Statik ve Dinamik RAM
Salt Oku Bellek (ROM)
Programlanabilir Lojik Dizisi (PLA)
Programlanabilir Dizi Lojiği (PAL)
Asenkron Ardışıl Devreler

BIL-204: Lojik Devreler II

Dersi veren öğretim üyesi:

Yrd. Doç. Dr. Fatih Gökçe

Süleyman Demirel Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği Bölümü

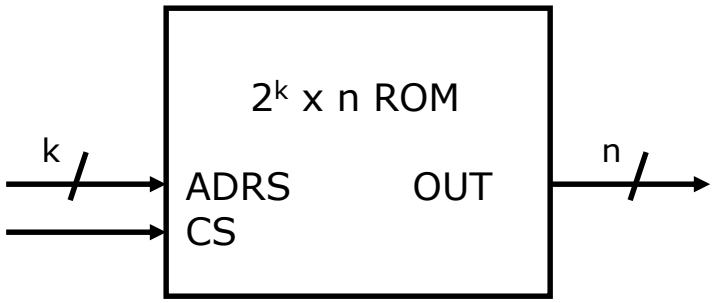
Bu ders notu "M. Morris Mano, Digital Design, 2nd Edition" kitabının çevirisi olan "Seta Boğosyan, Metin Gökaşan, Salman Kurtulan, Sayısal Tasarım, Literatür Yayıncılık" kitabından ve METU-CENG-232-Logic Design ders notlarından faydalanılarak hazırlanmıştır.

Statik ve Dinamik RAM

- Enerji kesildiğinde sakladıkları veri kaybolur (*volatile* hafıza).
- Statik RAM:
  - Bilgisayar ön belleği (cache) ve ekran kartlarında yaygın olarak kullanılır.
  - Her bir bit için bir tutucu (latch) veya flip-floptan oluşurlar (4-6 transistör gerekir).
  - Daha hızlıdırlar, daha fazla güç harcarlar.
  - Depolanmış bilgi, birimi besleyen enerji devam ettiği sürece geçerli olur.
- Dinamik RAM:
  - Bilgisayar ana belleğini oluşturan RAM'lerde yaygın olarak kullanılır.
  - Bilgi, kapasite elemanlarında elektrik yükleri biçiminde saklanır (1 bit için 1 kapasitör, 1 transistör gerekir.).
  - Daha yavaştırlar, daha az güç harcarlar.
  - Daha az fiziksel alanda daha fazla bilgi saklamaya imkan verirler.
  - Byte başına maliyetleri daha düşüktür.
  - Depolanmış bilgiyi koruyabilmek için kapasitörlerin belli aralıklarla sürekli olarak şarj edilmeleri (refresh) gerekir (Her birkaç milisaniyede bir.). Dinamik ismi buradan gelir.

Salt Oku Bellek (ROM)

- Bir *salt oku bellek (sadece okunur bellek, salt okunur bellek)* veya *ROM*, içeriği kolayca değiştirilemeyen özel bir hafıza türüdür.
  - Sadece okuma özelliği nedeniyle blok diyagram gösteriminde RAM'lerde bulunan WR ve DATA (veri girişi) girişleri bulunmaz.
  - ROM'lara veri özel donanımlarla yazılır, elektriğin kesilmesi durumunda veri silinmez (*nonvolatile* hafıza).
- ROM'lar değişmeyen veya nadiren değişen verileri saklamak için kullanışlıdır.
  - Logaritma ve bölme gibi bazı işlemlerin hızını arttırmak amacıyla, belli değerleri içeren tablolar ROM'lara kaydedilerek aritmetik devreler oluşturulabilir.
  - Bilgisayarlardaki BIOS.



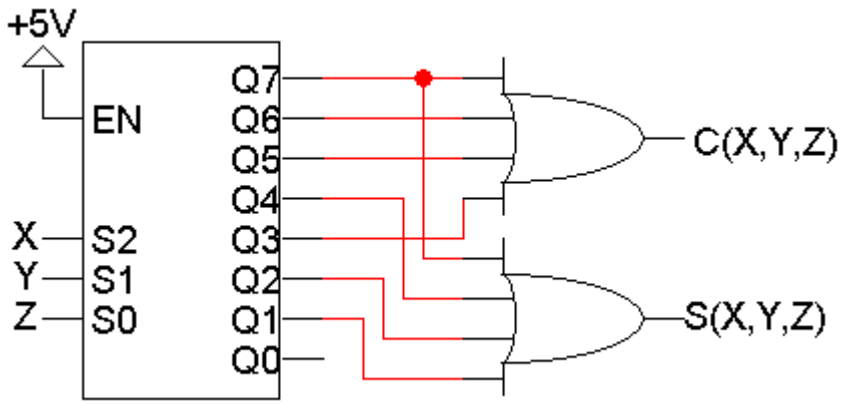
Hafızalar ve Fonksiyonlar

- ROM'lar aslında kombinasyonel devrelerdir, ardışıl değil!
  - Bir adres, veri ROM'a yazıldıktan sonra daima aynı veriyi içerir.
  - ROM, giriş olarak adres değerini alan, çıkışında adres değerine uygun olarak bir çıkış verisi üreten bir kombinasyonel devre olarak düşünülebilir.
- Bir ROM tablosu basit olarak bir doğruluk tablosudur.
  - Tablo her bir ROM adresinde saklanan veriyi gösterir.
  - Bu veri kombinasyonel olarak adresi giriş olarak kullaarak üretilebilir.

Adres $A_2A_1A_0$	Veri $V_2V_1V_0$
000	000
001	100
010	110
011	100
100	101
101	000
110	011
111	011

Dekoder kullanarak devre tasarımı

- Lojik-1 de dekoder (kod çözücü) kullanarak kombinasyonel lojik devre tasarımını görmüştük.

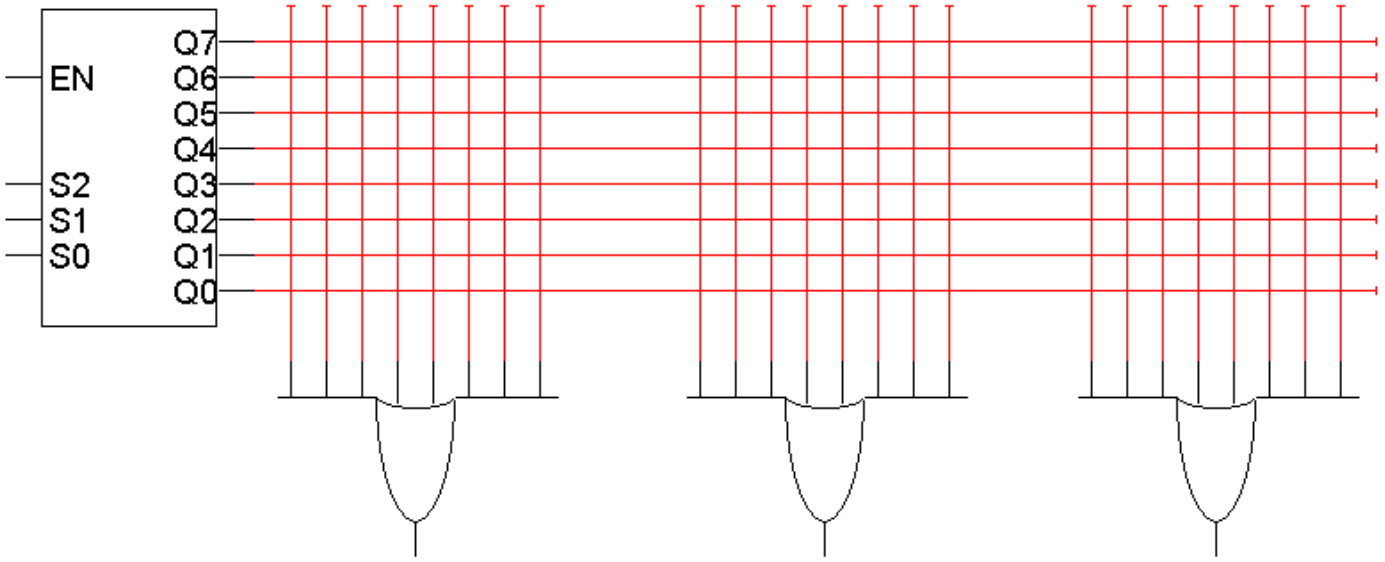


X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- Örneğin yukarıdaki devreyi, 1 bitlik tam toplayıcı devresi için toplam ve elde çıkış değerlerini kaydettiğimiz bir hafıza olarak düşünebilirsiniz.

