DRAM Bellek Gecikmelerini Azaltabilmek için Sık Kullanılan Dizelerin Yedek Dizeye Kopyalanması: Yedek Dize Yöntemi

Eyüphan İpek*, Hasan Hassan*, Doç. Dr. Oğuz Ergin*

* TOBB Ekonomi ve Teknoloji Üniversitesi, Bilgisayar Mühendisliği Bölümü, Ankara

eipek@etu.edu.tr, hhassan@etu.edu.tr, oergin@etu.edu.tr

Özet: DRAM gecikmeleri bellek işlemleri için harcanan toplam süre için önemli bir etkiye sahiptir. Çalışmamızda önerdiğimiz yedek dize yöntemi, erişilecek hedef dizenin çoklanması ile erişim süresinin kısalmasına fayda sağlayan bir mekanizmadır. Mekanizmamız alt dize içerisindeki dizelere erişimin yüksek zamanda yerellik özelliği gösterdiği gözlemi üzerine kurulmuştur. Alt dize içerisinde erişilen bir dizeye art arda erişimin tekrar tekrar gerçekleştiği gözlenmiştir. Bu gözlemden yararlanarak, erişilecek dizenin bir kopyasının yedek bir dizede daha tutulması DRAM erişimini hızlandıracaktır.

Aynı veriyi tutan DRAM'in iki dizesinin aynı anda aktif hale gelmesi erişim gecikmesini azaltacaktır çünkü iki dizenin kapasitörleri birleşerek algılanacak verinin sinyal seviyesini güçlendirecektir. Her bir alt dizeye fazladan bir (yedek) dize eklenmesini ve bu yedek dizeye seçilen bir dizenin verilerinin saklanmasını önermekteyiz. Yazılımsal mekanizmamız daha sonraki erişimler için o an erişilen dizenin verilerini yedek dizeye kopyalamaktadır. Yedek dizenin hedef dize verisini içerdiği durumda, mekanizmamız hedef ve kopyalanmış dizeyi aynı anda aktif hale getirerek DRAM erişiminin düşük gecikmelerle tamamlanmasını sağlamaktadır. Eğer yedek dize içerisindeki veri erişilmek istenen dizeden farklıysa, mekanizmamız o an erişilen dizenin verilerinin yedek dizeye kopyalanıp kopyalanmaması kararını verir. Çalışmamız esnasında geniş bir test verisi ile mekanizmamızı geliştirdik, ortalama DRAM erişim gecikmelerinin azaldığını ve sistem performansının genel olarak daha iyi bir duruma ulaştığını gözlemledik.

Anahtar Sözcükler: Yedek Dize, DRAM Gecikmesi, Alt Dize, Erişim Süresi, Hedef Dize, Bellek İşlem Süresi

Duplicating Frequently Accessed Rows for Reducing DRAM Memory Latency: Copied Row Approach

Abstract: DRAM latency has vital impact on overall execution time of many workloads. We propose SpareRow, a mechanism which reduces DRAM access latency by utilizing a duplicate of the accessed DRAM row to boost the access operation. Our mechanism is based on the key observation that high temporal locality exists among the rows of each subarray. We observe that a row from any subarray is typically accessed repeatedly for a couple of times without accessing any other rows from the same subarray. We exploit that observation by efficiently creating a copy of that repeatedly accessed row and using it to access the DRAM row faster.

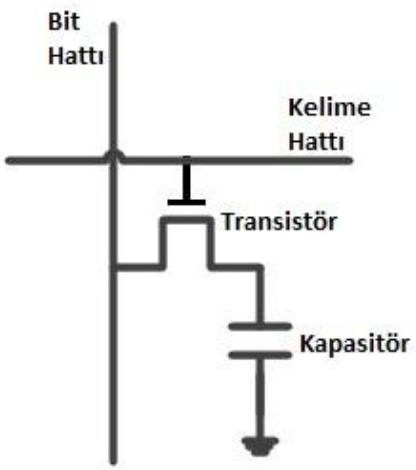
Simultaneously enabling two DRAM rows, which store the same data, reduces the access latency because the capacitors of the opposing cells from the two rows are effectively combined to strengthen the signal used for sensing the data. We propose to add an extra (spare) row to each subarray to selectively store a duplicate of one of the rows of the subarray. Our software-transparent mechanism utilizes the spare row to create a duplicate of the accessed row and accelerate the access operations to that row for the subsequent access. 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, our mechanism decides whether to copy the accessed row to the spare row or not. We evaluate SpareRow on a large set of workloads and show 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

1. Giriş

Dinamik RAM'lerin (DRAM) tasarımcılar tarafından sıkça kullanılmasının nedenlerinden biri bellek yongasının yoğunluğunun yüksek olmasıdır. Birim veri başına ödenen meblağ diğer bellek tiplerine göre azdır. Teknolojinin daha yoğun bellek yongaları üretmeye elverişli olmasına rağmen DRAM işlem süreleri neredeyse sabit kalmaktadır [1]. İşlemcilerin hızlarının artmasına karşın DRAM işlem sürelerinin neredeyse sabit olması sistem performansını kısıtlayan bir kalem olarak kalmıştır [2].

