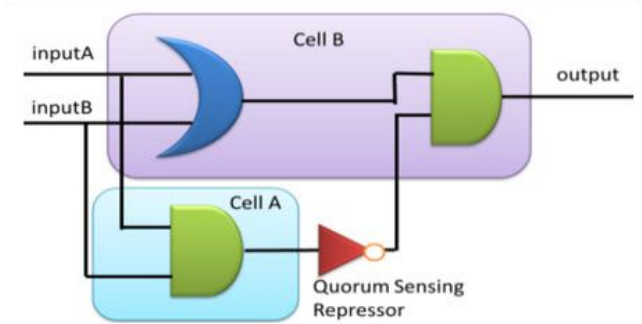
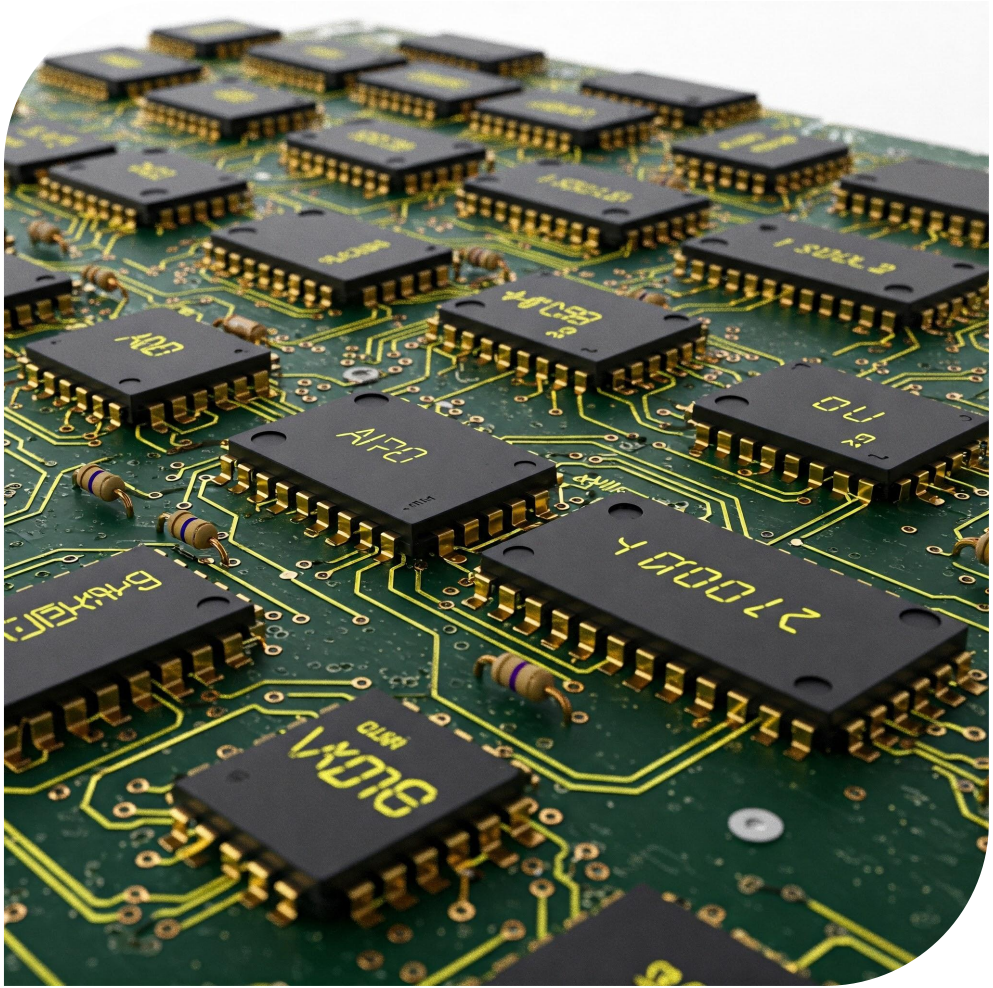
Birleşik Mantık Tasarımı



Suhap SAHİN

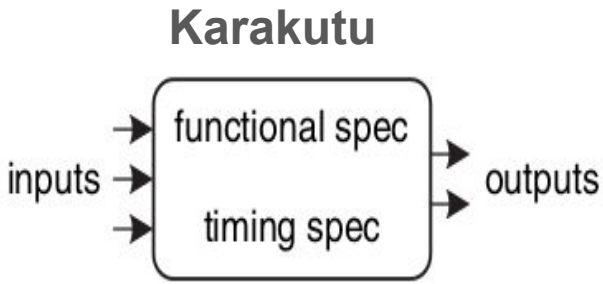
Birleşik vs. Sıralı Mantık

- Sayısal devreler birleşik ve sıralı olarak sınıflandırılır.
- Birleşik devre çıkışları sadece mevcut giriş değerlerine bağlıdır.
- Sıralı devre çıkışları hem mevcut hem de önceki giriş değerlerine bağlıdır.
- Birleşik devreler hafızasızdır.
- Sıralı devrelerin hafızası vardır.



Birleşik Mantık
(Combinational Logic)

ayrık degiskenler:
giris ve cıkıslar
fonksiyonel tanımlamalar:
giris çıkıs arasındaki iliski
zamansal tanımlamalar:
girislerdeki degisiklige cıkısların tepki süresi



Birleşik Mantık
(Combinational Logic)

Element:

Giris, çıkıs ve belirli tanımlamalara sahip devre

Node:

Ayrık degiskenleri ileten bir tel

Element:

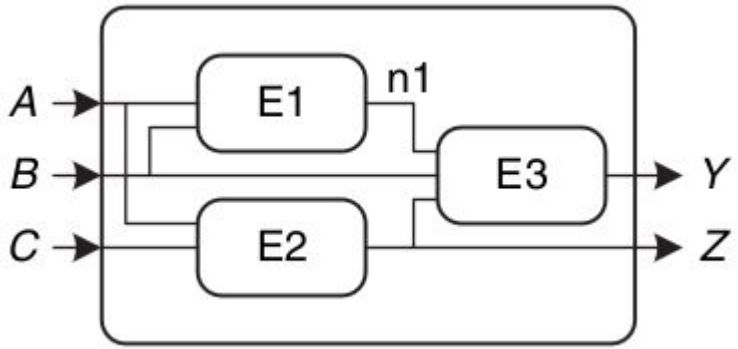
E1, E2, E3

Node:

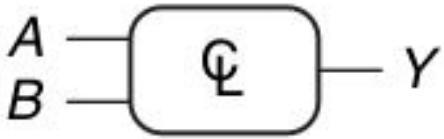
A,B,C (giris)

Y,Z (çıkıs)

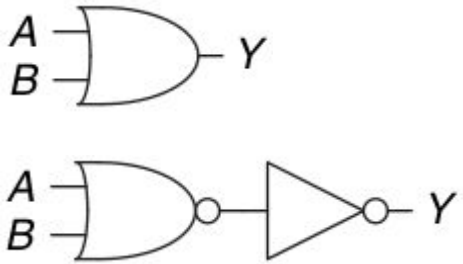
n1 (ara)



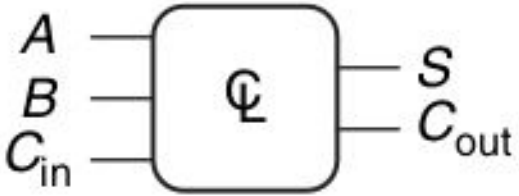
Birleşik Mantık
(Combinational Logic)



$$Y = F(A, B) = A + B$$

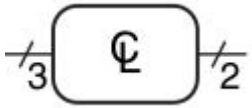


Birleşik Mantık
(Combinational Logic)



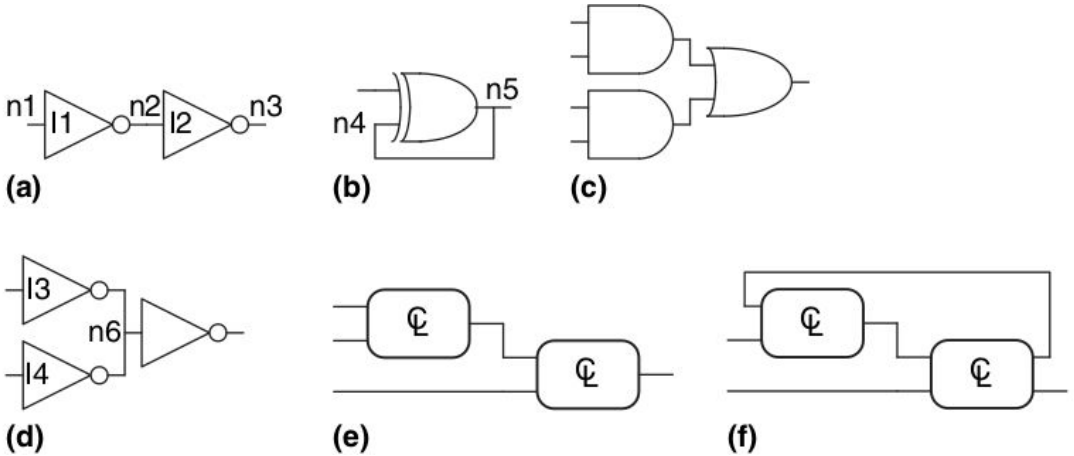
$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$



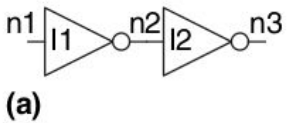
Birleşik Mantık Özellikleri

- Devrenin herbir elemanı bir birlesimsel mantık devresidir.
- Bir giris, çıkıs ve iç baglantı dügümlerinden olusmustur.
- Devredeki herbir yol, devreki her dügümü bir kez ziyaret eder ve döngüsel yol içermez.



Birleşik Mantık Özellikleri

- Devrenin herbir elemanı bir birlesimsel mantık devresidir.
- Bir giris, çıkıs ve iç baglantı dügümlerinden olusmustur.
- Devredeki herbir yol, devreki her dügümü bir kez ziyaret eder ve döngüsel yol içermez.

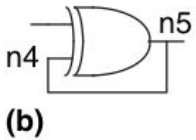


(a)

- I1 ve I2 tersleyiciler: Devrenin herbir elemanı bilesimsel mantık devresidir.
- Devre n1,n2 ve n3 dügümlerine sahiptir.
- Devre döngüsel yol içermiyor
- Devre bilesimsel bir mantık devresidir.

Birleşik Mantık Özellikleri

- Devrenin herbir elemanı bir birlesimsel mantık devresidir.
- Bir giris, çıkıs ve iç baglantı dügümlerinden olusmustur.
- Devredeki herbir yol, devreki her dügümü bir kez ziyaret eder ve döngüsel yol içermez.

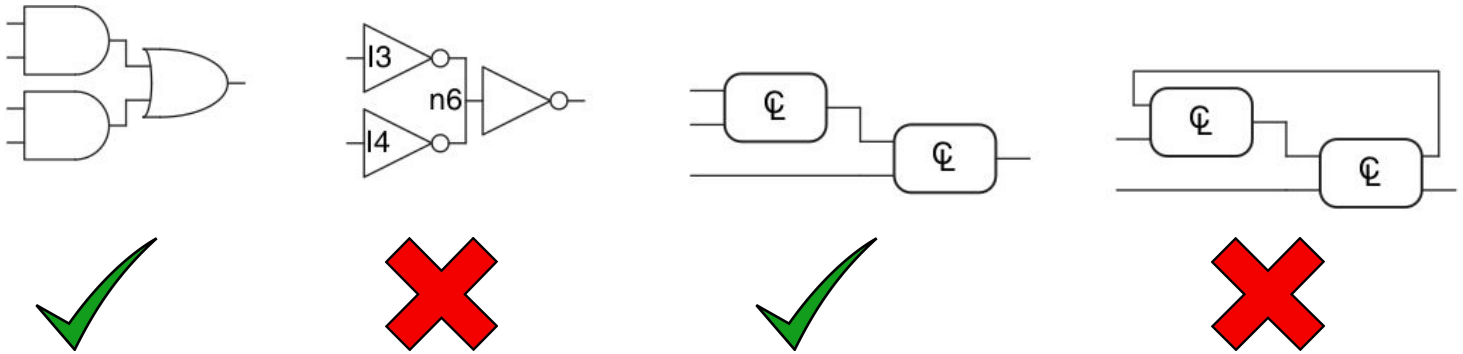


(b)

- Devre döngüsel yol içeriyor
- Devre bilesimsel bir mantık devresi **DEGiLDiR**.