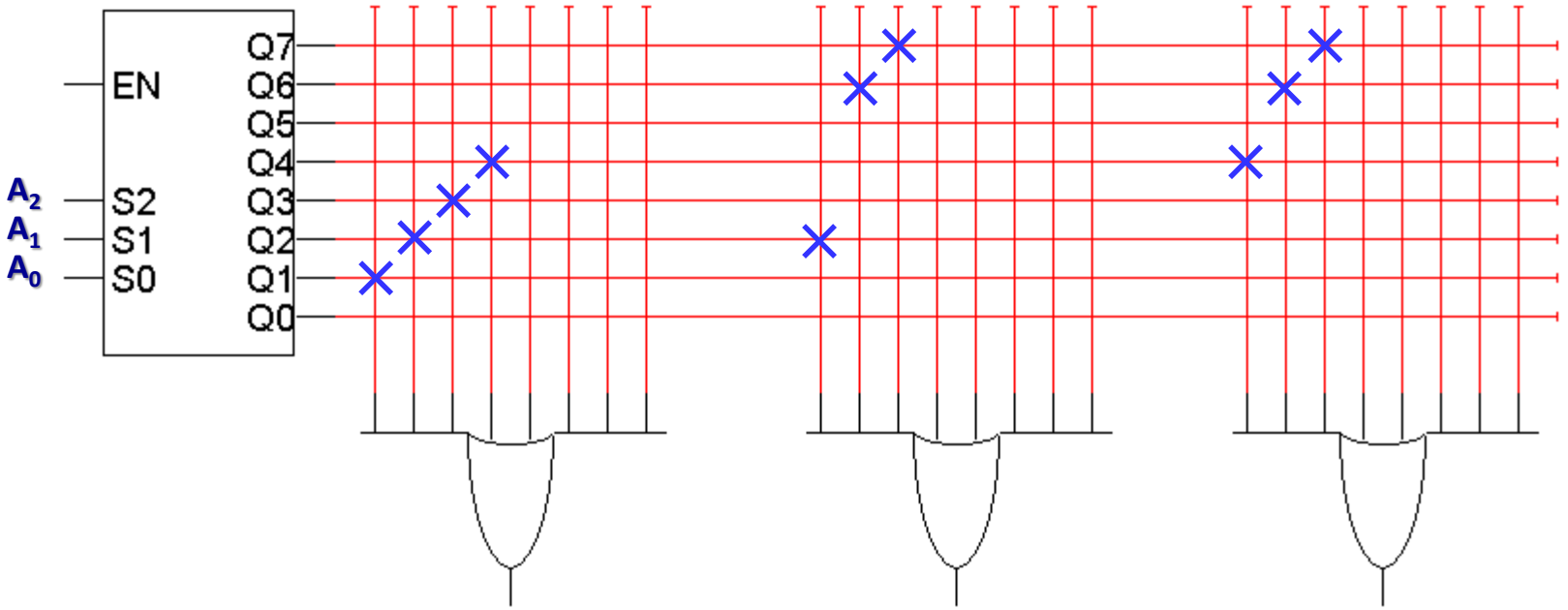
ROM'un yapısı

- ROM'lar fonksiyonların dekoderlerle gerçekleştirilmesi kavramına dayanırlar:
  - Boş, henüz programlanmamış bir ROM, sadece dekoder ve OR kapılarını içerir/sağlar.
  - Dekoder çıkışları ile OR kapıları girişleri arasındaki bağlantılar ***programlanabilmektedir***. Böylece farklı fonksiyonlar gerçekleştirilebilmektedir.
- ROM'u programlamak için, yapılması gereken sadece dekoder çıkışları ile OR kapıları girişleri arasındaki bağlantıların istenen şekilde yapılmasıdır.



ROM Örneği

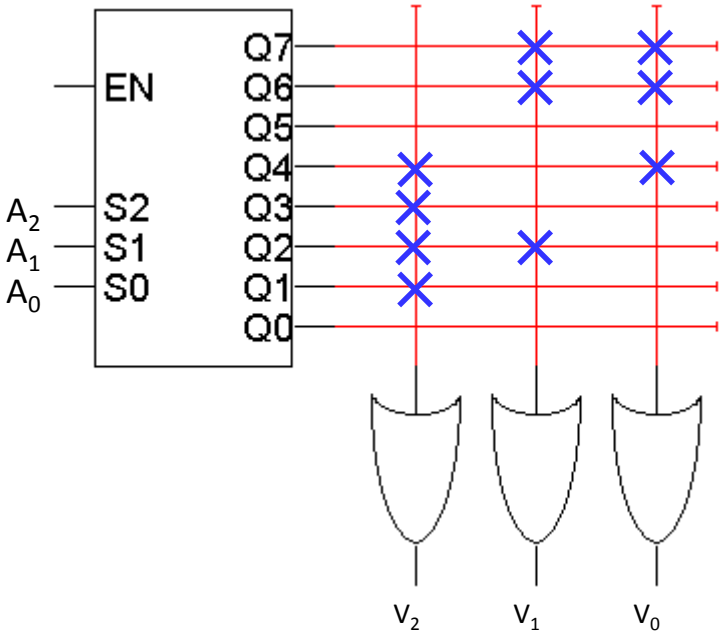
- 3 farklı fonksiyonun, $V_2V_1V_0$, 8 x 3 ROM kullanarak gerçekleştirilmesi.
- X'ler ilgili dekoder çıkışıyla OR kapısı girişi arasında bağlantı olduğunu gösterir. Bu işaret yoksa bağlantı yok anlamındadır.



Aynı örnek tekrar …

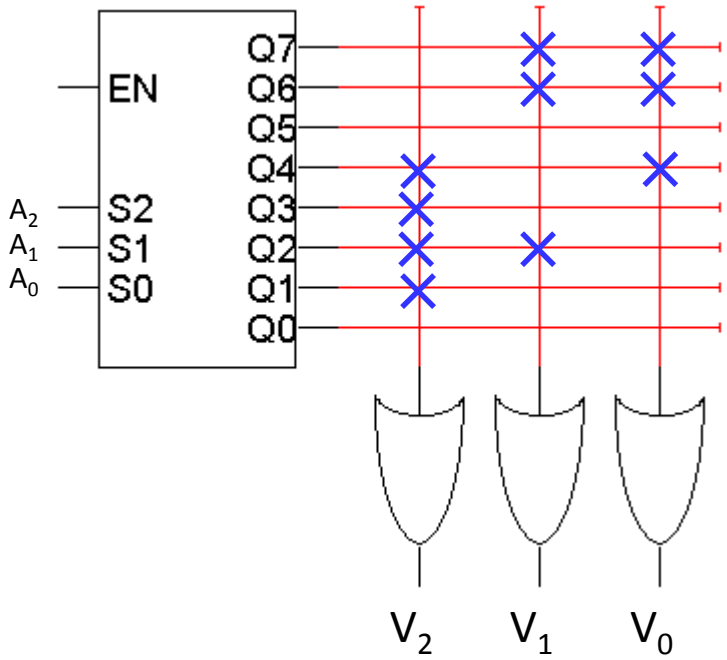
- Sadelik amacıyla, 8 x 3 ROM için, *"kısaltılmış"* OR kapıları kullanılarak alternatif bir gösterim kullanılabilir:

$V_2 = \Sigma m(1,2,3,4)$
$V_1 = \Sigma m(2,6,7)$
$V_0 = \Sigma m(4,6,7)$



Bu niçin bir "Hazıfa"dır?

