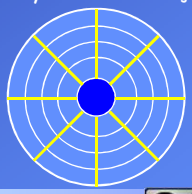
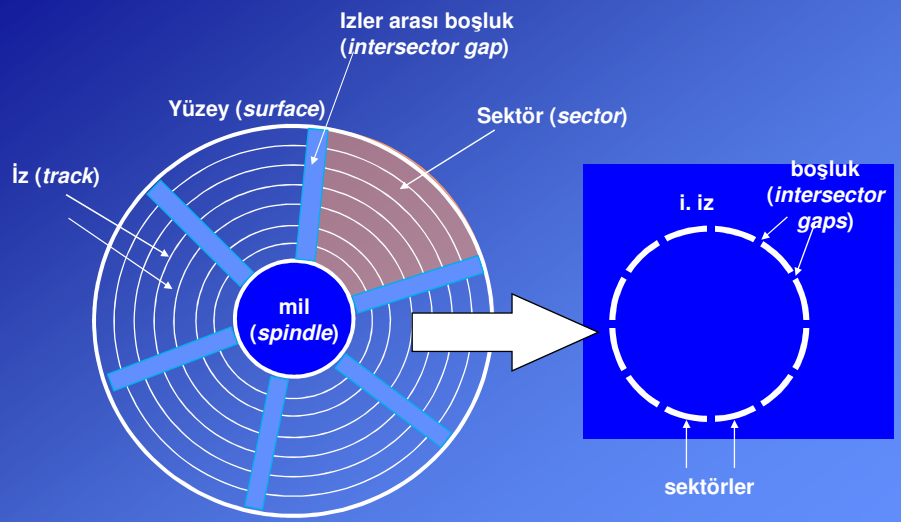
7 Dış Bellek (*External Memory*), Disk Sistemi

7.1 Manyetik Disk (Dış Bellek)

- Her plakada iki yüzey bulunur.
- Veriler, yüzeylerdeki **iz** (***track***) adı verilen eş merkezli halkalar (izler) üzerine yazılır (okunur).
- Bir yüzeyde binlerce iz bulunur.
- Komşu izler belli büyüklükte boşluklarla (*intertrack gap*) birbirlerinden ayrılırlar. Böylece okuma/yazma kafasının doğru izi okuması/yazması sağlanır.
- Veriler sektörler şeklinde aktarılır.
- Her izde yüzlerce sektör bulunur.
- Güncel sistemlerin çoğunda 512 sekizli uzunluğunda sabit uzunlukta sektörler kullanılır.
- Komşu sektörler de belli büyüklükteki boşluklarla (*intersector gap*) ayrılırlar.

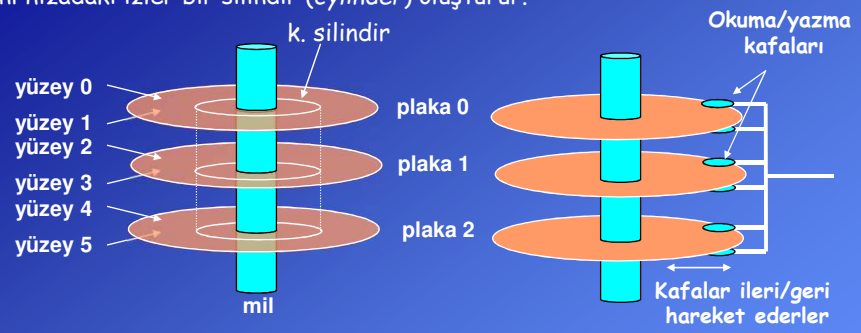


Manyetik disk düzeni:



Fiziksel Karakteristikler:

Aynı hizadaki izler bir silindir (*cylinder*) oluşturur.



Kapasite = (byte/sektör) x (ort. sektör/iz) x (iz/yüzey) x (yüzey/plaka) x (plaka/disk)

Örnek:

512 byte/sektör
300 sektör/iz (ortalama)
20,000 iz/yüzey
2 yüzey/plaka
5 plaka/disk

Kapasite = 512 x 300 x 20000 x 2 x 5
= 30,720,000,000 byte
≈ 28.6 GB

Disk Performansı

Bir diskin ortalama erişim süresi üç bileşenden oluşur:

Erişim süresi (Ta) = Konumlanma süresi (Ts) + Dönüş gecikmesi (Tr) + Aktarım Süresi(Tt)

- **Ortalama konumlanma süresi (*Seek time*) Ts:**

Okuma/yazma kafasının ilgili ize konumlanması için geçen süre. Yaklaşık 9ms (3-15ms)