DRAM hücreleri yongada çok küçük alanlar kaplayan bir kapasitör ve bir transistörden oluşmaktadır. Veriler yongadaki kapasitörler içerisinde yük olarak saklanmaktadır. DRAM hücresine ait donanım yapısı Şekil 1'de verilmiştir. Yongada yer alan transistörler kelime hattından gelen aktif komutuyla işlem yapılacak bellek hücresini bit hattına ve algı yükselteçlere bağlar. Algı yükselteçler, DRAM hücrelerinden büyük, kapasitörleri okuyan/ kapasitörlere yazan devre birimleridir.



Şekil 1. DRAM Hücresi Donanım Yapısı

DRAM gecikme zamanları üzerinde birçok güncel çalışma mevcuttur. Bu çalışmalarda da vurgulandığı gibi DRAM gecikmelerinin iki temel kaynağı vardır. Birincisi, yoğunluğu arttırabilmek için binlerce hücrenin aynı bit hattıyla (bitline) aynı algı yükselteci paylaşmasıdır. Bu durum bit hattının kapasitansını arttırmaktadır, bit hattının kapasitansının artması ise hücredeki veriye erişim süresini geciktirecektir [3]. İkincisi, yonga yoğunluğunu arttırabilmek için kapasitör ve transistör boyutlarının küçültülmek istenmesidir. Kapasitörlerin boyutlarının küçülmesi içerisinde saklanacak yükün azalmasına neden olacaktır. Bu nedenle algı yükselteçlere akacak yük miktarı da azalacaktır. Bu konuda daha önce yapılan çalışmalara göre küçük kapasitörler daha uzun erişim sürelerine neden olmaktadır [4].

Bu yayında yer alan fikrimizi destekleyebilmek için "Ramulator"[5] adı verilen ve akademik çalışmalar için geliştirilmiş bellek benzetim yazılımını kullanmıştır. Test aşamasında benzer çalışmalarda da analiz amaçlı yer alan veri dizilerini içeren komut dosyaları kullanılmıştır.

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

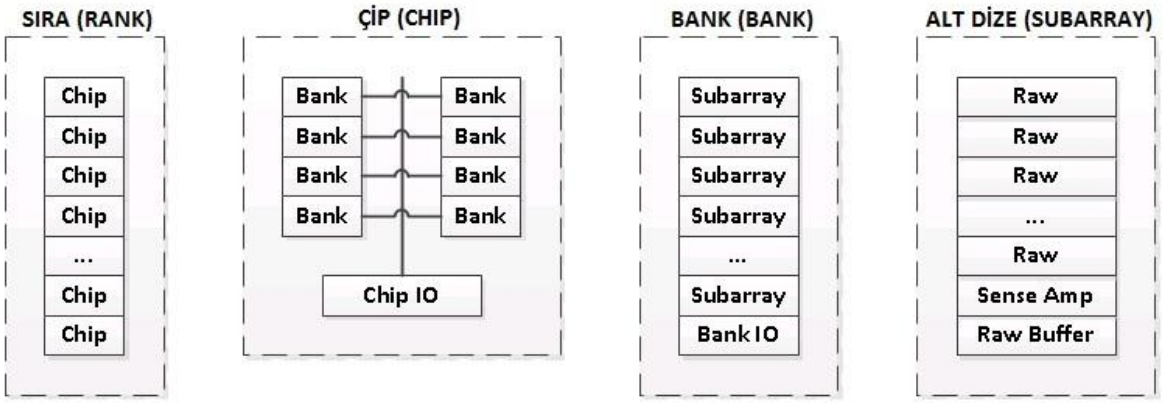
2. DRAM Donanım Mimarisi

DRAM mimarisi kanallar (channel), sıralar (rank), çipler (chip), banklar (bank), alt dize (subarray) [6], dizeler (row), kolonlar (column) ve bellek hücrelerinden oluşur [7]. Bellek kontrol birimi gelen talepler doğrultusunda DRAM mimarisindeki kanallara erişir. Şekil 2'de bir kanala ait alt birimlere yer verilmiştir. Bellek kontrol birimi komutları işlerken en fazla bir dizeyi aktif hale getirir. Komşu dizelerin oluşturduğu DRAM mimari birimine alt dize denir, alt dize hakkında daha detaylı bilgiye bu yayının Alt Dize bölümünden ulaşabilirsiniz. Daha önce yapılan benzer çalışmalarda da DRAM mimarisi detaylı bir biçimde anlatılmıştır.