Birleşik Mantık Özellikleri

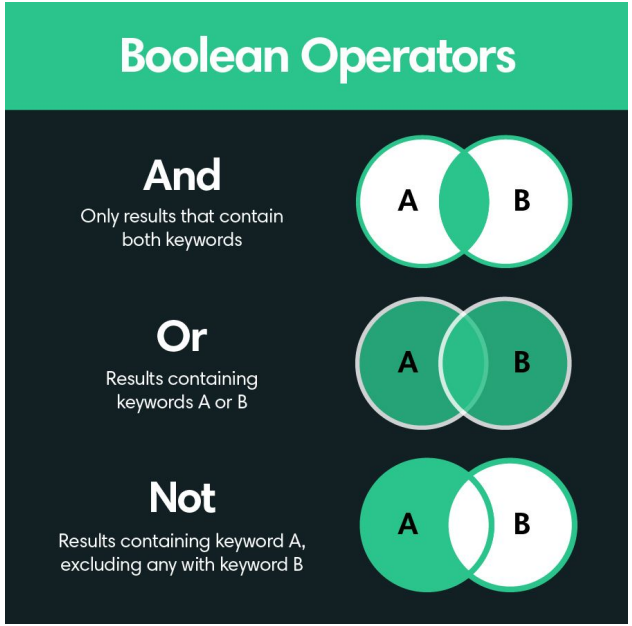
- Devrenin herbir elemanı bir birlesimsel mantık devresidir.
- Bir giris, çıkıs ve iç baglantı dügümlerinden olusmustur.
- Devredeki herbir yol, devreki her dügümü bir kez ziyaret eder ve döngüsel yol içermez.



Boole Denklemleri

Mantıksal işlemleri matematiksel olarak ifade eden cebirdir.

Temel olarak **0 (yanlış)** ve **1 (doğru)** değerleri kullanılır.



Temel Boole İşlemleri

İşlem	Sembol	Anlamı
VE	· **veya** ∧	İkisi de 1 ise sonuç 1
VEYA	+ **veya** ∨	En az biri 1 ise sonuç 1
DEĞİL	~ veya '	Değerin tersini alır (1 → 0, 0 → 1)

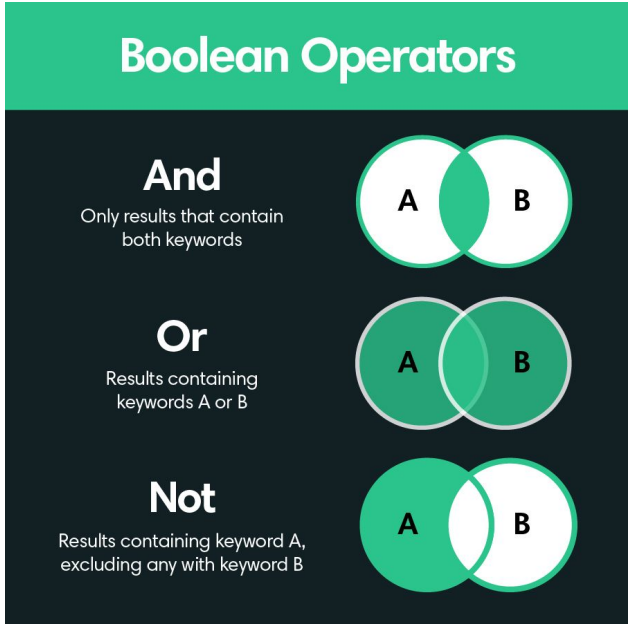
Boole Denklemleri

Temel Boole Denklemleri

- **Kimlik:**
  `A + 0 = A`
  `A · 1 = A`
- **İptal Etme:**
  `A + 1 = 1`
  `A · 0 = 0`
- **Tersleme:**
  `A + ~A = 1`
  `A · ¬A = 0`

Öncelik Sırası

1. **DEĞİL (~)**
2. **VE (·)**
3. **VEYA (+)**



Boole Denklemleri

Kullanım Alanları

- Dijital devre tasarımı
- Bilgisayar programlama
- Mantıksal karar verme sistemleri

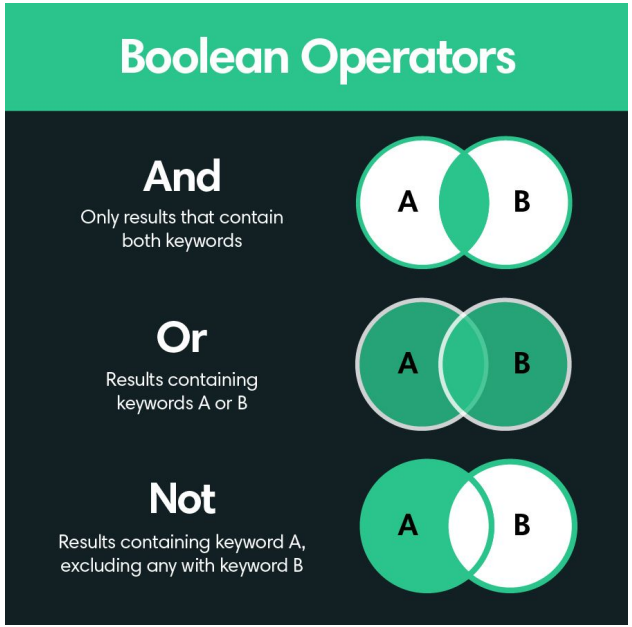
Örnek Boole Denklem Çözümü

Verilen Denklem:

`Y = ~A · (B + C)`

Çözüm

- `B + C` → B VEYA C
- `¬A` → A'nın tersi
- Y'nin 1 olması için `A sıfır` ve `(B VEYA C)`'nin en az birinin 1 olması gerekir



Sum Of Product Form ve Sigma Notation

A	B	Y	minterm	minterm name
0	0	0	~A~B	m_0
0	1	1	~AB	m_1
1	0	0	A~B	m_2
1	1	0	AB	m_3

$Y = \sim AB$

$Y = \sim AB + AB$

$F(A,B) = \Sigma(m_1, m_2)$

$F(A,B) = \Sigma(1,3)$

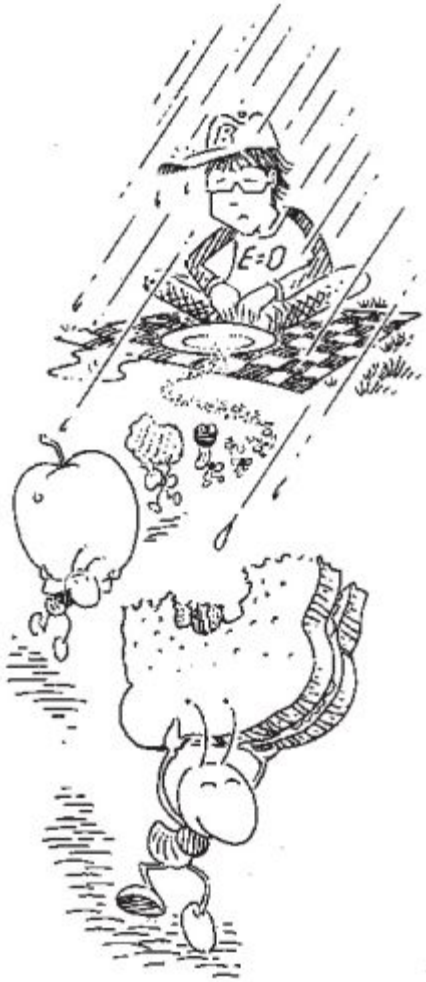
A	B	Y	minterm	minterm name
0	0	0	~A~B	m_0
0	1	1	~AB	m_1
1	0	0	A~B	m_2
1	1	0	AB	m_3

Sum Of Product Form

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$Y = \sim A \sim B \sim C + A \sim B \sim C + A \sim BC$

$Y = \Sigma(0,4,5)$



Piknik

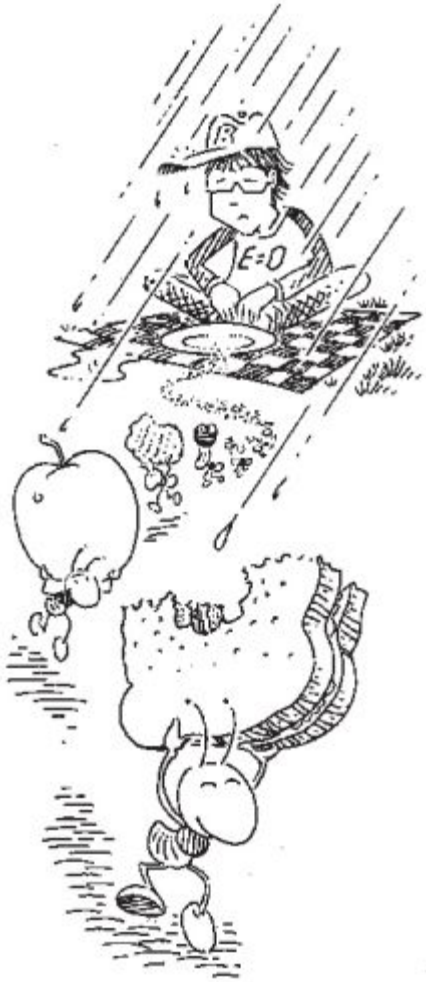
- **Karınca çok olması**
- **Yagmurun yagması**

Piknik yapmak TRUE

K	Y	P
0	0	1
0	1	0
1	0	0
1	1	0

P =~K~Y

P = Σ(0)



Piknik

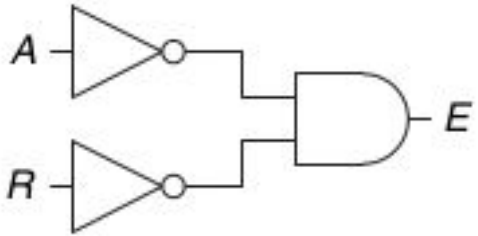
- **Karınca çok olması**
- **Yagmurun yagması**

Piknik yapmak TRUE

K	Y	P
0	0	1
0	1	0
1	0	0
1	1	0

P =~K~Y

P = Σ(0)



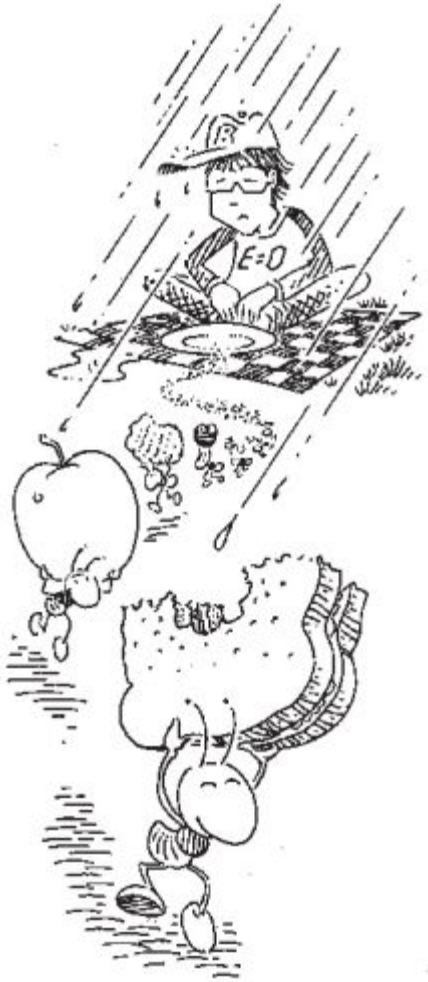
Product Of Sum Form ve Pi Notation

A	B	Y	minterm	maxterm name
0	0	0	A + B	M_0
0	1	1	A + ~B	M_1
1	0	0	~A + B	M_2
1	1	1	~A + ~B	M_3

$Y = (A + B)(\sim A + B)$

$Y = \Pi(M_0, M_2)$

$Y = \Pi(0,2)$



Piknik

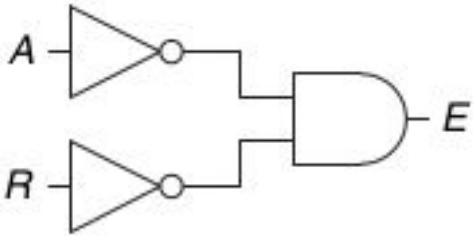
- **Karınca çok olması**
- **Yagmurun yagması**

PiCNiC yapmak TRUE

K	Y	P
0	0	1
0	1	0
1	0	0
1	1	0

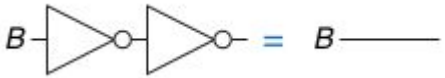
$$P = (K + \sim Y)(\sim K + Y)(\sim K + \sim Y)$$

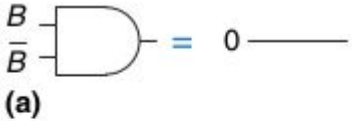
$$P = \Pi(1,2,3)$$

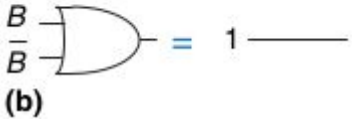


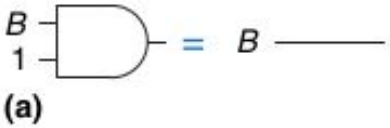
Bir Değişkene ait Teoremler

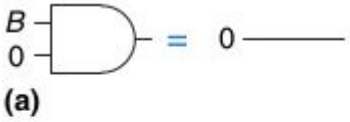
Teorem		Dual		Name
T1	B . 1 = B	~T1	B + 0 = B	Identity
T2	B . 0 = 0	~T2	B + 1 = 1	Null Element
T3	B . B = B	~T3	B + B = B	Idempotency
T4		~~B = B		Involution
T5	B . ~B = 0	~T5	B + ~B = 1	Complements

