5 Doğrudan Bellek Erişimi (*Direct Memory Access -DMA*)

DMA yöntemi, G/Ç birimleri ile bellek arasında yoğun (büyük bloklar halinde) veri aktarımı yapmak için kullanılır.

Örneğin: disk, grafik kartları, ses kartları, ağ kartları ile bellek arası aktarım

DMA yöntemi aşağıdaki işlemler için de kullanılır.

Çok çekirdekli/işlemcili sistemlerde işlemciler arası veri aktarımı için

Aynı sistem içinde bellekten belleğe veri aktarma için

Hatırlatma: **Yoklamalı ve kesmeli** çalışmada veri aktarımını MİB (program) yapmaktadır ve aktarılan tüm veriler MİB üzerinden geçmektedir.

Bu yöntemlerde MİB G/Ç birimini (veya belleği) programla okur ve veriyi belleğe (veya G/Ç birimine) yazar.

DMA yönteminde sistemde doğrudan **bellek erişimi denetçisi** (*DMA Controller* – DMAC) bulunur. DMAC, MİB gibi davranarak bellek adresleme işlemlerini yürütebilir.

MİB, DMAC'ı koşullayarak (programlayarak) tüm giriş çıkış işlerini ona havale eder.

Gerek olduğunda DMAC sistem yolunu alarak veri aktarımını yapar. G/Ç birimi ile bellek arasındaki veri aktarımı MİB üzerinden geçmez

DMAC aktarım yaparken MİB bellek gerektirmeyen iç işlemleri yürütebilir.

5.1 Genel Bakış

Veri aktarımı gerekli olduğunda G/Ç birimi, denetçiyi (DMAC) uyarır.

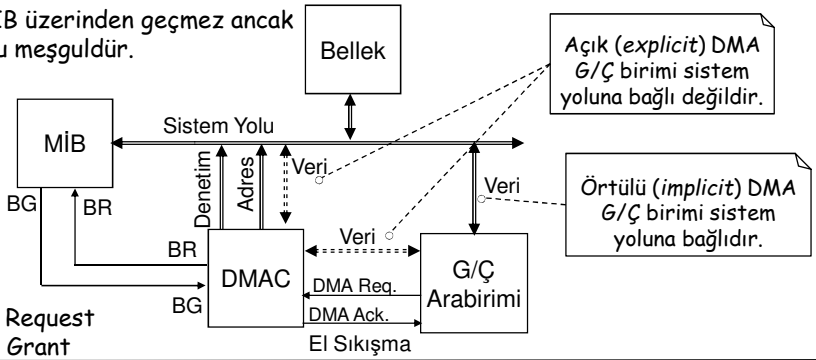
DMAC, sistem yolunu kullanmak için MİB'e istekte bulunur (BR).

MİB o anda devam eden yol çevrimini tamamlar (komutu değil), kendini sistem yolundan yalıtır ve BG çıkışını etkin yaparak DMAC'a yanıt verir.

Artık bellek erişimi için gerekli olan tüm işaretleri üretmek ve veri aktarımını gerçekleştirmek DMAC'ın sorumluluğundadır.

MİB kendi iç işlerine devam edebilir.

Veriler MİB üzerinden geçmez ancak sistem yolu meşguldür.



5.2 DMAC (*DMA Controller*) tipleri:

a) Açık DMA (*Flow-through*, *Explicit*): Bellek ile G/Ç arabirmi arasında aktarılan veriler DMAC üzerinden geçer.

Önce DMAC veriyi G/Ç arabirminden (ya da bellekten) okur ve belleğe (ya da G/Ç arabirmine) yazar.

b) Örtülü DMA (*Fly-by*, *Implicit*): Aktarılan veriler DMAC üzerinden geçmez.

DMAC yol kullanım hakkını alınca kaynak (veya varış adresini) ve denetim işaretlerini (R/W, VMA, vb.) çıkarır.

DMAC, bellek ve G/Ç arabirmini aynı anda etkin yapar.

Veri aktarımı bellek ile G/Ç arabirimi arasında doğrudan yapılır. Veri kaynaktan okunup varışa **bir saat çevriminde** yazılır.

Bu nedenle örtülü DMA yönteminde veriler açık DMA yöntemine göre daha kısa sürede aktarılabilir.

Ancak bu yöntemde aktarım sırasında yolda sadece tek bir adres bilgisi olduğundan bellekten belleğe (iki farklı adres arasında) aktarım yapmak mümkün değildir.

Örtülü DMA yönteminde sadece bellek ile G/Ç arabirimleri arasında aktarım yapılabilir.

5.3 DMA Aktarım Kipleri (*DMA Transfer Modes*)**:**

a) Blok aktarımı (*Burst mode*): DMAC yolu aldıktan sonra daha önceden belirlenmiş olan blok boyu kadar veri aktarır ve bu sürede yolu elinde tutar.

Blok aktarımı bitince yol MİB'e verilir.

Blok boyu MİB (program) tarafından DMAC koşullanırken belirlenir.

MİB uzun süre sistem yolundan yalıtılmış durumda kalabilir.

Bu aktarım şekli program veya veri dosyalarının yüklenmesinde kullanılabilir, çünkü MİB işlerini sürdürebilmek için zaten bu dosyalara gerek duyacaktır.

b) Çevrim çalma (*Cycle stealing*): DMAC yolu ister ve bir sözcük aktarıp yolu MİB'e geri verir.