3. Alt Dize (Subarray)

DRAM bankları işlem sürelerindeki gecikmeleri azaltabilmek için alt dize adı verilen mimari bileşenine ayrılmıştır. Okuma ya da yazma işlemlerinden önce dize içerisindeki veri ara dize'ye (row buffer) kopyalanmaktadır. Algı yükselteçlere akan elektriksel yük, dize'nin sahip olduğu veriyi algılamaktadır. Daha sonra algı yükselteçler boşalan kapasitörlere operasyon önceki değerleri yükler. Banklardaki her bir dize bit hattı üzerinden algı yükselteçlere bağlıdır. Düşey olarak yer alan hücreler (aynı hizada yer alan farklı dize hücreleri) aynı bit hattını paylaşırlar ve aynı algı yükseltece

bağlıdırlar. Daha yoğun bir yonga için tercih edilen bu tasarım kararının olumsuz yanları da vardır; bit hattının direncinin ve kapasitansının kendine bağlı hücre sayısının artmasıyla doğru orantılı olarak değişir. Bu olumsuzluğun üstesinden gelebilmek için genel olarak 512 dizenin bir ara gelmesiyle oluşan yapıya alt dize denir, 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.



Şekil 2. DRAM Donanım Mimarisi

4. DRAM Komut İşlemleri

DRAM'de işlenecek komutlar geldiği zaman DRAM hücreleri ve diğer mimari birimleri komut tipine göre farklı elektriksel koşullardan geçmektedir. Aşağıda mimari birimlerinin komut esnasında karşılaştığı durumlar özetlenmiştir.

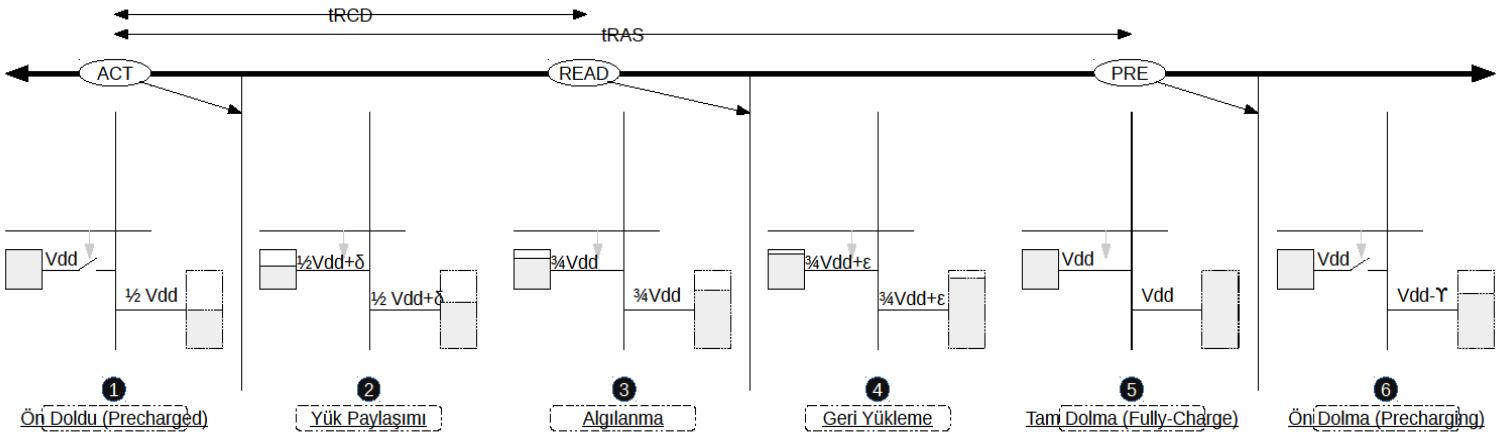
Ön Dolum Durumu (PRECHARGED): Şekil 3'deki 1 numaralı bölümde görüldüğü gibi henüz kelime hattı (word line) aktif halde değildir. Bu nedenle hücre ve bit hattı arasında her hangi bir bağlantı yoktur. Veri hücresi tamamen yüklü durumdadır. Bit hattının yük seviyesi ise besleme voltajının yarısı kadardır. ACTIVATE komutu gelene kadar mimari birimleri bu durumda kalmaktadır.

Aktif Hücrenin Yük Paylaşımı: Komutun işleneceği hücreye bağlı olan kelime hattı aktif hale geldikten sonra başlayan durumdur. Şekil 3'deki 2 numaralı bölümde hücre içerisindeki yükün, bit hattına akmaya başladığı görülmektedir. Hücre içindeki yük azaldıkça ya da arttıkça, bit hattı üzerindeki yük miktarı ters oranda değişmektedir. Algı yükselteç hat üzerindeki yükü anlamlandırmaya başladığı seviyeye ulaşana kadar hücreden yük akışı devam eder. Hat üzerine yük akışının artması algı yükseltecin daha hızlı algılayabilmesini sağlamaktadır [8].