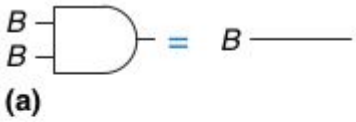


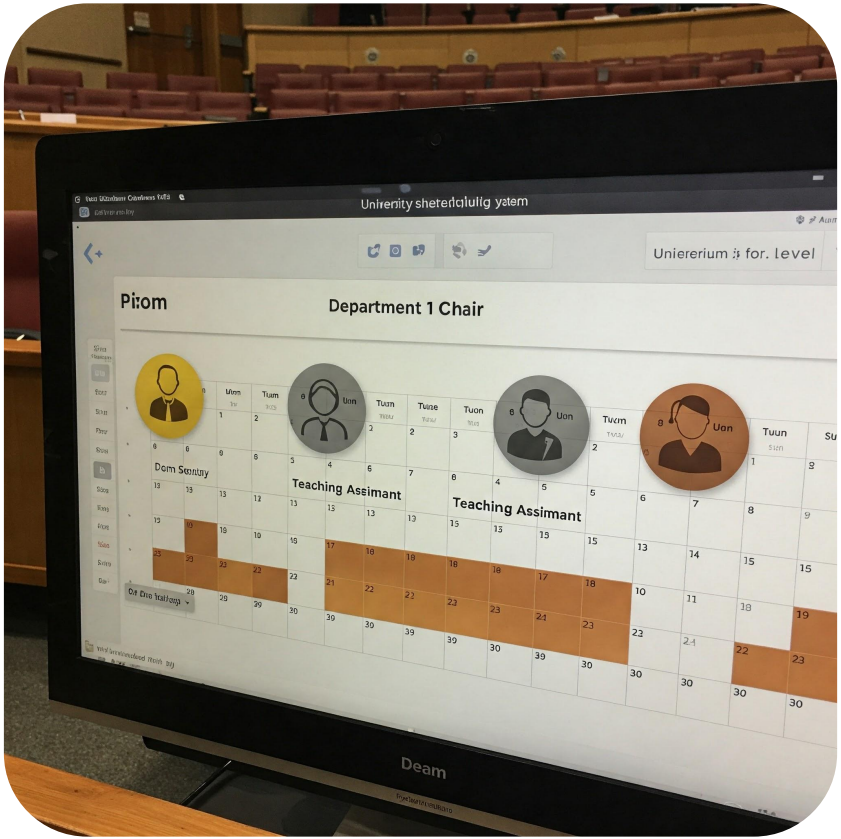












Konferans Salonu Rezarvasyonu

Aşağıdaki kişiler konferans salonunu kullanmaktadır.

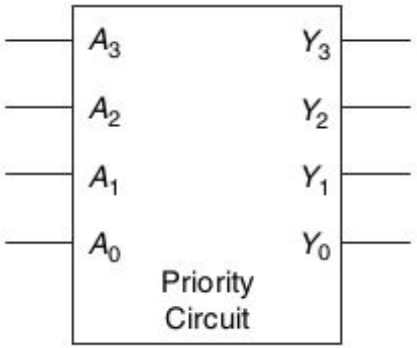
- Dekan
- Bölüm Başkanı
- Öğretim Görevlisi
- Yurt Müdürü

Çakısma meydana gelmemesi için gerekli lojik devreyi ciziniz.

Devreye ait dogruluk tablosunu ve Boolean denklemlerini yazınız.

Dört Girişli Öncelik Devresi

A0 → Y0: Yurt Müdürü
A1 → Y1 : Ögretim Üyesi
A2 → Y2: Bölüm Baskanı
A3 → Y3: Dekan

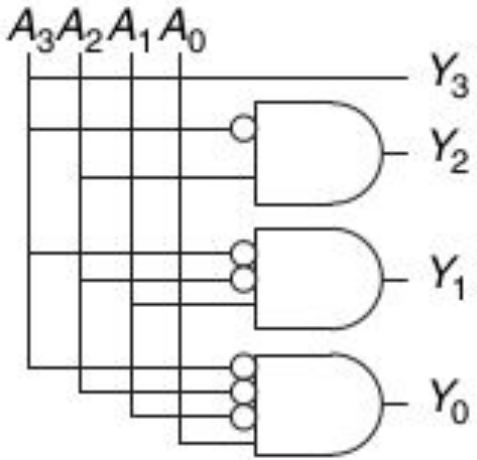


A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

Dört Girişli Öncelik Devresi

A0 → Y0: Yurt Müdürü
A1 → Y1 : Ögretim Üyesi
A2 → Y2: Bölüm Baskanı
A3 → Y3: Dekan

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0




```
Y3 = A3

Y2 = ~A3 * A2

Y1 = ~A3 * ~A2 * A1

Y0 = ~A3 * ~A2 * ~A1 * A0
```

Acil Durum Sistemi

- Bir bina için acil durum alarm sistemi tasarlanmak isteniyor.
- Sistemde farklı öncelikle sahip alarm kanaklarının olması isteniyor.

Acil Durum Sistemi

- Alarm kaynakları
  - Yangın Alarmı ➜ En yüksek öncelik 3
  - Gaz Kaçağı Alarmı ➜ Öncelik 2
  - Hırsız Alarmı ➜ Öncelik 1
  - Genel Bilgilendirme ➜ En düşük öncelik 0

Acil Durum Sistemi

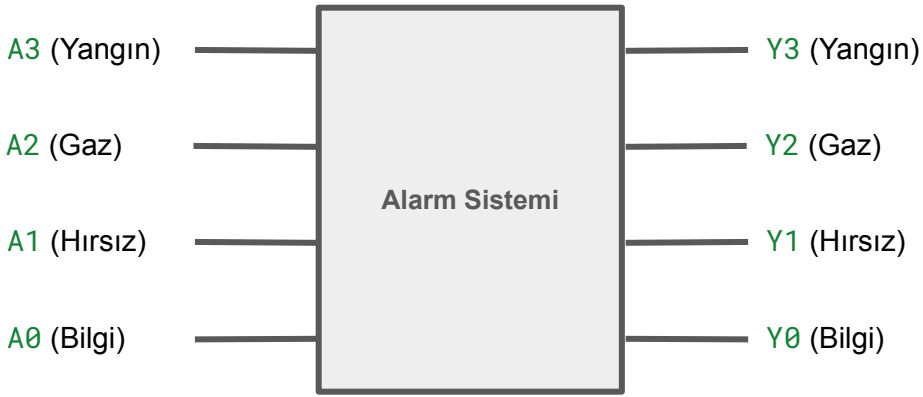
A3	A2	A1	A0	Y3	Y2	Y1	Y0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0

```
Y3 = A3

Y2 = ~A3 * A2

Y1 = ~A3 * ~A2 * A1

Y0 = ~A3 * ~A2 * ~A1 * A0
```

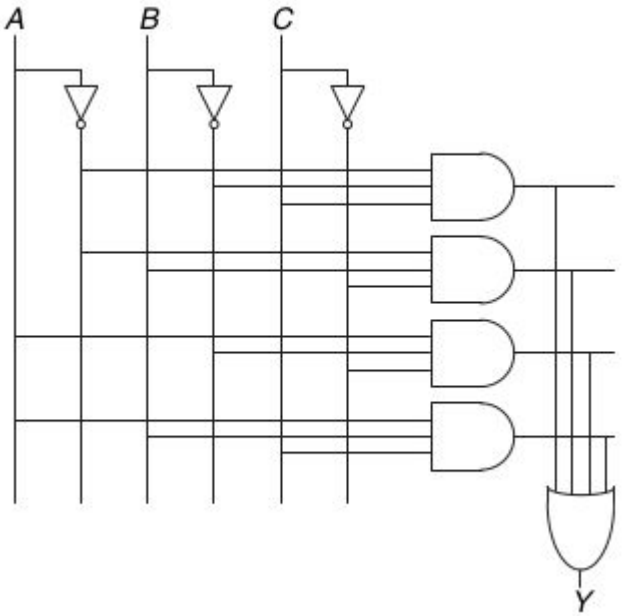


Çok Seviyeli Birleşik Mantık

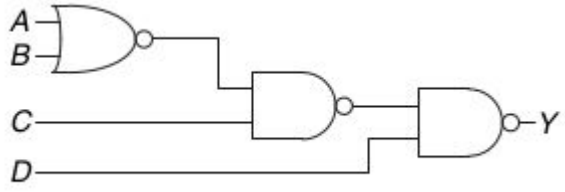
Sum of products formunda girisler AND lojik sevyelerinden ve çıkıslar OR lojik sevyelerinden olustugu için iki sevyeli lojik seklinde adlandırılır.

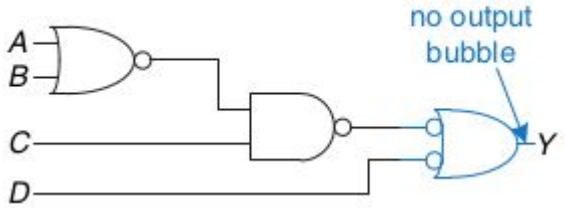
$$Y = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

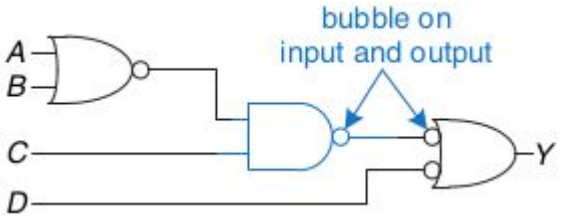


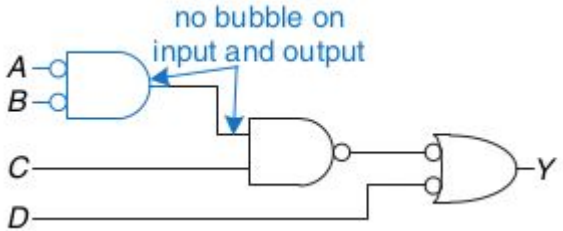
Tersleyici İtme



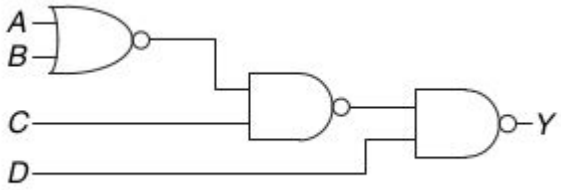


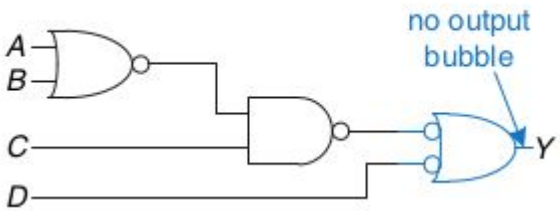
$$Y = \overline{A}\,\overline{B}C + \overline{D}$$

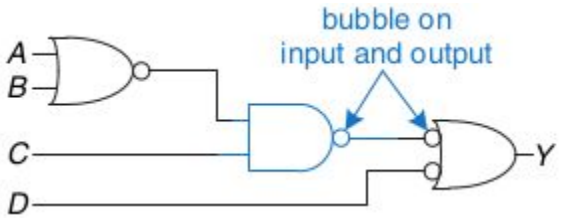


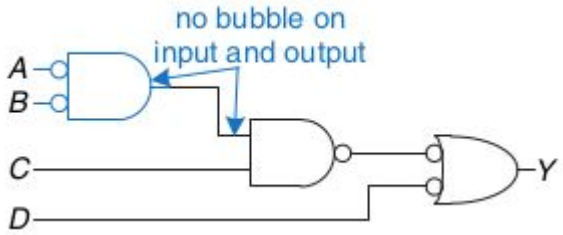


Tersleyici İtme

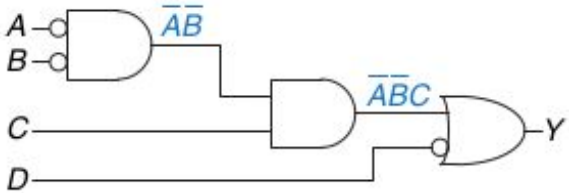








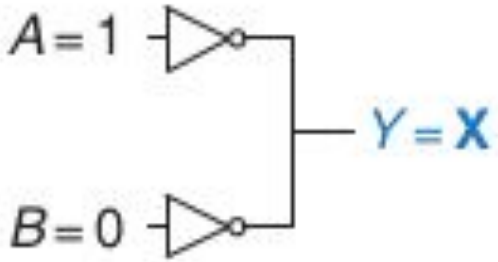
$Y = \overline{A}\overline{B}C + \overline{D}$



$Y = \overline{A}\overline{B}C + \overline{D}$

İllegal X Değeri

X→ HIGH/LOW degeri
(forbidden zone)

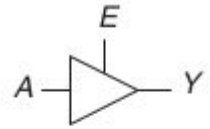


X degeri ile sürülen devreler, devre elemanlarının özelliklerine göre bazen LOW bazende HIGH degeri olarak islem yapabilirler.

