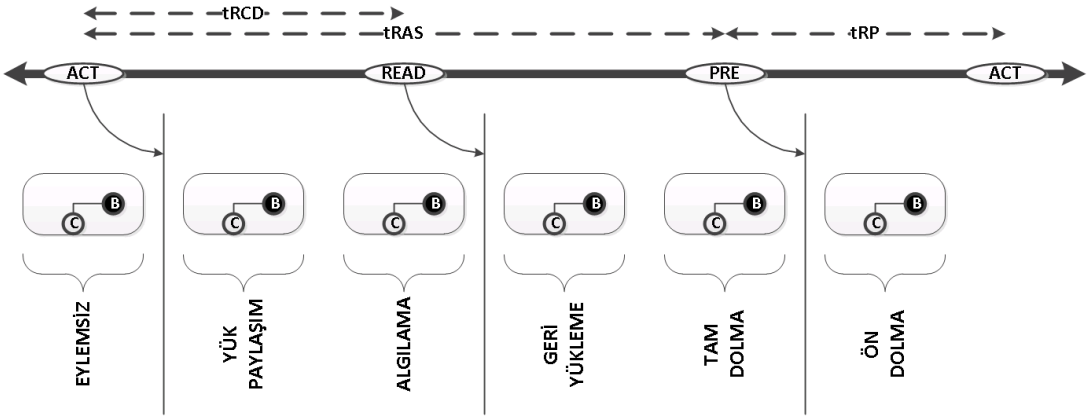
3.3.2 Aktif et sonrası zamansal durumlar

Eylemsiz durumda olan hücreye aktif et buyruğu geldikten sonra hücre bir takım elektriksel durumlardan geçmektedir. Aktif et buyruğuyla birlikte hücrenin yük paylaşımı ve algılama için harcadığı süre tRCD ile gösterilir. Şekil 3-6'da görüldüğü gibi bu süre hücrenin bit hattına bağlandıktan sonra içerisindeki değerin ara dizeden (row buffer) okunmaya elverişli hale gelmesine kadar geçen süredir.

Okuma buyruğunun gelmesiyle birlikte hücre işlem öncesi duruma geçmeye çalışır. Aktif et buyruğu ile tam dolma buyruğu arasında geçen süre tRAS olarak gösterilir.

İşlem tamamlandıktan sonra yeni bir işlem yapılabilmesi için bit hattının ön dolma aşamasından geçmesi gerekir. Ön dolma aşaması, hücre eylemsiz duruma geçene kadar olan süre tRP olarak gösterilir.

tRCD, tRAS ve tRP süreleri "Yedek Dize" yöntemi ile kazanım elde edilen işlem süreleridir.



Şekil 3-6: Aktif et buyruğu sonrasında zamansal akış.

3.4 Ddr3 Donanım Paketleri (Ddr3-1600k)

Her bir DRAM konfigürasyonu için buyruk işlemleri sırasında geçirilen standart süreler mevcuttur. Bu bölümde, testler esnasında kullandığımız DRAM paketi DDR3-1600K olduğu için ilgili DRAM'e ait veriler üzerinden gidilmiştir. JEDEC standardında DDR3-1600K için verilen tRCD, tRAS ve tRP değerleri Çizelge 3-4'de verilmiştir. "Yedek Dize" yöntemi ile bu sürelerde iyileşme sağlanmıştır.

Çizelge 3-4: Ddr3-1600k hız paketli sdram zamanlama tablosu.

Okuma komutundan ilk veriye geçen süre	tAA	13,75	20	ns
ACT komutundan sonra RD/WR komutuna kadar olan gecikme	tRCD	13,75	-	ns
PRE komut periyodu	tRP	13,75	-	ns
ACT komutundan sonra ACT/REF komutu gelme periyodu	tRC	48,75	-	ns
ACT komutundan sonra PRE komutu gelme periyodu	tRAS	35	9*tREFI	ns
Ortalama Saat Periyodu	tCK(Ort)	1,25	<1,5	ns

4. RAMULATOR

Ramulator [5], Carnegie Mellon Üniversitesi tarafından hazırlanan DRAM simülatördür. Bu simülatör endüstriyel ve akademik çalışmalarda DRAM kullanımını ve DRAM'lerin sistem etkilerini görebilmek için tasarlanmıştır. Getirdiği yenilik simülasyon sürelerinin istenildiği ölçüde uzatılabilmesidir. Bir diğer özelliği ise piyasada kullanılan DRAM'lerin çoğunun bu program ile simüle edilebiliyor olmasıdır. Hızlı ve tutarlı simülasyon sonucu alabilmek amacıyla geliştirilen RAMULATOR, simüle ettiği tüm DRAM'ler için standartlara uygun tasarlanmıştır. Bu yazılım, araştırmacıların hedefleri doğrultusunda istedikleri gibi değişiklik yapabileceği bir açık kaynak kod olduğu için sıkça tercih edilen bir benzetim programıdır. Araştırmacılara sunulan modeller DDR3/4 [22, 23], LPDDR3/4 [24, 25], GDDR5 [26], WIO1/2 [27,28] ve HBM [29] modellerine uygun hazırlanmıştır. Ayrıca bir çok güncel çalışmada da bu simülatör kullanılmıştır/kullanılmaktadır.

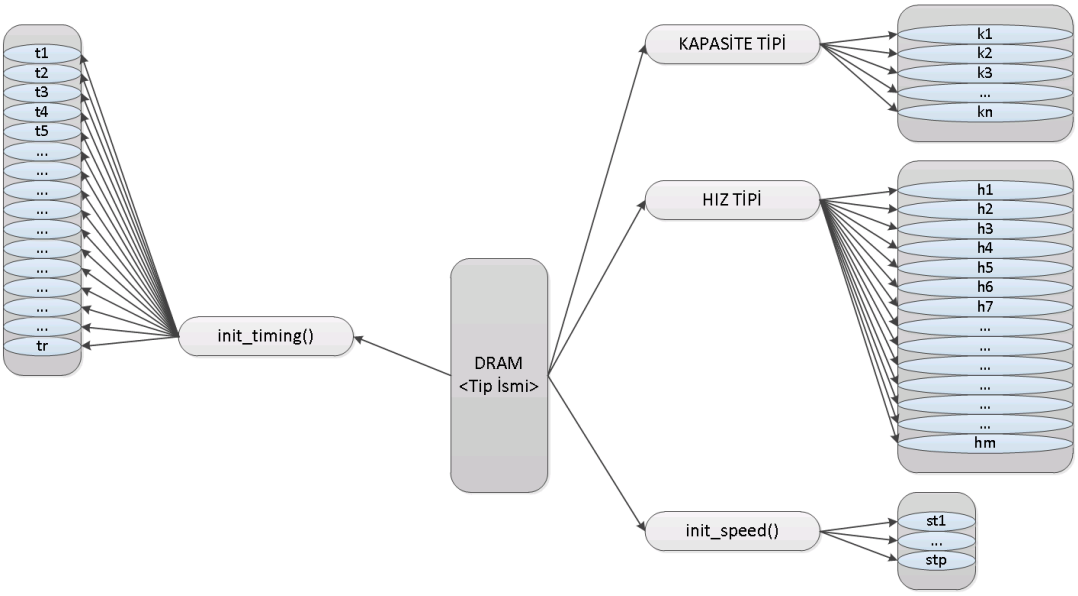
SALP[11], AL-DRAM[10], TL-DRAM[9], RowClone[14] ve SARP [30] yayınları sundukları mekanizmalarda elde ettikleri kazanımları bu simülatörün çıktıları ile göstermişlerdir. Ayrıca sürdürülebilir simülasyon süreleri için simülasyon hızından feragat etmemiştir. Aksine aynı amaçla kullanılan en hızlı simülatörden 2,5 kat daha hızlı çalışabilmektedir. Program girdi olarak piyasada farklı amaçlarla DRAM kullanan birimlerden elde edilen DRAM erişim listelerini yani "Trace" [31,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55] dosyalarını kullanmaktadır.

4.1 Ramulator Çalışma Mekanizması

4.1.1 Dram tipi seçimi ve ağaç yapısı

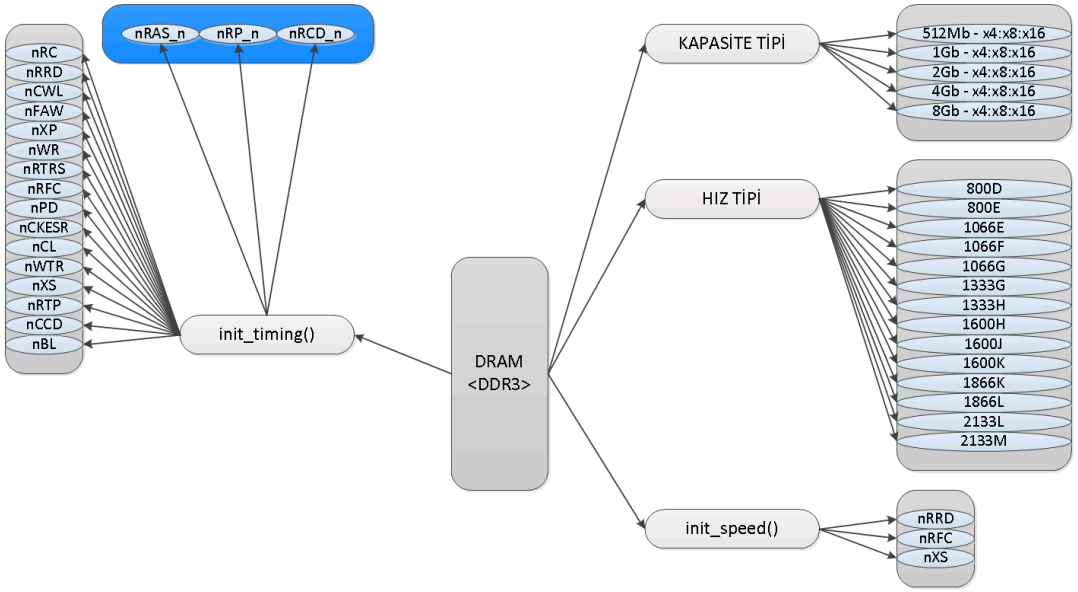
Ramulator çalışma mekanizması temelde DDR3 bellek kullanılarak başlanmış daha sonra diğer tip DRAM'ler için de uyarlanmıştır. Kim, Y. "Ramulator: A Fast and Extensible DRAM Simulator" [32] yayınında Ramulator çalışma mekanizmasını DDR3 üzerinden sunmuştur. Tekrar programlanabilir ve seçilebilir bir ağaç yapısı

kullanılmıştır. Bir diğer deyişle DRAM tipi bellek tanımlanmıştır, fakat kullanıcı tarafından tanımlanan DRAM tipine göre ağaç yapısında çocuk ve ana düğümleri oluşturmaktadır (Şekil 4-1).



Şekil 4-1: Ramulator parametre ağaç yapısı.

Örneğin, DRAM tipi DDR3 seçilen bir simülasyonda kanal, sıra, bank, satır ve kolon olarak tek bir ana düğüm 5 farklı çocuk düğüme ayrılmaktadır (Şekil 4-2). Diğer tip DRAM'ler için de aynı yapı mevcuttur fakat tipten tipe dallanma farklılaşabilmektedir.



Şekil 4-2: Ddr3 özelinde ramulator parametre ağaç yapısı.

Benzetimi yapılacak bellek tipi konfigürasyon dosyası içerisinde belirtilir. Seçilmiş bellek tipi, Ramulator içerisinde simülasyona hazır şekilde ilklenerek sıralı buyrukları işlemeyi bekler. Şekil 4-3'de DDR3 belleği için ilklendirme yapan kod parçası mevcuttur. Bu kod parçasında yer alan init_timing fonksiyonu DDR3'e ait standartta yer alan işlem sürelerini ilklemektedir. "Yedek Dize" yöntemi ile bu ilkleme fonksiyonunda da güncelleme yapılmıştır.

```
DDR3::DDR3(Org org, Speed speed) :
    org_entry(org_table[int(org)]),
    speed_entry(speed_table[int(speed)]),
    read_latency(speed_entry.nCL + speed_entry.nBL)
{
    init_speed();
    init_prereq();
    init_rowhit(); // SAUGATA: added row hit function
    init_rowopen();
    init_lambda();
    init_timing();
}
```

Şekil 4-3: Ddr3 ilkleme kod parçası.

4.1.2 Zamansal sıra oluşturma algoritmaları

Ramulator'un sunduğu bir diğer avantaj da bellek denetim mekanizmasının da simülasyonla sunulmasıdır. Denetim birimi ana düğüm ile irtibat halinde çalışır. Genel olarak yaptığı iş, gelen buyruklar için zamansal bir sıra oluşturmak ve bu buyrukları işlemektir. Ramulator, araştırmacılara denetim biriminin kullanacağı zamansal sıra oluşturma algoritması da seçtirmektedir. Temelde FRFCFS ve FCFS olmak üzere 2 tip sıralama algoritması benzetim yazılımında sunulmuştur. Ramulator bünyesinde yer alan planlama algoritmaları aşağıda özetlenmiştir [33].

- FRFCFS (First Ready, First Come First Served): Önceliği "bulma" durumu gösteren satırın işlenmesi olan, böyle bir durum yoksa da en eski işlem talebini ilk yapma eğilimi gösteren algoritmadır. Bu algoritmanın amacı, ara dize (row buffer) "bulma" oranını yükselterek DRAM performansını arttırmaktır.
- FCFS (First Come First Served): En eski işlem talebini ilk yapma eğilimi gösteren algoritmadır.

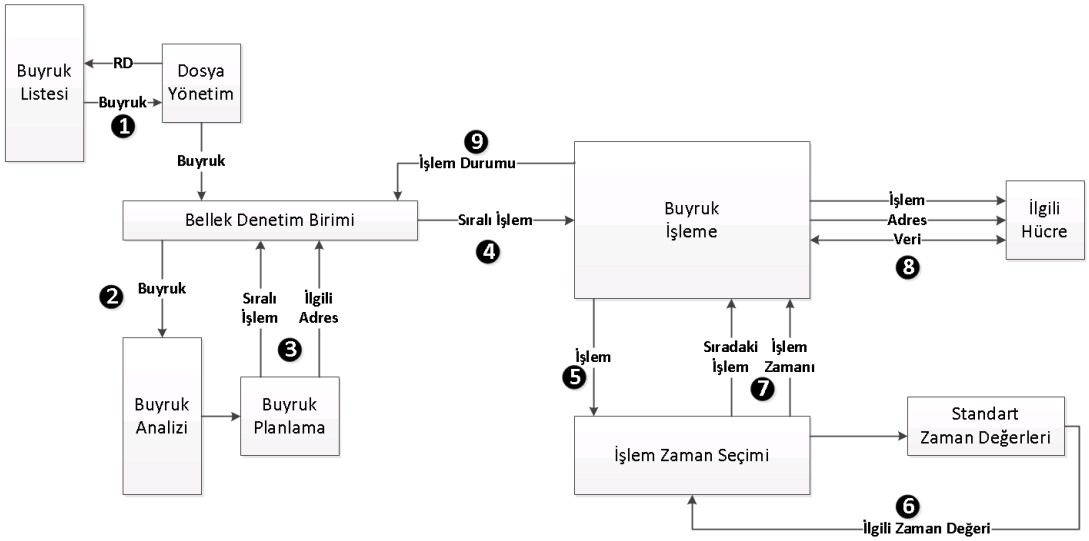
Genel olarak önceliklendirme aşağıdaki hususlara göre yapılmaktadır:

- İşlem talebinin geliş zamanı,
- Ara dizenin (row buffer) "bulma/bulamama" durumu,
- İşlem talep tipi (okuma, yazma, aktif etme vs.),
- Çekirdeklerin meşguliyet durumları.

4.2 "Yedek Dize" Mekanizmasında Ramulator'ın Yeri

Ramulator güncel bir çok yayında yer almıştır. Araştırmacılar, sundukları fikirleri destekleyebilmek için üzerinde test işlemlerini sürdürebilecekleri bir benzetim yazılımına ihtiyaç duymaktadırlar. Ramulator, bu ihtiyacı en iyi şekilde karşılayabilecek yeterliliktedir. Ayrıca kaynak kodu sunmaları sayesinde araştırmacılar için fikirlerini geliştirebilecekleri bir platform haline de gelmiştir. Tezin bu bölümünde ramulator benzetim yazılımının çalışma mekanizması ve yedek dize yöntemini ilgilendiren özellikleri anlatılmıştır.