- **Ortalama dönüş gecikmesi (*Rotational latency*) Tr:**

Okuma/yazma kafasının, iz içinde ilgili sektörün başına konumlanması için geçen süre.
Kafa diskte ilgili izin üstüne konumlandıktan sonra disk denetçisi gerekli olan sektörün kafanın altına gelmesi için plakanın dönmesini bekler.
Bu bekleme süresi ortalama olarak diskin bir turunu tamamlaması için gerekli olan sürenin yarısı kadardır:

$$T_r = \frac{1}{2} r$$ r: Diskin bir tur dönüş süresi (saniye)

Genellikle disklerin dönüş hızları tur/dakika (RPM: *Revolution per minute*) olarak verilir. Buna göre dönüş gecikmesi saniye cinsinden aşağıdaki gibi hesaplanabilir:

$$T_r = \frac{1}{2} \frac{60}{RPM}$$

Örnek: 7200 RPM disk bir turunu 8.3 ms'de tamamlar. Buna göre ortalama dönüş gecikmesi yaklaşık 4ms'dir.
10000 rpm: 3ms, 15000 rpm: 2ms.

- **Aktarım Süresi (*Transfer time*) (Tt)**

İki farklı şekilde ifade edilebilir:

Bir sektörü aktarmak için geçen süre (Tts) veya

Belli miktarda sekizli aktarmak için geçen süre(Ttb).

a) Bir sektörü okumak için geçen süre (Tts):

$$T_{ts} = \frac{1}{ort.sektör/iz} \frac{60}{RPM} \text{ [saniye]}$$

Örnek: Bir diskin dönüş hızı 7200 RPM ise ve bir izinde ortalama olarak 400 sektör varsa bir sektörlük aktarım hızı aşağıdaki gibi hesaplanır:

Tts = 60/7200 RPM x 1/400 sektör/iz x 1000 ms/sec = 0.02 ms

b) Veri aktarım süresi (Ttb):

b: Aktarılacak byte sayısı, N: Bir izdeki byte sayısı

$$T_{tb} = \frac{b}{N} \frac{60}{RPM} \text{ [saniye]}$$

Örnek:

- Disk dönüş hızı = 7200 RPM
- Ortalama konumlanma süresi = 9 ms.
- Bir izdeki ortalama sektör sayısı = 400.

Buna göre:

- Dönüş gecikmesi = 1/2 x (60 s/7200 RPM) x 1000 ms/s = 4 ms.
- Aktarım süresi = 60/7200 RPM x 1/400 sektör/iz x 1000 ms/sec = 0.02 ms
- Erişim süresi = 9 ms + 4 ms + 0.02 ms

Erişim süresinin belirleyici bileşenleri konumlanma süresi ve dönüş gecikmesidir.

Disklerin Gelişimi:

Kapasite artışı, inç kareye (veya santimetre kare) düşen bit sayısı ile ölçülen alansal (*areal density*) yoğunluktaki artış ile ifade edilir.

Areal density = $\frac{Tracks}{Inch}$ on a disk surface $\times$ $\frac{bits}{Inch}$ *on a track*

1988 yılı civarında, alansal yoğunluktaki artış yılda%29 oranında olduğundan her üç yılda bir disk kapasiteleri iki katına çıkıyordu.

1988, 1996 yılları arasında yıllık artış oranı %60'a yükseldi.

1997, 2003, yılları arasında yıllık artış oranı %100'e yükseldi ve kapasite her yıl ikiye katlanmaya başladı.

2003 yılından sonar yıllık artış %30'a düştü.

2011 yılında ticari ürünlerdeki en yüksek yoğunluk 400 milyar bit/ in^2'dir.

Gigabyte başına ödenen maliyet de alansal yoğunluktaki artışa paralel olarak düşmüştür.

Gigabyte başına maliyet, 1983 2011 yılları arasında 1,000,000 kat iyileşmiştir.

Kaynak: John L. Hennessy, David A. Patterson "Computer Architecture, A Quantitative Approach", 5 ed., Morgan Kaufmann, 2011.

Disk – DRAM Karşılaştırması **(Kaynak:** Hennessy, Patterson)

DRAM gecikmesi, diskin gecikmesinden yaklaşık olarak 100,000 kat daha azdır.

DRAM maliyeti (gigabyte başına) disk maliyetinden 30 ila 150 kat daha yüksektir.

2011 yılında fiyatı yaklaşık olarak 400$ olan 600 GB kapasiteli bir disk 200 MB/s hızında veri aktarabilmektedir.