X degeri, simülatorlerde baslangıç degeri ile sürülmemis devre elemanlarını sürmek için kullanılır.

X degeri dogruluk tablolarında “don’t care” durumları için kullanılır

Kayan Z Değeri



E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

Z→ HIGH/LOW DEGiL degeri
floating, high impedance, high Z

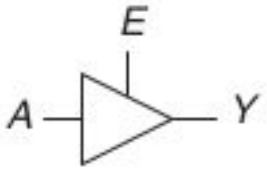
Z degeri ile sürülen devreler, devre elemanlarının özelliklerine göre bazen LOW bazende HIGH degeri olarak islem yapabilirler.

Lojik islemler sürerken görülen Z degeri hata anlamına gelmez.

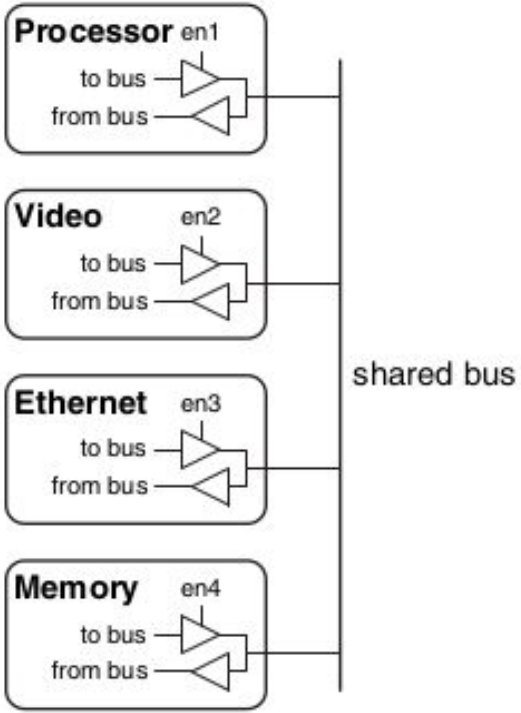
Devre girisine voltaj baglamamak veya baglanmamıs bir girisin 0 degerine sahip oldugunu varsaymak Z degerine sebep olur.

Devreye dokunmak, statik elektrik sebebiyle devrenin Z ile sürülmesine yeterli olabilir.

Ortak Yol Kullanımı

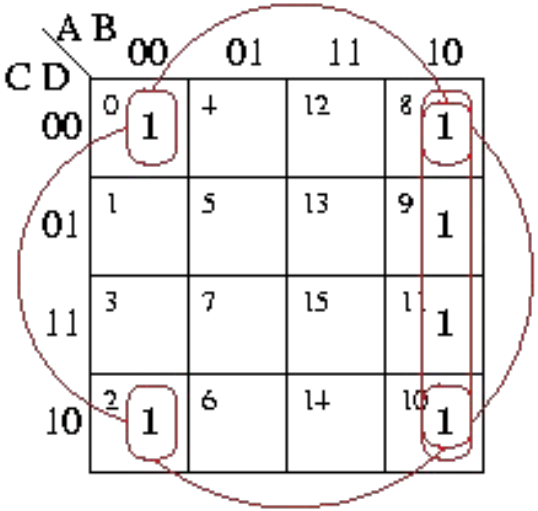


E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

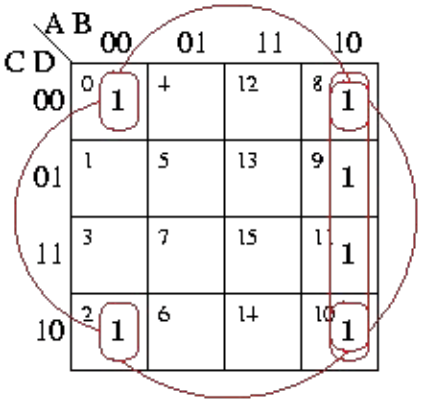


Haritalam Yöntemi ile Sadeleştirme (Karnaugh Map Minimization)

- Görsel bir sadelestirme yöntemidir.
  - Yakınlık özelligini kullanır.
  - En küçük deyimi bulur.
  - Kullanımı kolay ve hızlıdır.
- Problemler:
  - Belirli sayıda degiskene uygulanabilir. (4 ~ 8)
  - Dogruluk tablosundan haritaya geçirirken yanlıslar yapılabilir.
  - Haritadaki hücreler dogru bir sekilde gruplanmayabilir.
  - Son deyim yanlıs okunabilir.



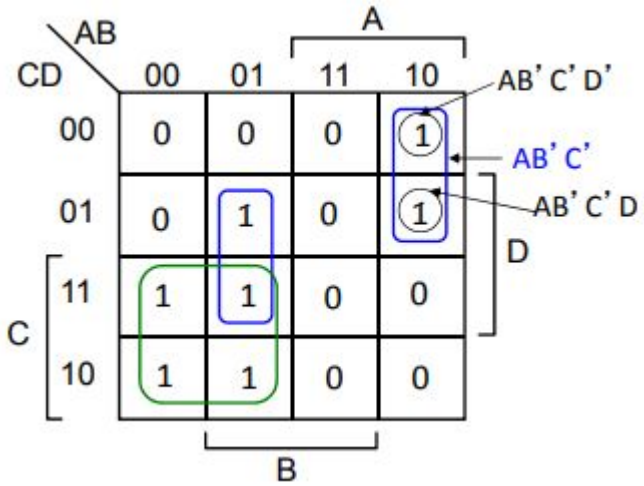
Haritalam Yöntemi ile Sadeleştirme (Karnaugh Map Minimization)



- Harita belli sayıda hücreden olusan bir 2 boyutlu dizgedir.
  - Her kare dogruluk tablosundaki bir satıra karsılık gelir
- Hücrelerin yerlesimi
  - Bitisik terimlerde sadece 1 degisken degeri farklıdır. örn. m6 (110) and m7 (111)

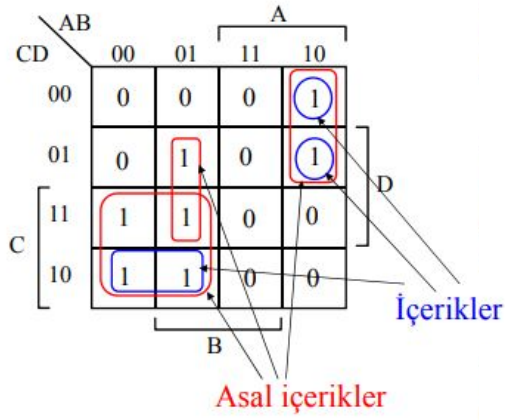
Gruplama - Bitişiklik İlkesinin Uygulanması

- iki hücre aynı degere sahip (1) ve birbirlerine komsu ise, deyimler bitisiktir.
- Gruplar üst üste gelebilir
- Grup sayısı 2 nin kuvveti olmalıdır (1, 2, 4, 8)
- 1 ler veya 0 lar gruplandırılabilir.

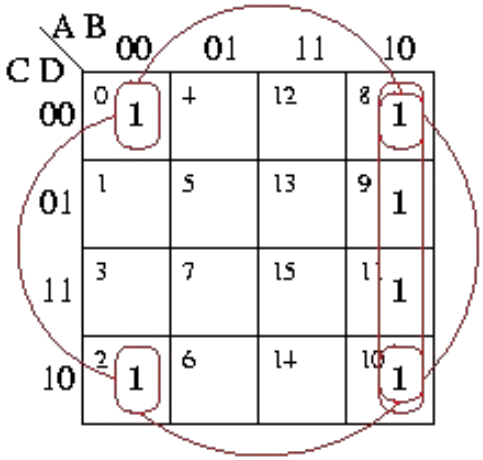


İçerikler(Implicants) ve Asal İçerikler (Prime Implicants)

- Daha büyük bir grubun parçası olan tek bir hücre ya da bir grup hücreye içerik denir.
- En büyük gruba asal içerik denir. (Tek bir hücre de asal içerik olabilir.)



K Haritaları Gruplama Kuralları



Tüm 1 leri kapsayacak en az çemberi kullanılmalıdır.

Çemberin içindeki tüm kareler 1 içermelidir.

Gruplamalarda seçilen kutu sayısı 1,2,4,8,16,…olmalıdır.

Herbir çember olabildigince büyük olmalıdır.

Karsılıklı köse ve kenarlardaki kareler birbirlerine komsu kare sayılırlar.

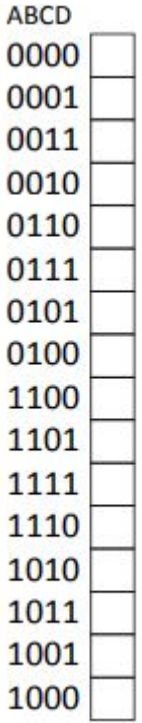
Harita içindeki 1 degeri en az çemberi saglamak için birden fazla çember içinde olabilir.

Bitisik terimlerde sadece 1 degisken degeri farklıdır. örn. m6 (110) and m7 (111)

Gray Kod

GEMi kelimesinin iNEK kelimesine dönüsümü;
Kural her bir adımda sadece bir harf degisebilir.

GEMi,
iEMi,
iEMK,
iNMK,
iNEK



Doğruluk Çizelgesi ve Bitişiklik (adjacency)

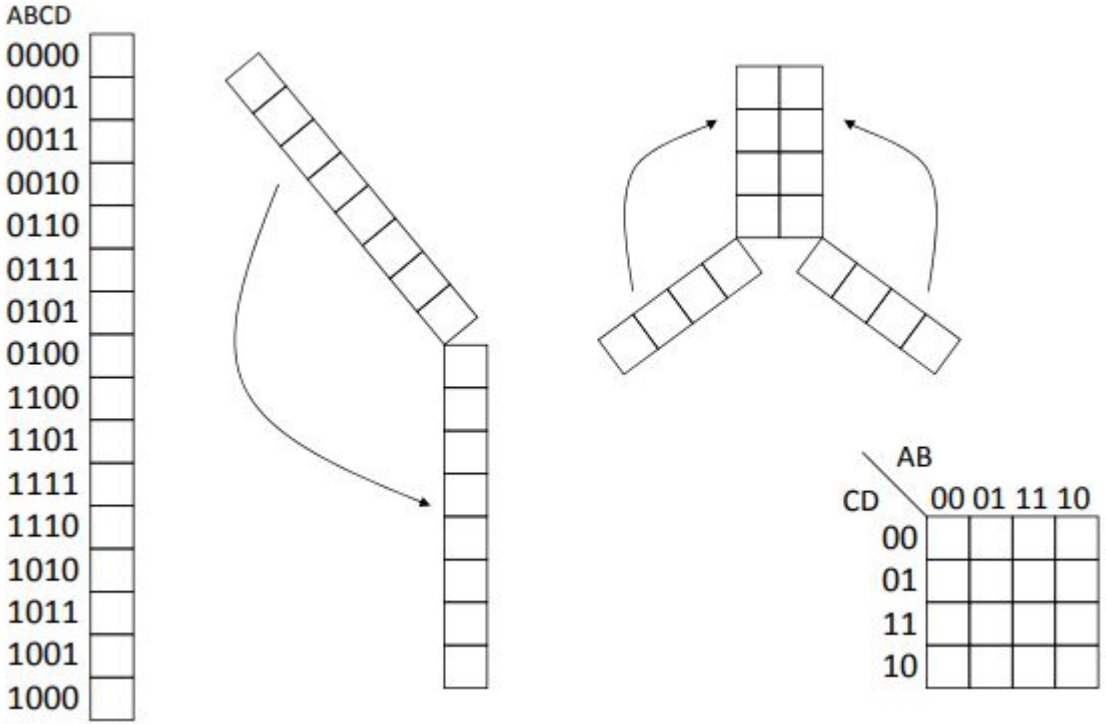
Standart dogruluk çizelgesi

A	B	C	D	minterm
0	0	0	0	m0
0	0	0	1	m1
0	0	1	0	m2
0	0	1	1	m3
0	1	0	0	m4
0	1	0	1	m5
0	1	1	0	m6
0	1	1	1	m7
1	0	0	0	m8
1	0	0	1	m9
1	0	1	0	m10
1	0	1	1	m11
1	1	0	0	m12
1	1	0	1	m13
1	1	1	0	m14
1	1	1	1	m15