Ramulator, DRAM bellek tiplerini standartlarda yer alan akış diyagramlarında olduğu gibi çalıştıran bir benzetim yazılımıdır. Bu bölümde ramulator yazılımının çalışma prensibi detaylı bir biçimde anlatılmıştır. Bu anlatılanların yanı sıra tezin yedek dize bölümlerinde güncellenen ya da eklenen mekanizma blokları sunulacaktır. Yedek dize yöntemi ile geliştirilen yeni mekanizma Şekil 4-4'de yer alan 4 numaralı adımdan sonrasındaki akışı etkilemektedir. Şekil 4-4'de ramulator benzetim yazılımının çalışma mekanizması numaralarla sıralandırılmıştır. Ramulator, girdi dosyası içerisindeki buyrukları (Şekil 4-4 adım 1) dosya yönetim bloğu tarafından alır ve bellek denetim birimine aktarır. Bellek denetim birimi girdi dosyasından aldığı buyrukları işlenebilirlik sırasına sokar (Şekil 4-4 adım 2,3). Daha sonra sıralı işlemler buyruk işleme birimine gelir (Şekil 4-4 adım 4). İlgili işlem için zaman değeri (bu zamanlama değerleri JEDEC [18] standardındaki değerlerdir) seçilir (Şekil 4-4 adım 5,6). Seçili zaman değeri ve buyruk tipine göre ilgili hücreden/hücreye okuma/yazma işlemi gerçekleştirilir (Şekil 4-4 adım 7,8). İşlemin tamamlandığı ve yeni işlem için uygunluk durumu bellek denetim birimine bildirilir (Şekil 4-4 adım 9).



Şekil 4-4: Standart ramulator çalışma mekanizması.

1 numaralı adımda bahsedilen dosya yönetim tarafından alınan dosyalar Çizelge 4-1'de verilmiştir. Bu girdi dosyaları (benchmark), akademik alanda ya da günlük hayatta kullanılan algoritmaların çalışmaları esnasında DRAM erişim davranışlarının tutulduğu dosyalardır . Çizelge 4-1'de de görülen bu girdi dosyaları yedek dize mekanizmasını test aşamasında kullanılmıştır. Bu dosyalar, güncel bir çok yayının test ve sonuç aşamalarında kullanılmıştır. Böylece araştırmacılar hem mekanizmalarını test edebilmekte hem de denetimli deney imkanı sayesinde diğer araştırmalarla kendi bulgularını kıyaslayabilmektedir.
[31,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55]

Çizelge 4-1: Ramulator çalışmalarında kullanılan girdi dosyaları.

Benchmark	Alan	Yaptığı İş
401.bzip2	Bilgisayar	Yüksek kalite veri sıkıştırıcı
403.gcc	Bilgisayar	AMD Opteron işlemciler için C dilinde derleyici
429.mcf	Ulaşım	Toplu taşımalarda kullanılan araç planlama mekanizması optimizasyon kodu
433.milc	Fizik	Süperbilgisayarlarda kullanılan kuantum kromodinamiği mekanizması kodu
434.zeusmp	Fizik	Mıknatıssal hidrodinamik hakkında akış dinamiği mekanizması kodu
435.gromacs	Kimya	Moleküler dinamik simülatör kodu
436.cactusADM	Fizik	Einstein izafiyet denklemlerini çözen kod
437.leslie3d	Kimya	Hesaplamalı akışkanlar dinamiği alanında karıştırma, patlama, akustik ve akışkan mekaniği kodu
444.namd	Biyoloji	Büyük biyomolekül yapılar için hazırlanan sistem simülatörü
445.gobmk	Bilgisayar/ Yapay Zeka	Go oynayan yapay zeka
447.dealII	Matematik	Uyumlayıcı sonlu element yönteminde kullanılan kısmi türev denklemlerinde
450.soplex	Bilgisayar/Matematik	Simplex doğrusal program çözücü
456.hmmer	Bilgisayar/Biyoloji	DNA gen araştırmalarında kullanılan bir algoritma
458.sjeng	Bilgisayar/ Yapay Zeka	Satranç ve türevi bir takım oyunları oynayabilen program
459.GemsFDTD	Fizik	3 boyutlu Maxwell denklemleri çözebilen hesaplamalı elektromanyetik mekanizması kodu
462.libquantum	Fizik	Kuantum mekanik işletlerini çözen program
464.h264ref	Medya	Yüksek kalite video sıkıştırıcı
470.lbm	Malzeme Bilimi	3 boyutlu sıkıştırılamayan akışkanlar modellemesi yapan Lattice Boltzmann Method simülatörü
471.omnetpp	Bilgisayar	Ethernet alt yapısı üzerinde ayrık olay simülatörü
473.astar	Bilgisayar/ Yapay Zeka	2 boyutlu haritada yol bulan yapay zeka algoritması
481.wrf	Meteoroloji	Hava tahmini yapan simülatör
482.sphinx3	Medya	Konuşma tanıma algoritması
483.xalancbmk	Bilgisayar	XML formatını HTML, metin, ve diğer XML formatlarına dönüştüren çevirici

Ramulator çalışma mekanizmasında 4 numaralı aşamadan sonra başlayan buyruk işlem fonksiyonu Şekil 4-5'de verilmiştir. Bu fonksiyon, aldığı adres vektör bilgisi üzerinde aldığı buyruğu işlemektedir. Standartta belirlenmiş süreler içerisinde işlemleri tamamlar ve sıradaki işleme geçiş yapar. Yedek dize yöntemi kapsamında bu fonksiyon içerisine BULMA/BULAMAMA mekanizması ve yedek dize adreslerinin tutulduğu bir doğruluk tablosu eklenmiştir.

```
void issue_cmd(typename T::Command cmd, const vector<int>& addr_vec)
{
    assert(is_ready(cmd, addr_vec));
    channel->update(cmd, addr_vec.data(), clk);
    rowtable->update(cmd, addr_vec, clk);
    if (record_cmd_trace){
        // select rank
        auto& file = cmd_trace_files[addr_vec[1]];
        string& cmd_name = channel->spec->command_name[int(cmd)];
        file<<clk<<','<<cmd_name;
        // TODO bad coding here
        if (cmd_name == "PREA" || cmd_name == "REF")
            file<<endl;
        else{
            int bank_id = addr_vec[int(T::Level::Bank)];
            if (channel->spec->standard_name == "DDR4" ...
             || channel->spec->standard_name == "GDDR5")
                bank_id += addr_vec[int(T::Level::Bank) - 1] * ...
                channel->spec->org_entry.count[int(T::Level::Bank)];
            file<<','<<bank_id<<endl;
        }
    }
    if (print_cmd_trace){
        printf("%5s %10ld:", channel->spec->command_name[int(cmd)].c_str(), clk);
        for (int lev = 0; lev < int(T::Level::MAX); lev++)
            printf(" %5d", addr_vec[lev]);
        printf("\n");
    }
}
```

Şekil 4-5: Ramulator buyruk işleme birimi kod parçası.

Tez kapsamında sunulan yedek dize yöntemi birim zamanda işlenen işlem sayısını arttırmaya yönelik bir çalışmadır. Bu yöntemle zamansal değişkenler konusunda iyileşme hedeflenmektedir. Ramulator içerisinde bellek tipine göre standartta tanımlı erişim süreleri tablo halinde mevcuttur. Buyruk işleme aşamasında Şekil 4-4'de yer alan 6 numaralı adımda ilgili işlem için standartta yer alan süreler buyruk işleme birimine aktarılır. Bu süre, kaç saat vuruşu boyunca bu işlemde kalınacağını göstermektedir.

5. YEDEK DİZE YÖNTEMİ

Ana bellek, elektronik sistemlerin olmazsa olmaz parçalarından biridir. Bellek birimlerinin gelişimi ve etkin kullanımı için bellek kapasitesi, bellek idaresi, işlem hızları gibi konular üzerine bir çok güncel çalışma mevcuttur [9,10,11,12,13,14]. Bilgisayar teknolojisinin ilerleyebilmesi için bilgisayar bileşenlerinin her birinin yakın oranlarla gelişim sağlaması gerekmektedir. İşlemci hızlarının artmasına karşı bellek erişim hızlarının neredeyse sabit kalması, ilerleyişin bir süre sonra olumsuz etkilenmesine neden olabilir[7,8]. DRAM kullanımının giderek yaygınlaşmasından ötürü DRAM işlem gecikmelerinin azaltılması güncel ve önemli konular arasına girmiştir. Bellek odaklı çalışmalar yapan bir çok grup bu alanla ilgili öngörülerini çalışmalara bir yol haritası belirleyebilmek için paylaşmaktadır [3]. Araştırmacılar temel bir gaye üzerine farklı yaklaşımlarla bellek konusunda olumlu getiriler elde etmektedirler.

Bu çalışma kapsamında sunulan "Yedek Dize" yöntemi de bellek birimlerinin gelişimi üzerine yapılmış bir çalışmadır. “Yedek Dize” mekanizması, DRAM işlem gecikme sürelerini azaltmayı hedeflemektedir. Bu çalışma kapsamında hem bellek denetim biriminin çalışma mekanizmasının hem de bellek donanım mimarisinin değişmesi ile daha iyi neticeler elde edilebileceği gösterilmiştir.

Tezin bu bölümünde fikrin çıkış noktası hakkında bilgiler, "Yedek Dize" yönteminin çalışma mantığı, donanım mimarisine olan etkisi, bellek işlemlerini üzerindeki olumlu etkisi ve Ramulator benzetim yazılımında yapılan çalışmalar anlatılmaktadır.

5.1 Yedek Dize Yaklaşımı

Çalışmamızın temel amacı DRAM işlem sürelerinin azaltılmasıdır. Bu amaç doğrultusunda güncel çalışmalar incelenmiştir, araştırmaların genelinde eğilimin "Aktive Etme" buyruğundan sonra "Okuma/Yazma" buyruğuna daha hızlı geçiş sağlayabilecek ya da "Okuma/Yazma" sonrasında durum makinasının "Eylemsiz" durumuna daha hızlı ulaşabilecek yöntemler olduğu saptanmıştır. Bu tez kapsamında

bahsedilen bu iki kazanımı elde edebilecek yeni bir mekanizma olan "Yedek Dize" yöntemi tasarımı gerçeklenmiştir. DRAM işlem sürelerinin hızlanabilmesi için ya donanımsal ya da yazılımsal değişiklikler yapmak gerekebilir. Yedek dize yönteminde hem donanım hem yazılım değişikliği ile iyileşme elde edilmiştir.

Bir hattan akan yükün hızı, hattın ve yük kaynağının kapasitansıyla ilişkilidir. Kapasitansların değişmesi hız değişimine neden olabilir. DRAM belleklerde veri hızını arttırabilmek için her hattın ya da hücre kapasitansının değiştirilmesi gereklidir. Bit hatlarının kısaltılması ile bit hattı kapasitansının değiştirilebileceği ve DRAM işlem gecikmelerinde iyileşme sağlanabileceği "TL-DRAM" yayınında sunulmuştur [9]. Bu tez çalışması kapsamında bit hattının kapasitansının sabit kaldığı varsayılarak, hücre kapasitansını değiştirmeye yönelik bir yöntem geliştirilmiştir. Akan yük miktarının bit hattı ve hücre kapasitansıyla ilişkisi Denklem (5.1)'de verilmiştir [17]. Denkleme göre besleme gücü ve bit hattı kapasitansı sabit düşünüldüğünde akan birim yükü arttırabilmek için hücre kapasitansını arttırmak gerekmektedir. Bu amaç doğrultusunda "Yedek Dize" yöntemi ile hücre kapasitansının arttırılması hedeflenmiştir.

$$\Delta V = \frac{\mathrm{Vdd}}{2}\frac{Ccell}{Ccell+Cbit} = \frac{\mathrm{Vdd}/2}{1+\mathrm{Cbit}/\mathrm{Ccell}} \tag{5.1}$$

Bellek mekanizması göz önüne alındığında bit hattından akan yükün artması, algı yükselteçlerin DRAM hücrelerindeki veriyi daha hızlı saptamasını sağlayacaktır [13]. Hücre kapasitansı silikon düzeyde yapılabilecek yarı iletken malzeme değişikliğiyle sağlanabileceği gibi donanımda bu amaçla rezerve kullanılacak alanlarla da sağlanabilir. Aynı veriye sahip iki hattın aynı anda "Aktive Etme" durumuna geçmesi ile bit hattına bağlanan hücre kapasitansı arttırmış olacaktır. Bu yöntemle, Denklem (5.1)'e göre birim zamanda bit hattı üzerinden akan yük miktarı artacaktır.

Matematik, fizik, kimya, bilgisayar vb. alanlarda geliştirilen yazılımların DRAM erişim kayıtları Bölüm 4.2'de "Yedek Dize" Mekanizmasında Ramulator'ın Yeri başlığı altında verilmiştir. Bu dosyalar analiz edildiğinde ardışık buyrukların kabul edilebilir seviyede zamansal yerellik özelliği gösterdiği saptanmıştır. İlgili analize ait sonuçlar tezin Bölüm 6.2 Zamansal Yerellik başlığı altında sunulmuştur.

Bit hattına bağlı hücre kapasitansının artırılabilmesi için birden çok hücrenin aynı anda bit hattına bağlanması gerekmektedir. Birden çok hücrenin bit hattına aynı anda anahtarlanabilmesi için sakladıkları verilerin aynı olduğu garanti edilmelidir. Yedek dize yöntemi ile zamansal yerellik özelliği gösteren adresin verilerini yedek dizede saklamak ve hedef dize ile birlikte yedek dizeyi bit hattına bağlamak "Yedek Dize" mekanizmasının temel özelliğidir.

5.2 Yedek Dize Yöntemi

5.2.1 Yedek dize yöntemi çalışma mekanizması

"Yedek Dize" yöntemi gerçekleme aşamasında, JEDEC standardında verilen tipik DRAM akış mekanizması üzerine değişiklikler yapılmıştır. Belirlenen yeni çalışma mekanizması genel hatlarıyla Şekil 5-1'de verilmiştir. Bu mekanizma Şekil 4-4'de gösterilen ramulator çalışma mekanizması ile 4. adımdan sonra farklılaşmaktadır.

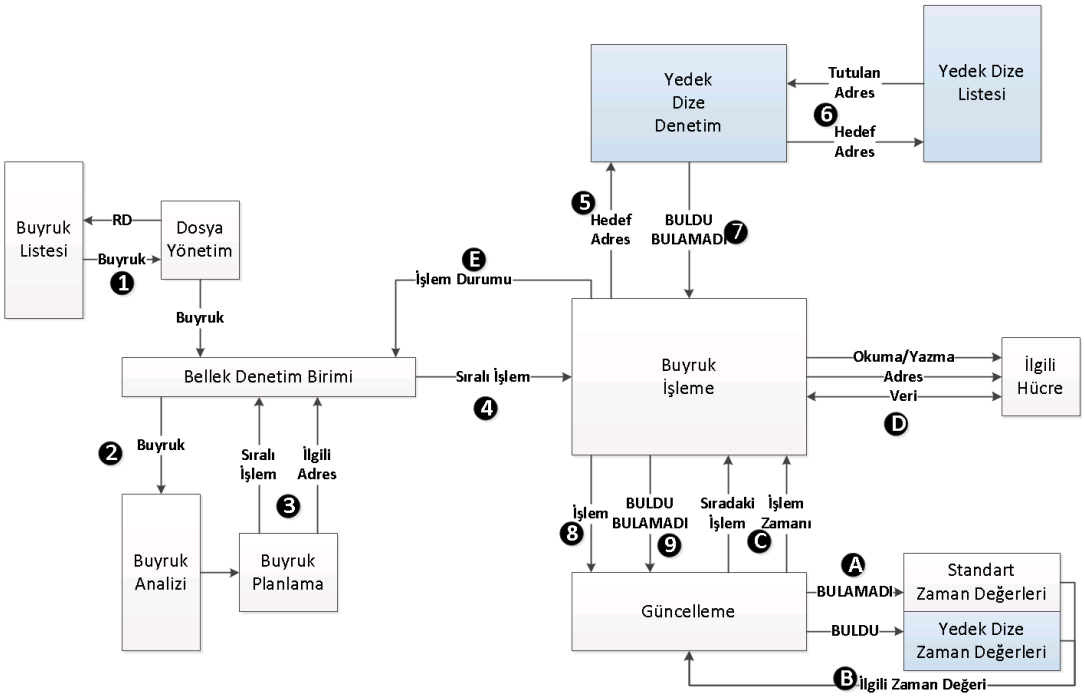
"Yedek Dize" mekanizması, normal DRAM çalışma mekanizmasından farklıdır. Normal çalışma şeklinden bir farkı yapıda fazladan denetlenmesi gereken yedek bir satırın var oluşudur. Yedek dize yönteminde zamansal yerellik özelliği ile yakın zamanda işlenmesi beklenen adresi tutar. Her bir alt dize için bir adet yedek dize tutulmaktadır. Bu iş için her 512 satırdan 1 tanesi rezerve edilmiştir.