- Bu kombinasyonel devre salt oku bellek olarak kullanılabilir.
  - Her biri 3-bit genişlikte 8 sözcükten oluşan veriyi kaydeder.
  - Dekoder girişleri sözcüklerden birine işaret eden adresi oluşturur.
  - Böylece, her bir giriş kombinasyonu bir adrese karşılık gelir ve 3-bitlik ilgili verinin okunmasını sağlar.



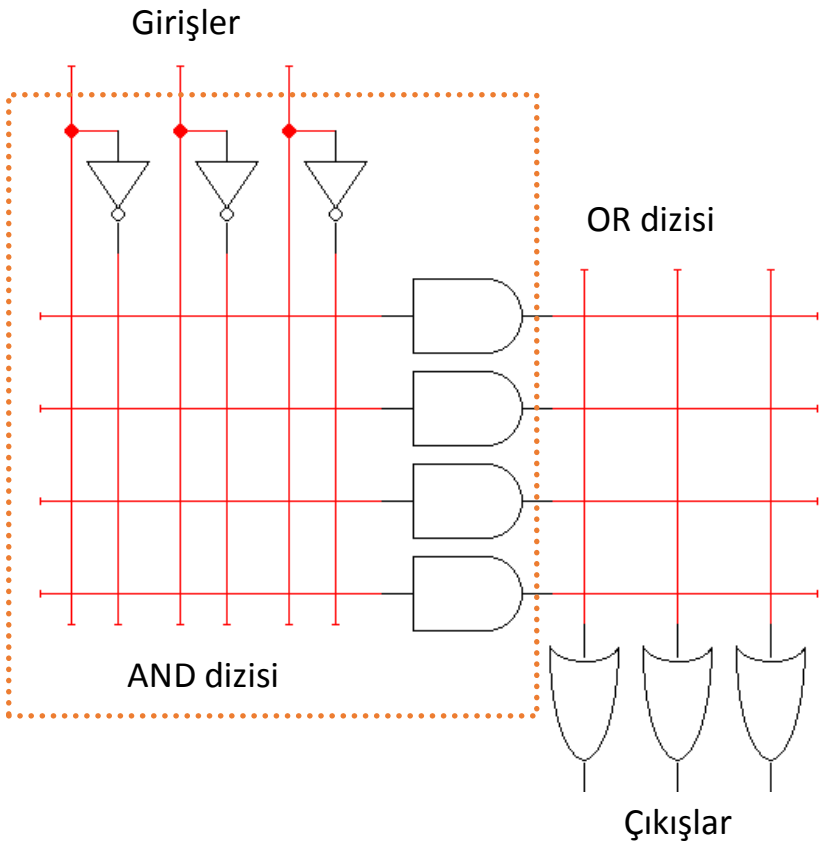
Address $A_2A_1A_0$	Data $V_2V_1V_0$
000	000
001	100
010	110
011	100
100	101
101	000
110	011
111	011

Programlanabilir Lojik Diziler (PLA)

- ROM potansiyel olarak verimsizdir, çünkü olası tüm minterimleri üretebilecek bir dekoder içerirler ve herhangi bir sadeleştirme yapılmaz.
- ROM kullanarak n-girişli bir fonksiyonu gerçekleştirmek için gerekenler:
  - n x 2^n dekoder (n DEĞİL kapısı ve 2^n tane n-girişli VE kapısı.
  - 2^n girişli bir OR kapısı.
- Programlanabilir lojik dizi (PLA), ROM'un dekoder kısmının da programlanabilir olmasını sağlar. Tüm minterimleri oluşturmak yerine, sadece gereken çarpım terimleri oluşturulur (sadeleştirmeye bağlı olarak bu çarpım terimlerinin minterim olmasına da gerek olmayabilir.).

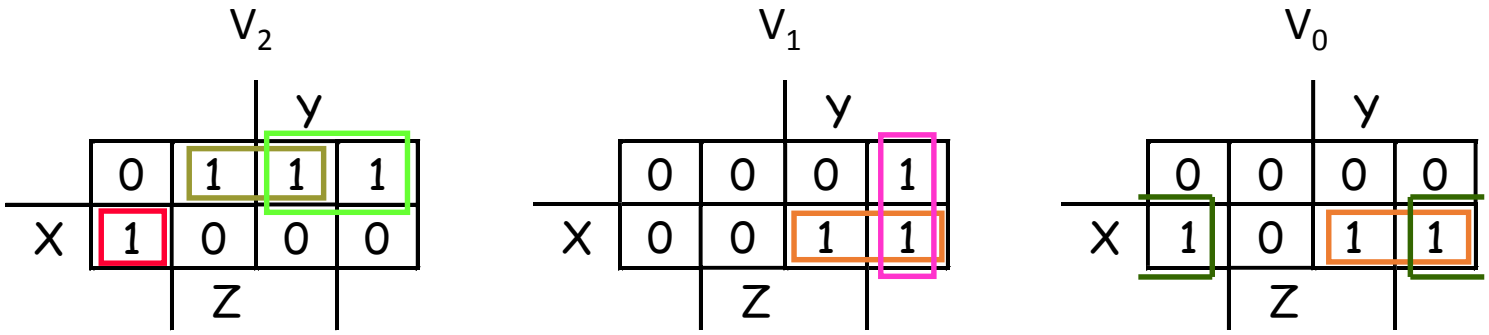
Programlanmamış 3 x 4 x 3 PLA

- Sağda programlanmaya hazır bir 3 x 4 x 3 PLA (3 girişli, 4 değişkene kadar çarpım terimi oluşturmaya imkan sağlayan, 3 çıkışlı) bulunmaktadır.
- Soldaki kesikli çizgileri içindeki kısım ROM'daki dekoderin yerini alır.
- "AND dizisi" kısmında yapılacak bağlantılarla ROM'daki gibi 8 farklı minterim yerine, 4 farklı çarpım terimi oluşturulabilir.
- Bu çarpım terimlerinden gerekli olanlar ilgili çıkışı oluşturmak için "OR dizisi" ile toplanırlar.



Standart K-map (Karnaugh diyagramı) yöntemi ile sadeleştirme

- Normal K-map yaklaşımında tek tek fonksiyonlardaki çarpım terimlerinin sayısı minimize edilir.
- Elimizdeki 3 farklı fonksiyon için, 6 farklı çarpım terimi elde edilecektir.