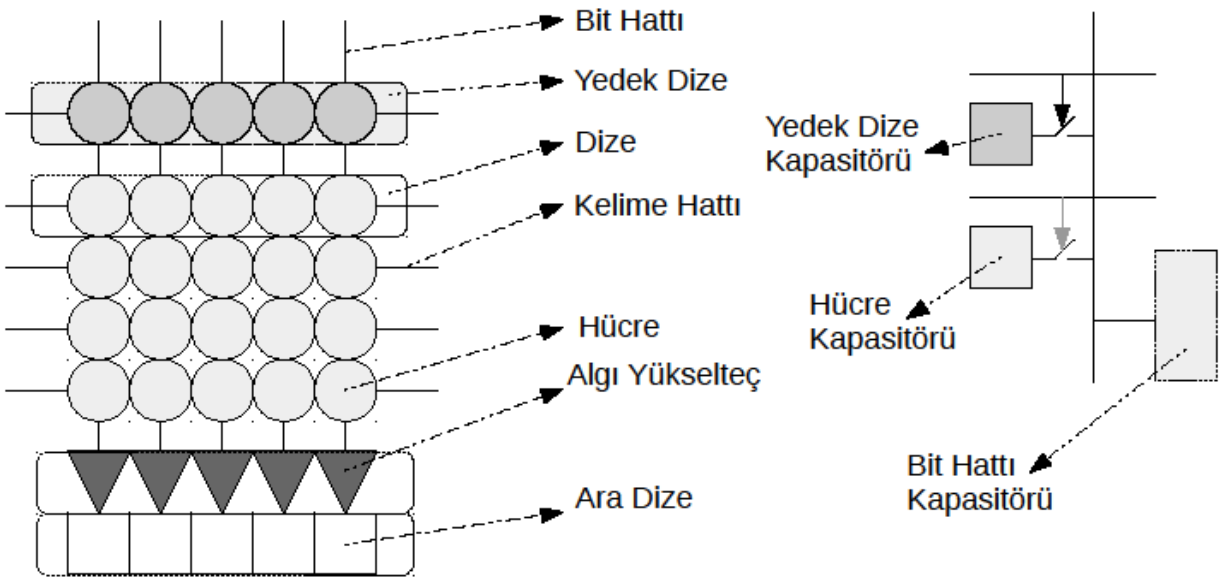
Aktif Hücre Yükünün Algılanması: Bit hattının üzerindeki yükün algı yükselteç tarafından algılanmaya başladığı andır (Şekil 3'de 3 numaralı bölüm). Algı yükselteç aktif hale gelmektedir, veri ara dizeye (row buffer) ulaşmış olur. Hücre içerisindeki veri, bit hattı üzerine aktığı için kaybolmuş durumdadır. Hücre içerisindeki verinin tekrar kullanılabilmesi için ön dolum işlemi başlamaktadır.

Geri Yükleme ve Tam Dolum (Fully-Charge): Hücrenin tekrar eski yük değerine ulaşabilmesi için algı yükseltecin bir diğer özelliği olan pozitif geri besleme özelliği aktif hale gelir. Algı yükselteç üzerinden bit hattı aracılığıyla aktif hücre üzerine yük akmaya başlar (Şekil 3'de 4 numaralı bölüm). Hücre işlem öncesi yükünü tekrar kazanır (Şekil 3'de 5 numaralı bölüm). Bu aşamada hücre ve bit hattı üzerindeki yük seviyesi eşittir.