DMAC tüm veri aktarılıncaya kadar her sözcük için yeniden istekte bulunur.

Bu yöntem MİB'in uzun süre etkisiz kalmaması gereken durumlar için uygundur.

Hatırlatma: MİB, komut alma, operand alma ve eğer gerekli ise operand yazma çevrimlerinde belleğe erişir.

Bunun dışındaki komut çözme çevriminde ve iç saklayıcılar üzerindeki hesaplamalar sırasında MİB belleğe erişmez.

Bu çevrimlerde DMAC aktarım yaparken MİB de kendi işine devam edebilir.

Bu kipte veriler blok aktarıma göre daha uzun sürede aktarılır.

DMA Aktarım Kipleri (*devamı*) :

c) Saydam aktarım (*Transparent mode, Hidden DMA*): DMAC sistem yoluna sadece MİB tarafından kullanılmadığı zamanlarda erişir.

DMAC veya ek bir donanım birimi sürekli sitem yolunu ve MİB'in yürüttüğü komutları gözler.

MİB, sistem yolunu belli sayıda saat çevrimi süresince kullanmayan bir komut yürütüyorsa DMAC bunu sezer ve bu sürelerde sistem yoluna erişerek aktarımı gerçekleştirir.

Bu yöntemde MİB yoldan yalıtılarak yavaşlatılmaz.

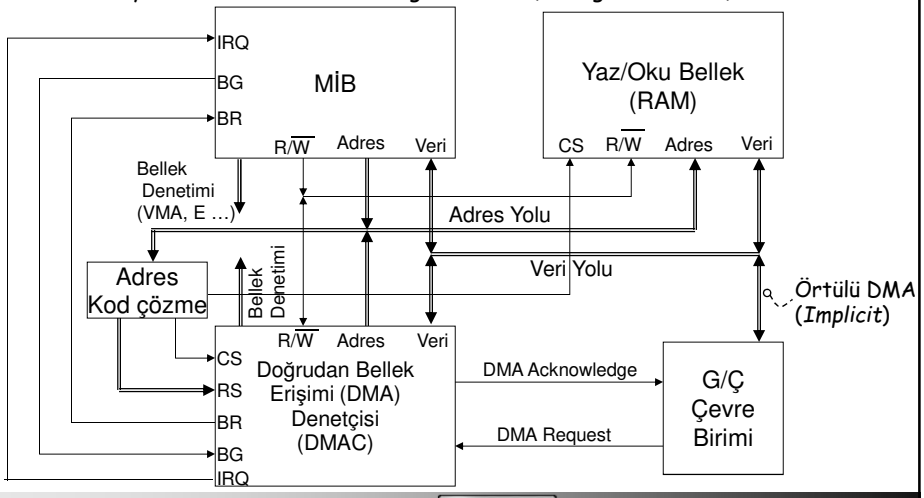
Veri aktarımı diğer kiplere göre daha uzun sürebilir.

Bu yöntemin maliyeti yüksektir, çünkü MİB'in yürüttüğü komutları gözleyip yorumlayabilecek ek bir donanıma gerek vardır.

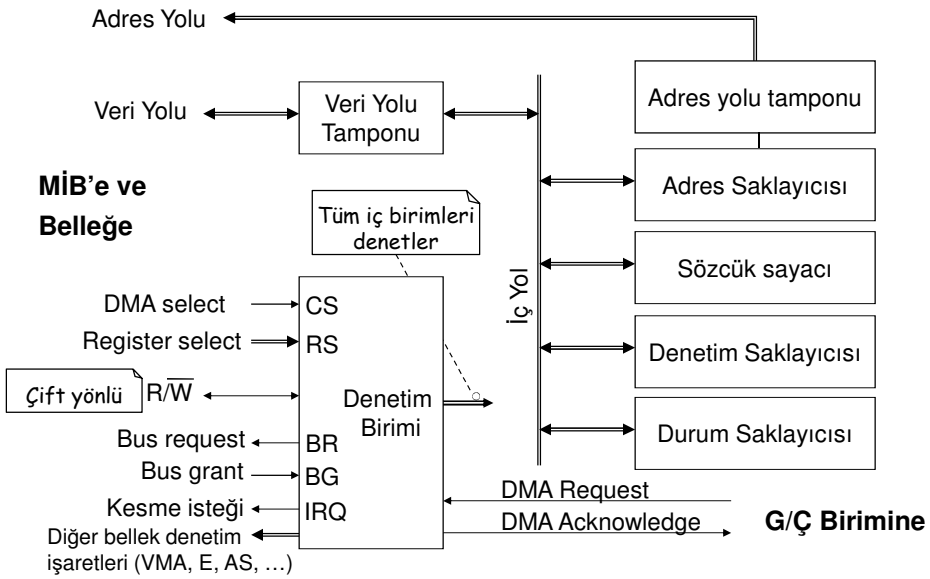
Blok aktarma ve çevrim çalma en yaygın kullanılan yöntemlerdir.