$V_2 = \Sigma m(1,2,3,4)$
$V_1 = \Sigma m(2,6,7)$
$V_0 = \Sigma m(4,6,7)$

PLA sadeleştirmesi

- PLA için, çarpım terimlerinin sayısını tüm fonksiyonları birlikte değerlendirerek minimize etmeliyiz.
- V_2, V_1 ve V_0'ı sadece 4 çarpım terimiyle ifade edebiliriz:

$V_2 = xy'z' + x'z + x'yz'$

		y		
	0	1	1	1
x	1	0	0	0
		z		

$V_1 = x'yz' + xy$

		y		
	0	0	0	1
x	0	0	1	1
		z		

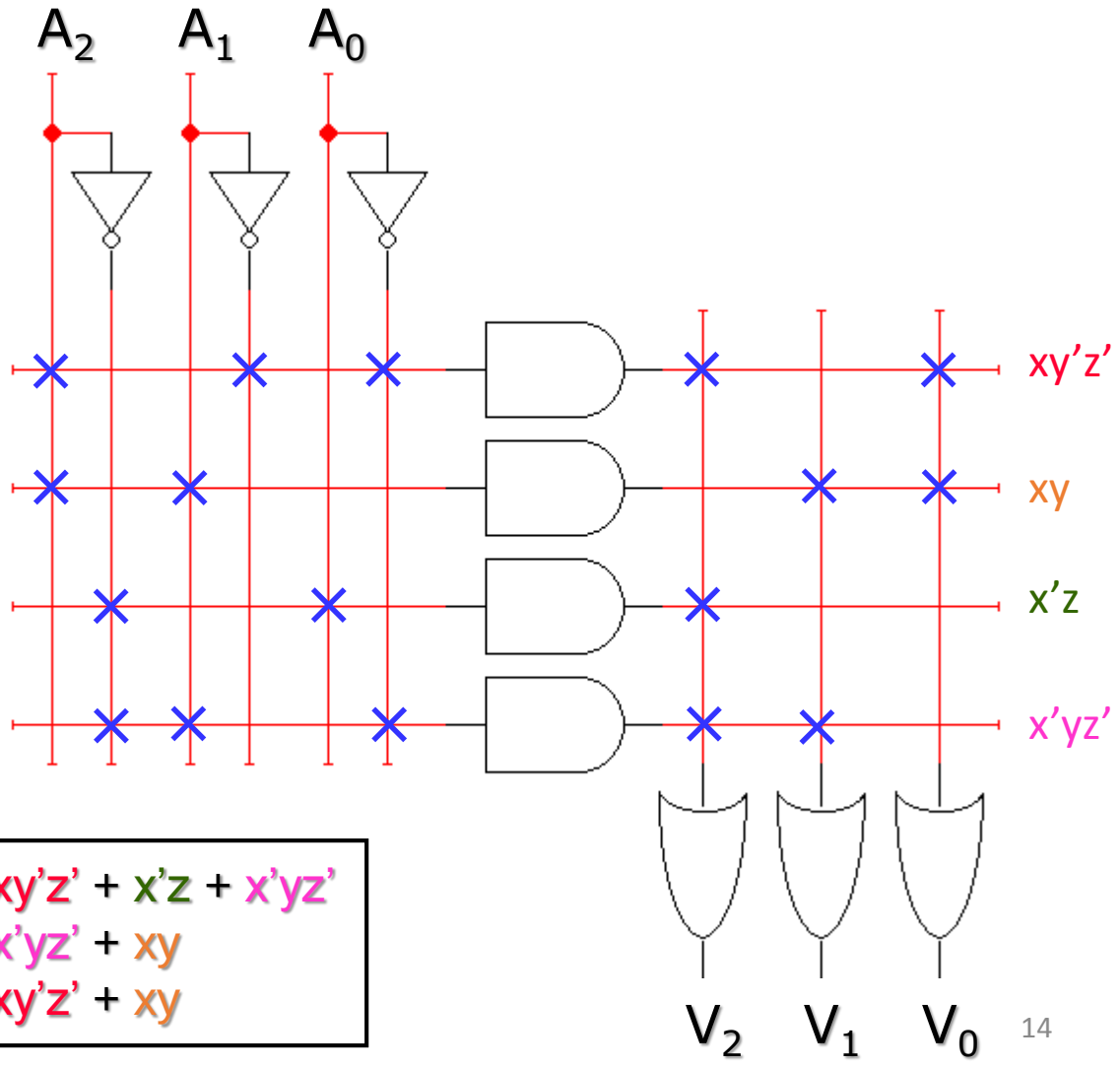
$V_0 = xy'z' + xy$

		y		
	0	0	0	0
x	1	0	1	1
		z		

$V_2 = \Sigma m(1,2,3,4)$
$V_1 = \Sigma m(2,6,7)$
$V_0 = \Sigma m(4,6,7)$

PLA örneği

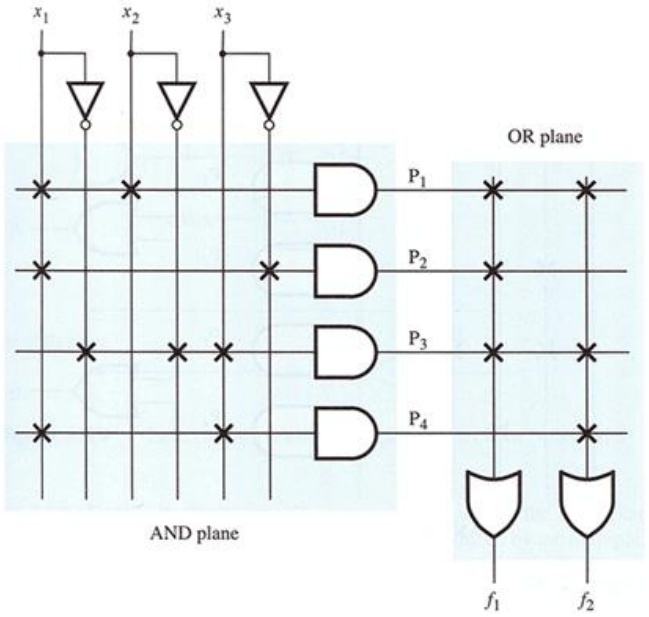
- Bu 3 fonksiyonu 3 x 4 x 3 PLA kullanarak gerçekleştirebiliriz:



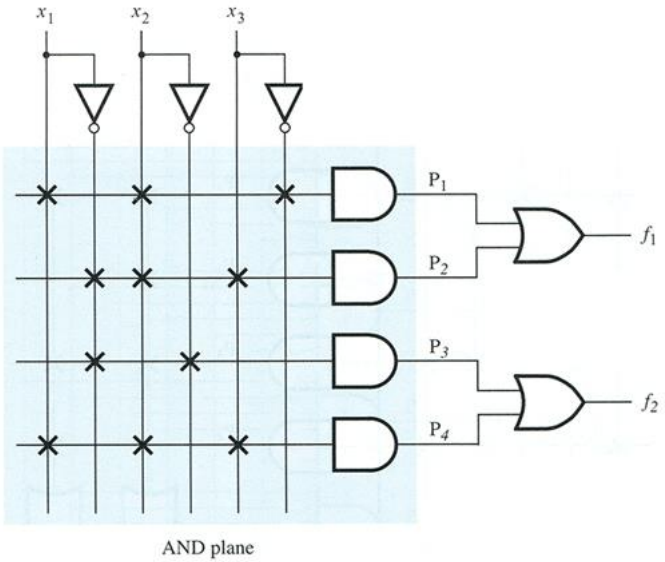
$V_2 = \Sigma m(1,2,3,4) = xy'z' + x'z + x'yz'$
$V_1 = \Sigma m(2,6,7) = x'yz' + xy$
$V_0 = \Sigma m(4,6,7) = xy'z' + xy$