Girdi dosyası içerisinde yer alan buyruk listesi, bellek denetim biriminin çalıştırdığı dosya yönetim modülü ile sırayla okunur (Şekil 5-1 adım 1). Belirli sayıda buyruk okunduktan sonra analiz ve planlama için buyruk analiz modülüne okunan buyruklar aktarılır (Şekil 5-1 adım 2). Buyruk analiz modülü, okuma yazma tipine göre ve adres bilgilerine göre buyruğu parçalara ayırır ve buyruk planlama modülüne aktarır. Buyruk planlama modülü seçili planlama çözüm yöntemine göre buyrukları sıralar. Sırayla işlenecek işlemleri ve adresleri bellek denetim birimine iletir (Şekil 5-1 adım 3). Bellek denetim birimi sıralı gelen işlemleri buyruk işleme modülü ile işler (Şekil 5-1 adım 4). Bu modül işleyeceği işlemi sırayla alır. Bu aşamadan sonra yedek dize mekanizması standarttan farklılık gösterir. Aldığı sıralı işlemleri yakın geçmişte işlemiş ihtimalini sorgulamak adına hedef adresi yedek dize denetim birimine aktarır (Şekil 5-1 adım 5).

Yedek dize denetim bloğu içerisine, buyruk işlem modülüne sıralı gelen işlemlerin adreslerinin yedek dizedeki verinin adresiyle aynı olup olmadığını denetleyen bir

mekanizma eklenmiştir. Eğer hedef adres ile yedek dizede verisi tutulan adres aynı ise BULMA olarak işaretlenir. Eğer hedef adres ile yedek dizede verisi tutulan adres farklı ise bu durum BULAMAMA olarak adlandırılır (Şekil 5-1 adım 6,7).

Karar mekanizması sonucu "BULMA" ise hedef ve yedek satırlar eş zamanlı olarak "Aktive Etme" durumuna geçer (Şekil 5-1 adım 8,9). Zamanlama değeri olarak "Multiple Clone Row DRAM" [14] yayınında aynı anda aktif hale gelen iki satır için hesaplanmış süreler kullanılmaktadır (Şekil 5-1 adım A,B). İlgili işlem tamamlandığında zamansal yerellik özelliğine göre o an işlenen veri tekrar yedek dizeye kopyalanmaktadır (Şekil 5-1 adım C,D).

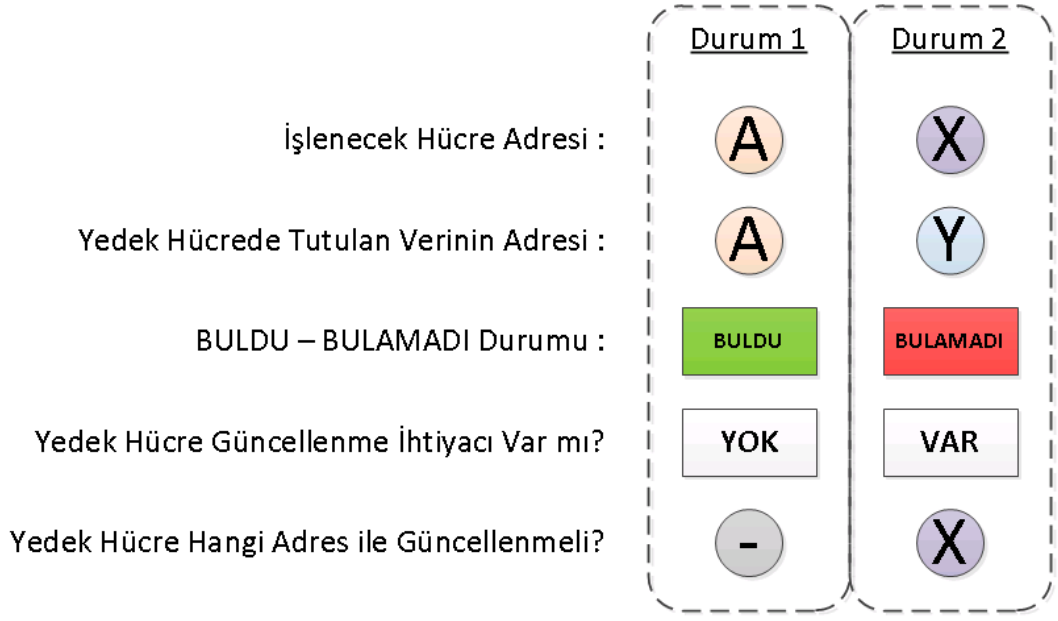


Şekil 5-1: Yedek dize dahil edilmiş ramulator çalışma mekanizması.

BULAMAMA durumunda sadece o an işlenecek adres "Aktive Etme" durumuna geçer, yedek dize "Eylemsiz" durumda kalır (Şekil 5-1 adım 8,9). Zaman parametresi olarak JEDEC standardında verilen süreler kullanılır (Şekil 5-1 adım A,B). İlgili işlem tamamlandıktan sonra o an işlenen adreste tutulan veri yedek dizeye kopyalanır, daha sonra her iki satır kapanarak "Eylemsiz" duruma geçerler (Şekil 5-1 adım C,D).

Son aşama olarak işlemin tamamlandığı bilgisi bellek denetim birimine iletilir ve sıradaki adres işlenmek üzere beklenir (Şekil 5-1 adım E).

BULMA ya da BULAMAMA mekanizması “Yedek Dize” yönteminin karar mekanizmasıdır. Karar mekanizmasının daha net anlaşılabilmesi için Şekil 5-2’de yer alan iki durum paylaşılmıştır. Şekil 5-2’de yer alan “Durum 1” koşulunda, işlenecek hedef hücre adresi “A” olarak gelmektedir. Bu aşamada, yedek dize denetim birimi “A” adresinin yedek dize satırındaki adresle aynı olup olmadığı kararını vermesi gerekmektedir. Tuttuğu adres tablosunu denetleyerek ilgili alt dizede yer alan yedek dizede hangi adresin verilerinin tutulduğuna bakar. “Durum 1”de yedek dizede de tutulan adres “A”dır. Hedef ve yedek dizelerde aynı verilerin tutulduğunu doğrulayan yedek dize denetim birimi karar mekanizması sonucu olarak “BULMA” döner. Her iki dizede birbirinin aynısı olduğu için yedek dizeyi güncelleme ihtiyacı yoktur. “Durum 2”de ise hedef adres “X” olarak yedek dize denetim birimine aktarılmıştır. Yedek dizede tutulan verinin “Y” adresine ait olduğunu tespit eden yedek dize denetim birimi karar mekanizması sonucunu “BULAMAMA” olarak döner. Zamansal yerellik gözlemi nedeniyle yakın gelecekte “X” adresinin gelme olasılığının yüksek olmasından dolayı yedek dize “X” adresinde yer alan veri ile güncellenmelidir.



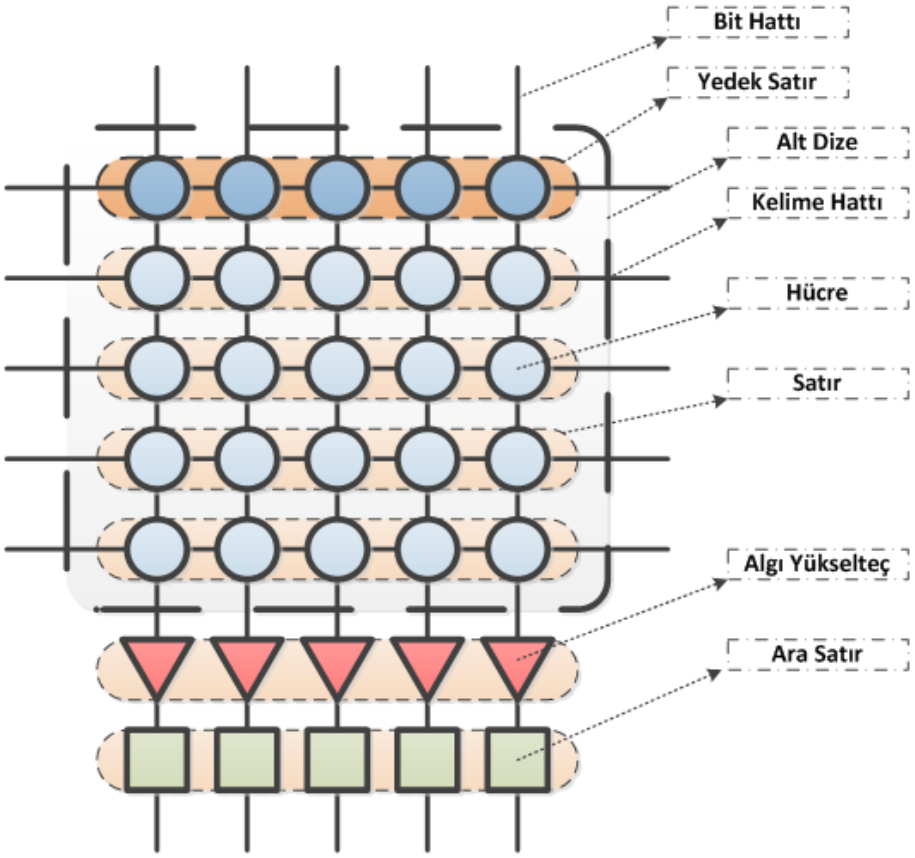
Şekil 5-2: Bulma/bulamama mekanizması.

5.2.2 Donanım mimarisi içerisinde yedek dize

Yedek dize yönteminin bank içerisindeki alt dize (subarray) seviyesine kadar donanımsal bir etkisi görülmemektedir. Şekil 5-3'de görüldüğü gibi DRAM

içerisinde her bir alt dizede bir yedek dize ayrılmıştır. Bir alt dize içerisinde klasik DRAM donanımında bir kelime hattına bağlı olarak kullanılan sıradan bir dize bu yöntem için rezerve edilmiştir. Hücre seviyesinde yedek dize ile sıradan bir satırın farkı yoktur. İşlenecek buyruk geldiği zaman, hedef dize ile yedek dizede tutulan adresin hedef adres olup olmadığı denetlenir.

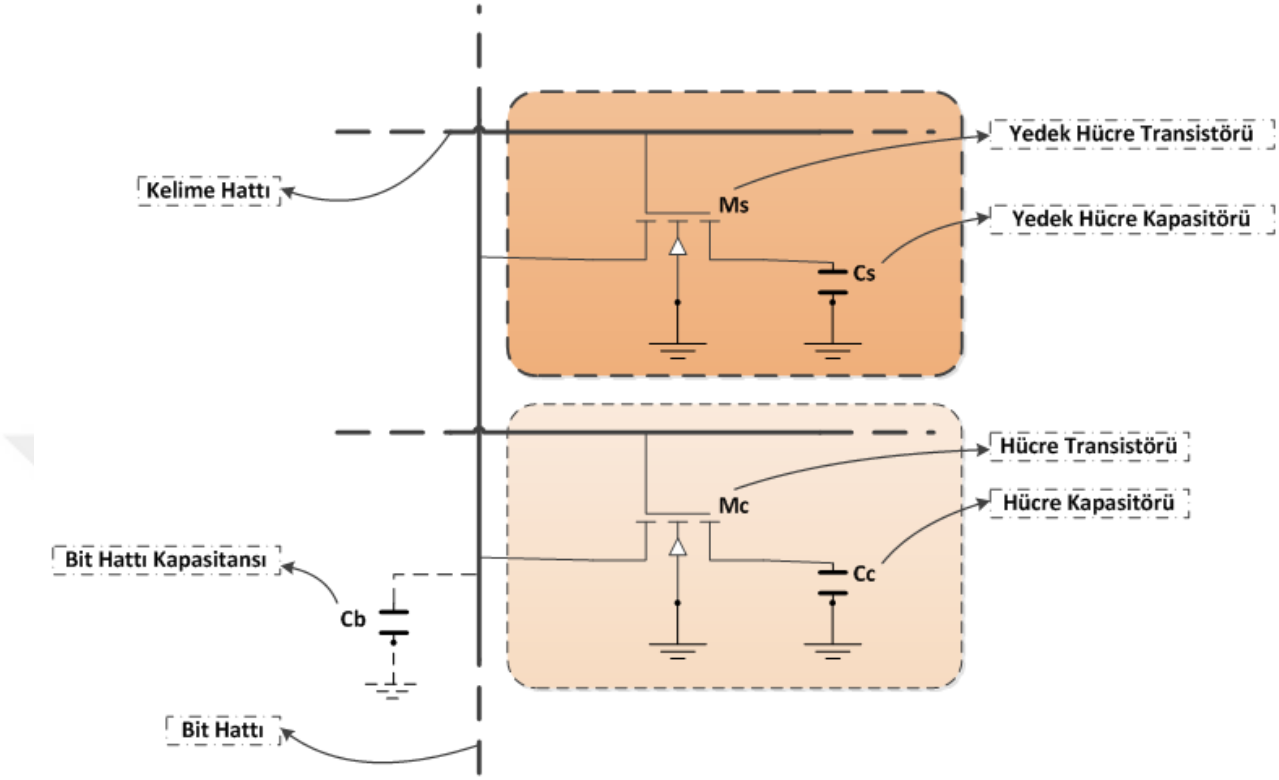
Eğer adresler aynı ise Şekil 5-4'de görünen yedek dize hücresi ve hedef hücre aynı bit hattına bağlanır. Böylece bölüm 5-1 Yedek dize yaklaşımı başlığı altında anlatılan kapasitans artışına bağlı yük akış hızının değişmesi sağlanır. Hücrelerdeki yükün bit hattına daha hızlı akması, bit hattına bağlı algı yükseltecin daha hızlı veriyi algılamasını sağlayacaktır. Algı yükselteç çıkışlarından kullanıcı arayüzüne bağlı ara satıra veri daha hızlı aktarılacaktır.



Şekil 5-3: Yedek dize mimarisi.

Yedek dize sayısı, DRAM'deki alt dize sayısıyla doğru orantılıdır. Genelde banklar 512 satırlı alt dizelerden oluşmaktadırlar [11]. Bu nedenle her 512 satırdan 1 tanesi yedek dize olarak kullanılmaktadır. Veri saklanabilecek bir dizenin yedek dize olarak kullanılması, DRAM bellek kapasitesinde kayba neden olacak gibi görülse de

çıktılar incelendiğinde zamansal yerellik özelliği ile bu alan kaybının yürütme zamanı olarak kazanca dönüştüğü görülmektedir.



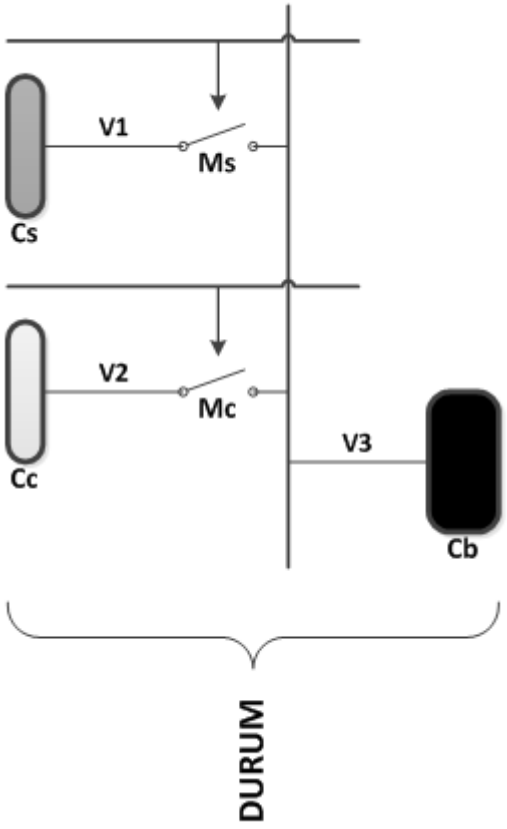
Şekil 5-4: Yedek dize ile dram veri hücresi yapısı.

5.2.3 Buyruk işleme sırasında yedek dize

Standarta uygun buyruk işlem sıraları tezin JEDEC bölümünde anlatılmıştır. Yedek dize yönteminin, standart akışta yer alan "Eylemsiz", "Aktive Etme", "Bank Aktifleme", "Okuma", "Yazma" ve "Ön Dolma" aşamalarına etki ettiği sonuçlarda gözlenmiştir. Durum makinasının diğer tüm durumları standarttaki ile aynıdır.

Karar mekanizması sonucu “BULMA” için buyruk işlem aşamaları bu bölümde anlatılmıştır. Eğer iki dizedeki veriler birbirlerinden farklı ise buyruk işlemleri JEDEC’de de anlatıldığı şekilde standartta uygun ilerlemektedir. Fakat standarttan farklı olarak işlem sonunda yedek dizede tutulan veriyi o an işlenen hedef dize verisiyle güncellemektedir. Durum makinasında durum geçişleri ve geçiş şartları standartla aynıdır. Bu nedenle Şekil 3-1’de verilen durum makinası BULAMAMA/BULMA kararları için aynı kalmıştır.