2011 yılında fiyatı yaklaşık olarak 200$ olan 4 GB kapasiteli bir DRAM modülü 16,000 MB/s hızında veri aktarabilmektedir.

Maliyeti DRAM'dan daha düşük ve hızı manyetik diskten daha yüksek olan veri saklama birimlerinin geliştirilmesi için çalışılmakta olmasına rağmen günümüze kadar bu birimlerin yerini tamamen alabilecek elemanlar oluşturulamamıştır.

Bu konuda başarıya en yakın eleman "**Flash bellektir**".

Bu yarı iletken bellekler aynı diskler gibi uçucu değildir (*nonvolatile*).

Flash bellek diskten yaklaşık olarak 100 ila 1000 kat daha hızlıdır.

Flash bellek maliyeti (gigabyte başına) diskten 15 ila 25 kat daha fazladır.

Flash bellek maliyeti (gigabyte başına) DRAM'dan 15 ila 20 kat daha düşüktür.

Flash bellekler taşınabilir cihazlarda yaygın olarak kullanılmaktadır çünkü güç tüketimleri disklerden çok daha düşüktür.

Disk ve DRAM'dan farklı olarak Flash belleklerin kullanım ömürleri kısadır (yaklaşık olarak 1 milyon defa yazma). Bu nedenle özellikle sunucu tipi bilgisayarlarda kullanılmamaktadırlar.

7.2 RAID: (*Redundant Array of Independent/Inexpensive Disks*)*
Fazlalıklı Bağımsız/Ucuz Diskler Dizisi

Veriler paralel çalışan birden çok diske dağıtılır.

Amaç: **Performansı** ve **güvenirliği** arttırmak.

- Paralel ve bağımsız diskler performansı arttırır.
- Fazlalık bilgi, hataları sezmek ve düzeltmek için kullanılır.

Düzeyler:

RAID 0 – RAID 6: 7 ana düzey ve bunların bileşiminden oluşan bileşik düzeyler var.

RAID 0 gerçek bir RAID sistemi değildir, çünkü fazlalık içermemektedir.

En çok RAID 3 ve 5 kullanılmaktadır.

Ortak özellikler:

1. Birden fazla fiziksel disk vardır. İşletim sistemleri bunları bir bütün, tek bir mantıksal disk olarak görür (gösterir).
2. Mantıksal olarak peş peşe gelen veriler, belli büyüklükteki bloklar (şerit –"*strip*") halinde farklı fiziksel disklere paralel olarak yerleştirilirler.
3. Fazlalık olarak, eşlik bilgileri (*parity*) yerleştirilir. Disklerden biri fiziksel olarak bozulduğunda bu diskteki bilgi tekrar oluşturulabilir.

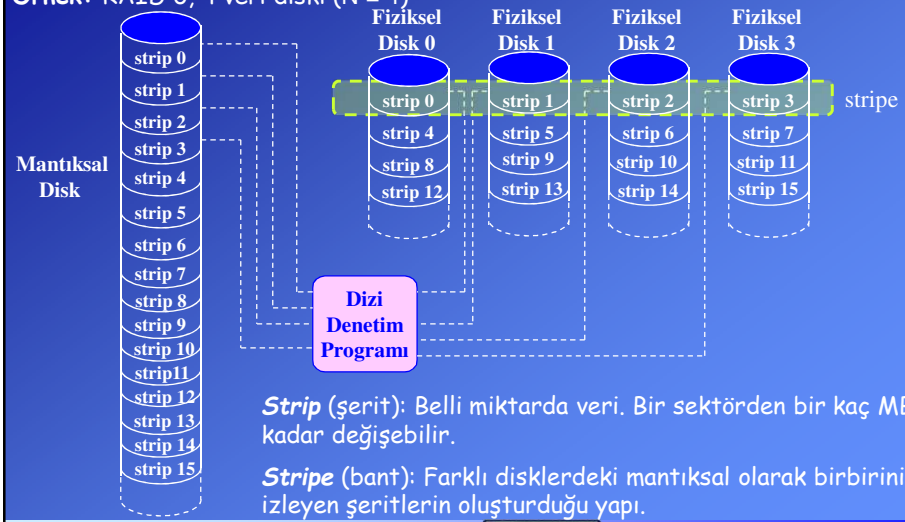
* Kaynak: W. Stallings, " Computer Organization and Architecture", 8/e, 2010.

RAID 0

Fazlalık ve eşlik biti yok. Hatalar düzeltilemez. Tam bir RAID sistemi değildir.

Veriler paralel erişilebilen fiziksel disklere dağıtılır, performans artımı sağlanır.

