

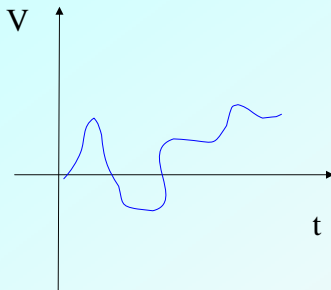
SAYISAL DEVRELER

Yrd.Doç.Dr. Feza BUZLUCA
İstanbul Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü
www.buzluca.info/sayisal

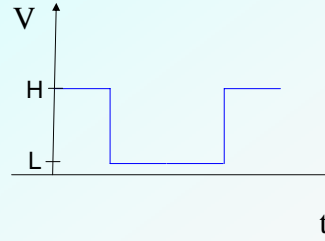
Analog - Sayısal (Dijital) İşaretler:

Gerçek dünyada karşılaştığımız bir çok fiziksel büyüklüğün (akım, gerilim, sıcaklık, ışık şiddeti vb.) değeri sürekli bir aralık içinde değişmektedir. Sınırlar arasındaki her türlü olası değeri alabilen bu tür işaretlere **analog** işaretler denir.

İkili (binary) **sayısal** işaretler ise belli bir anda sadece olası iki değerden birini alabilirler: 0 - 1, yüksek - alçak, doğru - yanlış, açık - kapalı.



Analog işaret



İkili sayısal işaret

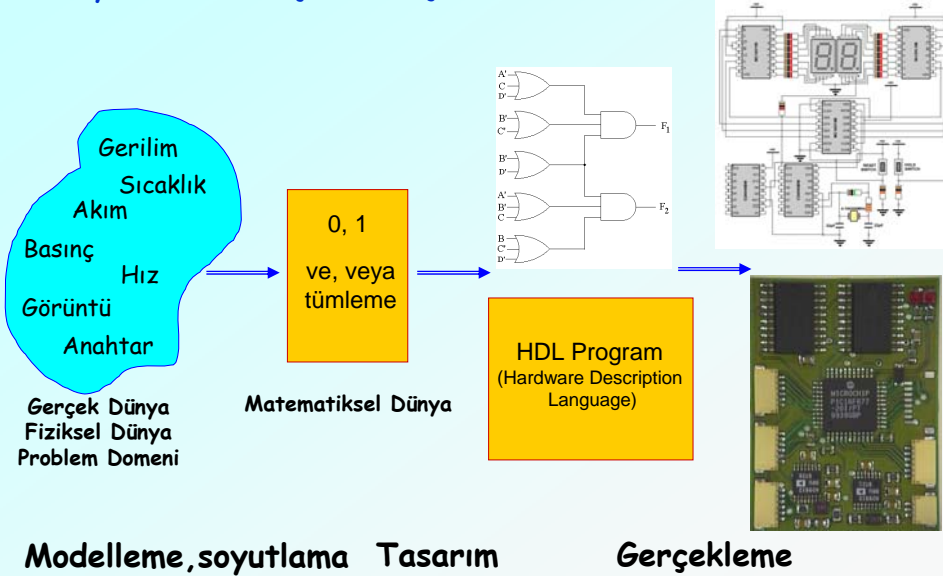
Sayısal Sistemlerin Avantajları:

Eskiden analog sistemlerin kullanıldığı bir çok alanda günümüzde daha avantajlı olduğundan sayısal sistemler kullanılmaktadır.

Örnekler: Fotoğrafçılık, video, ses kayıtları, otomobil motorları, telefon sistemleri vb.

Sayısal Sistemlerin Avantajları:

- Bir sayısal sisteme belli bir giriş kümesi defalarca uygulandığında hep aynı çıkış kümesi elde edilir. Burada aynı giriş kümesinin uygulanması demek her defasında aynı değer dizisinin aynı sırada uygulanması demektir. Analog sistemler ise çevre koşullarından daha çok etkilenirler.
- Sayısal tasarım (lojik tasarım) dayandığı matematiksel temeller açısından daha kolaydır. Ayrıca sayısal sistemleri test etmek ve hatalardan arındırmak da analog sistemlere göre daha kolaydır.
- Esneklik ve programlanabilirlik. Günümüzde sayısal sistemleri programlanabilir bilgisayarlar şeklinde gerçeklemek mümkündür. Bu sayede aynı tasarım yeni gereksinimlere göre yeniden programlanarak tekrar kullanılabilir.
- Sayısal verileri bilgisayar ortamında saklamak ve işlemek mümkündür.
- Sayısal sistemler daha hızlı çalışmaktadır.
- Sayısal sistemler küçülmekte ve ucuzlamaktadır.
- Sayısal sistemler gelişmeye devam ediyor.

Sayısal Devre Gerçekleme Aşamaları:

Sayısal Kodlama:

Sayısal sistemler ikili sayısal işaretler üzerinde işlemler yaptıklarından sadece iki farklı değeri işleyebilirler.