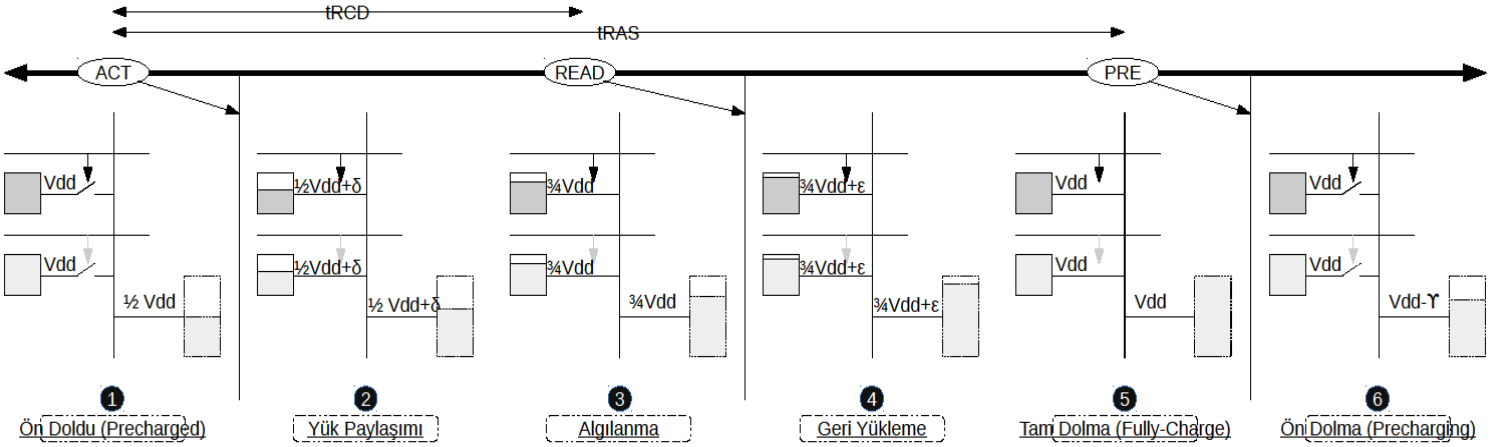
Ön Doluyor (PRECHARGING): Geri yükleme ve tam dolum aşamalarının sonunda hücreyi bit hattına bağlayan transistör tekrar pasif duruma geçer (Şekil 3'de 6 numaralı bölüm). Hücre ile bit hattı arasındaki bağlantı kopar. Bit hattı üzerindeki yük, besleme voltajının yarısına ulaşana kadar DRAM'in durumu ön doluyor (PRECHARGING) olarak tanımlanmaktadır. Bit hattı bu seviyeye ulaştığında DRAM ön doldu (PRECHARGED) duruma geçmektedir ve yeni komut beklemektedir.



Şekil 3. DRAM Komut İşlemleri



Şekil 4. Yedek Dizeli Alt Dize (Subarray) Yapısı (Sol Tarafta), Yedek Dizeli Hücre Gösterimi (Sağ Tarafta)



Şekil 6. Yedek Dizeli DRAM Komut İşlemleri

5. Daha Önce Yapılan Benzer Çalışmalar

DRAM gecikme sürelerini azaltmak için zamansal yerellikle sık kullanıldığı görülen dizenin özel bir dizede yedeklenmesi fikri ilk defa bu yayında sunulmaktadır. Buna rağmen DRAM gecikmelerini daha iyi seviyelere getirmeye çalışan birçok güncel yayın mevcuttur. Bunlardan bizim çalışmamıza ışık tutan ve kendi çalışmamıza en yakın gördüğümüz yayınlar aşağıdakilerdir.

Çoklu Kopyalanan Dizeli DRAM (Multiple Clone Row DRAM: A Low Latency and Area Optimized DRAM): Yük akış hızının arttırılması fikri üzerine geliştirilmiş bir yöntemdir, her bir dizenin yedeği donanım üzerinde rezerve edilmiştir. Hedef dize ile aynı veriyi paylaşan rezerve dizenin aynı anda aktif olmasıyla gecikme süreleri daha iyi seviyelere ulaşmaktadır. Bu çalışmanın dezavantajı, DRAM yongasının en az yarısının rezerve olarak kullanılması ve tutulabilecek veri boyutunun azalması anlamına gelmektedir.

Sıralı-Gecikme DRAM (Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture): TL-DRAM yayınında bit hattının kapasitansını azaltmaya yönelik bir fikir sunulmuştur. Bit hattını daha kısa parçalara bölerek, yük paylaşımını daha hızlı gerçekleştirilebileceğini ve hücrelerin içlerindeki veriyi algı yükselteçlerin daha hızlı algılayabileceğini göstermişlerdir.

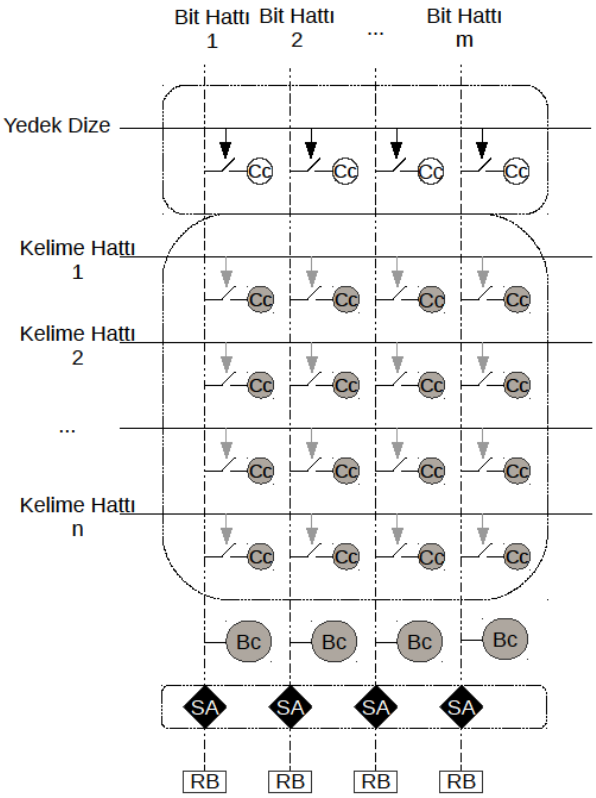
Uyarlamalı-Gecikme DRAM (Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case): AL-DRAM kullanıcılara gerçek zamanlı olarak çalışma koşullarının değişmesiyle zamanlama parametrelerini ayarlayabilme imkânı sağlamaktadır. Değişen sıcaklık koşullarına göre düşük ya da yüksek sıcaklıklarda en iyi performansla çalışacak zamanlama parametrelerini ayarlayarak gecikme sürelerinin azaltılabileceğini göstermişlerdir.