Hücrelerin işlemler esnasındaki durumları Şekil 5-5'deki gibi simgeleştirilmiştir. Bu şekilde Cs, Cc ve Cb ilgili hattın kapasitans değerlerini simgelemektedir. Ms ve Mc ise yedek ve hedef hücreyi bit hatlarına anahatarlayan transistörlerdir. Hatlar üzerindeki yük miktarı ise V1, V2 ve V3 olarak gösterilmiştir. Yedek dizeye ait bir hücre Cs kapasitansa bağlıdır, bu hücreyi Ms anahtarı bit hattına bağlamaktadır ve kapasitör çıkışındaki anlık yük V1'dir. Hedef dizeye ait kapasitör değeri Cc, anahtar Mc ve anlık yük miktarı V2'dir. Cb ise bit hattının kapasitansını simgelemektedir. Hat üzerinde herhangi bir kapasitör olmamasına rağmen, hat üzerinde yük tutabilen bir tel olduğu için küçük bir kapasitans etkisi yaratmaktadır. Bit hattı kapasitansını anahtarlayan bir transistör mevcut değildir. Bit hattı üzerindeki anlık yük ise V3 ile gösterilmiştir.

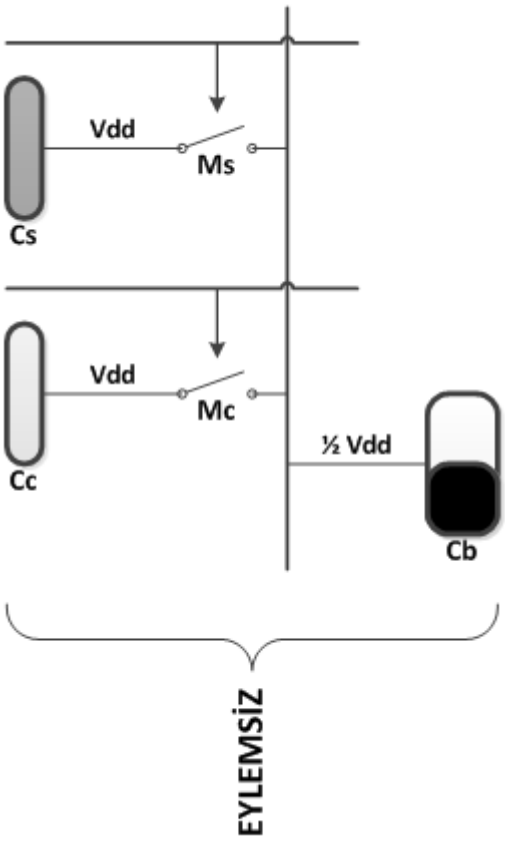


Şekil 5-5: Yedek dize mekanizmasında hücre elektriksel durumu.

5.2.3.1 Buyruk işlem akışı

Bu bölümde "BULMA" kararına sahip bir durum için elektriksel akış diyagramı gösterilmiştir. Eylemsiz durumda bekleyen hücreye okuma talebi gelmesi ve bu hedef dizenin yedek dize mekanizması ile işlenmesi aşağıdaki gibidir.

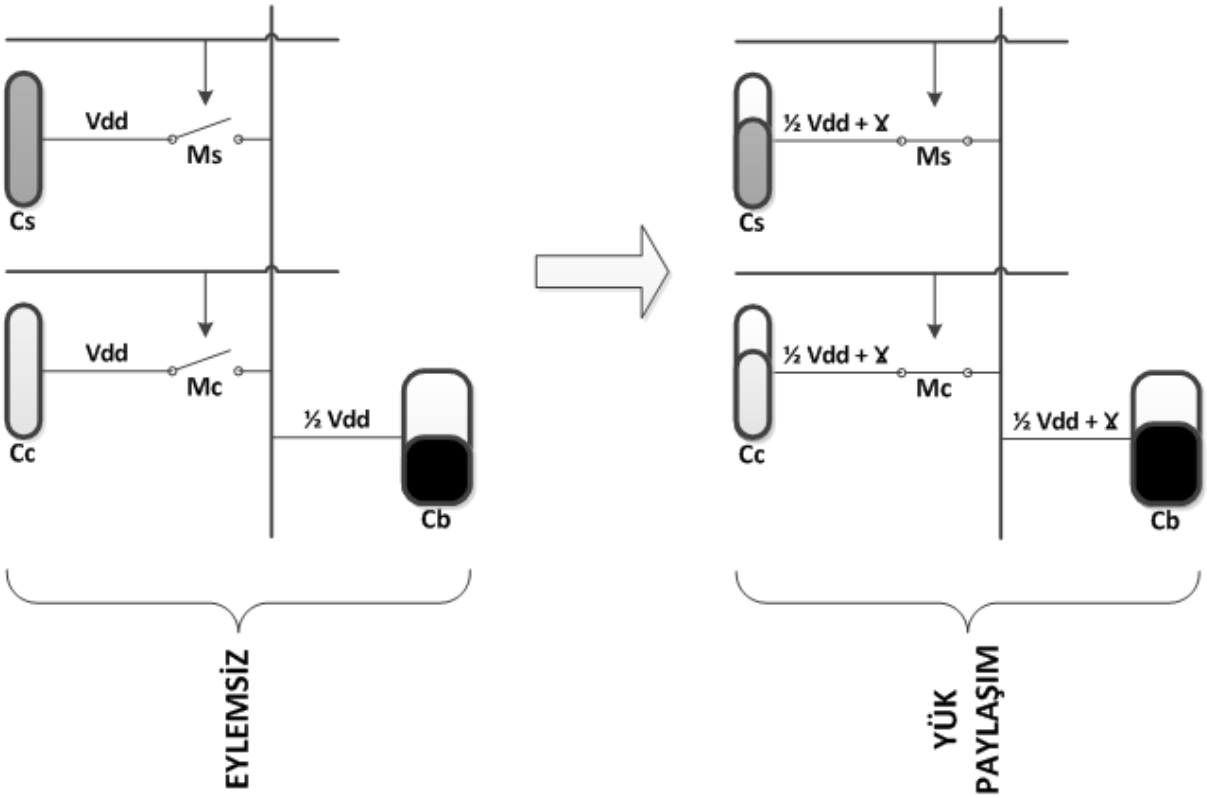
Eylemsiz: Şekil 5-6'da görüldüğü gibi hem yedek hem de hedef dizeye ait kelime hattı aktif halde değildir. Ms ve Mc anahtarladıkları kapasitörleri hatta bağlamamış durumdadır. Yedek ve hedef hücre kapasitörleri tamamen doludur, bit hattının yük seviyesi besleme voltajının yarısı kadardır. DRAM bu duruma ya başlangıç durumundan sonra herhangi bir buyruğunun gelmesi esnasında ulaşır ya da herhangi bir işlemi tamamladıktan sonra ön dolum durumunu tamamladığında ulaşır. "Eylemsiz" durumundan çıkabilmesi için "ACTIVATE" buyruğu alması gerekmektedir, aksi takdirde hep bu durumda kalacaktır.



Şekil 5-6: Yedek dize eylemsiz durumu.

Yük Paylaşım: "ACTIVATE" buyruğunu alan bellek denetim birimi, hedef adresin bağlı olduğu alt dizede yedek dize olarak ne tutulduğuna bakmaktadır. Alt dizeler listesinde ilgili alt dizede tutulan adres ile hedef adres aynı ise BULMA değilse BULAMAMA olarak buyruk işlem modülüne geri bildirim yapar. Buyruk işlem birimi BULAMAMA durumunda yük paylaşımı aşamasını standarttaki gibi gerçekleştirmektedir. Eğer BULMA durumu söz konusu ise hem yedek hem de hedef hücrelere bağlı transistörler bit hattına hücreleri bağlar. Şekil 5-7'de "Eylemsiz"

durumundan BULMA kararına sahip “Yük Paylaşımı” durum geçişi verilmiştir. "Yük Paylaşımı" aşamasında bir hattına anahtarlanan hücrelerin içerisindeki yükler bit hattına akmaya başladığı görülmektedir. Her bir hücreden akan yük hızı standart olsa da aynı anda iki hücreden akan yük, birim zamanda bit hattına akan yük miktarını arttıracaktır (Denklem 5.1). Bu nedenle birim zamanda algı yükselteç üzerine akan birim yük miktarı da artacaktır.

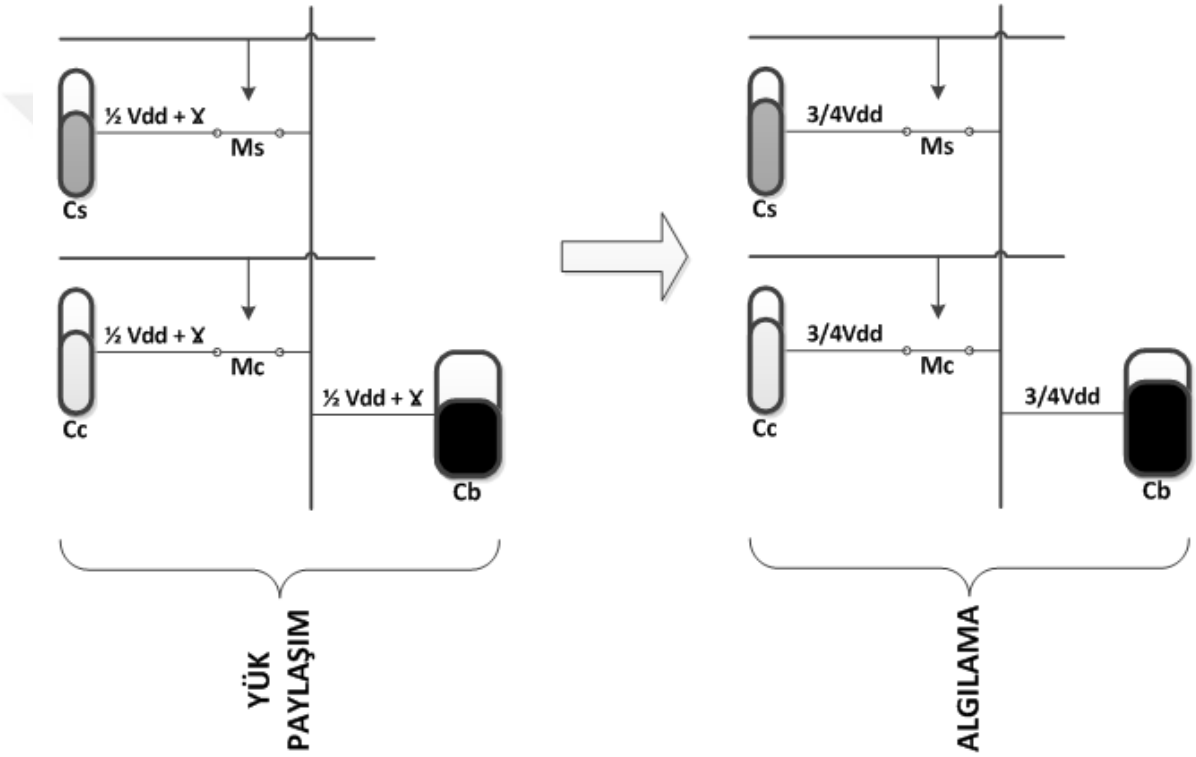


Şekil 5-7: Yedek dize eylemsiz durumdan yük paylaşımına geçiş durumu.

Algılama: "ACTIVATE" sonrası hücreler içerisindeki yükler bit hattına bağlı algı yükselteçlere akmaya başlar. Algı yükselteçler, bit hatları üzerindeki yük seviyesine bakar. Algı yükselteçler, belirli bir eşik değere sahiptirler. Aktif hücrelerden akan yükün bu eşik değere ulaşmasını beklenir. Eğer bit hattı üzerindeki yük eşik değere ulaşırsa algı yükselteç çıkışına "1" ya da "0" sürmeye başlar, buna "Algılama" durumu denilir (Şekil 5-8).

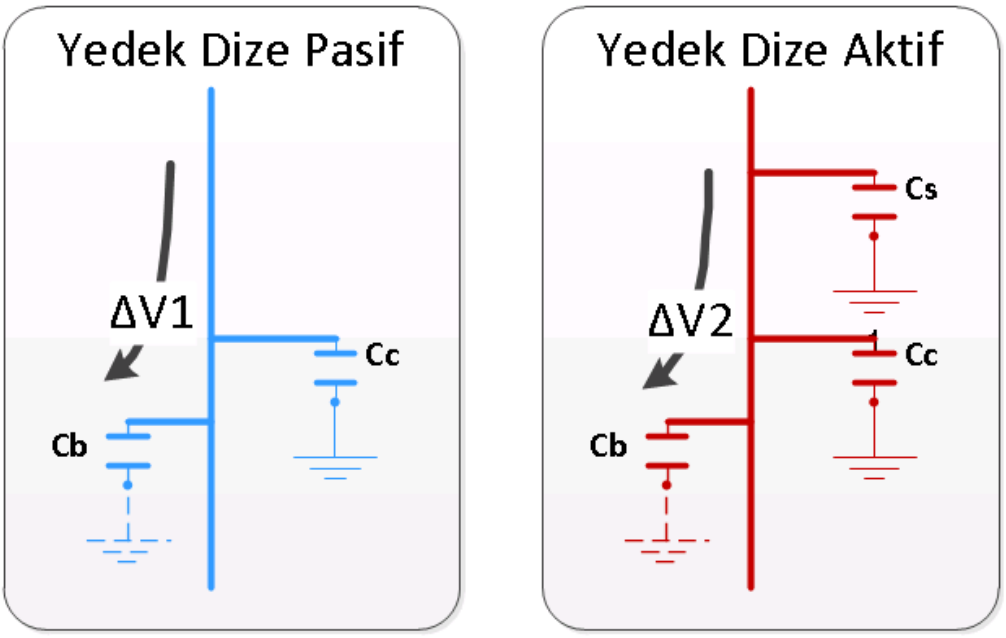
Bit hattının pozitif yönde eşik değeri geçmesi ile algı yükselteç çıkışı "1", negatif yönde eşik değeri geçmesi ile çıkış "0" gözlenir. Şekil 5-10'da hücre içerisindeki yükün "1" olduğu varsayılmıştır. Algı seviye grafiğinde iki farklı senaryo söz konusudur. Bu iki durum Şekil 5-9’da “Yedek Dize Aktif” ve “Yedek Dize Pasif”

olarak gösterilmiştir. Şekil 5-10'da yedek dize yöntemi aktif (kırmızı) ya da pasif (mavi) durumları için bit hattı yük miktarı ve algı yükselteç çıkışındaki yük seviyesi gösterilmiştir. Pasif senaryoda bit hattına akan yük miktarı algı yükseltecin eşik değerine t2 anında ulaşmaktadır. t2 anına kadar besleme voltajının yarısı kadar olan algı yükselteç çıkışı, t2 anından hemen sonra veri "1" yönünde çıkış vermeye başlar. Aktif senaryoda ise hem hedef hem yedek hücre aktif hale geldiği için bit hattına akan yükün hızı normal senaryodan daha hızlı olacaktır. Bu nedenle eşik değere daha kısa sürede erişecektir. t1 anında eşik değere ulaşan algı yükselteç girişi, o andan itibaren veri "1" yönünde çıkış vermeye başlar.

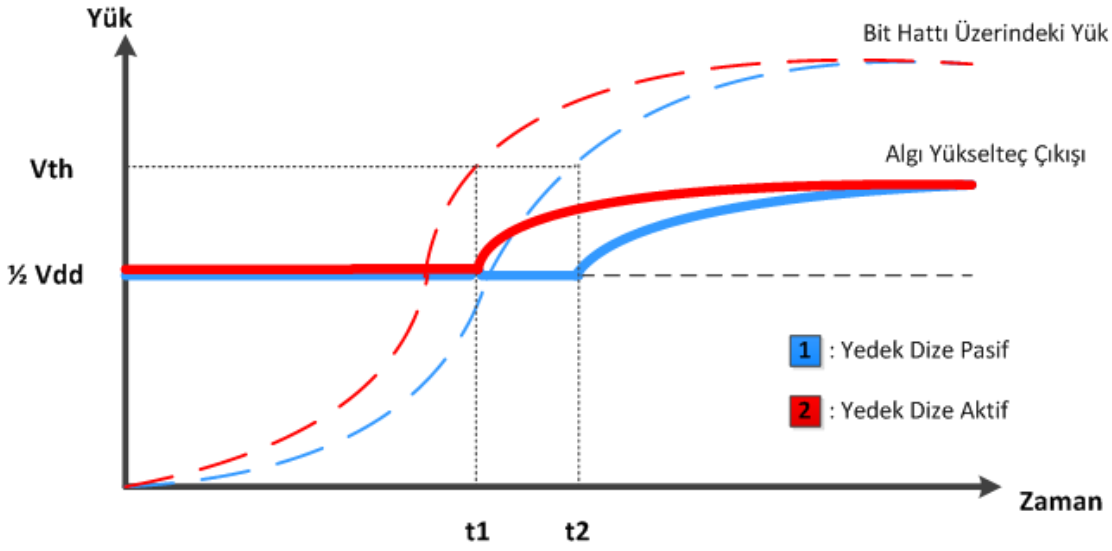


Şekil 5-8: Yedek dize yük paylaşım durumundan algılama durumuna geçiş.