Örnek: RAID 0, 4 veri diski (N = 4)



Strip (şerit): Belli miktarda veri. Bir sektörden bir kaç MB'a kadar değişebilir.

Stripe (bant): Farklı disklerdeki mantıksal olarak birbirini izleyen şeritlerin oluşturduğu yapı.

RAID 0 (devamı)

Verim artışı (İki olası durum):

1. Bir G/Ç isteği, birbirini mantıksal olarak izleyen çok sayıda şerit içeriyorsa, N adet şerit aynı anda paralel olarak işlenebilir (N : paralel veri disklerinin sayısı).
   Böylece veri aktarım süresi büyük oranda azalmış olur.
2. Eğer aynı anda oluşan iki farklı G/Ç isteği, farklı veri blokları ile ilgiliyse büyük olasılıkla bu veriler farklı fiziksel disklerde olacaklardır.
   Böylece, bu iki istek paralel olarak aynı anda yerine getirilir ve G/Ç kuyruğunda bekleme süresi azaltılmış olur.

Şerit (*strip*) boylarının performansa etkisi:

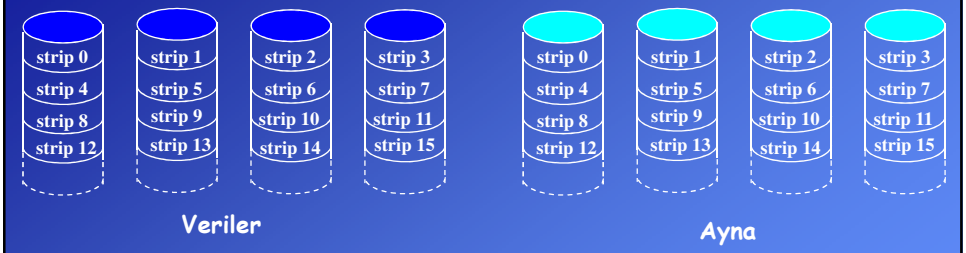
a) Eğer mantıksal olarak **peş peşe gelen büyük bloklara erişiliyorsa**
   Aktarım hızı önemlidir: Dosya kopyalama, video oynatma.
   Bu durumda **küçük şeritler** performansı arttırır.
   Mantıksal olarak birbirini izleyen veriler mümkün olduğu kadar farklı fiziksel disklere dağılır ve paralel erişim sağlanır.

b) Eğer **sık G/Ç istekleri varsa ve küçük bloklara erişiliyorsa** (örneğin rasgele, kısa ve sık veri tabanı sorguları)
   Büyük şeritler tercih edilir. Tek bir G/Ç isteği bir diske denk düşer, böylece çok sayıda farklı istek farklı disklerde aynı anda yerine getirilebilir.

RAID 1

Veriler aynalanır (*mirroring*). Her veri iki ayrı diske yazılır.

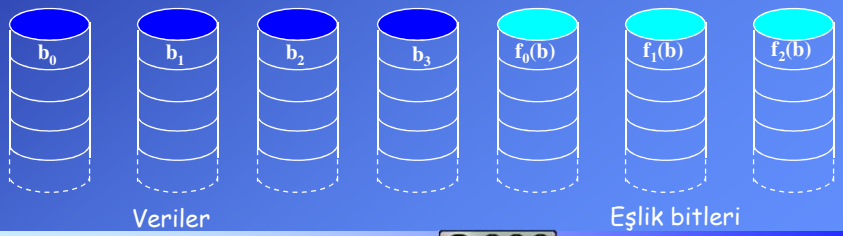
Örnek: RAID 1, 4 adet veri diski (N = 4)



- Veri disklerinin sayısı: N Toplam disk sayısı : 2·N
- Okuma isteği iki diske de aynı anda gönderilir. Daha hızlı olandan veri alınır.
- Yazarken veri iki diske de paralel yazılır. Bu durumda daha yavaş olan disk beklenir.
- Bir disk bozulduğunda veri diğerinden alınır.
  Fiziksel olarak hangi diskin bozulduğu bellidir.
- Fazla disk kullanıldığı için RAID 1 'in maliyeti yüksektir.