Gray kodları

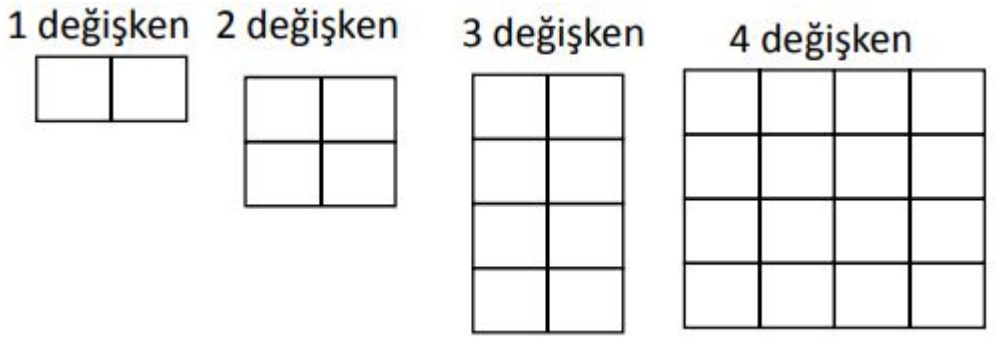
A	B	C	D	minterm
0	0	0	0	m0
0	0	0	1	m1
0	0	1	1	m3
0	0	1	0	m2
0	1	1	0	m6
0	1	1	1	m7
0	1	0	1	m5
0	1	0	0	m4
1	1	0	0	m12
1	1	0	1	m13
1	1	1	1	m15
1	1	1	0	m14
1	0	1	0	m10
1	0	1	1	m11
1	0	0	1	m9
1	0	0	0	m8

Gray Kodlardan K Haritasına Dönüştürme

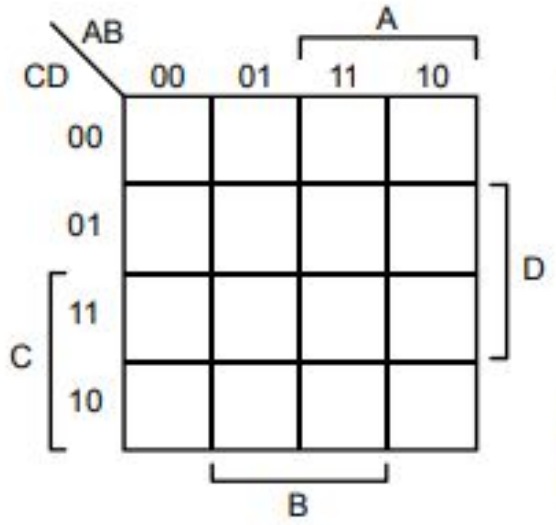


Harita (K-Map)

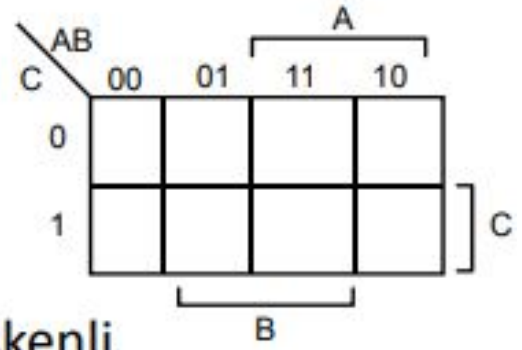
1 degiskenli harita $2^1 = 2$
2 degiskenli harita $2^2 = 4$
3 degiskenli harita $2^3 = 8$
4 degiskenli harita $2^4 = 16$
hücreye sahiptir



Harita (K-Map)



4 değişkenli



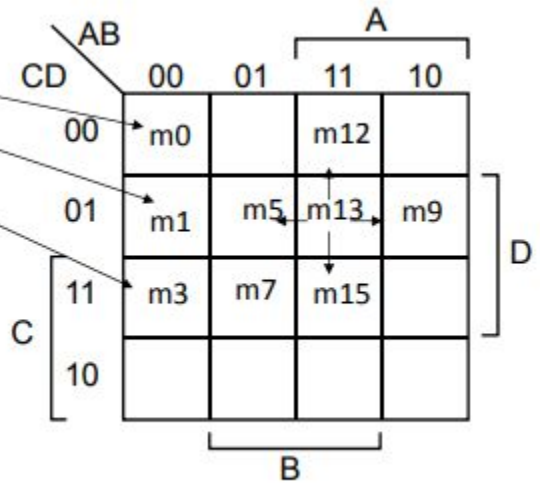
3 değişkenli

En üst ve en alttaki hücreler, en sağ ve en sol hücreler de bitişiktir.

Doğruluk Tablosundan Haritaya

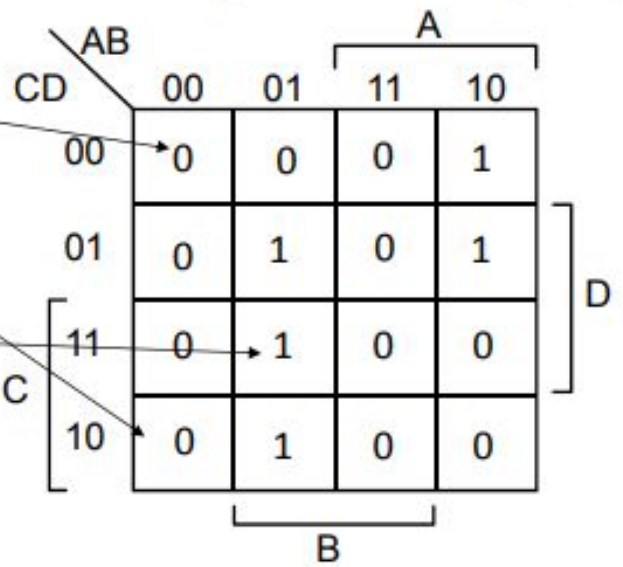
Dogruluk tablosundaki satırların sayısı ile haritanın hücrelerinin sayısı aynı olmalıdır.!

A	B	C	D	F
0	0	0	0	x
0	0	0	1	x
0	0	1	0	x
0	0	1	1	x
0	1	0	0	x
0	1	0	1	x
0	1	1	0	x
0	1	1	1	x
1	0	0	0	x
1	0	0	1	x
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x



Harita ile Sadeleştirme

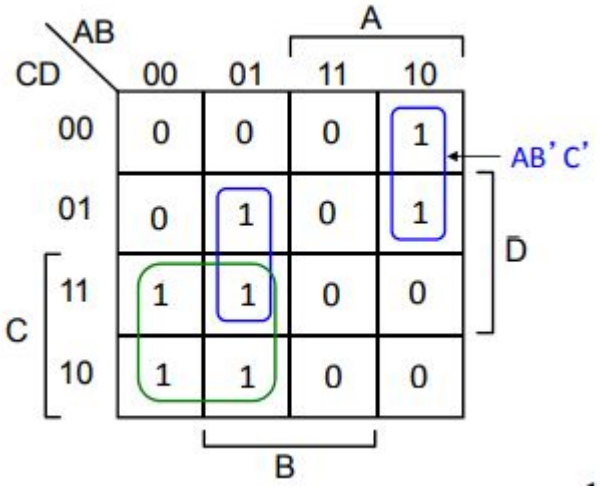
A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



10 ve 11 sütunlarının yerine dikkat!

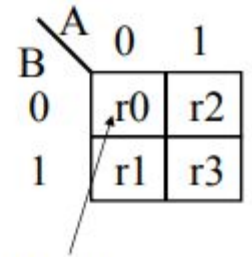
Grupların okunması

	1leri gruplama	0ları gruplama
Değişken değişiyor	*Dahil etme*	*Dahil etme*
Değişken sabit 0	tümleri	kendisi
Değişken sabit 1	kendisi	tümleri

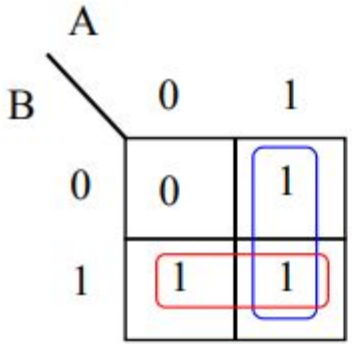


2 Değişkenli Harita

Satır	A	B	F(A,B)
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1



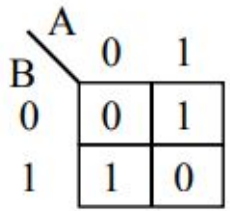
Satır 0,
A=0, B= 0



F(A,B) = A + B

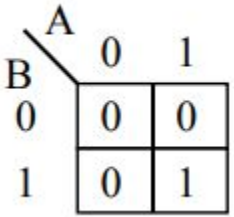
2 Değişkenli Harita

Satır	A	B	F1(A,B)
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0



$F1(A,B) = A'B + AB'$

Satır	A	B	F2(A,B)
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1



$F2(A,B) = AB$

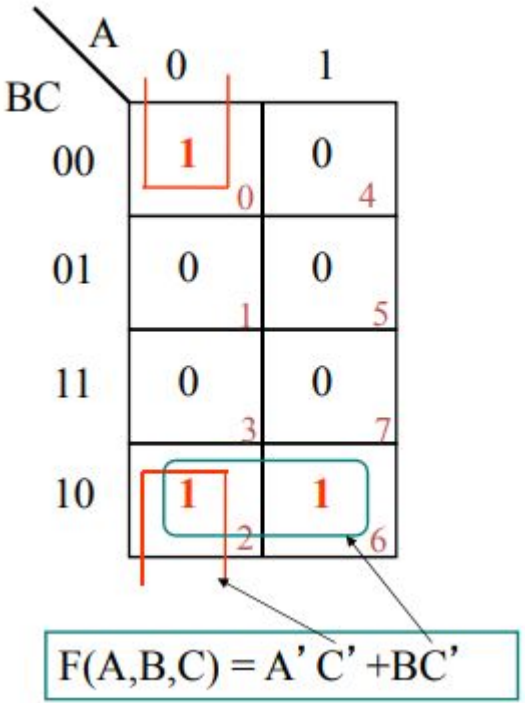
3 Değişkenli Harita

Satır	A B C	F(A,B,C)
0	0 0 0	1
1	0 0 1	0
2	0 1 0	1
3	0 1 1	0
4	1 0 0	0
5	1 0 1	0
6	1 1 0	1
7	1 1 1	0

$F(A,B,C) = \Sigma m(0,2,6)$

$F'(A,B,C) = \Sigma m(1,3,4,5,7)$

$F(A,B,C) = \pi M(1,3,4,5,7)$



K Haritaları

$$Y = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C = \overline{A}\,\overline{B}(\overline{C} + C) = \overline{A}\,\overline{B}$$

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

$\overline{A}\,\overline{B}$

Y \ AB / C	00	01	11	10
0	$\overline{A}\overline{B}\overline{C}$	$\overline{A}B\overline{C}$	$AB\overline{C}$	$A\overline{B}\overline{C}$
1	$\overline{A}\overline{B}C$	$\overline{A}BC$	ABC	$A\overline{B}C$

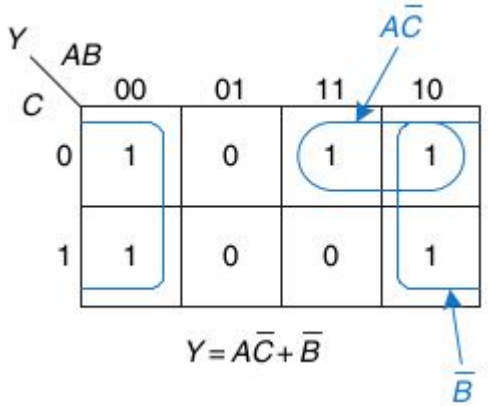
Y \ AB / C	00	01	11	10
0	1	0	0	0
1	1	0	0	0

Y \ AB / C	00	01	11	10
0	1	0	0	0
1	1	0	0	0

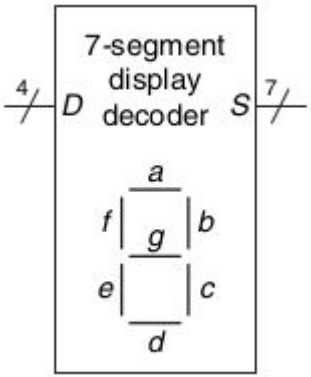
Örnek: K Haritaları

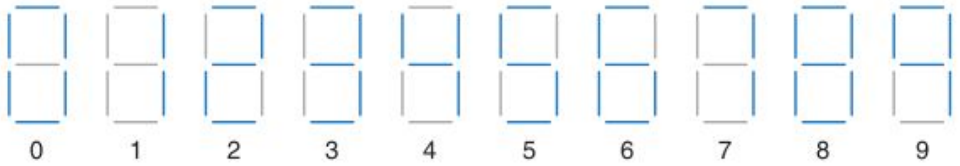
$$Y = F(A, B, C)$$

Y, AB \ C	00	01	11	10
0	1	0	1	1
1	1	0	0	1



Yedi Parçalı Gösterge Sürücüsü

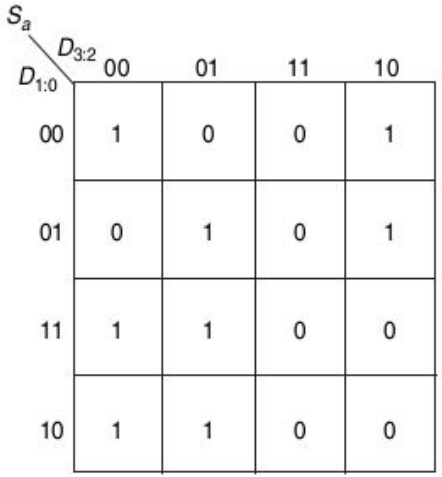


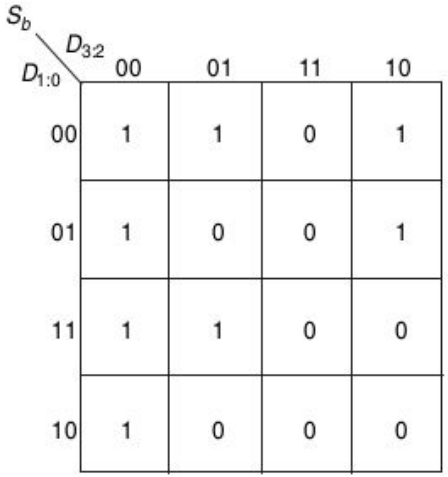


$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
others	0	0	0	0	0	0	0

Yedi Parçalı Gösterge Sürücüsü

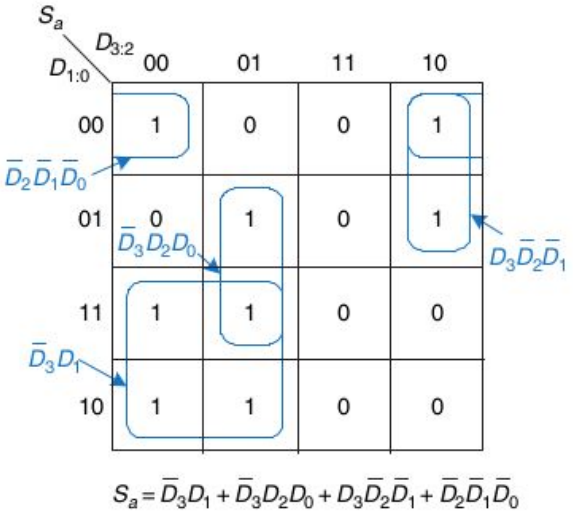
$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
others	0	0	0	0	0	0	0