6. Yedek Dize Yöntemi

Yedek dize DRAM işlem gecikme sürelerini azaltmayı hedefleyen bir yöntemdir. Her bir alt dizede (subarray) yedek bir dize tutulmaktadır. Bu bölümde yedek dize yönteminin donanım mimarisinde ve DRAM komut işlemlerinde getirdiği değişimleri ve bu mekanizmanın çalışma şekli anlatılmaktadır.

6.1. Donanım Mimarisinde Yedek Dize

Yedek dize yönteminde, her bir alt dize (subarray) içerisinde bir tane yedek dize ayrılmıştır. Şekil 4'de görülen bir alt dize içerisinde klasik DRAM donanımında bir kelime hattına bağlı olarak kullanılan bir dize bu yöntem için rezerve edilmiştir. İşlenecek bir komut geldiğinde, hedef dize ile aynı veriyi taşıyan yedek dize de bit hattına bağlanmaktadır. İki hücrenin kapasitans değerinin artması bit hattına akacak yükün büyüklüğünün de artacağı anlamına gelmektedir (Şekil 5). Bu nedenle bit hattına akan yükün büyüklüğü arttığı için algı yükselteçlerin algılama süresi kısalacaktır [10]. Veriye erişim süresi azalacaktır. Genelde 512 dizeden oluşan alt dize içerisinde 1 dizenin rezerve kullanılması donanımda olumsuz bir etki gibi görülse de, zamanda yerellik prensibi ile aranan verilerin sıklıkla yedek dizede görülmesi DRAM gecikme sürelerinin kısalmasını sağlayacaktır ve rezerve alan avantaja dönüşecektir.



Şekil 5. Yedek Dizeli DRAM Donanım Yük Kapasiteleri

6.2. Yedek Dize Yöntemiyle DRAM Komut İşlemleri

Gelen komutun işleneceği hedef dizede tutulan veri ile yedek dizede saklanan veri aynı ise ilgili komut işlem aşamaları bu bölümde anlatılmıştır. Eğer iki dizedeki veriler bir birlerinden farklı ise komut işlemleri bölüm 4'de de anlatıldığı şekilde standartta uygun ilerlemektedir. Fakat standarttan farklı olarak işlem sonunda yedek dizede tutulan veriyi o an işlenen hedef dize verisiyle güncellemektedir.

Ön Dolum Durumu (PRECHARGED): Şekil 6'da 1 numaralı bölümde görüldüğü gibi hem yedek hem de hedef dizeye ait kelime hattı aktif halde değildir. Veri hücreleri tamamen yüklü durumdadır. Bit hattının yük seviyesi ise besleme voltajının yarısı kadardır. ACTIVATE komutu gelene kadar mimari birimleri bu durumda kalmaktadır.

Aktif Hücrenin Yük Paylaşımı: Bu aşamada ilk olarak bellek kontrol birimi yedek dizede tutulan verinin hedef dizedeki veri ile aynı olup olmadığını kontrol etmektedir. Eğer veriler aynı ise komutun işleneceği hücreye bağlı olan kelime hattı ve yedek dizeye ait kelime hattı aktif hale gelir. Şekil 6'da 2 numaralı bölümde hücrelerin içerisindeki yüklerin, bit hattına akmaya başladığı görülmektedir. Hücre içerisindeki yükler standarda uygun bir biçimde aksa da birim zamanda bit hattına akan yük miktarı artacaktır. Hat üzerine yük akışının artması algı yükseltecin daha hızlı algılayabilmesini sağlamaktadır.

Aktif Hücre Yükünün Algılanması: Bit hattı üzerine akan yük arttığı için algı yükselteç daha hızlı aktif hale gelmektedir (Şekil 6'da 3 numaralı bölüm). Hem hedef dize hem de yedek dize hücrelerinin içerisindeki veriler, bit hattı üzerine aktığı için kaybolmuş durumdadır. Daha sonra bu hücrelerin içerisindeki verilerin tekrar kullanılabilmesi için geri yükleme işlemi başlamaktadır.