"Algılama" durumu esnasında bit hattına anahtarlanan hücrelerin içerisindeki yük, bit hattına aktığı için içerisindeki veri kaybolmuş olur. Belleklerin temel özelliklerinden olan verinin tekrar tekrar kullanılması önemlidir. Bu nedenle hücre içerisine işlem öncesindeki yük seviyesinin tekrar yüklenmesi gerekmektedir. DRAM bu işlemi "Geri Yükleme" aşamasında yapmaktadır.



Şekil 5-9: Standart ve yedek dize hücrelerinden yük akışı.

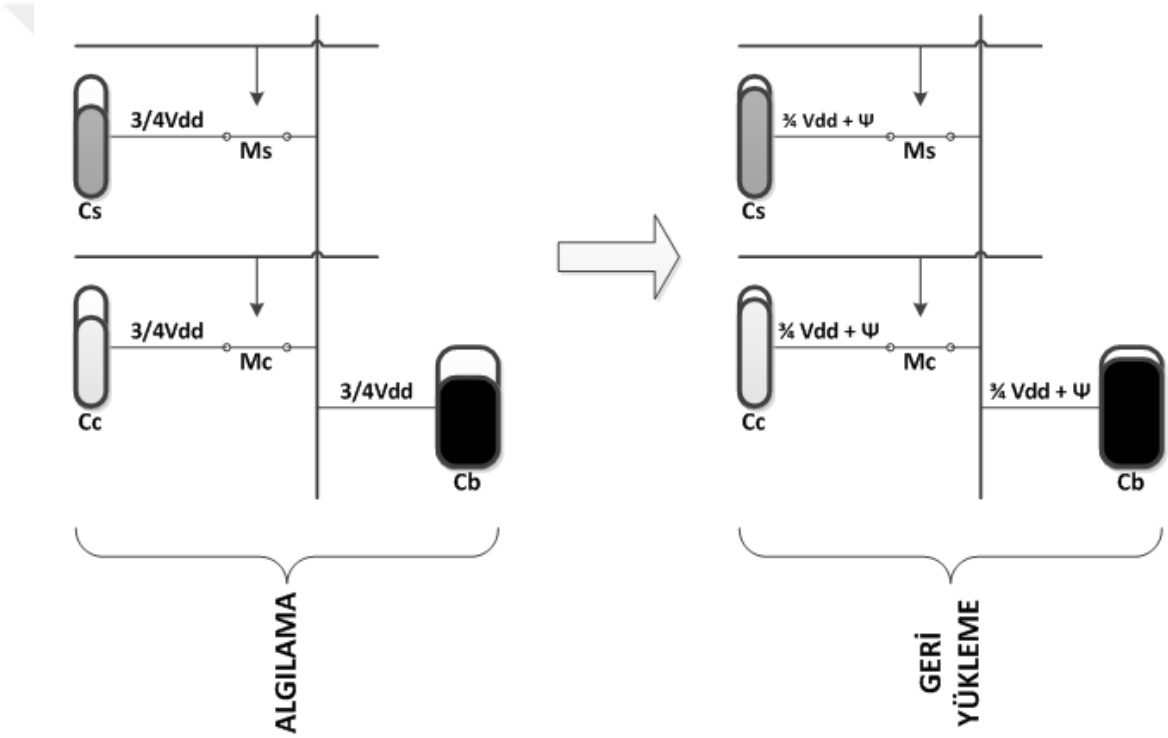


Şekil 5-10: Yedek dize zamana bağlı algılama seviyesi grafiği.

Geri Yükleme: Hücrelerden bit hattına yük akışı sonrası, hücreler ilk durumdaki verilerini kaybetmiş olur. Eğer ilk durumdaki veri tekrar hücrelere yüklenmezse, bir sonraki işlemde hücre içerisinden yanlış veri okunabilir. Bellekte saklanan verinin güvenirliliği için bu mekanizmanın "Algılama" durumu sonrasında Şekil 5-11'deki gibi "Geri Yükleme" durumuna geçmesi gerekmektedir. "Geri Yükleme" durumunda algı yükseltecin karakteristik özelliklerinden biri olan pozitif geri besleme özelliği aktif hale gelir. Pozitif geri besleme ile bit hattı üzerinden açık olan hücrelere yük

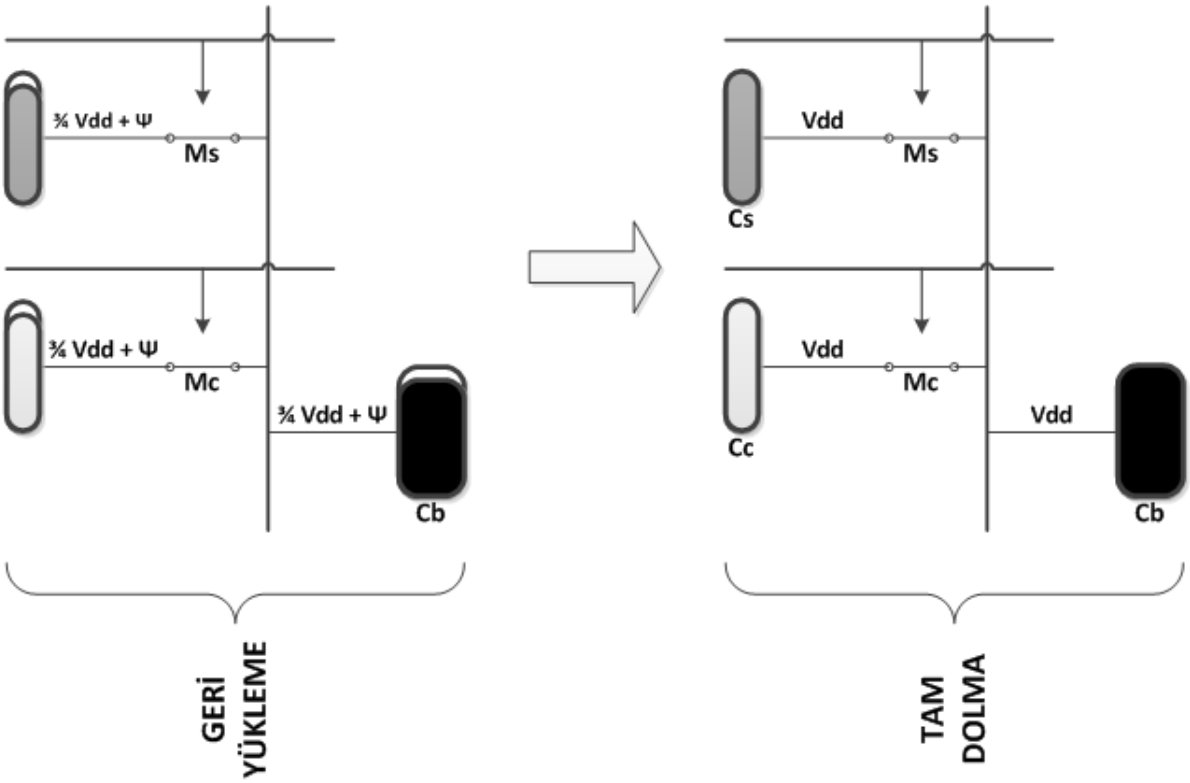
akışı başlar. "Geri Yükleme" durumu hücrelerin "Tam Dolma" durumuna ulaşmasıyla tamamlanır.

BULMA durumunda hem yedek hem de hedef hücre üzerine geri besleme yapılmaktadır. BULAMAMA durumunda ise yedek hücre hiç açılmamış durumda olduğu için bu hücreye geri besleme söz konusu değildir. Fakat zamansal yerellik özelliği doğrultusunda o an işlenen adresin kapalı olan yedek hücreye kopyalanması gerekmektedir. Bu nedenle eğer karar mekanizmasının sonucu BULAMAMA ise "Geri Yükleme" aşamasında yedek hücrenin açılarak o an işlenen hücre içerisindeki yük ile dolması sağlanır. Bu aşamadan sonra "Eylemsiz" duruma geçene kadar yedek hücre ve hedef hücre BULMA koşuluyla aynı akışı göstermektedir.



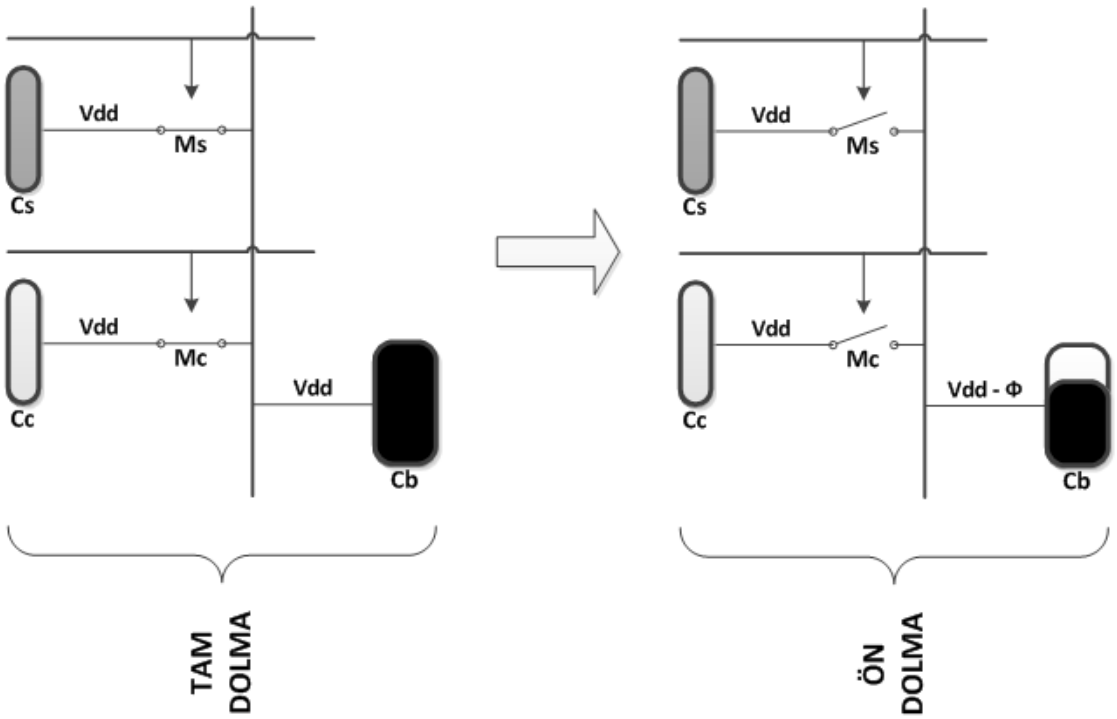
Şekil 5-11: Yedek dize algılama durumundan geri yükleme durumuna geçiş.

Tam Dolma: Hücreler "Geri Yükleme" durumundayken hücre ilk durumdaki yük değerine ulaşana kadar bit hattı üzerinden beslenmektedir. Hücre içerisindeki kapasitör ilk haline ulaştığında "Tam Doldu" durumunu tamamlamış olur. Şekil 5-12'de görüldüğü gibi hücrelerin içerisindeki kapasitörler ve bit hattı üzerindeki yük miktarı tam dolmuş seviyeye ulaşır. Hücreler bir sonraki işlem öncesi yükünü tekrar kazanır, veri güvenilirliği sağlanmış olur.

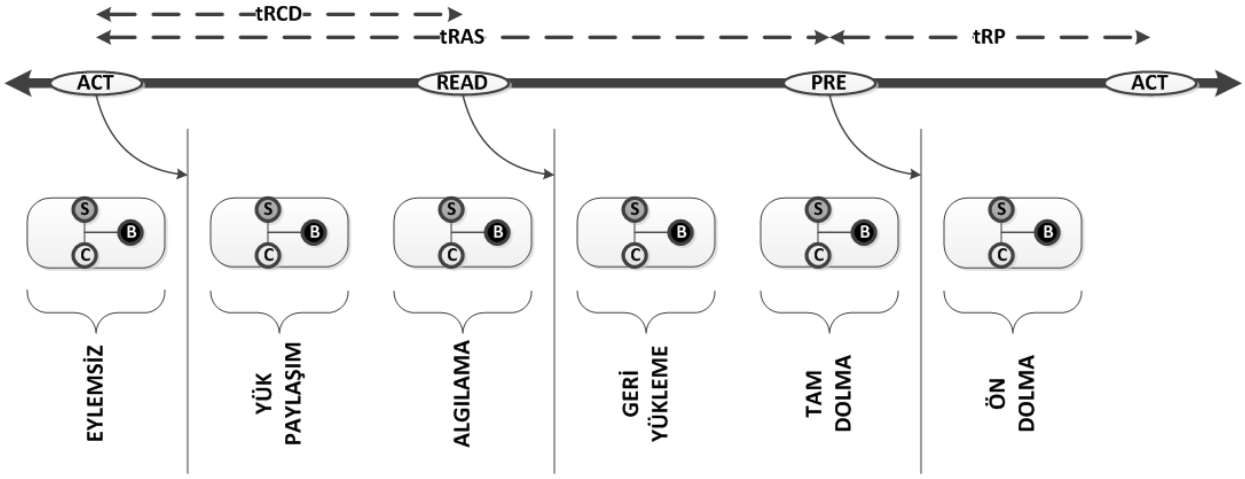


Şekil 5-12: Yedek dize geri yükleme durumundan tam dolma durumuna geçiş.

Ön Dolma: "Tam Dolma"'ya ulaşan hücrelerin "Eylemsiz" duruma gelmeden önceki son durumudur. Şekil 5-13'de görüldüğü gibi "Ön Dolma" durumunda hücreleri bit hattına anahtarlayan transistörler tekrar pasif duruma geçer. Hücreler ile bit hattı arasındaki bağlantı kopar. Hücreler ve anahtar "Eylemsiz" durumundaki hallerini alırlar. Fakat bit hattının da "Eylemsiz" duruma uygun hale gelebilmesi için "Ön Dolma" aşamasında bu hattın üzerindeki yük miktarı değişmektedir. Bellek, bit hattı üzerindeki yük besleme voltajının yarısına ulaşana kadar "Ön Dolma" durumundadır. Daha sonra "Eylemsiz" duruma geçiş yapar.



Şekil 5-13: Yedek dize tam dolma durumundan ön dolma durumuna geçiş.



Şekil 5-14: Yedek dize aktif et buyruğu sonrasında zamansal akış.

Çizelge 5-1: Standart ve mcr zamanlama değerleri.

		Standart			MCR
Parametre	Sembol	Minimum	Maksimum	Birim	Değer
Okuma komutundan ilk veriye geçen süre	tAA	13,75	20	ns	Aynı
ACT komutundan sonra RD/WR komutuna kadar olan gecikme	tRCD	13,75	-	ns	9,94
PRE komut periyodu	tRP	13,75	-	ns	9,94
ACT komutundan sonra ACT/REF komutu gelme periyodu	tRC	48,75	-	ns	Aynı
ACT komutundan sonra PRE komutu gelme periyodu	tRAS	35	9*tREFI	ns	21,46
Ortalama Saat Periyodu	tCK(Ort)	1,25	<1,5	ns	Aynı

5.2.3.2 Buyruk işleme zamanlaması

Buyruk işlem akışı mekanizmasının tarif edilen şekilde ilerleyebilmesi için bellek denetim birimi tarafından buyruk işleme modülüne buyrukların gelmesi gereklidir. Hücrenin "Eylemsiz" durumdan "Yük Paylaşım" durumuna geçebilmesi için "ACTIVATE" buyruğunun gelmesi gerekmektedir. Algı yükselteç çıkışında bit hattına bağlı hücre verisi görülmeye başladığında "READ" buyruğunun gelmesi ile ara dize dış dünyaya hücre verisini aktarır. Bellek hücrelerin ilk durumdaki yüklerine ulaşması sonrasında "PRECHARGE" buyruğu ile "Ön Dolma" aşamasına geçiş yapar. Tüm bu buyruklar arasında geçmesi gereken minimum süreler mevcuttur. JEDEC standardında ilgili süreler verilmiştir (Çizelge 3-4)[18]. Yedek dize ile iyileşen süreler ise "Multiple Clone Row DRAM" yayınından edinilmiştir. Standartta verilen ve yedek dize ile elde edilen zaman değerleri Çizelge 5-1'de verilmiştir. Ayrıca zamanlamaların buyruk işleme aşamalarıyla ilişkileri Şekil 5-14'de sunulmuştur.