5.4 Örnek Bir MİB-DMAC Bağlantısı:

İletişim başlamadan önce DMAC MİB tarafından programlanır (ilgili saklayıcılarına gerekli bilgiler yüklenir).
Bu aşamada DMAC sistemde bir bellek ya da G/Ç birimi gibi davranır.
DMAC sistem yolunu aldıktan sonra belleği adresler (MİB gibi davranır).



Örnek Bir DMAC İç Yapısı:



5.5 DMA yönteminde veri aktarımının adımları:

1. DMA denetçisi (DMAC), MİB tarafından programlanır.
   G/Ç adresi, bellek adresi, oku/yaz, veri miktarı, çalışma şekli vs. DMAC denetim saklayıcılarına yazılır.
   Bu aşamada DMAC MİB tarafından adreslenen bir bellek ya da G/Ç birimi gibi davranır. Bu aşamada DMAC'ın R/W' hattı giriştir.
2. G/Ç birimi DMAC'dan istekte bulunur ("hazır" ya da "al") (DMA Request).
3. DMAC, BR'yi etkin yaparak MİB'ten sistem yolunu ister.
4. MİB yol çevrimini (komutu değil) tamamlar,
   sistem yolundan çekilir (3. konuma – yüksek empedans- geçer),
   yolu verdiğini BG ile bildirir.
   MİB DMAC'tan gelen istekleri kesmelerde olduğu gibi **maskeleyemez** (engelleyemez).

DMA yönteminde veri aktarımının adımları (devamı):

5. Bu aşamada sistem yolunun sahibi DMA denetçisidir. Belleği (MİB'in yaptığı şekilde) uygun işaretlerle adreslemek denetçinin görevidir.
   Bu adımda DMAC'ın R/W hattı çıkıştır.
   DMAC adres, R/W ve bellek erişimi için gerekli diğer denetim işaretlerini çıkartır.

   a) Örtülü (*Fly-by, implicit*) DMA:
   DMAC G/Ç arabirimini uyarır (DMA Acknowledge).
   G/Ç birimi veri aktarımını yapar (örtülü "*implicit*").

   b) Açık (*Flow-through, explicit*) DMA :
   DMAC G/Ç arabirimini uyarır (DMA Acknowledge).
   DMAC veriyi G/Ç biriminden alır ve belleye yazar
   veya
   DMAC bellekten veriyi okur, G/Ç birimini uyarır (DMA Acknowledge), veriyi G/Ç birimine yazar.

DMA yönteminde veri aktarımının adımları (devamı):

6. Eğer G/Ç biriminden yeni istek geldiyse (DMA Request) , DMAC BR'yi etkin tutar.
   Burst mode: Blok tamamlanıncaya kadar BR etkin kalır, yol bırakılmaz.
   Cycle stealing: Bir çevrim MİB'e bırakılır, sonra yol tekrar istenir.
   DMA denetçisi veri aktarımı yaparken, MİB yol erişimi gerektirmeyen işlemleri (örneğin komut çözme, iç saklayıcılar üzerindeki hesaplamalar) yapmaya devam edebilir. (Taramalı veya kesmeli aktarımda nasıldır?)
7. G/Ç biriminden yeni istek (DMA Request) gelmiyorsa veya tüm blok tamamlandıysa (DMAC sözcük sayacı sıfırlanmıştır) DMAC sistem yolundan çekilir, yani sürdüğü denetim hatlarını 3ncü konuma çeker.
   DMA BR'yi etkisiz yapar. Sistem yolu tekrar MİB'in kontrolüne geçer.

DMA yönteminde veri aktarımının adımları (devamı):

8. Eğer kesme üretecek şekilde koşullandıysa, DMAC MİB'e kesme göndererek tamamlan iş hakkında rapor verebilir.

   MİB kesme hizmet programında DMAC'ın durum saklayıcılarını okuyarak aktarım hakkında bilgi alabilir (kaç sözcük aktarıldı, hata oldu mu).

   DMA yönteminde kesmelerin yolun alınıp verilmesiyle ilgisi yoktur.

Örnek 1: DMA Yöntemi ile G/Ç İşlemleri

Problem:

Bir MİB'in komut çevrimi aşağıda süreleri verilen 5 adımdan oluşmaktadır.

1. Komut alma: 60 ns, 2. Komut çözme: 20 ns, 3. Operand alma: 60 ns, 4. Yürütme: 30 ns, 5. Kesme (eğer gerekli ise): 200 ns.

MİB, komut ve operand alma çevrimlerinde belleğe erişmekte ama komut çözme ve yürütme çevrimlerinde erişmemektedir.

Bellek erişim süresi ve G/Ç arabirimi erişim süresi her ikisi de 50 ns'dir.

Sistemde 2-hatlı (BR, BG) bir doğrudan bellek erişimi denetçisi (DMAC) bulunmaktadır. DMAC, G/Ç arabiriminden belleğe **çevrim çalma** (*cycle-stealing*) yöntemiyle veri aktarmaktadır. DMAC **örtülü** (*fly-by / Implicit*) olarak çalışmaktadır. Veri DMAC üzerinden geçmemektedir.

MİB programı koşmaya başladığı anda saati başlattığımızı varsayalım (Saat = 0).