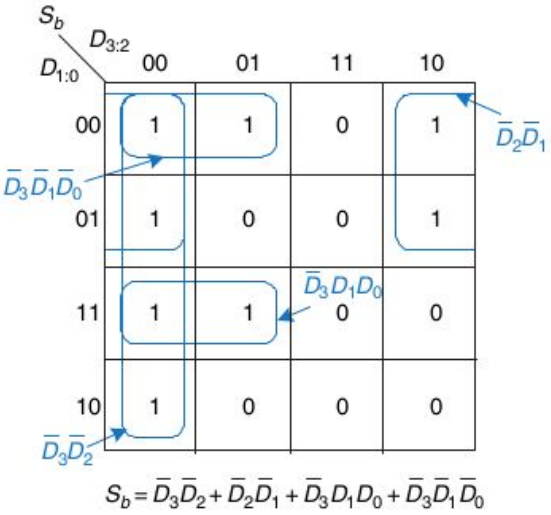




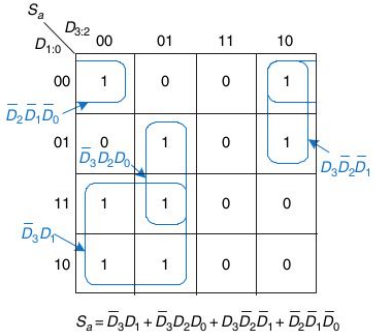
Yedi Parçalı Gösterge Sürücüsü

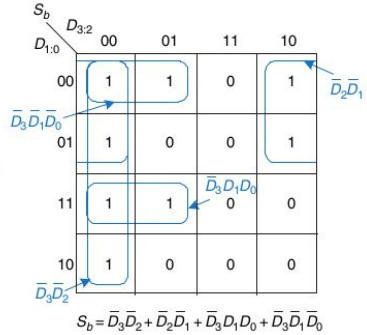
$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
others	0	0	0	0	0	0	0

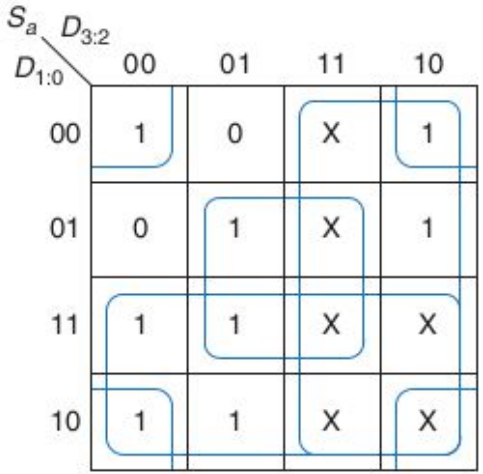




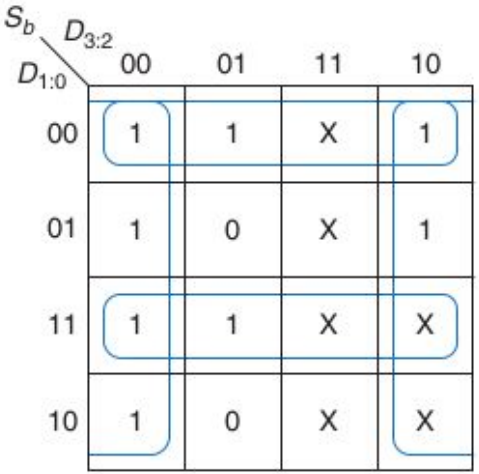
Önemsenmeyen Durumlar





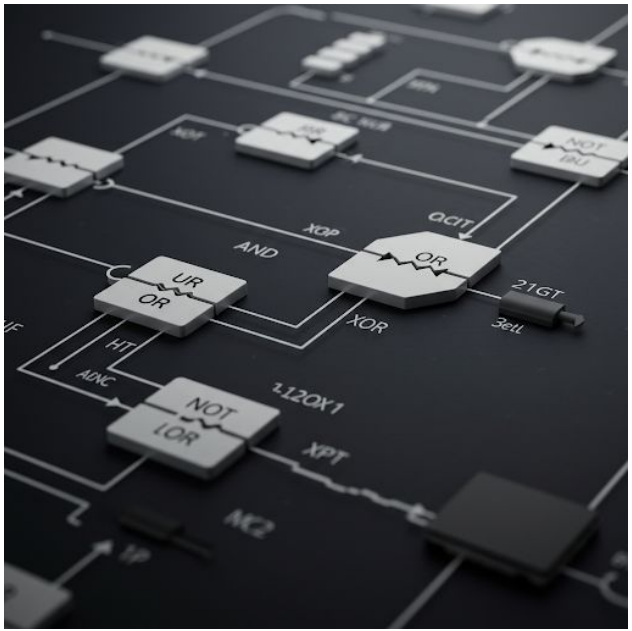


$S_a = D_3 + D_2D_0 + \bar{D}_2\bar{D}_0 + D_1$



$S_b = \bar{D}_2 + D_1D_0 + \bar{D}_1\bar{D}_0$

Birleşik Mantık Devreleri



- Çoklayıcı (MUX): Birden fazla girişi tek bir çıkışa yönlendirir.
- Dağıtıcı (DEMUX): Tek bir girişi birden fazla çıkışa yönlendirir.
- Kod Çözücü (Decoder): Kodlanmış girişi kodlanmamış çıkışa dönüştürür.
- Kodlayıcı (Encoder): Kodlanmamış girişi kodlanmış çıkışa dönüştürür.
- Karşılaştırıcı (Comparator): İki giriş değerini karşılaştırır.
- Toplayıcılar: İki veya daha fazla sayıyı toplar.
- ALU (Aritmetik Mantık Birimi): Aritmetik ve mantık işlemlerini gerçekleştirir.

Çoklayıcı / Multiplexer

A (Kapı)	B (Pencere)	C (Hareket)	F (Alarm)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Bir akıllı ev güvenlik sistemi, üç farklı sensörden gelen verileri (A, B ve C) kullanarak bir alarmı (F) tetikleyecektir.

Değişkenler:

- A: Kapı sensörü (0: Kapalı, 1: Açık)
- B: Pencere sensörü (0: Kapalı, 1: Açık)
- C: Hareket sensörü (0: Yok, 1: Var)
- F: Alarm (0: Pasif, 1: Aktif)

Alarm (F) aşağıdaki durumlarda tetiklenecektir:

- Kapı kapalı (A=0) ve pencere (B=1) veya hareket sensörü (C=1)
- Kapı açık (A=1), pencere kapalı (B=0), hareket yok (C=0)
- Kapı açık (A=1), pencere açık (B=1), hareket var (C=1)

Çoklayıcı / Multiplexer

A (Kapı)	B (Pencere)	C (Hareket)	F (Alarm)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

A \ BC	00	01	11	10
0	0	1	1	1
1	1	0	1	0

F = AB’C’ + BC + A’C+A’B

Bir akıllı ev güvenlik sistemi, üç farklı sensörden gelen verileri (A, B ve C) kullanarak bir alarmı (F) tetikleyecektir.

Değişkenler:

- A: Kapı sensörü (0: Kapalı, 1: Açık)
- B: Pencere sensörü (0: Kapalı, 1: Açık)
- C: Hareket sensörü (0: Yok, 1: Var)
- F: Alarm (0: Pasif, 1: Aktif)

Alarm (F) aşağıdaki durumlarda tetiklenecektir:

- Kapı kapalı (A=0) ve pencere (B=1) veya hareket sensörü (C=1)
- Kapı açık (A=1), pencere kapalı (B=0), hareket yok (C=0)
- Kapı açık (A=1), pencere açık (B=1), hareket var (C=1)

Çoklayıcı / Multiplexer

A (Kapı)	B (Pencere)	C (Hareket)	F (Alarm)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

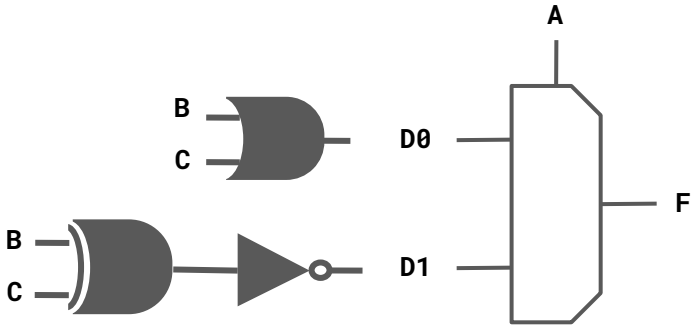
A \ BC	00	01	11	10
0	0	1	1	1
1	1	0	1	0

F = AB’C’ + ABC + A’C+A’B

A → S

D_0 = B + C (OR kapısı)

D_1 = B'C' + BC = (B ⊕ C)’ (XOR kapısının değili)



Dağıtıcı / DeMultiplexer

S1	S0	Y (Açılan Lamba)
0	0	Y0: Salon
0	1	Y1: Mutfak
1	0	Y2: Yatak Odası
1	1	Y3: Banyo

Bir akıllı ev sisteminde, tek bir kontrol sinyali kullanarak, evin 4 farklı odasındaki lambalardan sadece birini yakmak istenmektedir.

Odalar:

- Y0: Salon
- Y1: Mutfak
- Y2: Yatak Odası
- Y3: Banyo

Kullanıcı mobil uygulama üzerinden hangi odanın ışığını açmak istediğini seçiyor.

Kullanıcı "Mutfak" seçti → S1 = 0, S0 = 1

Y1 = 1 → Mutfak lambası açılır.

Diğer çıkışlar = 0 → Diğer odalardaki lambalar kapalı kalır.

Dağıtıcı / DeMultiplexer

S1	S0	Y3 Banyo	Y2 Yatak Odası	Y1 Mutfak	Y0 Salon
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Bir akıllı ev sisteminde, tek bir kontrol sinyali kullanarak, evin 4 farklı odasındaki lambalardan sadece birini yakmak istenmektedir.

Odalar:

- Y0: Salon
- Y1: Mutfak
- Y2: Yatak Odası
- Y3: Banyo

Kullanıcı mobil uygulama üzerinden hangi odanın ışığını açmak istediğini seçiyor.

S1 \ S0	0	1
0	0	1
1	0	0

Y1 = S1’S0

Kullanıcı "Mutfak" seçti → S1 = 0, S0 = 1

Y1 = 1 → Mutfak lambası açılır.

Diğer çıkışlar = 0 → Diğer odalardaki lambalar kapalı kalır.

Dağıtıcı / DeMultiplexer

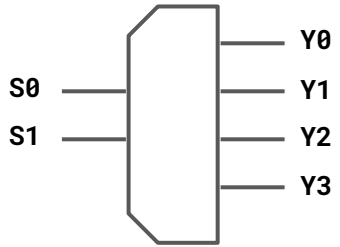
S1	S0	Y3 Banyo	Y2 Yatak Odası	Y1 Mutfak	Y0 Salon
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Bir akıllı ev sisteminde, tek bir kontrol sinyali kullanarak, evin 4 farklı odasındaki lambalardan sadece birini yakmak istenmektedir.

Odalar:

- Y0: Salon
- Y1: Mutfak
- Y2: Yatak Odası
- Y3: Banyo

Kullanıcı mobil uygulama üzerinden hangi odanın ışığını açmak istediğini seçiyor.