Bu nedenle sayısal devreler yardımıyla üzerinde işlem yapılacak olan fiziksel büyüklüklere (gerilim, sıcaklık vs.) ve her türlü veriye (harf, sayı, renk, ses) ikili sayılar karşı düşürülür.

Örneğin 8 basamaklı (8 bitlik "*Binary digit*") bir ikili sayı kullanarak 2^8 tane (256) farklı "şey" ifade edebiliriz. Bunlar 256 farklı renk, 256 sembol, 0 ile 255 arası tamsayılar, 1 ile 256 arası tamsayılar, -128 ile +127 arası tamsayılar olabilir.

Bir ikili değer (Örneğin 10001101) ne anlama geldiğine o değeri kullanacak olan sistem (donanım ya da yazılım sistemi olabilir) ya da kişi belirler. Bu değer bir sayı da olabilir bir renk de.

Özellikle sayıların kodlanması büyük önem taşır. Bu konu mikroişlemci sistemleri dersinde ele alınacaktır. Bu derste bazı temel kodlama yöntemlerine ilişkin bilgiler verilecektir.

BCD (*Binary Coded Decima*) İkili kodlanmış onlu sayılar:

0-9 arasındaki rakamlara 4 bitlik bir ikili kod karşı düşürülür.

Doğal BCD:

Sayı:	Kod:	Sayı:	Kod:	Örnek:
0:	0000	5:	0101	Sayı: 805
1:	0001	6:	0110	Kod:1000 0000 0101
2:	0010	7:	0111	
3:	0011	8:	1000	
4:	0100	9:	1001	

Artıklı kodlamadır. Çünkü 4 bit ile 16 farklı kodlama yapılabilmekte ancak bunlardan sadece 10 tanesi kullanılmaktadır.

BCD sayılar üzerinde işlem yapmak zor olduğundan günümüz bilgisayarlarında sayıları göstermek için bu kodlama kullanılmamaktadır.

Ağırlıklı Kodlama:

Bitlerin konumlarına birer ağırlık verilir.

Doğal ikili kodlama: Sayıların ağırlıklı kodlama ile 2 tabanında gösterilmesidir.

Örneğin: $11010 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 26$

Soldaki ilk basamağa yüksek anlamlı bit (*Most Significant Bit - MSB*)

Sondaki basamağa düşük anlamlı bit (*Least Significant Bit - LSB*) denir.

Bilgisayarlarda sayıları göstermek için doğal ikili kodlama kullanılır.

Hamming uzaklığı: n uzunluğundaki iki kod sözcüğünün arasındaki Hamming uzaklığı, o sözcüklerdeki aynı sırada olup değerleri farklı olan bileşenlerin sayısıdır.

Örneğin: 011 ile 101 arasındaki uzaklık 2 dir.

Bitişik kodlar: Bir birini izleyen sayılara karşı gelen kodlar arasındaki uzaklık 1 ise o kodlama bitişiktir.

Ayrıca son sayı ile ilk sayı arasındaki uzaklık da 1 olursa kod **çevrimlidir**.

Örnek: Çevrimli bir BCD kodu (doğal BCD'den farklı)

Sayı:	Kod:	Sayı:	Kod:
0:	0000	5:	1110
1:	0001	6:	1010
2:	0011	7:	1000
3:	0010	8:	1100
4:	0110	9:	0100

Gray Kodu: 2^n elemanlı bir küme için 2 tabanında **artıksız ve çevrimli** bir kodlama yapılırsa **gray kodu** elde edilir.

Örnek: 2 bitlik bir Gray kodu:

Sayı:	Kod:
0:	00
1:	01
2:	11
3:	10

Sayıların Bilgisayarda Gösterilimi

Bu derste tamsayıların gösterilimine ilişkin bilgiler verilecektir.

Kayan noktalı (*floating point*) sayıların gösterilimi Bilgisayar Mimarisi dersinde ele alınmaktadır (Bkz. <http://www.buzluca.info/mimari>).

Sayılar kodlanmadan önce **işaretsiz** ya da **işaretleli** sayılarla çalışılacağı belirlenmelidir. Çünkü işaretsiz ve işaretleli sayıların kodlanmasında farklı yöntemler kullanılmaktadır.

İşaretsiz (*Unsigned*) Sayıların Kodlanması:

Bilgisayarlarda işaretsiz tamsayıların ifade edilmesinde "doğal ağırlıklı ikili kodlama" kullanılır.

Örnek: $215_{10} = (1101\ 0111)_2 = 1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

$215/2 = 107$ kalan **1** (düşük anlamlı bit "*Least Significant Bit – LSB*") son basamak

$107/2 = 53$ kalan **1**

$53/2 = 26$ kalan **1**

$26/2 = 13$ kalan **0**

$13/2 = 6$ kalan **1**

$6/2 = 3$ kalan **0**

$3/2 = 1$ kalan **1**

$1/2 = 0$ kalan **1** (yüksek anlamlı bit "*Most Significant Bit – MSB*") ilk basamak

8 bit ile ifade edilebilecek en büyük işaretsiz sayı:

$1111\ 1111_2 = 255_{10}$

8 bit ile ifade edilebilecek en küçük işaretsiz sayı:

$0000\ 0000_2 = 0_{10}$

İşaretleli (*Signed*) Sayıların Kodlanması:

Pozitif ve negatif sayıları ayırt etmek için ikili sayının ilk basamaktaki en yüksek anlamlı bitine bakılır.