RAID 2

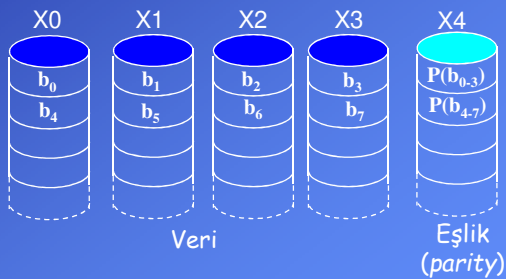
- Hata sezme/düzeltme için eşlik bitleri eklenir.
- Hata sezme için **Hamming kodu** kullanılır.
- Hamming kodlaması (Bölüm 7.3.1'de açıklanmıştır) belleklerde ve iletişimde hata sezme/düzeltme için kullanılır. İki bitlik hatalar sezilir, bir bitlik hatalar düzeltilir.
- RAID 2'de diskler **senkron** çalışır, tüm disklerdeki kafalar aynı yere konumlanır.
- **Küçük "strip"ler** (1 sözcük) kullanılır. Sürekli ve büyük blok erişimlerinde avantajlı.
- Aşağıdaki şekilde 4 bitlik veriye hata sezme için 3 bitlik eşlik eklenmiştir.
- Disklerin fiziksel olarak bozulduğunu anlamak mümkün olduğu için disklerde Hamming kodlaması gereksiz yere karmaşıklığı arttırmakta ve maliyeti yükseltmektedir.
- RAID 2'de RAID 1'e göre daha az disk kullanılmakla birlikte maliyet yüksektir.



RAID 3

- RAID 2'de olduğu gibi diskler **senkron** çalışır, tüm disklerdeki kafalar aynı yere konumlanır.
- **Küçük "strip"ler** kullanılır. Sürekli ve büyük blok erişimlerinde avantajlı.
- RAID 3, veri disklerinin sayısı ne olursa olsun hata düzeltme için sadece bir ek diske gerek duyar.
- Veri disklerinin sayısı: N Toplam disk sayısı: N+1
- Hata düzeltici bir kodlama yerine daha basit bir kodlama kullanılır ve tüm disklerde aynı konumda olan bitler için tek bir eşlik biti (*parity*) eklenir.

Örnek: RAID 3, 4 Veri + 1 Eşlik Diski (N = 4)



RAID 3 (devamı)

Eşlik (*parity*):

- Eşlik (*parity*) biti, veri bitleri "YA DA"lanarak (XOR "⊕" fonksiyonu ile) belirlenir.
  X0-X3 veri sözcükleri, X4 ise eşlik sözcüğü olmak üzere i. eşlik biti aşağıdaki gibi hesaplanır:
  X4(i) = X0(i) ⊕ X1(i) ⊕ X2(i) ⊕ X3(i) ; Böylece 1'lerin toplam sayısı çift olur.
- Normalde bu eşlik yöntemi sadece tek sayıdaki hataları sezebilir ama düzeltemez.
  Ancak fiziksel olarak hangi diskin bozulduğu belli ise eşlik bilgilerinden yararlanılarak o diskteki bilgiler yeniden oluşturulabilir.

  Örneğin; 1 numaralı disk bozulursa:
  Yukarıdaki denklemin her iki tarafına X4(i) ⊕ X1(i) eklenirse aşağıdaki ifade elde edilir.
  X1(i) = X0(i) ⊕ X2(i) ⊕ X3(i) ⊕ X4(i)
  Böylece X1 diskindeki tüm şeritlerin içeriği sağlam disklerin aynı sıradaki şeritlerinden tekrar elde edilebilir.
  Bu yöntem RAID3-RAID6 arasındaki tüm düzeylerde kullanılmaktadır.

RAID 3 (devamı)

Performans:

Hatırlatma: diskler **senkron** çalışır, tüm disklerdeki kafalar aynı yere konumlanır.

Okuma:

- Aynı satırda (*stripe*) (aynı iz/sektör) olan sözcükler aynı anda okunabilir.
  Örneğin, yansı 7.14'teki şekildeki sözcükler b_0, b_1, b_2, b_3 paralel olarak okunabilir.
- Farklı satırlarda (*stripe*) yer alan sözcükler sadece sırasal olarak okunabilirler.
  Örneğin b_0, b_5 sözcüklerini okumak için peş peşe iki okuma işlemi gereklidir.

  Örnek:
  Eğer disklerin bir sözcük okuma veya yazma için erişim süreleri **ta** ise,
  (b_0, b_1, b_2, b_3)'dan oluşan 4 sözcük okuma süresi: ta.
  (b_0, b_5)'dan oluşan 2 sözcük okuma süresi: 2·ta.