PLA ve PAL

- Programlanabilir lojik dizi (PLA)
  - Programlanabilir VE ve VEYA kapıları dizileri

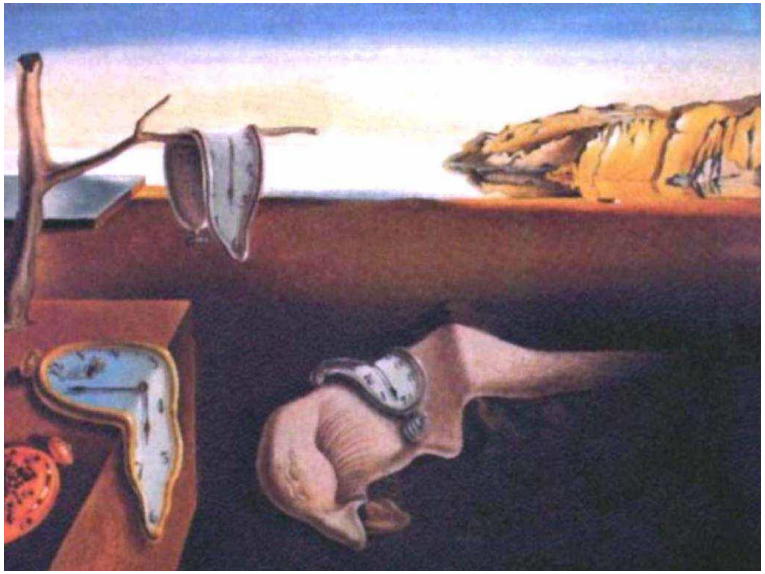


- Programlanabilir dizi lojiği (PAL)
  - Programlanabilir VE kapıları dizisi
  - Sabit VEYA kapıları



Şekiller http://beta.ivc.no/blog/2011/03/30/logic-devices/ 'den adapte edilmiştir.

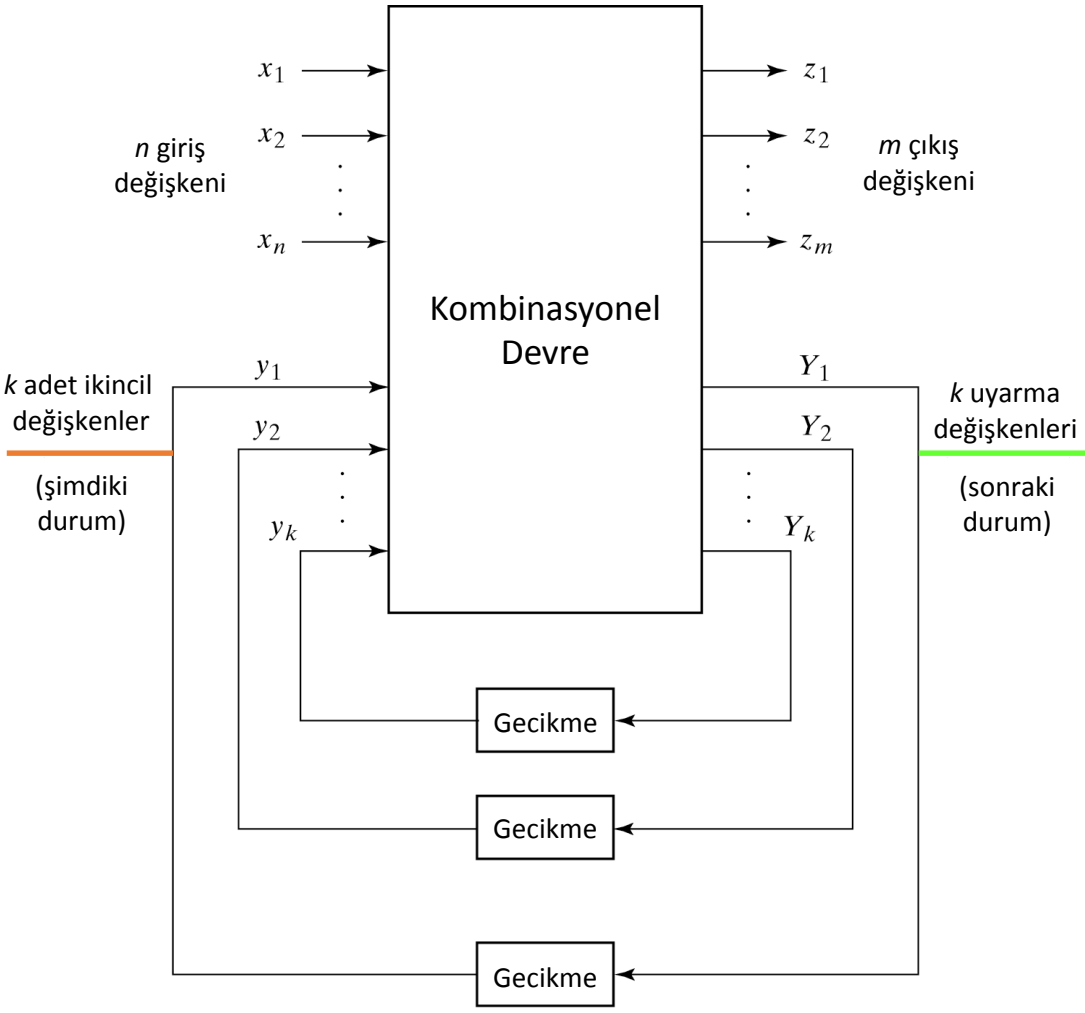
Asenkron Ardışıl Devreler



For most figures:

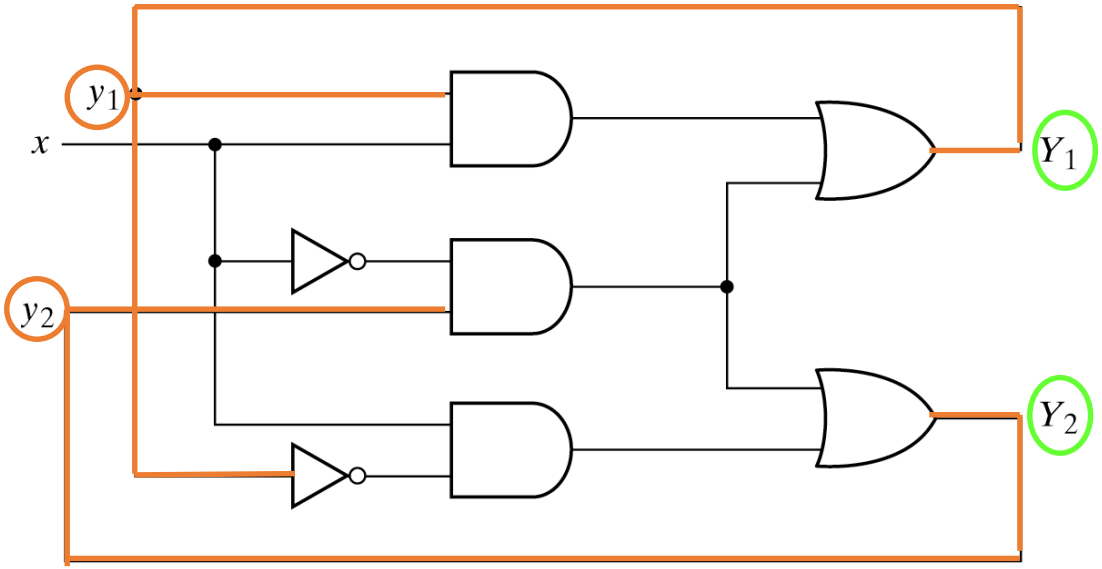
Asenkron Ardışıl Devreler

- Saat sinyali bulunmaz.
- Giriş sinyallerinde bir değişme olması iç durumlarda (internal states) değişikliğe neden olur.
- Senkron ardışıl devrelerin tasarım ve analizinden daha zordur.
- Hızın önemli olduğu durumlarda kullanışlıdır.
- Daha ekonomiktir.
- Yapısal olarak bakıldıklarında, (gecikmeli hatlar üzerinden bağlanmış olan) geri besleme hatlarıyla birlikte bir kombinasyonel devre içerirler.
- İkincil değişkenler: geri besleme hatlarının kapı girişlerine bağlanan uçları, örneğin yandaki şekildeki $y_1, y_2, \ldots y_k$.