Geri Yükleme ve Tam Dolum (Fully-Charge): Hücrelerin tekrar eski yük değerlerine ulaşabilmesi için algı yükseltecin pozitif geri besleme özelliği aktif hale gelir. Algı yükselteç üzerinden bit hattı aracılığıyla aktif hücrelere yük akmaya başlar (Şekil 6'da 4 numaralı bölüm). Hücreler bir sonraki işlem öncesi yükünü tekrar kazanır (Şekil 6'da 5 numaralı bölüm).

Ön Doluyor (PRECHARGING): Geri yükleme ve tam dolum aşamalarının sonunda hücreleri bit hattına bağlayan transistörler tekrar pasif duruma geçer (Şekil 6'da 6 numaralı bölüm). Hücreler ile bit hatları arasındaki bağlantı kopar. Bit hattı üzerindeki yük, besleme voltajının yarısına ulaşma yönünde eğilim sergiler.

6.3. Yedek Dize Yöntemi Çalışma Prensibi

İşlem yapılacak dize ile yedek dize içerisindeki veri aynı ise bu durum bellek kontrol birimi tarafında VURDU olarak adlandırılmaktadır. VURDU durumunda zamanlama parametreleri hızlı değerlerle güncellenir. Eğer işlem yapılacak dize ile yedek dize içerisindeki veri farklı ise bu duruma da ISKA denir. ISKA durumunda sadece hedef hücre aktif hale gelir. Bu nedenle zamanlama parametreleri olarak JEDEC [9] standardında yer alan süreler kullanılmıştır.

7. Sonuçlar

Bu bölümde "Ramulator" bellek benzetim yazılımı üzerinde geliştirdiğimiz yedek dize yöntemine ait çıktılar paylaşılmıştır. Paylaşılan sonuçlara birçok yayında da kullanılan test girdilerini kullanarak ulaşılmıştır. Test aşamasında kullandığımız alt yapı hakkındaki bilgileri, zamansal yerellik prensibine sonuçların uyumluluğu ve gecikme sürelerindeki iyileşme bilgileri aşağıda mevcuttur.

7.1. Test Alt Yapısı

Benzetim yazılımda kullanılan DRAM tipi DDR3 SDRAM'dir. Bu belleğe ait çip tipi DDR3-1600K olarak seçilmiştir. Kullanılan toplam veri alanı 2Gb'dir. İşlenecek komut sayısı üst limiti 20 milyar olarak ayarlanmıştır. Komut işlerken kullanılan planlama algoritması FRFCFS PriorHit'dir. Toplamda 29 farklı test girdisi ile testler yapılmıştır.

7.2. Zamanda Yerellik

Yedek dize yöntemiyle minimum rezerve alan kullanarak gecikme sürelerinde iyileşme sağlanmaya çalışılmıştır. Yapılan gözlemlerde de edinilen bilgiye göre işlenmek üzere gelen komutların zamanda yerelliğe uygun bir davranış gösterdiği dikkat çekmiştir. Tablo 1'de yedek dize üzerinde zamansal yerellik test sonuçları verilmiştir. Sonuçlarda ISKA durumu görülmeden art arda VURDU durumunun gözlenme sayıları paylaşılmıştır. Ardışık VURDU sayılarına göre aynı yedek dizeye ortalama 15,78 kere VURDU işlemi

gerçekleşmiştir. Sonuçlardan edinilen bilgilere göre yedek dize yöntemi ile saklanan verilerin tekrar tekrar kullanılma sıklığı yüksektir ve bu yöntem zamansal yerelliğe uygun bir mekanizmadır.

7.3. Gecikme Sürelerinde Görülen İyileşme

Gecikme sürelerinde görülen iyileşmeyi anlayabilmek için birim zamanda işlenen komut sayıları (Instruction per Cycle: IPC) incelenmiştir. Çıktılar yedek dize yöntemi uygulanarak ve uygulanmadan olmak üzere 2 kere alınmıştır. Tablo 2'de görülen sonuçlara göre yedek dize yöntemi kullanılan sonuçların kullanılmadan alınan çıktılara oranı verilmiştir. Tablodaki her verinin pozitif artış gösterildiği görülmektedir, bu nedenle yedek dize kullanılan tüm sonuçlarda gecikme sürelerinde azalma gözlenmiştir. Ortalama %2,38 oranında iyileşme gözlenmiştir. Gecikme sürelerinde en çok artış "zeusmp" test girdileriyle elde edilmiştir, bu testte birim zamanda % 8,83 oranında daha fazla komut işlenmiştir. Gecikme sürelerinde minimum iyileşme %0,024 ile "namd" test girdileriyle elde edilmiştir.