RAID 3 Performans (devamı)

Yazma:

- Tek bir sözcük bile yazılsa tüm disklere erişmek gerekir, çünkü eşlik bilgisini hesaplamak için yazma yapılan sözcük ile aynı sıradaki diğer verileri okumak gereklidir.
  Bu durum RAID 3'te ek bir soruna neden olmaz, çünkü diskler senkron çalıştığından zaten (farklı satırlara) bağımsız erişim mümkün değildir.
  Örneğin b_0, sözcüğünü değiştirirken b_1, b_2, b_3 sözcüklerinin okunması gerekir. Bu sözcükler aynı yerde (iz/sektör) olduğundan yazma ve okuma işlemleri aynı anda yapılır.
  Eşlik bilgisi hesaplandıktan sonra eşlik diskine yazılır.
- N adet sözcük farklı disklerde aynı yerlere (aynı iz/sektör), paralel olarak yazılabilir. Örneğin b_0, b_1, b_2, b_3 sözcüklerinin aynı anda değiştirilmesi mümkündür.
  Eşlik önceden hesaplanıp verilerle birlikte aynı anda yazılabilir.
  Diskler senkron çalıştığından eşlik diskinde konumlandırma ve dönüş gecikmesi oluşmaz.

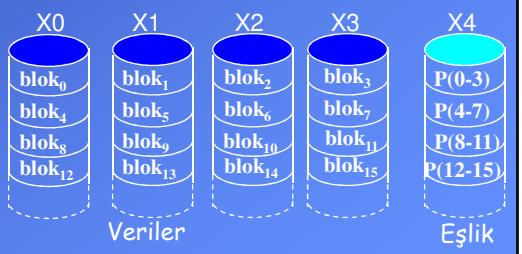
Özet:

- Senkron diskler ve küçük şeritler büyük miktardaki aktarımlar için uygundur (Dosya sunucuları "*file server*").
- Sık, bağımsız erişimlerde performans düşer.

RAID 4

- Diskler bağımsız çalışır (senkron değil).
  Farklı şeritlere denk düşen G/Ç işlemleri paralel olarak yapılabilir.
- Büyük "strip"ler (blok) kullanılır.
  Sık ve bağımsız okuma erişimlerinde avantajlı. Büyük ve sürekli veri aktarımları için uygun değil.
- Hata sezme/düzeltme için tek eşlik (*parity*) biti eklenir. Toplam disk sayısı : N+1
- Okuma işlemlerinde eşlik diskini okumaya gerek yoktur.
- Ancak her yazma işleminde eşlik diskine de yazmak gerekir.
- Diskler senkron olmadan bağımsız çalışsa da disklerin farklı yerlerine aynı anda yazmak mümkün olmaz çünkü eşlik diski tektir, beklemek gerekir.

- Yeni bir yazma işleminin yapılabilmesi için önceki yazmanın bitmesi beklenir.
- Eşlik diski performans açısından bir darboğaz oluşturur.



RAID 4 (devamı)

Yazma cezası (*write panalty*):

Her yazma işleminde ilgili veri bitleri ile birlikte eşlik bitini de güncellemek gerekir.

Örneğin; X0-X3 veri diskleri, X4 ise eşlik diski olsa
ve sadece X1'in bir şeridine yazılsa
i. eşlik biti (X4'(i)) aşağıdaki gibi hesaplanır:

X4'(i) = X0(i) ⊕ **X1'(i)** ⊕ X2(i) ⊕ X3(i) X1'(i), X4'(i), : Değişen veriler

Bu durumda 3 diskten okumak (X0, X2, X3), 2 diske yazmak (X4, X1) gerekir.

Tüm diskler meşguldür.

İşlemi kolaylaştırmak için sağ tarafa ⊕ X1(i) ⊕ X1(i) eklenir.

(Hatırlatma bir değer kendisiyle XOR işlemine girdiğinde sonuç lojik 0 olur.)

X4'(i) = X4(i) ⊕ X1'(i) ⊕ X1(i) elde edilir.

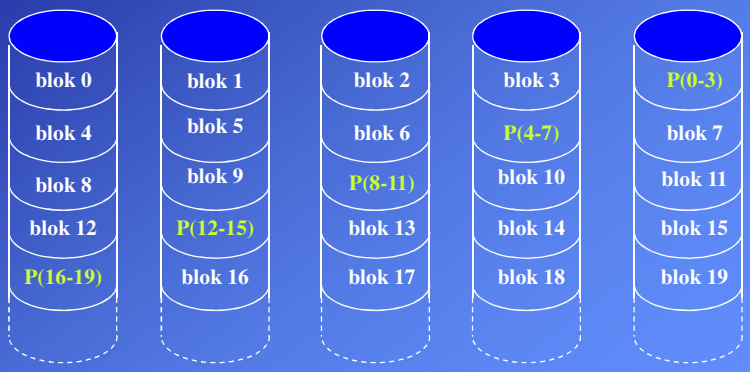
Bu durumda eşlik hesabı yapabilmek için iki okuma iki yazma gereklidir.