Kod Çözücü / Decoder

Bir sunucuda, gelen verinin hangi bellek bloğuna yazılacağını belirlemek.

Girişler: Bellek bloğunu seçer

- 000 → Bellek Bloğu 0
- 001 → Bellek Bloğu 1
- ...
- 111 → Bellek Bloğu 7

Çıkışlar (Y0-Y7): Her biri farklı bir bellek blokunu temsil eder.

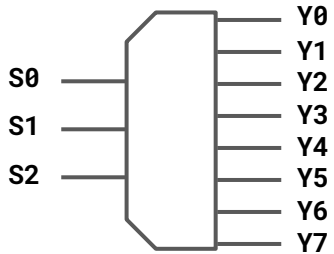
Örnek: Gelen verinin Bellek Bloğu 5'e yazılması istenirse;

Adres giriş: `101` ise Y5 = 1 → Bellek Bloğu 5 etkin

S2	S1	S0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

S2 \ S1S0	00	01	11	10
0	0	0	0	0
1	0	1	0	0

Y5 = S2S1'S0



Decoder VS DeMultiplexer

Özellik	Decoder	Demultiplexer (DEMUX)
Amaç	Girişteki ikili kodu çözmek	Veriyi seçilen çıkışa yönlendirmek
Giriş Sayısı	n giriş → 2^n çıkış	1 veri girişi + n seçim hattı
Çıkış	Yalnızca biri "1", geri kalanı "0"	Seçilen çıkışa veri gönderilir
Veri Girişi	Yok	Var (D)

Özellik	Decoder	DEMUX
Veri taşır mı?	❌ Hayır	✅ Evet
Kullanım amacı	Kod çözme (adresleme)	Veri yönlendirme
Kullanıldığı yerler	Bellek adresleme, kontrol devreleri	Veri yönlendirme, iletişim sistemleri
Giriş türü	Sadece seçim/kod girişleri	Veri girişi + seçim girişleri

Karşılaştırıcı / Comparator

Sınav Değerlendirme sistemi

- **Giriş A:** Öğrencinin sınavdan aldığı puan (örneğin 6 bit'lik sayılar: 0–63)
- **Giriş B:** Başarı barajı (örneğin 50 puan)
- **Çıkışlar:**

Bir **3-bit comparator** entegresi örneğin şu çıkışları üretir:

- **A > B**: `Geçer`
- **A = B**: `Eşit`
- **A < B**: `Kalır`

A (Puan)	B (Baraj)	Çıkış
55	50	A > B → Geçti ✅
50	50	A = B → Sınırda ⚠️
42	50	A < B → Kaldı ❌

Karşılaştırıcı / Comparator

A1	A0	B1	B0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Sınav Değerlendirme sistemi

Basitleştirmek için: 2-bit'lik sayılar kullanalım

- **A (Öğrenci Notu):** A1, A0
- **B (Baraj):** B1, B0

A1A0 \ B1B0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

A > B = A1·B1' + A1·A0·B0' + A0·B1'·B0'

Toplayıcılar

Özellik	Half Adder	Full Adder
Giriş Sayısı	2 (A, B)	3 (A, B, Carry-in)
Kullanım Yeri	Basit toplama	Zincirleme toplamalarda
Gerçek Dünya	Saat, sayaç, toplama devresi	ALU, CPU toplama, timer devresi

Bir dijital saat devresindeki saniye sayacının tasarımı

Saat ilk saniyeyi artırırken 2-bitlik sayıyı toplar. Ancak saniye 1 + 1 olduğunda sonuç `10` olur
→ 1 bit taşma oluşur Bu durumda Half Adder yetersiz kalır.

Saat 4-bitlik (0–15) çalışıyorsa;

- `A =  0111` (7)
- `B =  0001` (1)
- Sonuç: `1000` (8)
-

1. Bit 0: 1 + 1 + 0 → Sum: 0, Carry: 1
2. Bit 1: 1 + 0 + 1 → Sum: 0, Carry: 1
3. Bit 2: 1 + 0 + 1 → Sum: 0, Carry: 1
4. Bit 3: 0 + 0 + 1 → Sum: 1, Carry: 0

İlk bit dışında her bit için Full Adder gereklidir çünkü taşıma zincirleme olarak ilerler.

Toplayıcılar / Yarım Toplayıcı

A	B	Toplam	Elde
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- **Girişler:** A, B
- **Çıkışlar:**
  - **Toplam = A ⊕ B**
  - **Elde = A · B**

A \ B	0	1
0	0	1
1	1	0

Toplam = A’B+B’A = A ⊕ B

A \ B	0	1
0	0	0
1	0	1

Elde = AB

Toplayıcılar / Tam Toplayıcı

A	B	Elde Giriş	Toplam	Elde Çıkış
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- **Girişler:** A, B, Cin (carry-in)
- **Çıkışlar:**
  - **Toplam = A ⊕ B ⊕ Cin**
  - **Elde = (A·B) + (B·Cin) + (A·Cin)**

Cin \ AB	00	01	11	10
0	0	1	1	0
1	1	0	1	0

Toplam = A ⊕ B ⊕ Cin

Cin \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Elde = AB+ACin+BCin

ALU

İşlem	ALU Fonksiyonu	Açıklama
Sınav1 + Sınav2	Toplama	Aritmetik toplama (Sum)
Ortalama < 50	Karşılaştırma / Çıkarma	Kaldı mı kontrolü
Devamsızlık > 10 gün	Karşılaştırma	Şart kontrolü
Not >= 90	Mantık (AND, CMP)	Takdirname
1'si devamsız, notu < 50	Mantık (AND)	Koşullu kontrol

Sistem Görevi	**ALU İşlevi**
Notları toplamak	Toplayıcı
Ortalama hesaplamak	Toplayıcı / bölücü (bazı ALU'larda olur)
Sınıfı geçti mi kontrolü	Karşılaştırıcı (A > B)
Devamsızlıkla kaldı mı	AND + karşılaştırma
Takdirname verilecek mi?	Mantık işlemi (AND, OR)

Öğrencilerin sınav, proje ve devamsızlık bilgilerine göre; not ortalamasını ve geçti kaldı kararını veren sistem.

🛠️ ALU Nasıl Çalışır:

- 2 adet giriş (A, B)
  - A = 0110 (6), B = 0011 (3),
- bir komut (opcode): "Topla", "Çıkar", "AND", "OR"
  - Opcode = 01 (çıkarma)
- ALU sonucu: `0110 - 0011 = 0011` (3)

ALU

ALU'ya 2 giriş verilir: A ve B
Ayrıca bir kontrol sinyali (Opcode) gelir. Bu sinyal, ALU'nun hangi işlemi yapacağını belirler.

📋 ALU Kontrol Sinyalleri (Opcode)

Opcode	ALU İşlemi	Açıklama
0000	AND	A ∧ B (bit-bit mantıksal and)
0001	OR	A ∨ B
0010	ADD	A + B
0110	SUBTRACT	A - B
0111	SET ON LESS THAN (SLT)	A < B ise 1, değilse 0
1100	NOR	¬(A ∨ B)

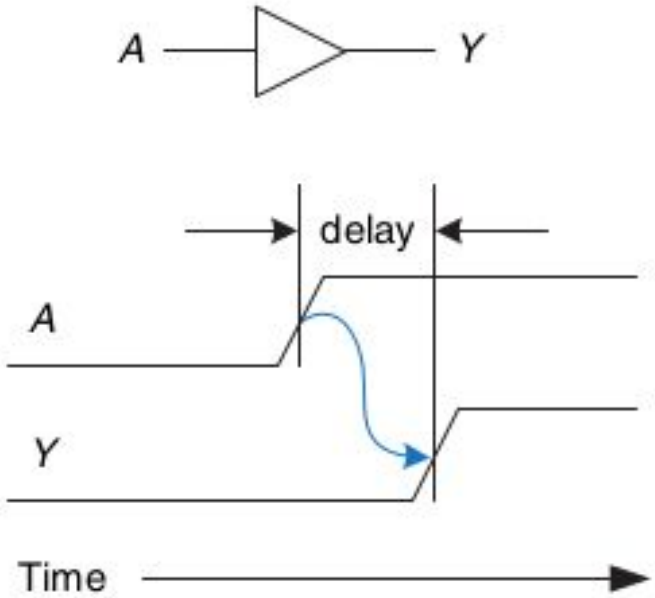
🎓 Uygulama Örneği: Not Hesaplama

Not A	Not B	Opcode	Açıklama
70	30	0010	70 + 30 = 100
85	60	0110	85 - 60 = 25
40	50	0111	40 < 50 → 1

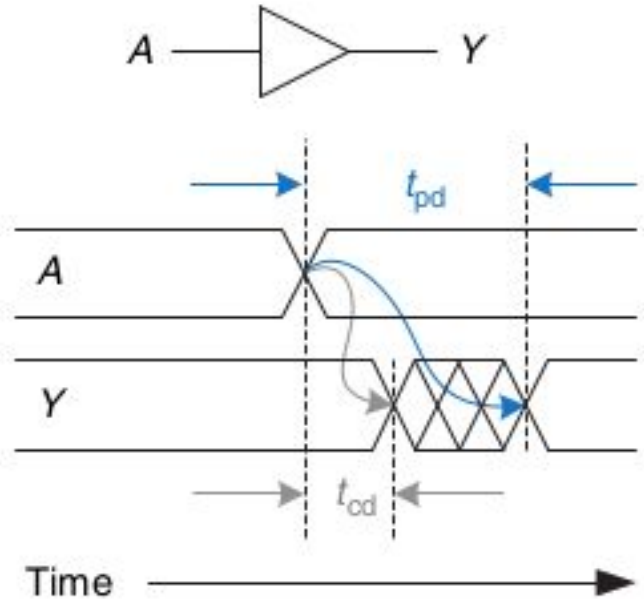
✅ ALU Kontrol Tablosu

A	B	Opcode	ALU Çıkışı	Açıklama
0101	0011	0000	0001	5 AND 3 = 1
0101	0011	0001	0111	5 OR 3 = 7
0101	0011	0010	1000	5 + 3 = 8
0101	0011	0110	0010	5 - 3 = 2
0011	0101	0110	1110 (negatif)	3 - 5 = -2 (2's tamamı)
0011	0101	0111	0001	3 < 5 → 1
0101	0011	0111	0000	5 < 3 → 0
0101	0011	1100	1000	NOR(5,3) = ¬ (0111) = 1000