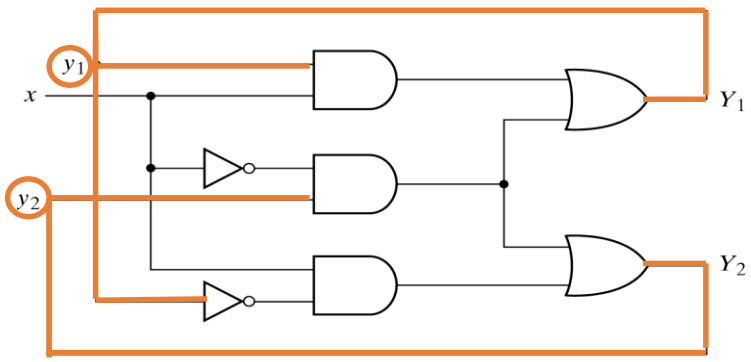
Örnek

- Bir giriş: x
- İki geri besleme hattı
- İki uyarma değişkeni:
  - Y_1 ve Y_2
- İki ikincil değişken:
  - y_1 ve y_2



Geçiş Tablosu (Transition table)

- $Y_1 = xy_1 + x'y_2$
- $Y_2 = xy_1' + x'y_2$
- Giriş (x) aynı zamanda durumun bir parçasıdır.
- Kararlı durumlar mavi yuvarlak içine alınmış olanlardır.
  - $Y_1 = y_1$ and $Y_2 = y_2$



$y_1 y_2$ \ x	0	1
00	0	0
01	1	0
11	1	1
10	0	1

$Y_1 = xy_1 + x'y_2$ için diyagram

$y_1 y_2$ \ x	0	1
00	0	1
01	1	1
11	1	0
10	0	0

$Y_2 = xy'_1 + x'y_2$ için diyagram

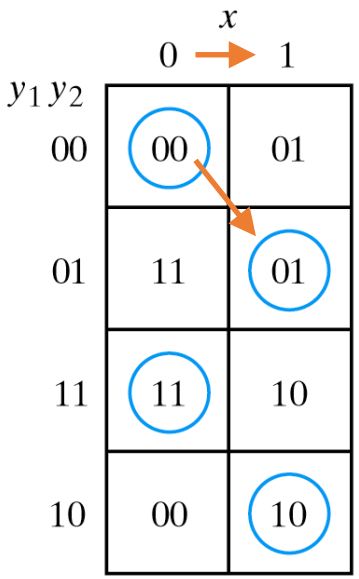
$y_1 y_2$ \ x	0	1
00	00	01
01	11	01
11	11	10
10	00	10

Geçiş Tablosu

Geçiş tablosunun yorumlanması

- 4 kararlı durum:
  - $y_1y_2x = \{000,011,110,101\}$

- Eğer $y_1y_2x = 000$ ve x: 0 → 1 ise
  - $Y_1Y_2x = 011$



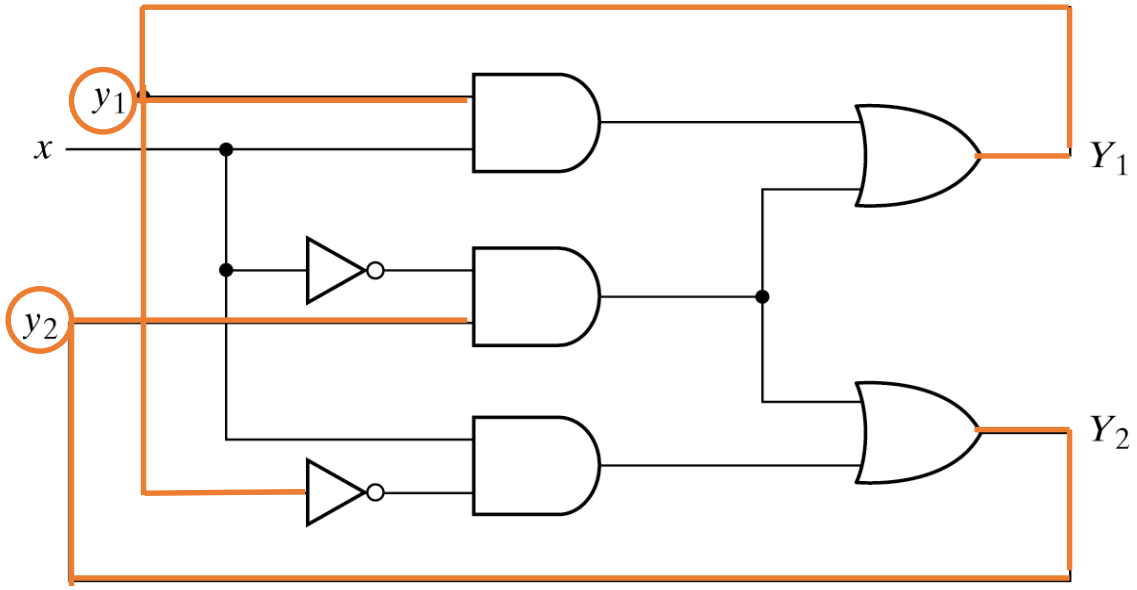
Geçiş Tablosu

Yanda Geçiş Tablosu verilen devrenin Durum tablosu

Şimdiki Durum		Sonraki Durum			
		$x = 0$		$x = 1$	
0	0	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	1	1	1	1	0

Analiz Prosedürü

- Devredeki tüm geri beslemeleri belirle
- Y_i (uyarma değişkenleri) ve y_i (ikincil değişkenler) atamalarını yap
- Tüm Y_i'ler için Boolean fonksiyonları belirle
- Her bir Y fonksiyonu için diyagram oluştur
- Geçiş tablosunu/Durum tablosunu oluştur
- Kararlı durumları yuvarlak içine al



Akış tablosu (Flow table)

- Akış tablosu iç durumların harflerle sembolize edildiği bir durum geçiş tablosudur.

	x = 0	x = 1
a	(a)	b
b	c	(b)
c	(c)	d
d	a	(d)

Bir giriş ile dört durum

	x_1x_2 = 00	01	11	10
a	(a), 0	(a), 0	(a), 0	b, 0 ← çıkış
b	a, 0	a, 0	(b), 1	(b), 0

İki giriş ve bir çıkış ile iki durum

- Soldaki tablo, her bir satırda sadece bir tane kararlı duruma sahip olduğu için ilkel akış tablosu (primitive flow table) adını alır.