RAID yönetim yazılımı, yeni eşlik değerini (X4') hesaplamak için önce eski veri şeridini (X1) ve eski eşlik şeridini (X4) okur.

Daha sonra güncel veri (X1') ve hesaplanan eşlik değeri (X4') yazılır.

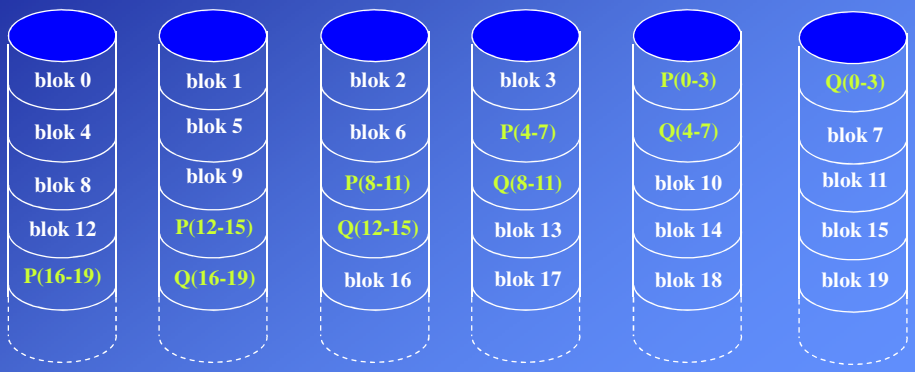
RAID 5

- RAID 4' e benzer. Diskler bağımsız çalışır (senkron değil).
- Büyük "strip"ler (blok) kullanılır. Sık ve bağımsız okuma erişimlerinde avantajlı.
- Hata sezme/düzeltme için eşlik tek eşlik biti eklenir. Toplam disk sayısı : N+1
- RAID 4'ten farklı olarak eşlik bilgileri disklere dağıtılır. Böylece her yazma işleminde aynı eşlik diskinin beklenmesi önlenmiş olur.



RAID 6

- İki eşlik bilgisi kullanılır ve bunlar disklerin farklı bölgelerine dağıtılır.
- İki eşlik bilgisi hata düzeltme miktarını arttırır; iki disk bozulduğunda da veriler yeniden üretilebilir.
- Veri disklerinin sayısı: N ise toplam disk sayısı : N+2

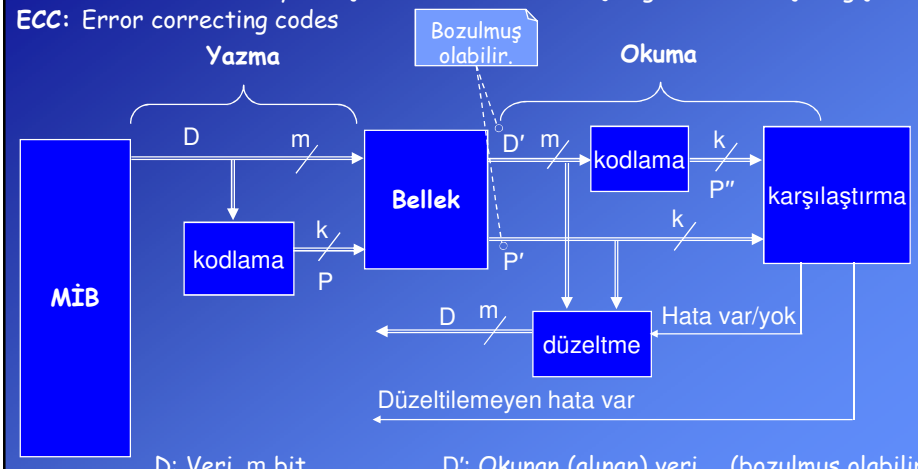


7.3 Belleklerde Hata Sezme/Düzeltme

Fiziksel Hata (*Hard error*): Kalıcı fiziksel (malzemede) bozukluk

***Soft error*:** Kalıcı olmayacak şekilde sadece bellek içeriğinin istem dışı değişmesi

ECC: Error correcting codes



D: Veri, m bit
D': Okunan (alınan) veri (bozulmuş olabilir.)
P: Eşlik, k bit
P': Okunan (alınan) eşlik (bozulmuş olabilir.)
P": Okunan veriden hesaplanan eşlik

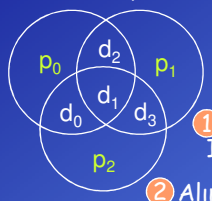
7.3.1 Bir bitlik hata düzelten Hamming Kodları (*Single Error Correction* – SEC)

Veri bitlerine, bir bitlik hatanın yerini bulmayı sağlayacak şekilde eşlik bitleri eklenir.