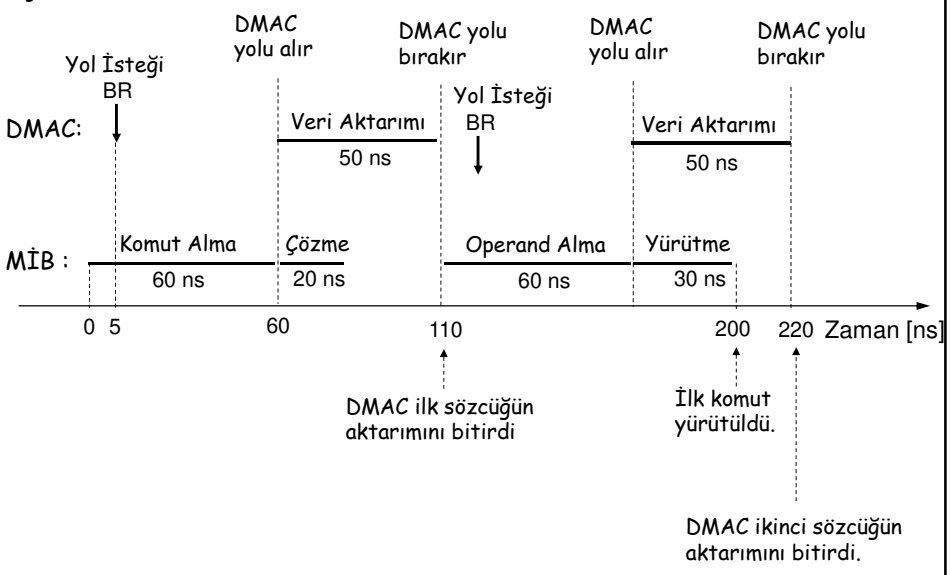
MİB ilk komutun alma çevrimindeyken (Saat = 5ns) DMAC veri aktarımını başlatmak için istekte (BR) bulunmaktadır.

a. DMAC ilk sözcüğün aktarımını ne zaman bitirir (Saat = ?)? Neden?

b. MİB ilk komutu yürütmeyi ne zaman bitirir (Saat = ?)? Neden?

c. DMAC 10 sözcüğün tamamının aktarımını ne zaman bitirir (Saat = ?)? MİB tüm programı yürütmeyi ne zaman bitirir (Saat = ?)?

Çözüm:



Çözüm (devamı):

a) MİB o andaki yol çevrimini tamamlar (komut alma) ve kendini sistem yolundan yalıtır.
DMA denetçisi ilk sözcüğü aktarır. DMA denetçisi örtülü olarak çalıştığı için veri aktarımı 50ns sürer.
Zaman= 60 + 50 = 110ns

b) MİB'in komut çözme ve operand alma çevrimleri DMA aktarımları ile paralel yürüyebilir.

DMA denetçisi **çevrim çalma** yöntemiyle çalıştığı için ilk sözcüğü aktardıktan sonra yolu MİB'e verir. MİB operandı aldıktan sonra komutu yürütür.

Zaman= 60 + 50 + 60 + 30 = 200ns

c) DMA denetçisi bir komut çevriminde **iki sözcüğü 220 ns'de** aktarmaktadır.
10 sözcük 5 x 220 =1100ns'de aktarılır.
Zaman = 5 * 220 = 1100ns

DMAC 10 sözcük aktarırken MİB 5 komut yürütmüş olur.
On sözcük aktarıldıktan sonra MİB geri kalan her komutu 170ns'de yürütür.
Zaman= 1100 + 5*170 = 1950ns

Bu süreleri 4.17 numaralı yansıdaki kesmeli G/Ç örneği ile karşılaştırınız.

Örnek 2: DMA ve Kesme

Problem:

Bir MİB'in komut çevrimi aşağıdaki 5 durumu (çevrimi) içermektedir:

1. Komut alma ve Çözme: 60ns, 2. Operand alma: 70ns, 3. Yürütme: 30ns, 4. Sonuç yazma: 60ns, 5. Kesme: 200ns.

MİB belleğe sadece "3. Yürütme" çevriminde erişmemektedir. Diğer çevrimlerde (1, 2, 4, 5) belleğe erişilmektedir.

Bellek erişim ve G/Ç arabirimi erişim sürelerinin her ikisi de 50 ns'dir.

Bu sistemde bir kesme kaynağı (KK) cihaz bulunmaktadır. Kaynağa ilişkin kesme hizmet programının (KHP) süresi 2500 ns'dir.

Sistemde 2-hatlı bir doğrudan bellek erişimi denetçisi (*Direct Memory Access Controller*- DMAC) bulunmaktadır.

DMAC, G/Ç arabiriminden belleğe **çevrim çalma** (*cycle-stealing*) yöntemiyle veri aktarmaktadır. DMAC **örtülü** (*fly-by / Implicit*) olarak çalışmaktadır (veri DMAC üzerinden geçmemektedir)

MİB programı koşmaya başladığı anda saati başlattığımızı (Saat = 0) varsayalım.

Problem: (devamı)

Saat = 10ns'de KK kesme isteğinde bulunuyor.