Zamanlama



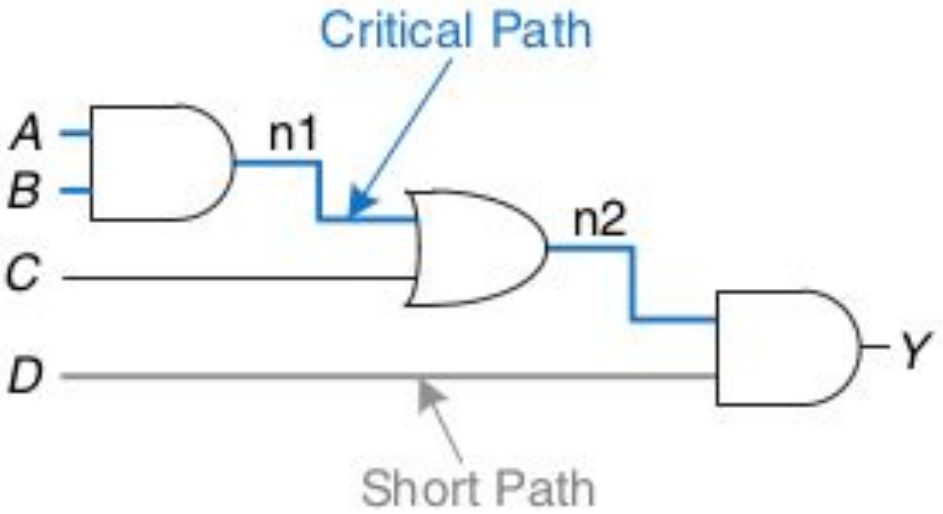
Yayılma ve Kirlenme Gecikmesi



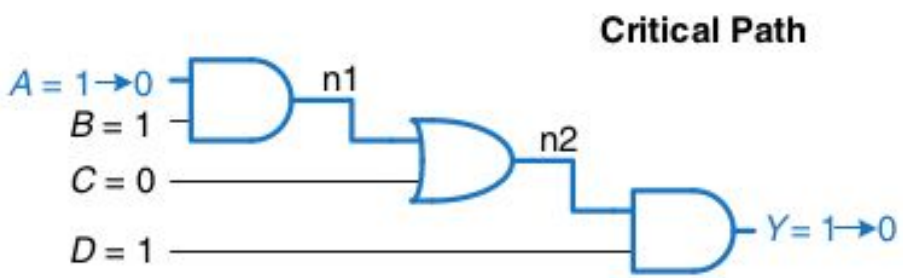
t_{pd}: Yayılma gecikmesi

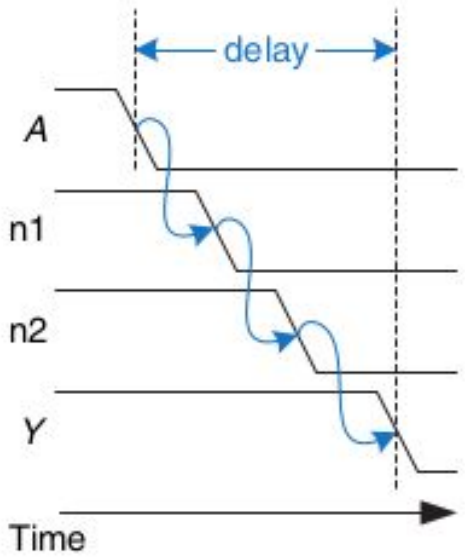
t_{cd}: Kirlenme gecikmesi

Yayılma Gecikmesi



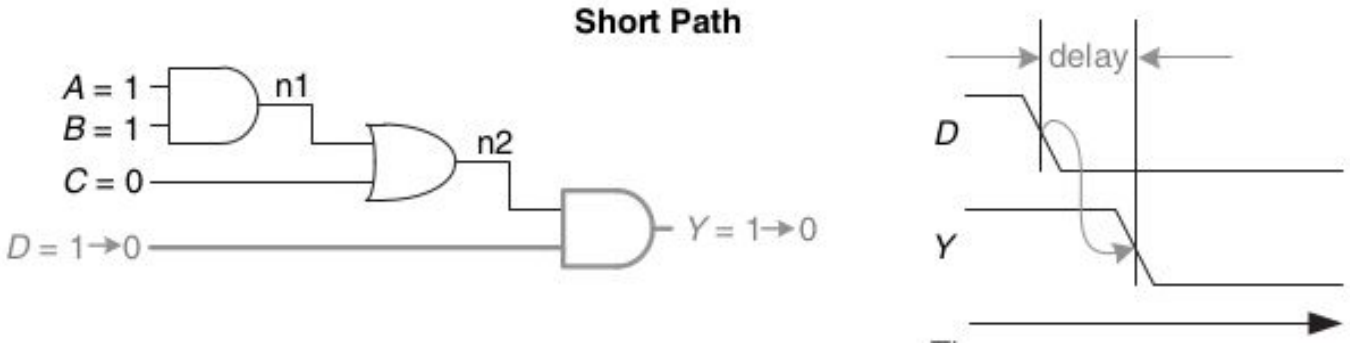
Yayılma Gecikmesi





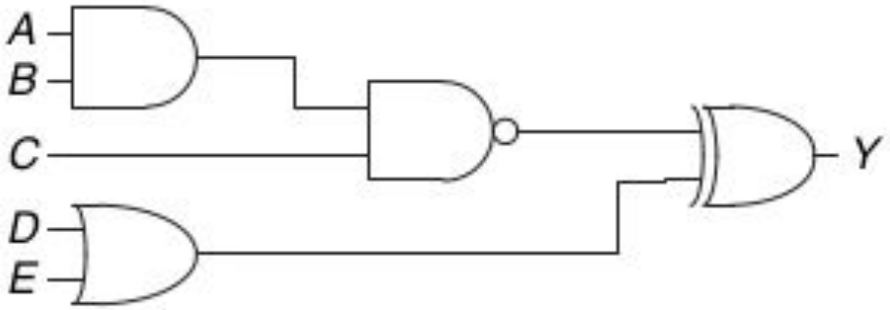
$$t_{pd} = 2t_{pd\_\mathrm{AND}} + t_{pd\_\mathrm{OR}}$$

Kirlenme Gecikmesi



$$t_{cd} = t_{cd\_\mathrm{AND}}$$

Gecikmelerin Tespiti Örnek

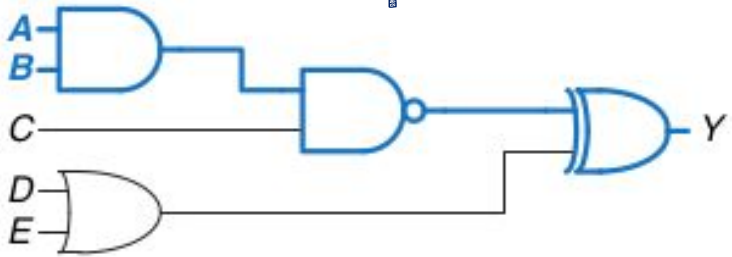


t_{pd}: 100 ps

t_{cd}: 60 ps

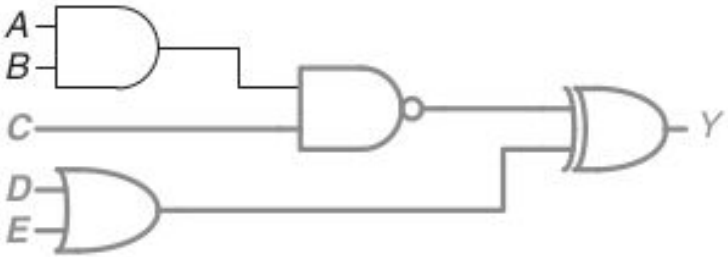
Gecikmelerin Tespiti Örnek

critical path



t_{pd}: 3 x 100 ps = 300 ps

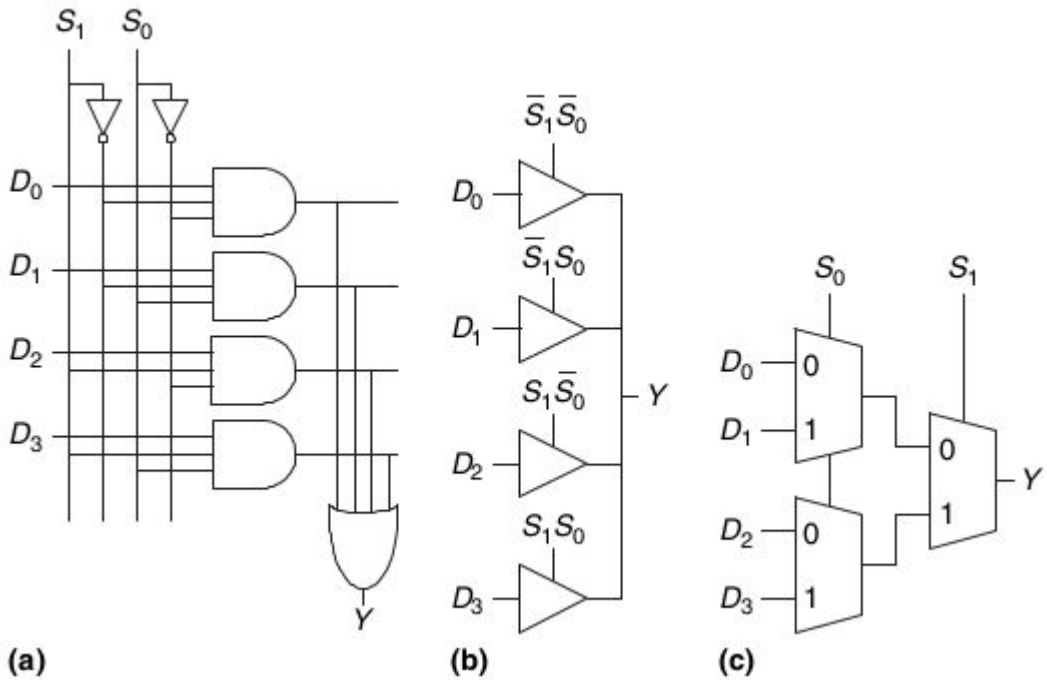
shortest path



t_{cd}: 2 x 60 ps = 120 ps

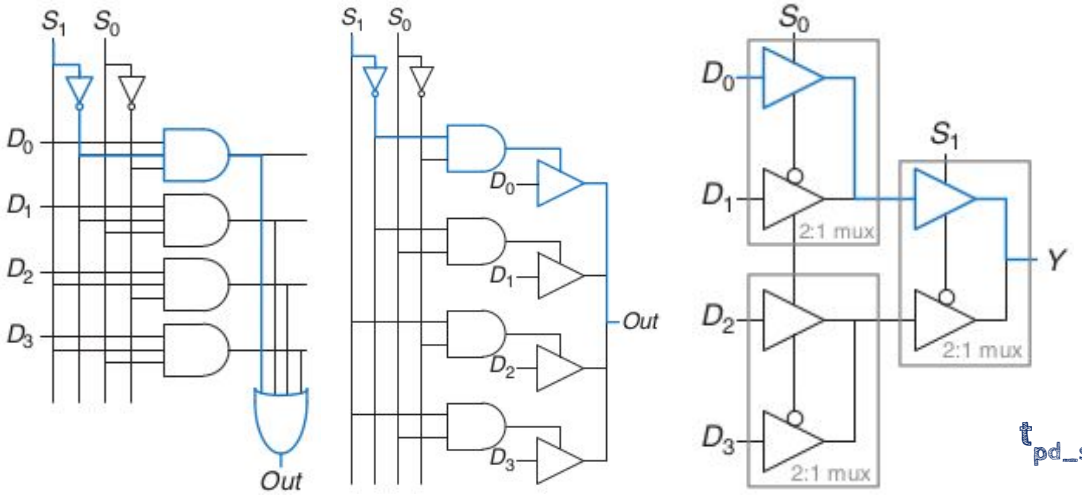
Çoklayıcı (Mux) Zamanlama

Asagıdaki Mux devrelerinin zamanlamalarını karsılastırın



Gate	t_{pd} (ps)
NOT	30
2-input AND	60
3-input AND	80
4-input OR	90
tristate (A to Y)	50
tristate (enable to Y)	35

Çoklayıcı (Mux) Zamanlama



Gate	t_{pd} (ps)
NOT	30
2-input AND	60
3-input AND	80
4-input OR	90
tristate (A to Y)	50
tristate (enable to Y)	35

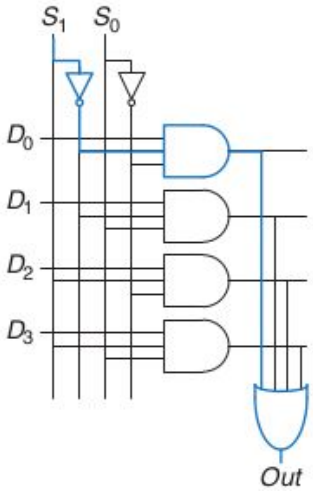
t_{pd_sy}: S ile Y arasındaki yayılma gecikmesi

t_{pd_dy}: D ile Y arasındaki yayılma gecikmesi

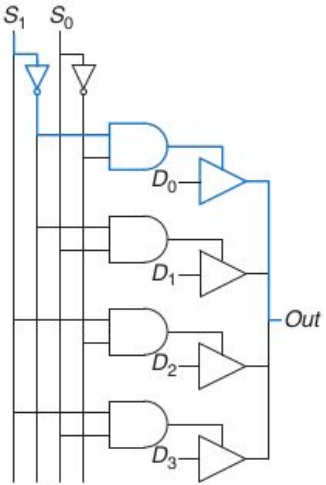
t_{pd}: $\max(t_{pd\_sy}, t_{pd\_dy})$

Çoklayıcı (Mux) Zamanlama

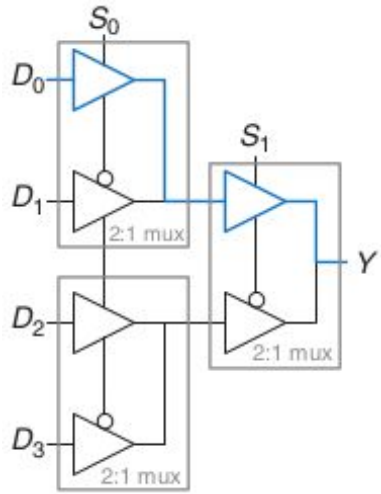
Gate	t_{pd} (ps)
NOT	30
2-input AND	60
3-input AND	80
4-input OR	90
tristate (*A* to *Y*)	50
tristate (enable to *Y*)	35



$$t_{pd\_sy} = t_{pd\_INV} + t_{pd\_AND3} + t_{pd\_OR4}$$
$$= 30\text{ ps} + 80\text{ ps} + 90\text{ ps}$$
$$= \mathbf{200\text{ ps}}$$
$$t_{pd\_dy} = t_{pd\_AND3} + t_{pd\_OR4}$$
$$= \mathbf{170\text{ ps}}$$



$$t_{pd\_sy} = t_{pd\_INV} + t_{pd\_AND2} + t_{pd\_TRI\_sy}$$
$$= 30\text{ ps} + 60\text{ ps} + 35\text{ ps}$$
$$= \mathbf{125\text{ ps}}$$
$$t_{pd\_dy} = t_{pd\_TRI\_ay}$$
$$= \mathbf{50\text{ ps}}$$



$$t_{pd\_s0y} = t_{pd\_TRLSY} + t_{pd\_TRI\_AY} = \mathbf{85\text{ ns}}$$
$$t_{pd\_dy} = 2\,t_{pd\_TRI\_AY} = \mathbf{100\text{ ns}}$$