Eşlik bitlerini farklı şekillerde hesaplamak mümkündür.

Örnek: 4:7 Hamming kodu. (Richard Wesley Hamming (1915-1998), ABD)

4 bitlik veriye, 3 bit eşlik eklenir, toplam 7 bit **kod sözcüğü** iletilmiş (yazılmış) olur.



d_i: veri biti
p_i: eşlik biti

$p_0 = d_0 \oplus d_1 \oplus d_2$
$p_1 = d_1 \oplus d_2 \oplus d_3$
$p_2 = d_0 \oplus d_1 \oplus d_3$

1. İletilen kod sözcüğü: $d_0\, d_1\, d_2\, d_3\, p_0\, p_1\, p_2$ 4 bit veri + 3 bit eşlik
2. Alınan (okunan) sözcük: $d_0'\, d_1'\, d_2'\, d_3'\, p_0'\, p_1'\, p_2'$ bozulmuş olabilir
3. Alıcı tarafta eşlikler yeniden hesaplanır:
   $p_0'' = d_0' \oplus d_1' \oplus d_2'$
   $p_1'' = d_1' \oplus d_2' \oplus d_3'$
   $p_2'' = d_0' \oplus d_1' \oplus d_3'$
4. Alınan eşlikler ile hesaplanan eşlikler karşılaştırılır (XOR):
   Sendrom sözcüğü
   $s_0 = p_0' \oplus p_0''$
   $s_1 = p_1' \oplus p_1''$
   $s_2 = p_2' \oplus p_2''$
5. Sendrom bitlerinin (s_i) hepsi sıfırsa hata yok demektir. Sendrom sıfırdan farklı ise hatalı bitin yeri belirlenir ve tümlenerek düzeltilir.

Sendrom etkileşim tablosu:
(*Syndrome impact table*):

Hangi sendrom bitinin hangi kod sözcüğü bitinden etkilendiğini gösterir.

	d_0	d_1	d_2	d_3	p_0	p_1	p_2
s_0	X	X	X		X		
s_1		X	X	X		X	
s_2	X	X		X			X

Sendrom Tablosu:

s_0 s_1 s_2	Anlamı
0 0 0	Hata yok
0 0 1	p_2 (bozulmuş)
0 1 0	p_1
0 1 1	d_3
1 0 0	p_0
1 0 1	d_0
1 1 0	d_2
1 1 1	d_1

Eşlik bitlerinin sayısının belirlenmesi:

Veri bitleri sayısı: m

Eşlik bitleri sayısı: k

Eğer k adet eşlik biti kullanılıyorsa sendrom sözcüğünün uzunluğu da k bit olur ve $[0\ ,\ 2^k - 1]$ aralığında değerler alabilir.

Sıfır değeri hata olmadığını gösterir.

Kalan $2^k - 1$ değer hangi bitte hata olduğunu gösterir.

Hata, m adet veri bitinde olabileceği gibi k adet eşlik bitinde de olabilir.

Buna göre: $\mathbf{m + k \le 2^k - 1}$ olmalıdır.

7.3.2 Bir bitlik hata düzeltme, iki bitlik hataları sezme: (*Single error correction – double error detection* SEC-DED)

Önceki ölümde anlatılan kodlama tek bitlik hataları düzelteme yeteneğine sahiptir (*single-error correcting code* "SEC").

Bu kodlamaya iki bitlik hataları sezme yeteneği kazandırmak için her kod sözcüğünün sonuna bir eşlik biti daha eklenir.

Bu eşlik biti, kod sözcüğündeki 1'lerin sayısını tek ya da çift yapacak şekilde seçilir.

İletilen kod sözcüğü: $d_0\, d_1\, d_2\, d_3\, p_0\, p_1\, p_2\, q$ 4 + 3 + 1 bit

d: Veri , p: Hata düzeltme eşik bitleri, q: Tek/çift eşlik

Eğer alıcı taraftaki sendrom sıfırdan farklı (hata var) ise ve ek eşlik biti "hata yok" sonucu veriyorsa çift sayıda hata olmuş demektir.

Bu yöntem iki bitlik hataları düzeltemez ancak en azından hatalı veri kullanılmadan silinebilir.

En yaygın kullanılan SEC-DED kodlaması 64 + 7 + 1 bitlik kodlamadır.

Bu kodlamada %12.5 fazlalık vardır.