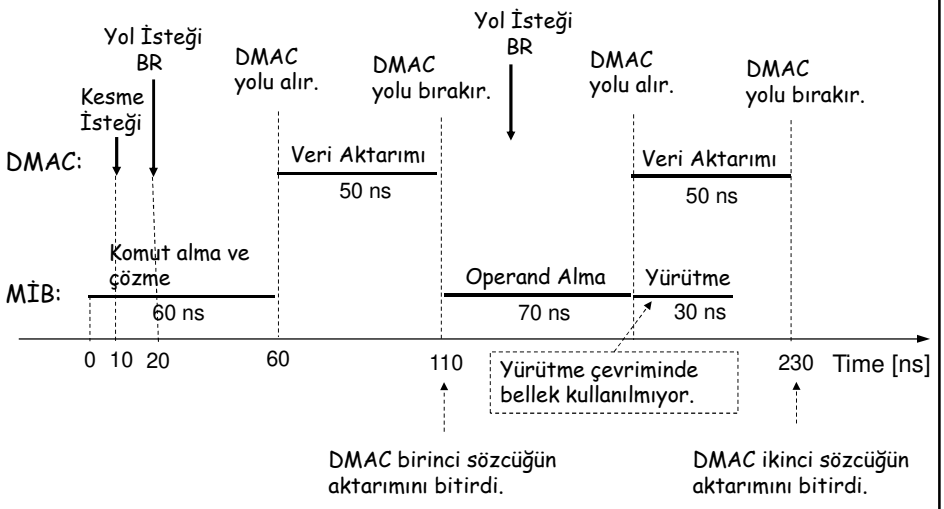
Saat = 20ns'de DMAC G/Ç arabiriminden belleğe veri aktarımı başlatmak istiyor. G/Ç arabiriminin her zaman veri aktarımına hazır olduğunu varsayıyoruz.

a. DMAC birinci, ikinci ve üçüncü sözcüklerin aktarımını ne zaman bitirir (Saat = ?)? Neden?

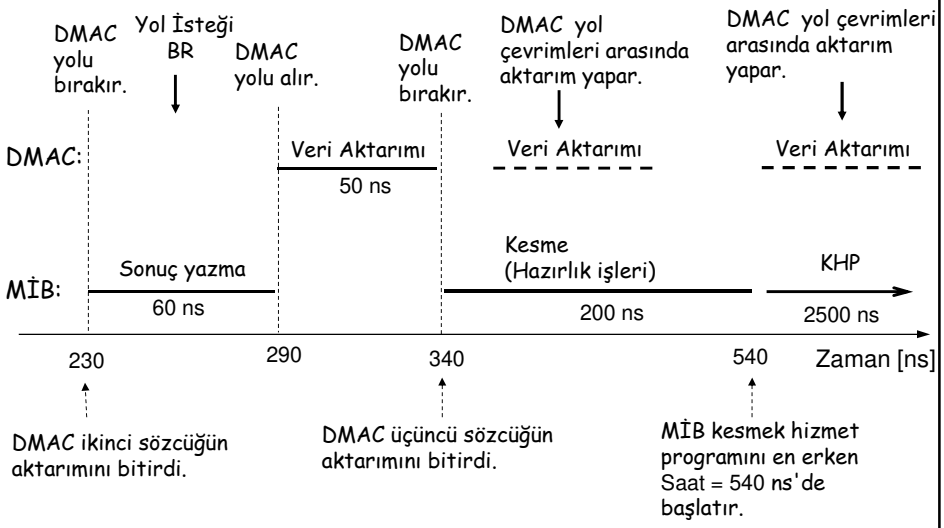
b. Kesme kaynağına ilişkin KHP ne zaman çalışmaya başlar (Saat = ?)? Neden?

c. KHP yürütülürken DMAC'ın hala aktaracağı sözcükler varsa ne olur?

Çözüm:



Çözüm (Bir önceki yansıdan devam):



Çözüm (devamı):

Hatırlatma: MİB kesme isteklerini komut tamamlandıktan sonra değerlendirir.
Eğer kesme isteği varsa ve kesmelere izin verildiyse MİB Kesme çevrimine girer.
Kesme çevriminde birçok yol çevrimi (bellek erişimi) oluşur: vektör numarası okunur, geri dönüş adresi ve durum bilgisi yığına yazılır.
Eğer DMA denetçisinin aktaracak verisi varsa bu yol çevrimlerinin arasında aktarım yapar.

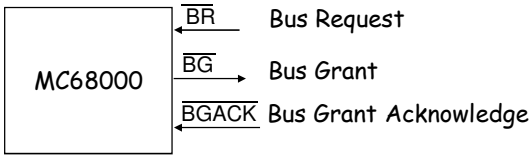
b. MİB kesmek hizmet programını en erken Saat = 540 ns'de başlatabilir.

c. KHP de komutlardan oluşan bir programdır.
Bu nedenle DMAC, KHP çalışırken de veri aktarımına devam edebilir.

5.6 Üç Hatlı Doğrudan Bellek Erişimi

Önceki bölümde anlatılan iki hatlı (BR, BG) DMA erişiminden farklı olarak üç hat ile çalışan merkezi işlem birimleri ve DMA denetçileri de vardır.
Birden fazla denetçinin bulunduğu sistemler için uygundur.

Örnek MC68000:
MC68000'in DMA hatları üç hatlı olarak hazırlanmıştır.