5.2.4 Ramulator üzerinde yapılan işler

Ramulator, benzetimini yaptığı belleklerin standardına uygun bir şekilde çalışmaktadır. "Yedek Dize" mekanizmasını benzetim yazılımında çalıştırabilmek için Ramulator kaynak kodu üzerinde değişikliğe gidilmiştir. Yapılan değişiklikler genel olarak; yedek dize karar mekanizmasının denetim mekanizmasına dahil edilmesi ve zamanlama tablolarının güncellenmesi için bir mekanizmanın tasarlanmasıdır.

Ramulator benzetim yazılımı, ALDRAM [10], DDR3 [22], DDR4 [23], GDDR5 [26], HBM [29], LPDDR3 [24], LPDDR4 [25], SALP [11], TLDRAM [9] ve WIO [27, 28] bellekler için benzetim yapabilme imkanı sağlamaktadır. Tüm bu modellerin sahip olduğu bir takım özellikler mevcuttur; kapasite tipileri, hız tipileri ve zamansal değiştirgeler. İlgili değerler, benzetim yazılımında yer alan kaynak kod içerisinde .cpp uzantılı dosyalarda tanımlanmıştır. Standart çalışma modunda ilgili model değiştirgeleri Şekil 4-1'de görülmektedir.

Seçilen DRAM tipine göre her bir DRAM'in kapasite seçenekler, hız seçenekleri ve zaman seçenekleri değişmektedir. Bu tezdeki testler DRAM DDR3 bellek üzerinde yapıldığı için Şekil 5-16'da DDR3 özelinde model özelliklerini görebilirsiniz. Şekilde JESD79-3C standardında anlatılan 15 farklı kapasite tipi, 14 farklı hız tipi ve diğer

zaman seçenekleri yer almaktadır. Ayrıca bu şekilde yedek dize yöntemi kapsamında tasarıma eklenen iki farklı zaman seçeneği özelliği de gösterilmiştir. "init_timing_normal()" fonksiyonunda atanan ve "init_timing_yedek()" fonksiyonunda atanan değerler arasında tRCD, tRP ve tRAS farkı olduğu görülmektedir. Diğer zaman değerleri normal çalışma mekanizmasında ve yedek dize yönteminde aynı kalmıştır. Bellek denetim birimi BULMA durumunda "init_timing_yedek()" fonksiyon değerlerini ve BULAMAMA durumunda "init_timing_normal()" fonksiyon değerlerini kullanmaktadır.

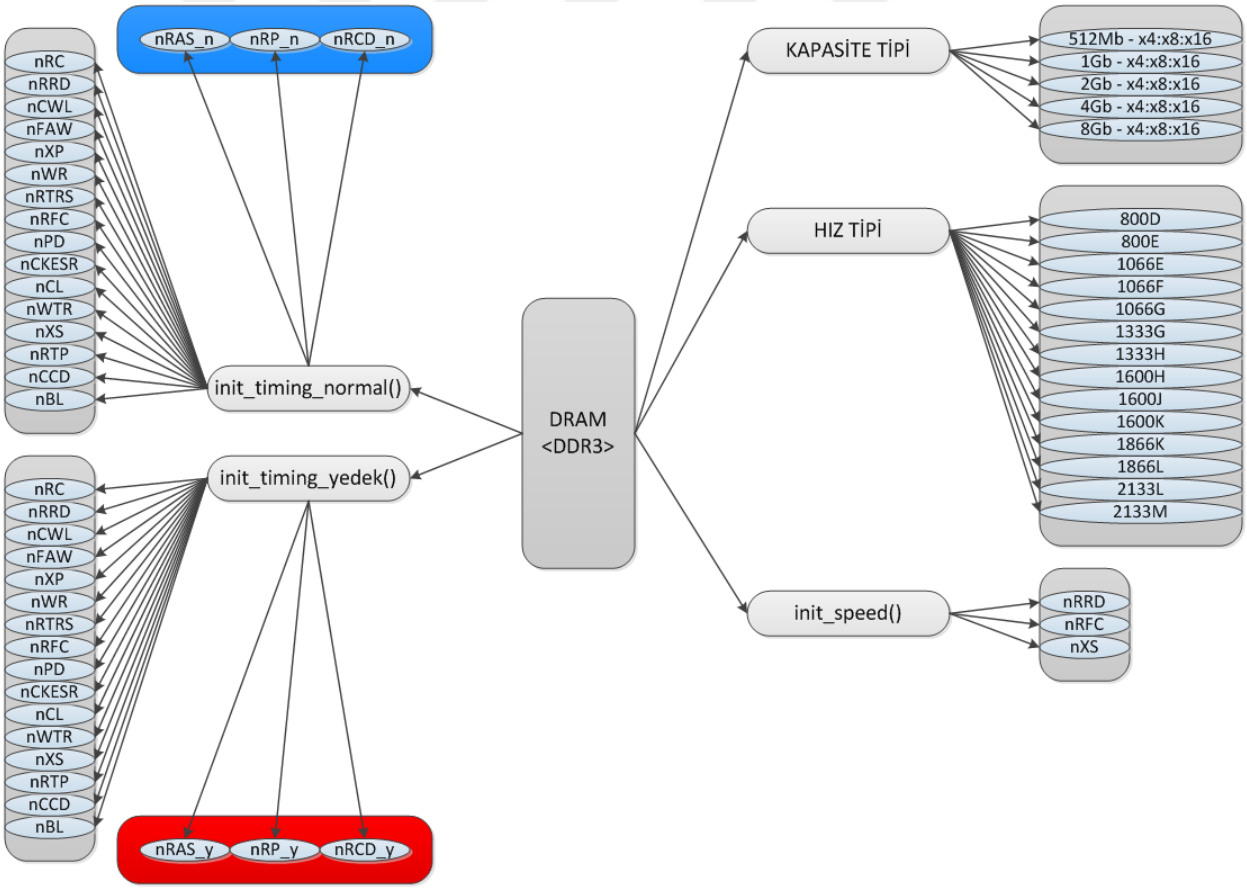
```
DDR3::DDR3(Org org, Speed speed) :
    org_entry(org_table[int(org)]),
    speed_entry(speed_table[int(speed)]),
    read_latency(speed_entry.nCL + speed_entry.nBL)
{
    init_speed();
    init_prereq();
    init_rowhit(); // SAUGATA: added row hit function
    init_rowopen();
    init_lambda();
    init_timing_normal();
    init_timing_fast();
}
```

Şekil 5-15: Yedek dize mekanizmasında ddr3 ilkleme kod parçası.

Benzetim yazılımı kaynak kodu içerisinde de BULMA/ BULAMAMA mekanizması için değişikliğe gidilmiştir. Standart tasarımda yer alan "init_timing()" fonksiyonu değerleri değiştirilmeden "init_timing_normal()" olarak adlandırılmıştır, BULAMAMA durumunda bu tablo kullanılmaktadır. Yedek dize yöntemiyle eklenen kod parçası "init_timing_yedek()" fonksiyonudur, BULMA kararı verildiğinde kullanılmaktadır. Orjinal ramulatordan farkı Şekil 5-15'de verilen kod parçasıyla gösterilmiştir.

Ramulator benzetim yazılımında yapılan denetim mekanizması değişikliği "Controller.h" dosyasına uygulanmıştır. Bu modül bellek denetim modülüdür. İşlemlerin yürütülmesi ve mekanizmanın devamlılığı için değişkenlerin güncellenmesi bu modülde yapılır. İşlem planlayıcısından gelen sıralı işlemleri işlemekle yükümlüdür. Girdi olan işlemi "issue_cmd()" fonksiyonu ile işler. "Yedek Dize", ramulator yazılımı içerisine dahil bir mekanizma olmadığı için bu mekanizmanın kullanım durumu ramulator kodu içerisinde seçilebilir halde gerçeklenmiştir. Şekil 5-17'de görüldüğü gibi, mod seçim kütüğüne 0 girilen

durumda ramulator standarta uygun, 1 girilen durumda yedek dize mekanizmasıyla çalışmaya başlamaktadır. Mod seçimi 1 olduğu durumda, BULMA/ BULAMAMA seçim mekanizmasının çalışması için issue_cmd() fonksiyonu güncellenmiştir. Şekil 5-18'de hedef adresi parçalayarak ilgili adrese ait bank, alt dize, vs. bilgiler edinilir. Bu bilgiler, yedek dize tablosunda tutulan yedek dize satırlarını adreslemek için kullanılır. Şekil 5-19'da yedek dize tablosundan ilgili alt dizeye ait yedek dize bilgisinin alındığı kod parçası verilmiştir. Bu iki kod parçası işlenecek adresi ve o an işlenecek alt dizede tutulan yedek dize adresini vermektedir. Bu aşamadan sonra Şekil 5-2'de de gösterilen BULMA/ BULAMAMA karar mekanizması devreye girer. Şekil 5-20'de BULMA/ BULAMAMA kararı veren kıyaslama kod parçası mevcuttur. Karar mekanizması sonucuna göre, bir sonraki saat vuruşları için yedek dize güncellemesi de bu kod parçası ile yapılmaktadır.



Şekil 5-16: Ddr3 özelinde yedek dize mekanizması kullanılan ramulator parametre ağaç yapısı.

```
// EI Modified //
void issue_cmd(typename T::Command cmd, const vector<int>& addr_vec, int id)
{
    int spareRow_enable = 1; // 0 = Orjinal Kod, 1 = SpareRow Kod
```

Şekil 5-17: Mekanizma seçim kod parçası.

```
if(cmd_spare_name == "PREA" || cmd_spare_name == "REF")
    current_subarr  = addr_vec_current[3];
else
    current_subarr  = (addr_vec_current[3] /512);

current_channel = addr_vec_current[0];
current_rank    = addr_vec_current[1];
current_bank    = addr_vec_current[2];
```

Şekil 5-18: Yedek dize mekanizmasında hedef adresi alan kod parçası.

```
if((current_channel     >= 0) && (current_channel   < max_channel_cnt) &&
       (current_rank    >= 0) && (current_rank      < max_rank_cnt) &&
       (current_bank    >= 0) && (current_bank      < max_bank_cnt) &&
       (current_subarr >= 0) && (current_subarr    < max_subarr_cnt) &&
       (id_idx >= 0) && (id_idx <8))
    addr_vec_spare = spare_vector_list[id_idx][current_channel][current_rank][current_bank][current_subarr];
else
    addr_vec_spare = {-1,-1, -1, -1, -1};
```

Şekil 5-19: Yedek dize mekanizmasında yedek adresi alan kod parçası.

```
if(spareRow_enable == 0)
{
    mode_sel = 0;
}
else
{
    if(addr_vec_current == addr_vec_spare)
        mode_sel = 1;
    else
        mode_sel = 0;

    if(cmd_spare_name == "PRE")
        spare_vector_list[id_idx][current_channel][current_rank][current_bank][current_subarr] = addr_vec_current;
}
```

Şekil 5-20: Bulma/bulamama durumuna göre zaman tablosu seçen kod parçası.

5.3 Hedeflenen Kazanımlar Ve Öngörülen Kayıplar

Araştırma esnasında yedek dize yönteminin istatistiki bir şekilde DRAM gecikme süreleri üzerinde olumlu gelişim sağladığı gözlenmiştir. Ayrıltılı inceleme yapıldığı zaman yedek dize yöntemi ile hem kazanım hem de kayıp elde edildiği görülmektedir. Kayıplar, donanımsal ve zamansal olarak iki türlü yaşanmıştır. Donanımsal kayıplar, bölüm 5.2.2 "Donanım Mimarisi Üzerinde Yedek Dize"de anlatıldığı gibi, her bir alt dize için 1 yedek satırın rezerve edilmesi donanımda standart şekilde veri saklanamayacak satırlar olmasına neden olmuştur.

Zamansal kayıplar ise yedek dize yöntemi kullanıldığı durumda BULAMAMA koşulu oluştuğunda meydana gelmektedir. BULMA durumunun tRAS, tRCD ve tRP sürelerine olumlu etkide bulunduğu diğer işlem zamanlarına etki etmediği bölüm 5.2.1 Yedek Dize Yöntemi Çalışma Mekanizması başlığı altında anlatılmıştır, gelişim gösteren süreler ve standarttaki değerleri Çizelge 5-1'de verilmiştir. BULAMAMA durumunda da "Geri Yükleme" durumunda kapalı durumdaki yedek hücreye o an işlenen hedef hücre verisinin yüklenmesi gerekmektedir. Bu durum standardın dışında bir durumdur ve hedef hücreye geri yükleme yapılma esnasında harcanan süreden daha fazla vakit kaybı yaşanmaktadır. Bu kayıp, kazanılan sürenin yanında kabul edilebilir seviyelerdedir. Ayrıca bellek denetim biriminin denetlediği ekstra bir tablo ortaya çıkmaktadır. Yedek dize mekanizması ile denetim birimi her gelen adresin, yedek dize listesinde olup olmadığını denetlemektedir.

Bu da yedek dize kullanılacak mimaride yonga içerisinde bu amaç için kullanılacak fazladan bir alana ihtiyaç duyulduğunu göstermektedir. Alternatif bir yöntem olarak bu adresler önbelleklerde veya işlemcinin içerisinde de tutulabilir.

6. SONUÇLAR

Tezin bu bölümünde, geliştirilen "Yedek Dize" yöntemini üzerinde yapılan testler ve elde edilen sonuçlar verilmiştir. Test ortamı olarak bölüm 4'te detaylı bir biçimde anlatılan "Ramulator"[5] bellek benzetim yazılımı kullanılmıştır. Bu test ortamında yedek dize yönteminin zamansal yerelliğe uygunluğu ve bu yöntemin tek çekirdekli ve çok çekirdekli mimarilerde başarımı ne kadar etkilediği araştırılmıştır.

6.1 Test Alt Yapısı

Ramulator benzetim yazılımında kullanılan SDRAM tipi DDR3'dür. Bu belleğe ait çip tipi bölüm 3.4'de anlatılan DDR3-1600K'dır. Toplam veri alanı olarak 2Gb'lik DDR3 seçilmiştir. İşlenecek buyruk sayısı için belirlenen üst limit, tek çekirdekli ve çok çekirdekli testler için aynı ve 20 milyar olarak ayarlanmıştır. Zamansal planlama algoritması olarak FRFCFS PriorHit seçeneği kullanılmıştır. Elde edilen sonuçlar için kullanılan girdi dosyaları birçok yayında da kullanılan bölüm 4.2'de verilen Ramulator girdi dosyalarıdır. Toplamda 23 farklı test girdisi ile testler yapılmıştır. Bu testler farklı kombinasyonlarla denenerek sonuçlarda elde edilen sapma oranları değerlendirilmiştir.

6.2 Zamanda Yerellik

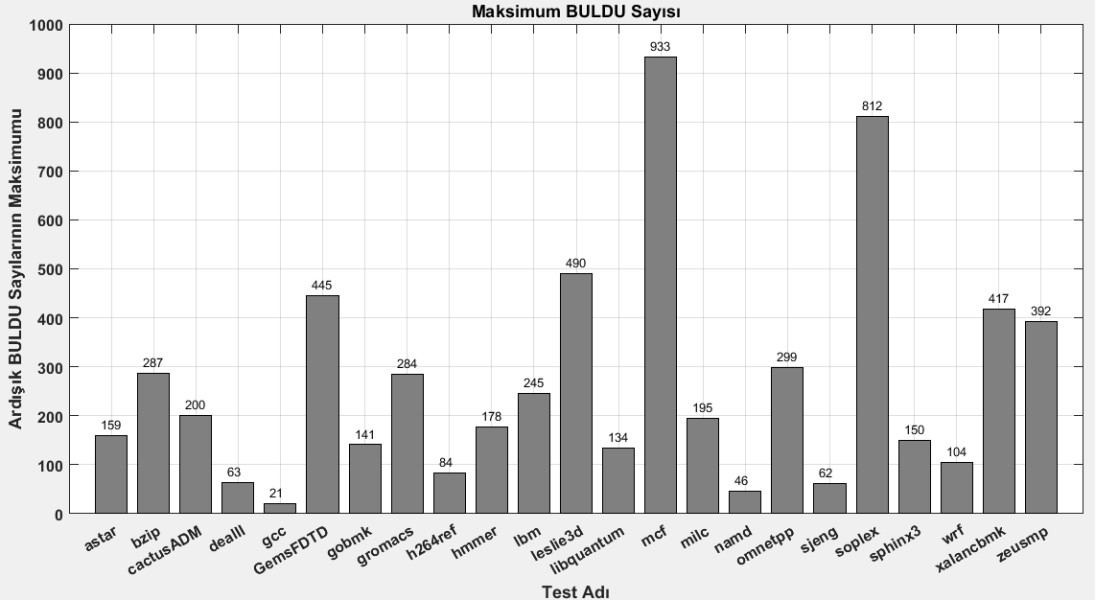
Yedek dize yöntemi, zamansal yerellik gözlemi üzerine geliştirilmiştir. Bu nedenle bu gözlemin tutarlı olup olmadığı Ramulator benzetim yazılımı üzerinde test edilmiştir. Tek çekirdek üzerinde yapılan zamansal yerellik testiyle edinilen bilgiye göre işlenmek üzere gelen buyrukların zamanda yerelliğe uygun bir davranış gösterdiği dikkat çekmiştir.