Asenkron Ardışıl Devre Tasarımı

	x_1x_2 = 00	01	11	10
a	a, 0	a, 0	a, 0	b, 0
b	a, 0	a, 0	b, 1	b, 0

y \ x_1x_2	00	01	11	10
0	0	0	0	1
1	0	0	1	1

(a) Geçiş Tablosu

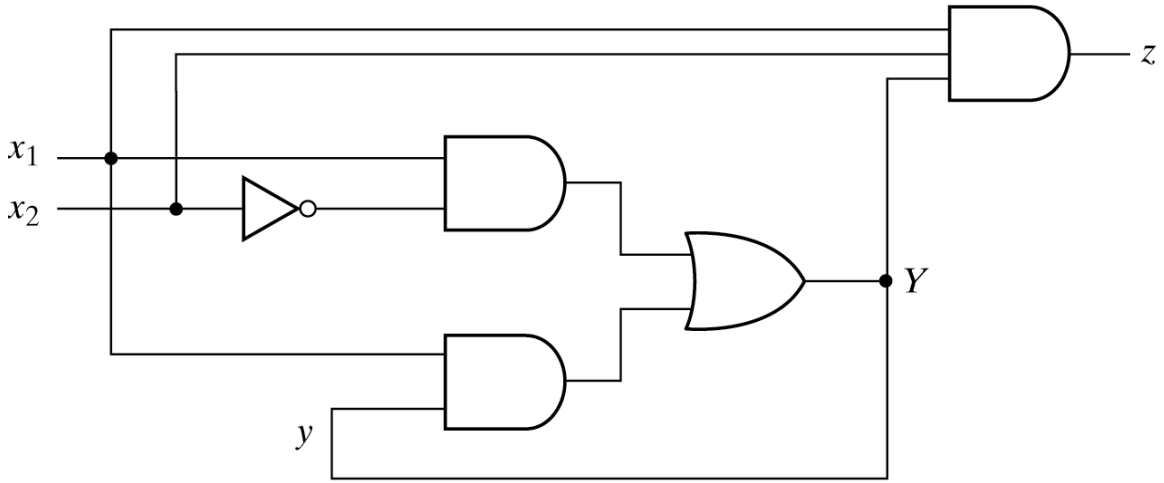
$Y = x_1x'_2 + x_1y$

y \ x_1x_2	00	01	11	10
0	0	0	0	0
1	0	0	1	0

(b) Çıkış için diyagram

$z = x_1x_2y$

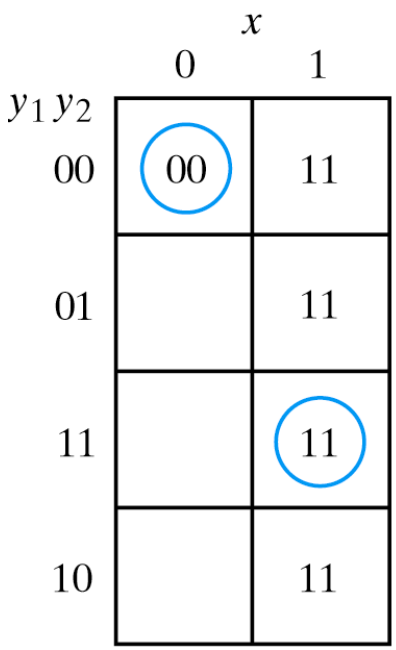
- Atamalar
  - a = 0
  - b = 1



(c) Lojik devre

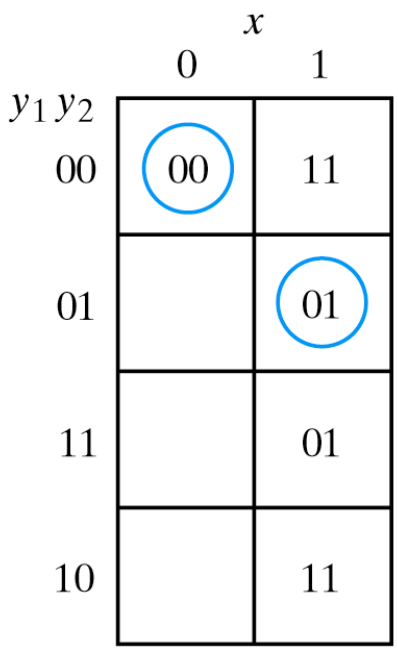
Yarış Koşulları (Race Conditions)

- İki veya daha fazla durum değişkeni, bir giriş değişkeninde oluşan değişikliğe cevap vermek üzere değer değiştiriyorsa bu devrede bir yarış koşulundan söz edilebilir.
- Örneğin:
  - y_1y_2=00 'den y_1y_2 = 11'ye geçerken
  - 3 olası geçiş bulunur:
    - 00 → 11
    - 00 →10 → 11
    - 00 → 01 → 11



(a) Muhtemel geçişler:

$00 \rightarrow 11$
$00 \rightarrow 01 \rightarrow 11$
$00 \rightarrow 10 \rightarrow 11$



(b) Muhtemel geçişler:

$00 \rightarrow 11 \rightarrow 01$
$00 \rightarrow 01$
$00 \rightarrow 10 \rightarrow 11 \rightarrow 01$

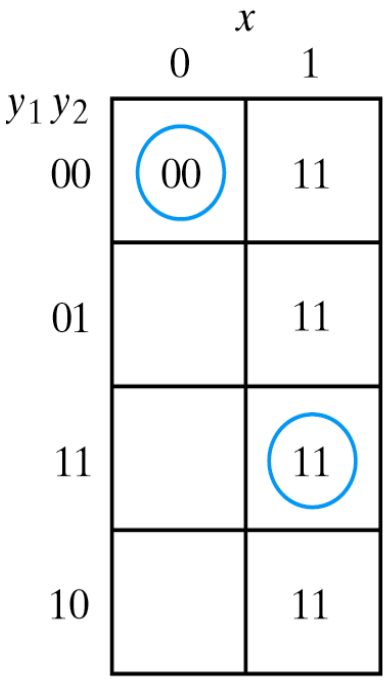
- İki tür yarış koşulu vardır: Kritik ve kritik olmayan

Kritik olmayan yarışa ilişkin örnekler

Kritik olmayan (Non-critical) Yarışlar

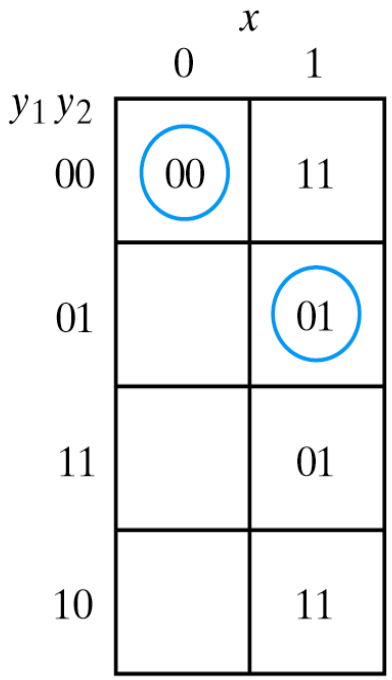
- 3 olası geçiş :
  - 00 → 11
  - 00 →10 → 11
  - 00 → 01 → 11

- Eğer olası tüm geçişler bizi aynı son duruma ulaştırıyorsa, bu kritik olmayan bir yarıştır.