MC68000'in DMAC ile çalışmasının adımları:

1. DMAC BR'yi etkin yaparak yolu ister.
2. 68000 hemen BG'yi etkin yapar.
   Bu işlemcinin yol çevrimini bitirdiği ve yolu verdiği **anlamına gelmez**.
   Yolu vermeye hazır olduğunu, bir sonraki yol sahibinin belirlenebileceğini belirtir.
   Sistemde birden fazla DMAC olabilir. Yolu kullanım hakkını kimin alacağını belirlemek gerekir (öncelikler).
   Ek bir donanım birimi (yol hakemi) yolun bir sonraki sahibini belirler.

MC68000'in DMAC ile çalışmasının adımları (devamı):

3. Eğer birden fazla DMAC varsa yol hakemi (*bus arbiter* bir sonraki yol sahibini belirler.

   İşlemciden gelen BG işaretini alan bir sonraki yol sahibi o andaki yol çevriminin bitmesini bekler (AS, DTACK ve BGACK etkisiz olmalıdır).

   AS'nin etkisiz olması bir önceki yol sahibinin yol çevrimini tamamladığını gösterir. (AS etkinken yol çevrimi bölünüp yol başka bir birime verilemez)

   DTACK'in etkisiz olması MİB'in veya G/Ç birimlerinin yolu kullanmadığını gösterir.

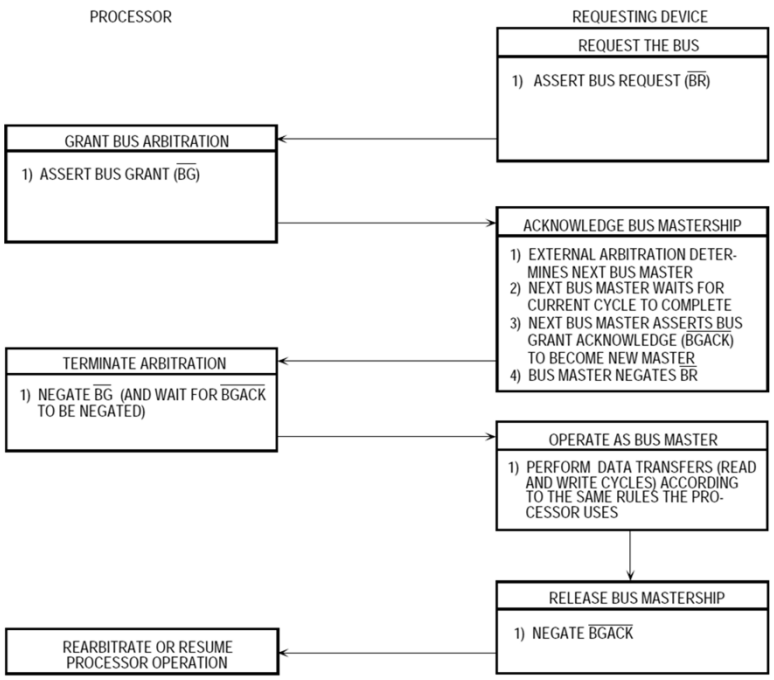
   BGACK etkisiz olması yolun bir önceki sahibinin yolu bıraktığını gösterir.

4. Yolun sahibi BGACK işaretini etkin yapar ve yolu kullandığı sürece bu konumda tutar.

   BR etkisiz yapılır, böylece sıradaki birim istekte bulunabilir.

5. Veri aktarımı tamamlanınca BGACK etkisiz yapılır.

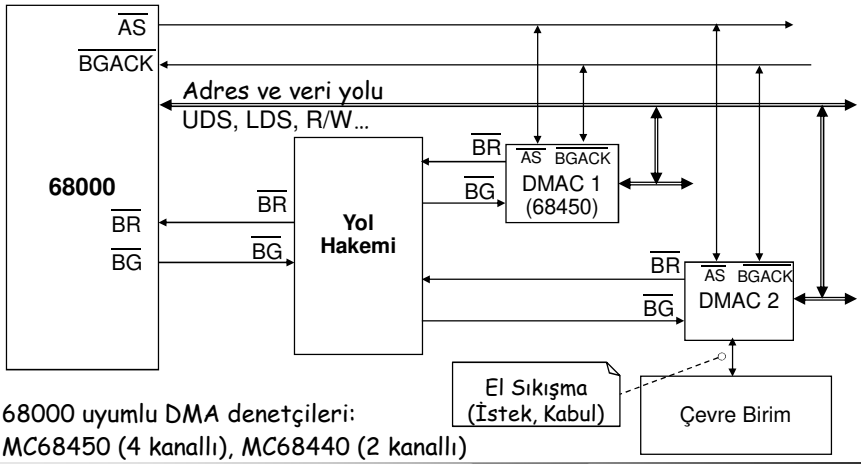
   BGACK girişi etkin olduğu sürece 68000 yol erişimi yapamaz.



Yol Hakemi (*Bus Arbiter*)

Sadece bir MİB ve bir DMA denetçisi olan sistemlerde sistem yolunun kullanılması konusunda DMA denetçisinin önceliği vardır.

Sistem yolunu kullanan birden fazla yol denetçisinin olduğu durumlarda bir yol hakemine (*bus arbiter*) gerek vardır.