Sonuçlarda art arda gelen hedef dizelerin BULAMAMA durumu görülmeden, BULMA durumunun gözlendiği sayıların testlere göre maksimum değerleri Şekil 6-1'de paylaşılmıştır. Bu şekilde de görüldüğü gibi her bir testte görülen maksimum BULMA sayıları en az 21'dir (gcc test girdisi). Aynı test kapsamında maksimum

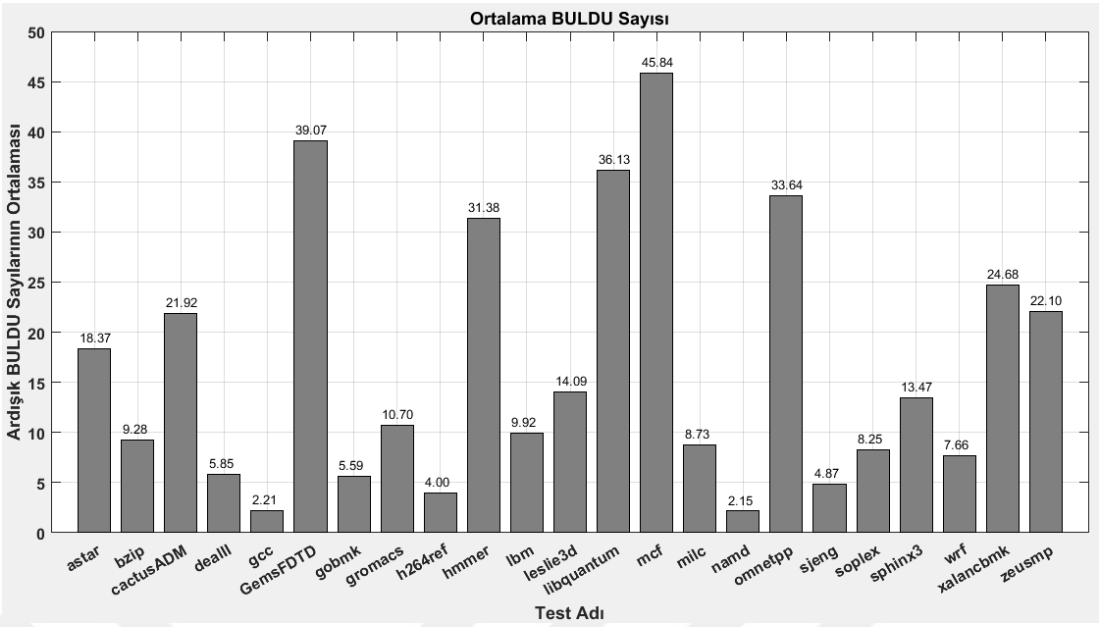
bulma sayısının en çok görüldüğü test olan mcf testinde, maksimum BULMA değeri 933 olarak ölçülmüştür. Maksimum BULMA sayısının buyruk sayısına bağlı değişebileceği düşüncesi ile Şekil 6-2'de ortalama BULMA sayıları hesaplanmıştır. Bu hesap ile art arda alınan BULMA değerlerinin sayısının toplam BULMA değerine oranı ölçülmüştür. Minimum ortalama BULMA 2,15 işlem ile namd girdi dosyası ve maksimum ortalama BULMA 45,84 ile mcf girdi dosyası ile elde edildiği analizi yapılmıştır.

BULMA ile BULAMAMA değerlerinin oranına bakıldığında, BULMA görülme oranları Şekil 6-3'de verilmiştir. Bu grafiğe göre minimum BULMA oranı %35,11 ile namd girdi dosyasıyla ve maksimum BULMA oranı %99,05 ile zeusmp girdi dosyasıyla elde edilmiştir.

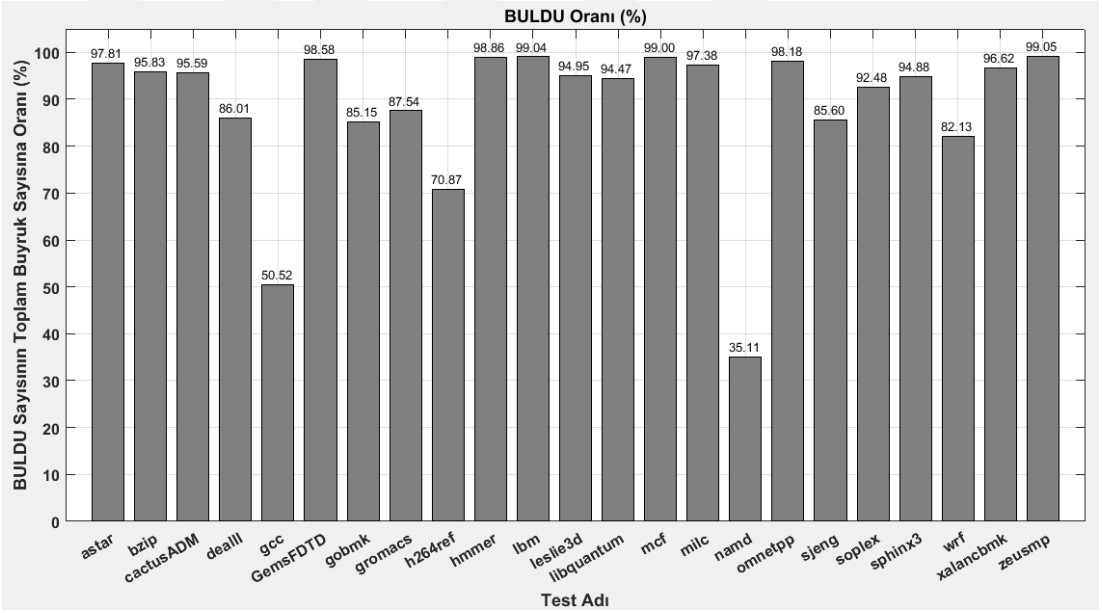
Tek çekirdekli ve çok çekirdekli mimarilerde yedek dize yöntemiyle elde edilen başarım da zamansal yerellik özelliği olarak bu değerin yeterli bir seviyede olduğunu göstermiştir.



Şekil 6-1: Maksimum bulma sayısı.



Şekil 6-2: Ortalama bulma sayısı.



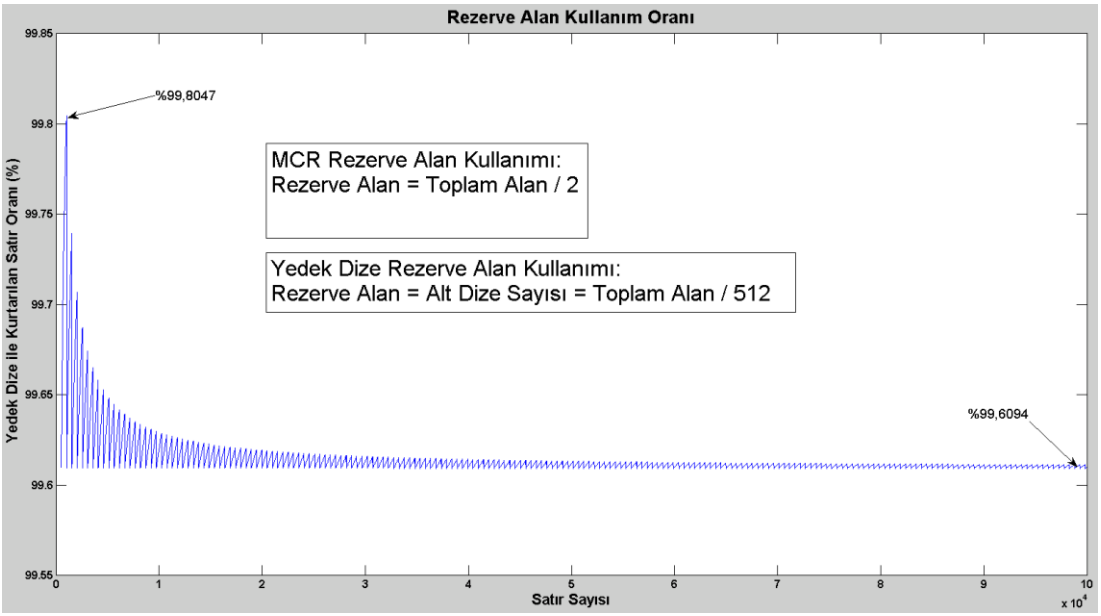
Şekil 6-3: Bulma oranı.

6.3 Donanımda Elde Edilen Kazanım

MCR yayınında da bahsedildiği gibi aynı anda aynı veriyi taşıyan iki hücrenin eş zamanlı bit hattına açılması işlem sürelerini kısaltabilmektedir. Bu yayında, sürede elde edilen gelişimin götürüsü olarak donanım gösterilmiştir. Her bir verinin yedeğinin başka bir satırda daha bulunması alanın yarısını rezerve olarak kullanmaya neden olmaktadır. Bu bölümde tez kapsamında sunulan "Yedek Dize" yöntemi ile

MCR yayınında yaşanan alan kaybından ne kadar tasarruf edilebileceği gösterilmiştir.

Matematiksel açıdan bakıldığında MCR yayınında yer alan rezerve alan toplam alanın yarısıdır. Yedek dize yönteminde rezerve alan alt dize başına 1 satır olarak belirlenmiştir. Bu nedenle her 512 satırdan 1 tanesi rezerve olarak kullanılmaktadır. Bir diğer deyişle alt dize sayısı kadar rezerve satır kullanılmaktadır. Şekil 6-4'e bakıldığında alanda yer alan satıra bağlı olarak elde edilen donanım tasarrufu değişmektedir. Maksimum %99,80 ve minimum %99,61 oranlarında donanım alanından tasarruf elde edildiği hesaplanmıştır.



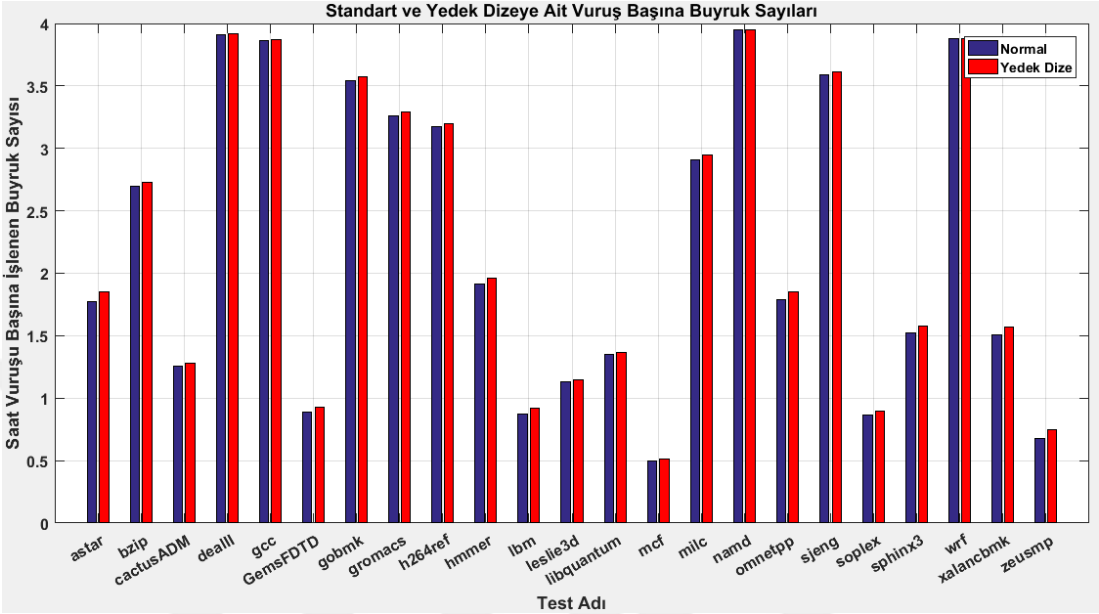
Şekil 6-4: Rezerve alan kullanım oranı.

6.4 Tek Çekirdekli İşlemciler Üzerinde Elde Edilen İyileşme

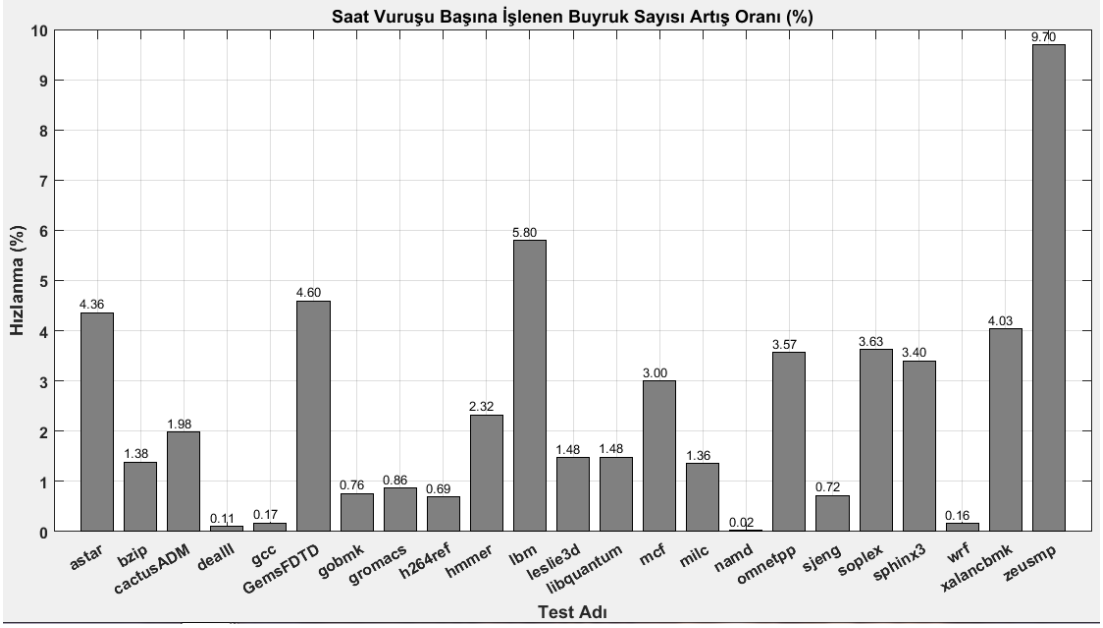
Birim zamanda işlenen buyruk sayısı (IPC), bellek işlem süreleri üzerindeki başarımı doğru orantılı şekilde göstermektedir. Ramulator çıktıları hem tek çekirdekli hem de çok çekirdekli mimariler için yedek dize uygulanan ve uygulanmayan sonuçlar olarak iki şekilde alınmış ve kıyaslanmıştır.

Şekil 6-5'de standartta yer alan ve yedek dize yöntemiyle sunulan mekanizmalar kullanılarak elde edilen IPC değerleri sunulmuştur. Tüm testlerde birim zaman başına işlenen buyruk sayısı birbirine çok yakın olarak gözlenmiştir. Başarımdaki artışı gözlemek adına bu iki değerin oranları Şekil 6-6'da verilmiştir.

Şekil 6-6'da görülen sonuçlara göre tek çekirdekli işlemcilerde yedek dize yöntemi başarımı arttırmıştır. Bellek işlem sürelerinde en çok artış "zeusmp" testiyle edinilmiştir. "zeusmp" testinde IPC oranı % 9,70 artmıştır. Bellek işlem sürelerinde en az artış %0,02 ile "namd" testiyle edinilmiştir.



Şekil 6-5: Standart ve yedek dizeye ait vuruş başına buyruk sayıları.



Şekil 6-6: Saat vuruşu başına işlenen buyruk sayısı artış oranı.

6.5 Çok Çekirdekli İşlemciler Üzerinde Elde Edilen İyileşme

Çizelge 6-1'de çok çekirdekli mimari testlerinde kullanılan girdi dosyaları verilmiştir. Testler 8 çekirdekli mimari üzerinde yapıldığı için her bir testte 8 adet girdi dosyası mevcuttur.