(a) Muhtemel geçişler:

$00 \rightarrow 11$
$00 \rightarrow 01 \rightarrow 11$
$00 \rightarrow 10 \rightarrow 11$



(b) Muhtemel geçişler:

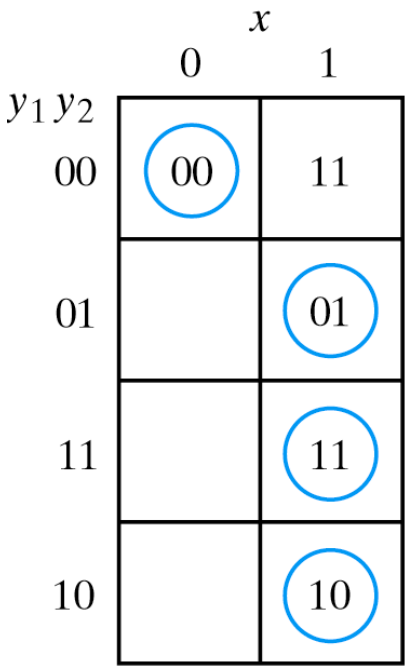
$00 \rightarrow 11 \rightarrow 01$
$00 \rightarrow 01$
$00 \rightarrow 10 \rightarrow 11 \rightarrow 01$

Kritik olmayan yarışa ilişkin örnekler

Kritik (Critical) Yarışlar

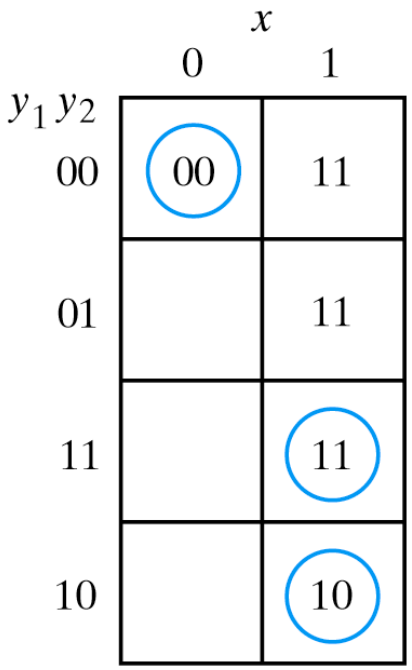
- 3 olası geçiş:
  - 00 → 11
  - 00 → 01
  - 00 → 10

- Eğer farklı geçişler farklı son durumlara ulaşmaya neden oluyorsa, bu kritik bir yarıştır.



(a) Muhtemel geçişler:

$00 \rightarrow 11$
$00 \rightarrow 01$
$00 \rightarrow 10$



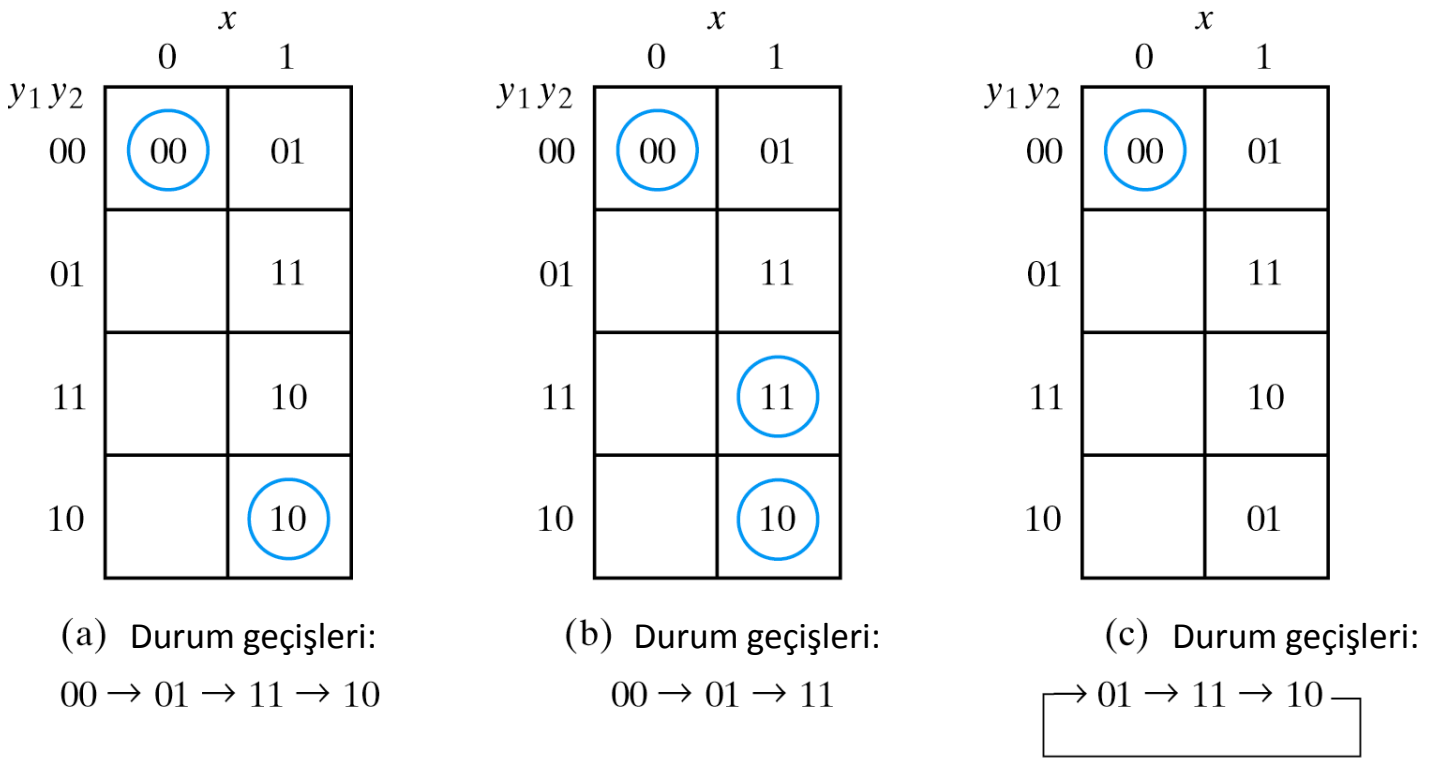
(b) Muhtemel geçişler:

$00 \rightarrow 11$
$00 \rightarrow 01 \rightarrow 11$
$00 \rightarrow 10$

Kritik yarışa ilişkin örnekler

Döngüler (Cycles)

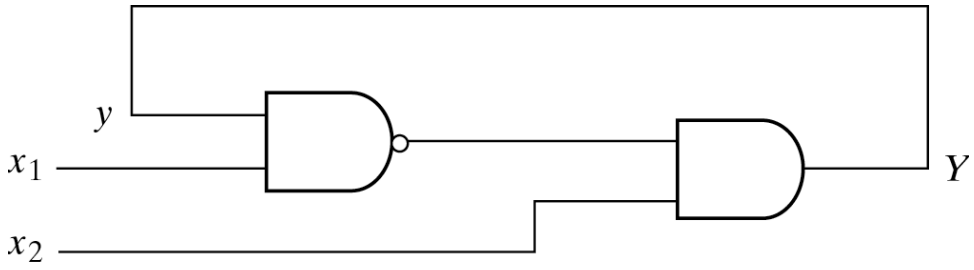
- Döngü: Devre, tek bir kararsız durumlar dizisini izliyorsa bu dizi bir döngüdür.
- Aşağıda döngü örnekleri verilmiştir:



Kararsızlık (Unstability)

- $Y = (x_1y)'x_2 = x_1'x_2 + x_2y'$
- Eğer $x_1x_2y = 111$ ise ➔ $Y = 0$
- Eğer $x_1x_2y = 110$ ise ➔ $Y = 1$

- Devrede Y sinyali 1 ve 0 arasında osilasyon yapar. Bu durumda Y sinyali bir kare dalgadır.



(a) Kararsız lojik devre

y \ x_1x_2	00	01	11	10
0	0	1	1	0
1	0	1	0	0

(b) Geçiş Tablosu