68000 uyumlu DMA denetçileri:
MC68450 (4 kanallı), MC68440 (2 kanallı)

Sistemin Çalışması:

DMA denetçisi üç hatlı (BGACK) yapıyı kullanıyorsa (örneğin MC68450)

1. DMA denetçisi isteğini yol hakemine BR ile iletir.
2. Yol hakemi isteği 68000'e (BR) iletir.
3. Yol hakemi 68000'den gelen kabul işaretini (BG) istekte bulunan en yüksek öncelikli DMA denetçisine iletir.
4. BG onayını alan DMA denetçisi AS ve BGACK işaretlerini gözleyerek bir önceki yol çevriminin tamamlanmasını ve yolun önceki sahibinin yolu bırakmasını bekler.

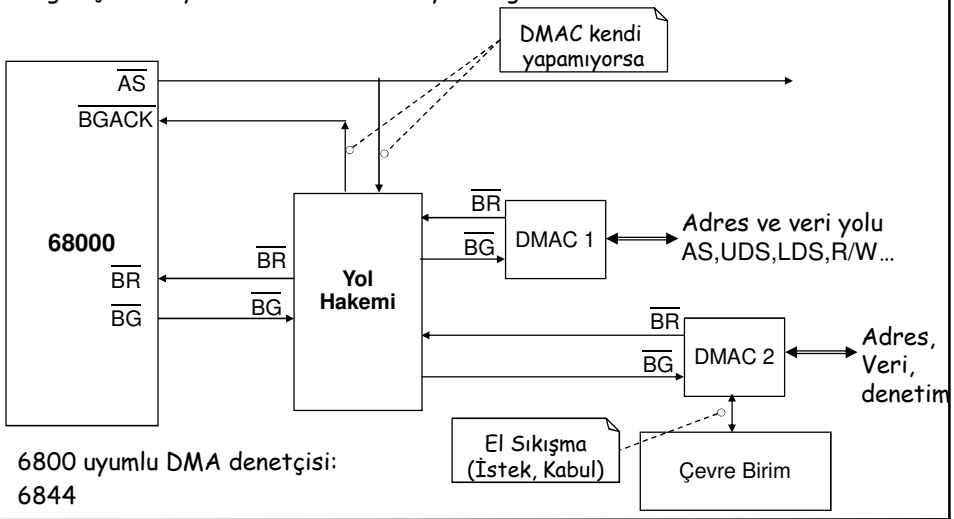
   AS bir önceki yol çevriminin bitip bitmediğini, BGACK ise bir önceki yol sahibinin yolu bırakıp bırakmadığını gösterir.
5. DMA denetçisi yolu aldıktan sonra BGACK işaretini kendisi etkin yaparak yolu ele geçirdiğini 68000'e ve diğer DMA denetçilerine bildirir.

   Yolu ele geçiren DMA denetçisi BR çıkışını etkisiz yaparak başka denetçilerin istekte bulunmasına olanak sağlar. Ancak aktarımı bitene kadar BGACK işaretini etkin tuttuğu için yol denetimi kendisinde kalır.

   Bundan sonra sistem yolunun sahibi DMA denetçisidir ve tüm bellek erişim işlerini yapmak (adresleme, AS, UDS, LDS, VMA, R/W …) onun sorumluluğundadır.
6. Tüm DMA denetçileri aktarımlarını tamamladıklarında 68000'in BGACK girişi etkisiz olur. Bu durumda 68000 tekrar sistem yolunu kullanmaya başlar.

İki hat kullanan (BR,BG) Denetçilerin 68000'e bağlanması:

Yol denetimi için sadece iki hat kullanan (BR, BG) ve BGACK işlevlerine sahip olmayan DMA denetçilerin 68000 mikroişlemcisine bağlanması durumunda BGACK ile ilgili işlemler yol hakemi tarafından yerine getirilmelidir.



6800 uyumlu DMA denetçisi: 6844

Sistemin çalışması:

DMA denetçisi sadece iki hat kullanıyorsa

1. DMA denetçisi isteğini (BR) yol hakemine iletir.
2. Yol hakemi isteği 68000'e (BR) iletir.
3. 68000'den kabul işareti (BG) geldikten sonra yol hakemi AS işaretini gözleyerek bir önceki yol çevriminin tamamlanmasını ve (eğer varsa) yolun önceki sahibinin isteğinin (BR) sona ermesini bekler.
   Bu yöntemde BGACK işaretini yol hakemi kendi sürdüğü için bu hattı yoklamasına gerek yoktur.
4. Yol hakemi istekte bulunan en yüksek öncelikli DMA denetçisine kabul işaretini (BG) iletilir.
5. Yol hakemi BGACK işaretini etkin yapar böylece yolun bir DMA denetçisi tarafından alındığını 68000'e bildirir.
6. Bu tür DMA denetçileri iletişim sürdüğü sürece istek çıkışlarını (BR) etkin tutarlar.
   Bu durumda yol hakemi de BGACK işaretini etkin tutar.
7. İletişim işi sona eren DMA denetçisi istek çıkışını (BR) etkisiz yapar.
   Eğer başka bir denetçiden gelen istek yoksa yol hakemi BGACK işaretini etkisiz yaparak 68000'in yolu almasını sağlar.