Çizelge 6-1: Çok çekirdekli mekanizma testleri.

Trace Files	Trace Name	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16
401	bzip2	x	x		x					x		x			x			
403	gcc	x	x						x	x				x				
429	mcf	x	x		x	x	x										x	
433	milc	x	x				x			x				x			x	
434	zeusmp	x		x	x							x		x			x	x
435	gromacs	x		x		x							x	x				
436	cactusADM	x		x	x								x		x			
437	leslie3d	x		x		x							x			x		
444	namd		x	x	x							x			x			
445	gobmk		x	x		x						x				x		
447	dealII		x	x	x								x		x			
450	soplex		x	x					x				x			x		x
456	hmmer				x				x	x		x		x	x		x	
458	sjeng																	
459	GemsFDTD				x	x			x		x			x		x	x	x
462	libquantum							x	x	x	x	x						
464	h264ref					x		x		x	x		x		x			
470	lbm						x	x		x	x	x				x	x	x
471	omnetpp						x	x	x		x	x			x		x	x
473	astar						x	x	x		x		x			x	x	x
481	wrf					x	x	x			x			x	x			
482	sphinx3					x	x	x			x			x		x		x
483	xalancbmk						x	x	x	x			x			x		x

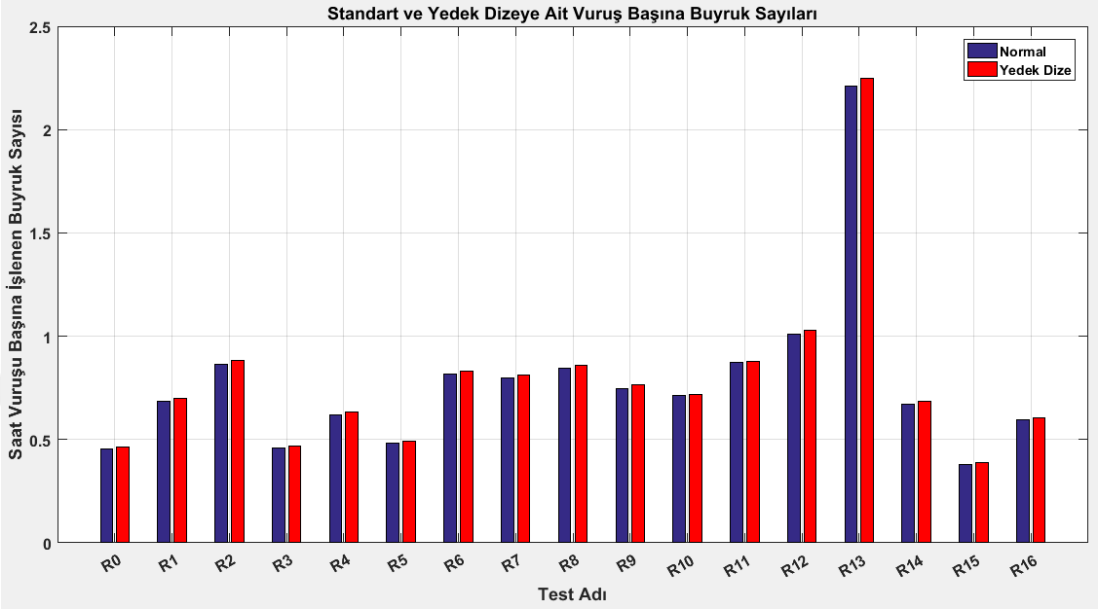
Yapılan testlere göre elde edilen birim zamanda işlenen buyruk sayısı Şekil 6-7'de verilmiştir. Standart ve yedek dize mekanizmaları elde edilen bu ikili değer grubu birbirlerine çok yakın olarak gözlenmiştir. Oransal yönden bakıldığında Şekil 6-8'de de görüldüğü gibi minimum hızlanma %0,915 ile R11 testi sonucunda ve maksimum hızlanma %2,313 ile R9 testi sonucunda elde edilmiştir.

6.6 Yorumlar

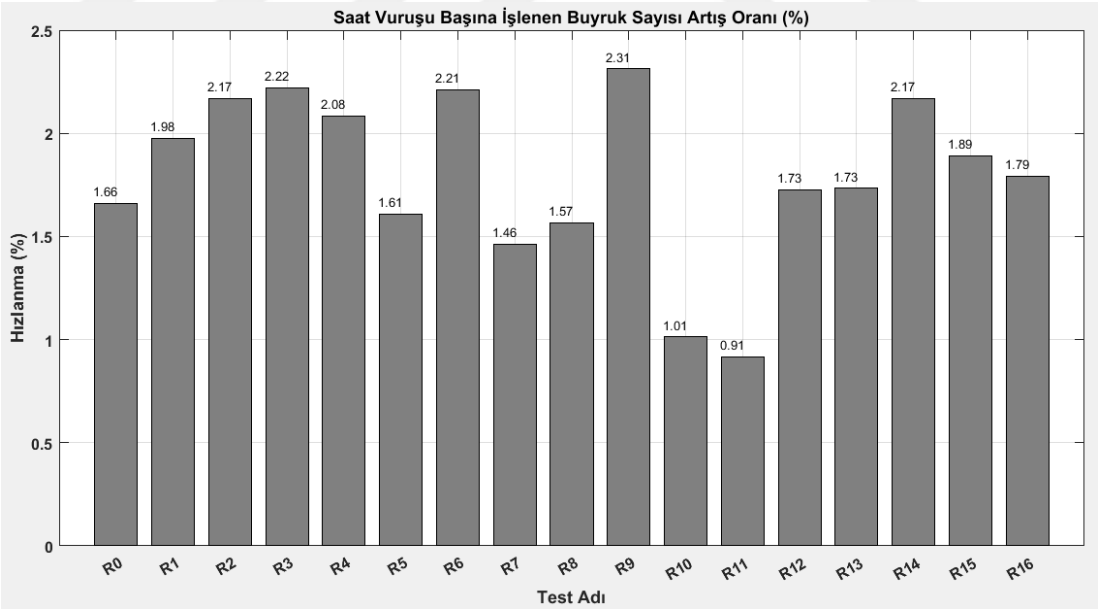
Yedek dize yönteminin amacı buyruk işlemlerinde belleğe daha hızlı erişim ve daha hızlı ön dolum sağlanmasıdır. Yedek dize mekanizması zamansal yerellik gözlemi üzerine kurulmuştur. Uygulanan yöntem bellek içerisinde bir alt dizeye yedek bir dize ekleme ve bu dizeye ilgili verileri kopyalama üzerine geliştirilmiştir.

Yapılan analizlerde, zamansal yerelliğin en çok görüldüğü test girdisi zeusmp olmuştur. En az görüldüğü test girdisi ise namd dosyası olmuştur. Zamansal yerelliğe bağlı olarak tek çekirdek ve çok çekirdekli mekanizmanın farklı kazanımlar sağladığı gözlenmiştir. Başarımın en fazla elde edildiği testler tek çekirdekli mimari üzerinde

yapılan testler sonucunda alınmıştır. Çok çekirdekli mimari ile yapılan testlerde beklenen başarıma ulaşamamıştır. Bunun nedeni ise bellek işlem birimine birden çok çekirdeğin sürekli erişmesi ve farklı çekirdeklerin zamansal yerellik özelliğini bozması olarak gözlenmiştir.



Şekil 6-7: Standart ve yedek dizeye ait vuruş başına buyruk sayıları (çok çekirdekli).



Şekil 6-8: Saat vuruşu başına işlenen buyruk sayısı artış oranı (çok çekirdekli).

Tek çekirdekli mimaride, birim zaman başına işlenen buyruk sayıları incelendiğinde zamansal yerellikle orantılı bir başarım elde edildiği saptanmıştır. Birim zamanda

işlenen buyruk sayılarına bakıldığında maksimum işlem zeusmp ile yapılmıştır. En az gelişim namd girdi dosyası testi ile elde edilmiştir. Tek çekirdekli mimarilerden alınan sonuçlara bakıldığında tüm testlerde kazanım elde edildiği görülmektedir.

Çok çekirdekli mimaride, birim zaman başına işlenen buyruk sayıları incelendiğinde zamansal yerellikle orantılı bir başarım elde edilememiştir. Bunun nedeni çekirdeklerin denetim birimi üzerinde yedek dize zamansal yerellik denetimini bozmasıdır.

KAYNAKLAR

[1] **Hennessy, J., Patterson, D.**, (2011). Computer Architecture, Fifth Edition: A Quantitative Approach 5th, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ISBN: 012383872X 9780123838728.

[2] "Early computers at Manchester University", Resurrection, The Computer Conservation Society, Summer 1992, ISSN 0958-7403, retrieved 7 July 2010.

[3] **Mutlu, O., Subramanian, L.**, (2014) "Research problems and opportunities in memory systems," SUPERFRI.

[4] Microelectronics: Circuit Analysis and Design", McGraw-Hill Companies Inc., New York, NY, USA, 2010, ISBN: 978-0-07-338064-3.

[5] "Ramulator" Kaynak Kodu, https://github.com/CMUSAFARI/Ramulator, (2015).

[6] **Lee, D., Kim, Y., Seshadri, V.**, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture" in HPCA, (2013).

[7] **Wilkes, M.**, "The memory gap and the future of high performance memories", Comp. Arch. News, ACM, (2001).

[8] **Wulf, W., McKee, S.**, "Hitting the memory wall: implications of the obvious", Comp. Arch. News, ACM, (1995).

[9] **Lee, D., Kim, Y., Seshadri, V.**, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture" in HPCA, (2013).

[10] **Lee, D., Kim, Y., Pekhimenko, G.**, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case" in HPCA, (2015).

[11] **Y. Kim et al.**, "A case for exploiting subarray-level parallelism (SALP) in DRAM." in ISCA, (2012).

[12] **Hassan, H., Pekhimenko, G., Vijaykumar, N.**, "ChargeCache: Reducing DRAM latency by exploiting row access locality", HPCA, (2016).

[13] **W. Shin, J. Yang, J. Choi, L.-S. Kim**, "NUAT: A non-uniform access time memory controller," in HPCA, (2014).

[14] **Choi, J., Shin, W., Jang, J.**, "Multiple Clone Row DRAM: A Low Latency and Area Optimized DRAM" in ISCA, (2015).

[15] **Y. Kim et al.,** "A case for exploiting subarray-level paralelism (SALP) in DRAM." in ISCA, (2012).

[16] **İpek, E., Hassan, H., Ergin, O.**, "DRAM Bellek Gecikmelerini Azaltabilmek için Yedek Dize Yöntemi", 19. Akademik Bilişim Konferansı (AB17) Bildiriler Kitapçığı, Aksaray, Şubat 2017.

[17] **D. H. Neil, H.E. Weste**, CMOS VLSI Design. A Circuit and Systems Perspective , 3rd ed. Addison-Wesley, 2005.

[18] "JEDEC79-3C", JEDEC Solid State Technology Association 2008, Arlington, VA 22201-2107, June 2008.

[19] "JEDEC79-2C", JEDEC Solid State Technology Association 2008, Arlington, VA 22201-2107, June 2008.

[20] "JEDEC79", JEDEC Solid State Technology Association 2008, Arlington, VA 22201-2107, June 2008.

[21] Jedec Website, <https://www.jedec.org>, 2017.

[22] JEDEC. JESD79-3 DDR3 SDRAM Standard, June 2007.

[23] JEDEC. JESD79-4 DDR4 SDRAM, Sept. 2012.

[24] JEDEC. JESD209-3 Low Power Double Data Rate 3 (LPDDR3), May 2012.

[25] JEDEC. JESD209-4 Low Power Double Data Rate 3 (LPDDR4), Aug. 2014.

[26] JEDEC. JESD212 GDDR5 SGRAM, Dec. 2009.

[27] JEDEC. JESD229 Wide I/O Single Data Rate (Wide/IO SDR), Dec. 2011.

[28] JEDEC. JESD229-2 Wide I/O 2 (WideIO2), Aug. 2014.

[29] JEDEC. JESD235 High Bandwidth Memory (HBM) DRAM, Oct. 2013.

[30] **K. Chang et al**. Improving DRAM Performance by Parallelizing Refreshes with Accesses. In HPCA, 2014.

[31] "401.bzip2 Benchmark", https://www.spec.org/auto/cpu2006/Docs/401.bzip2.html, 2016.

[32] **Kim, Y., Yang, W., Mutlu, O.**, "Ramulator: A Fast and Extensible DRAM Simulator" in CAL, (2015).

[33] **Mutlu, O.**, "Lecture 24: Memory Scheduling", Carnegie Mellon University, 2014.

[34] "403.gcc Benchmark", https://www.spec.org/cpu2006/Docs/403.gcc.html, 2016.

[35] "429.mcf Benchmark", https://www.spec.org/cpu2006/Docs/429.mcf.html, 2016.

[36] "433.milc Benchmark", https://www.spec.org/cpu2006/Docs/433.milc.html, 2016.

[37] "434.zeusmp Benchmark", https://www.spec.org/cpu2006/Docs/434.zeusmp.html, 2016.

[38] "435.gromacs Benchmark", https://www.spec.org/cpu2006/Docs/435.gromacs.html, 2016.

[39] "436.cactusADM Benchmark", https://www.spec.org/cpu2006/Docs/436.cactusADM.html, 2016.

[40] "437.leslie3d Benchmark", https://www.spec.org/auto/cpu2006/Docs/437.leslie3d.html, 2016.

[41] “444.namd Benchmark”, https://www.spec.org/cpu2006/Docs/444.namd.html, 2016.

[42] “445.gobmk Benchmark”, https://www.spec.org/cpu2006/Docs/445.gobmk.html, 2016.

[43] “447.dealII Benchmark”, https://www.spec.org/cpu2006/Docs/447.dealII.html, 2016.

[44] “450.soplex Benchmark”, https://www.spec.org/cpu2006/Docs/450.soplex.html, 2016.

[45] “456.hmmer Benchmark”, https://www.spec.org/cpu2006/Docs/456.hmmer.html, 2016.

[46] “458.sjeng Benchmark”, https://www.spec.org/cpu2006/Docs/458.sjeng.html, 2016.

[47] “459.GemsFDTD Benchmark”, https://www.spec.org/cpu2006/Docs/459.GemsFDTD.html, 2016.

[48] “462.libquantum Benchmark”, https://www.spec.org/auto/cpu2006/Docs/462.libquantum.html, 2016.

[49] “464.h264ref Benchmark”, https://www.spec.org/cpu2006/Docs/464.h264ref.html, 2016.

[50] “470.lbm Benchmark”, https://www.spec.org/cpu2006/Docs/470.lbm.html, 2016.

[51] “471.omnetpp Benchmark”, https://www.spec.org/cpu2006/Docs/471.omnetpp.html, 2016.

[52] “473.astar Benchmark”, https://www.spec.org/cpu2006/Docs/473.astar.html, 2016.

[53] “481.wrf Benchmark”, https://www.spec.org/auto/cpu2006/Docs/481.wrf.html, 2016.

[54] “482.sphinx3 Benchmark”, https://www.spec.org/cpu2006/Docs/482.sphinx3.html, 2016.

[55] “483.xalancbmk Benchmark”, https://www.spec.org/auto/cpu2006/Docs/483.xalancbmk.html, 2016.

ÖZGEÇMİŞ

Ad-Soyad : Eyüphan İpek

Uyruğu : T.C.

Doğum Tarihi ve Yeri : 30.01.1989 / ANKARA

E-posta : eyuphanipek@gmail.com

ÖĞRENİM DURUMU:

- **Lisans** : 2012, Bilkent Üniversitesi, Elektrik Elektronik Mühendisliği
- **Yüksek Lisans** : 2017, TOBB Ekonomi ve Teknoloji Üniversitesi, Bilgisayar Mühendisliği

MESLEKİ DENEYİM VE ÖDÜLLER:

Yıl	Yer	Görev
2016-halen	SDT AŞ.	Uz. Sayısal Tasarım Mühendisi
2015-2016	ASELSAN AŞ.	Sayısal Tasarım Mühendisi
2012-2015	MİKES AŞ.	Sayısal Tasarım Mühendisi
2013-2017	TOBB ETÜ	Araştırma Burslu Yüksek Lisans Öğrencisi

YABANCI DİL: İngilizce, Almanca

TEZDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

- **İpek, E., Hassan, H., Ergin, O.,** "DRAM Bellek Gecikmelerini Azaltabilmek için Yedek Dize Yöntemi", 19. Akademik Bilişim Konferansı (AB17) Bildiriler Kitapçığı, Aksaray, Şubat 2017.
- **İpek, E.,** "DRAM Bellek Gecikmelerini Azaltabilmek için Yedek Dize Yöntemi", 19. Akademik Bilişim Konferansı (AB17) Sunumları, Aksaray, Şubat 2017.