8. Kaynaklar

[1], [3], [4], [7]: Lee, D., Kim, Y., Seshadri, V., "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture" in HPCA, (2013).

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

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

[3], [4], [7]: Lee, D., Kim, Y., Pekhimenko, G., "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case" in HPCA, (2015).

[3], [6]: Y. Kim et al., "A case for exploiting subarray-level parallelism (SALP) in DRAM." in ISCA, (2012).

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

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

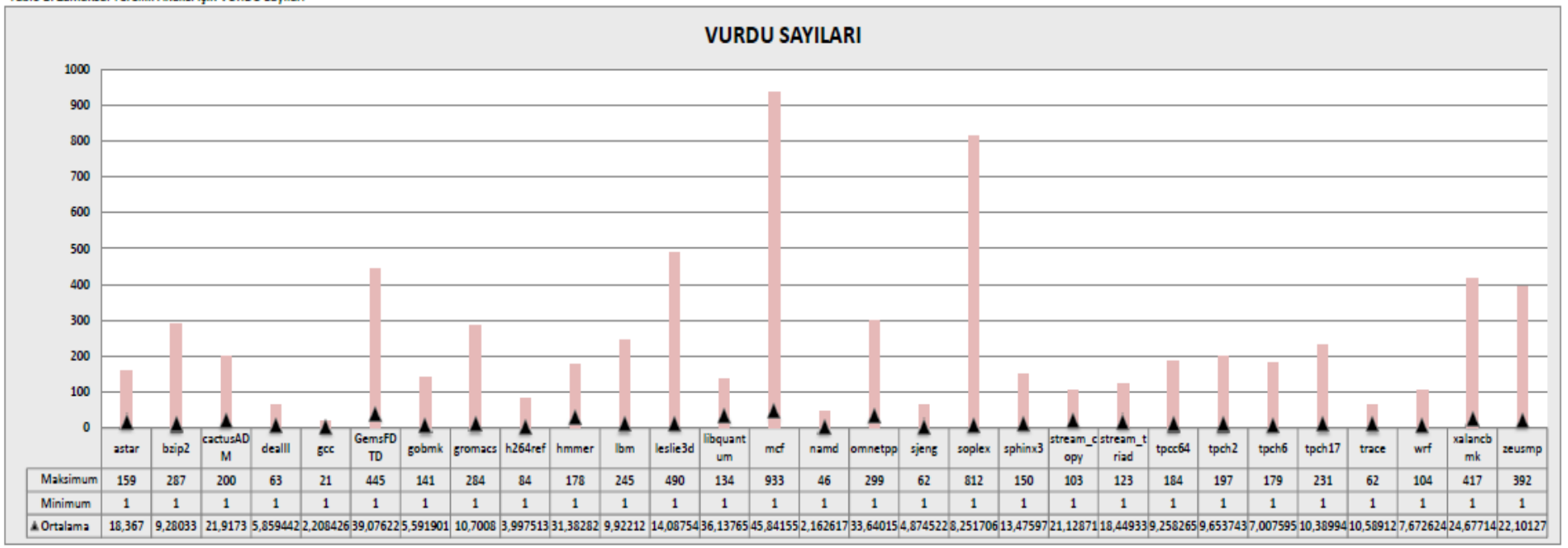
[5]: "Ramulator" Kaynak Kodu, https://github.com/CMUSAFARI/ Ramulator, (2015).

[4], [8]: Choi, J., Shin, W., Jang, J., "Multiple Clone Row DRAM: A Low Latency and Area Optimized DRAM" in ISCA, (2015).

[9]: JEDEC Standard No. 79-3C

[10]: D. H. Neil H.E. Weste, "CMOS VLSI Design. A Circuit and Systems Perspective", 3rd ed. Addison-Wesley, (2005).

Tablo 1. Zamansal Yerellik Analizi için VURDU Sayıları



	astar	bzip2	cactusADM	dealII	gcc	GemsFDTD	gobmk	gromacs	h264ref	hmmer	lbm	leslie3d	libquantum	mcf	namd	omnetpp	sjeng	soplex	sphinx3	stream_copy	stream_triad	tpcc64	tpch2	tpch6	tpch17	trace	wrf	xalancbmk	zeusmp
Maksimum	159	287	200	63	21	445	141	284	84	178	245	490	134	933	46	299	62	812	150	103	123	184	197	179	231	62	104	417	392
Minimum	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Ortalama	18,367	9,28033	21,9173	5,859442	2,208426	39,07622	5,591901	10,7008	3,997513	31,38282	9,92212	14,08754	36,13765	45,84155	2,162617	33,64015	4,874522	8,251706	13,47597	21,12871	18,44933	9,258265	9,653743	7,007595	10,38994	10,58912	7,672624	24,67714	22,10127

Tablo 2. Yedek Dizeli Mekanizma Sonuçlarının Standard Sonuçlara Oranı