5.7 G/Ç İşlemcisi (*I/O Processor*)

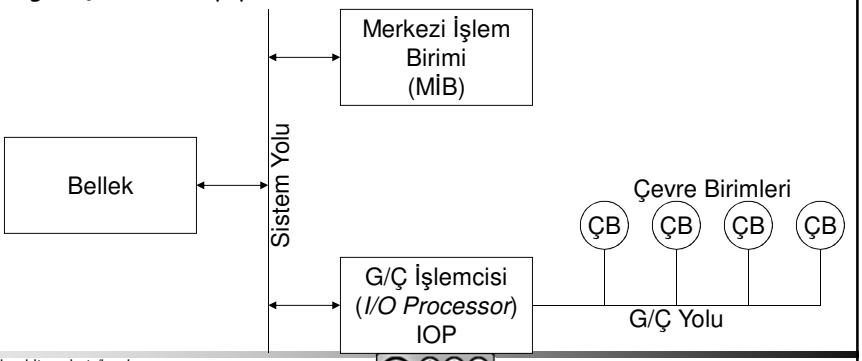
G/Ç işlemcileri; MİB, DMAC ve G/Ç arabirimlerinin bir araya gelmesiyle oluşmuşlardır. Bazı G/Ç işlemcileri yerel belleğe de sahiptirler.

Giriş çıkış işlemleri için özelleşmiş (özel komutlar) komut kümesine sahiptirler.

MİB, G/Ç işlemcisini G/Ç ile ilgili bir programı çalıştırmak üzere koşullar.

G/Ç işlemcisi kendisine atanan programın komutlarını alırken ve yürütürken DMA yöntemini kullanır.

Bu birimler veri aktarımı yanında, format dönüştürme, şifreleme, şifre çözme, hata düzeltme gibi işlemleri de yaparlar.



5.8 Bölünemez Yol Çevrimi (*Indivisible Bus Cycle*)

Semafor işlemlerini gerçekleştirmek için bölünemeyen tek bir yol çevriminde çalışan **Test-And-Set (TAS)** komutu kullanılır.

Tek bir komut ve tek bir yol çevrimi içinde

bellek okunur,

veri test edilir (sıfır ile karşılaştırılır),

değiştirilir ve

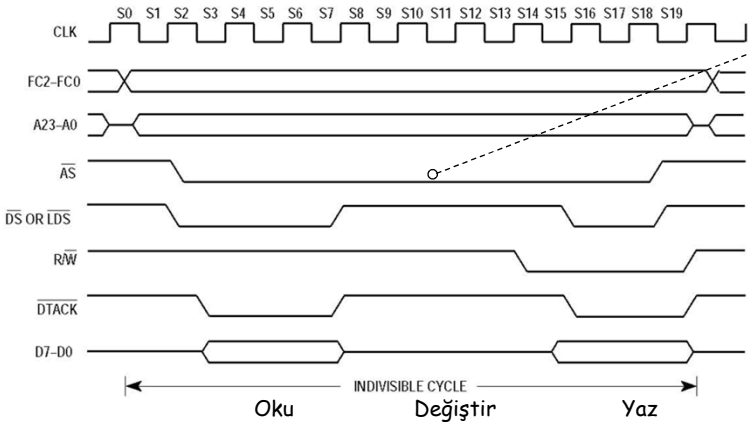
belleğe geri yazılır.

TAS komutu ve bu komutun kullandığı oku-değiştir-yaz (*read-modify-write*) yol çevrimi iki nedenden dolayı bölünemez.

- Üç işlem (oku/test, değiştir, yaz) tek bir komutla yapıldığından bu işlemler kesme istekleri ile bölünemez (Kesme istekleri komutları bölemez).
- Tüm yol çevrimi boyumca AS (*address strobe*) etkin durumdadır. DMA denetçileri (DMAC) ve diğer işlemciler bu yol çevrimini bölemezler ve belleğe erişemezler.

Örnek: MC68000 tabanlı sistemlerde oku-değiştir-yaz (*read-modify-write*) yol çevrimi

Oku-değiştir-yaz (*read-modify-write*) yol çevrimi zamanlama diyagramı:
(TAS "Test and set" komutunda kullanılır)



AS sürekli etkin olduğu için bir DMAC yolu alamaz.

Üç işlem aynı komut içinde yapıldığı için kesme ile de bölünemez.

Read-Modify-Write Cycle Timing Diagram

TAS **Test and set an operand**
Yazım: TAS <ea>
İşlem: [CCR] ← tested([operand]); [destination(7)] ← 1
Operandı test eder: Verinin değerine göre Z ve N bayraklarını etkiler.
Operandın en yüksek anlamlı bitini 1'ler.
Bu işlemler bölünemez.
Semafor işlemleri için kullanılır.

Örnek1: TAS komutu olmadan kritik bölge denetimi (sakıncalı !)

```
TEST    TST.B   BAYRAK      (Bölünebilir)
        BMI     TEST        Negatif mi (MSB=1?)
KRITIK  OR.B    #$80,BAYRAK Semafor kapatılıyor
        ......              Kritik bölge
        ......
SON     CLR.B   BAYRAK      Semafor açılıyor
```

Bu iki komut bölünebildiği için bu program sağlıklı olarak çalışmaz.

Örnek2: TAS komutu ile kritik bölge denetimi (doğru)

```
TEST    TAS     BAYRAK      Semafor test ediliyor ve gerekiyorsa kilitleniyor
        BMI     TEST
KRITIK  ......              Kritik bölge
        ......
SON     CLR.B   BAYRAK      Semafor açılıyor
```