"0" ile başlayan sayıların pozitif,

"1" ile başlayan sayıların negatif olduğu kabul edilir.

Pozitif sayıların kodlanmasında (işaretsiz sayılarda olduğu gibi) "doğal ağırlıklı ikili kodlama" kullanılır. Dikkat edilmesi gereken nokta sayının 0 ile başlamasıdır.

Buna göre 8 bit ile temsil edilebilecek pozitif işaretleli sayılar:

0000 0000 ile 0111 1111 arasında (yani 0 ile +127 arasında) değişecektir.

Negatif sayıların kodlanmasında **2'ye tümeleme** yöntemi kullanılmaktadır.

Bu yöntemde pozitif bir sayının 2'ye tümleyenini hesaplandığında o sayının negatif gösterilimi elde edilmiş olur.

Bir sayının 2'ye tümleyenini elde etmek için

- Önce sayı 1'e tümlenir, yani 0'lar 1, 1'ler 0 yapılır,
- 1'e tümlenmiş sayıya 1 eklenir.

2'ye tümeleme yöntemi aritmetik işlemlerde kolaylık sağladığı için tercih edilmektedir.

Negatif Sayılara Örnekler:

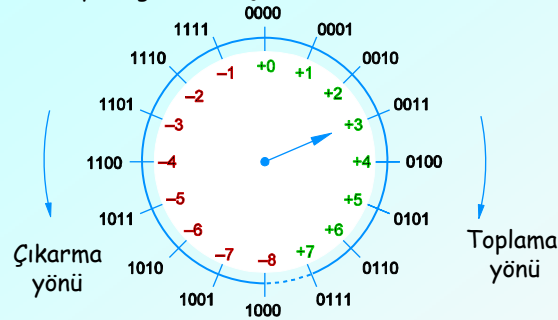
8 bitlik $+5_{10}$: 0000 0101	4 bitlik $+7_{10}$: 0111
1'e tümeleme	: 1111 1010	1'e tümeleme	: 1000
1 ekleme	: $\begin{array}{r} + \\ \hline 1 \end{array}$	1 ekleme	: $\begin{array}{r} + \\ \hline 1 \end{array}$
Sonuç -5_{10}	: 1111 1011	Sonuç -7_{10}	: 1001

2'ye tümeleme işlemi bir sayının **işaretini değiştirmek** için kullanılır.
Bir negatif sayıya 2'ye tümeleme işlemi uygulandığında o sayının pozitif değeri elde edilmiş olur.

Negatif bir sayının pozitif yapılması:

8 bitlik -5_{10}	: 1111 1011
1'e tümeleme	: 0000 0100
1 ekleme	: $\begin{array}{r} + \\ \hline 1 \end{array}$
Sonuç -5_{10}	: 0000 0101

İşaretili sayılar bir grafik üzerinde gösterilebilir.
Aşağıda 4 bitlik sayılar gösterilmiştir.



4 bit ile ifade edilebilecek mutlak değeri en büyük negatif sayı

$$1000 = -8$$

Mutlak değeri en küçük negatif sayı 1111 = -1

8 bit mutlak değeri en büyük negatif sayı 1000 0000 = -128

Mutlak değeri en küçük negatif sayı 1111 1111 = -1

İkili Sayıların Uzatılması

Sayısal istemlerde ikili sayılar için belli uzunlukta yerler (bellek gözleri) ayrılır.

Bazı durumlarda daha az bit ile ifade edilebilen bir sayıyı daha büyük bir yere yazmak ya da daha uzun bir sayı ile işleme sokmak gerekebilir.

Bu durumda kısa olan sayı uzatılır.

Örneğin 4 bittten 8 bite veya 8 bittten 16 bite uzatma.

Uzatma işleminde sayının işaretli ya da işaretli olmasına göre farklı yollar izlenir.

İşaretsiz Sayılar: Sayının başına (yüksek anlamlı kısmına) gerektiği kadar sıfır '0' eklenir.

Örnek: 4 bitlik 3_{10} : 0011 8 bitlik 3_{10} : 0000 0011

Örnek: 4 bitlik 9_{10} : 1001 8 bitlik 9_{10} : 0000 1001

İşaretli Sayılar: Sayının başına (yüksek anlamlı kısmına) sayının işareti gerektiği kadar eklenir. Buna **işaret uzatma** (*sign extension*) denir.

Örnek: 4 bitlik $+3_{10}$: 0011 8 bitlik $+3_{10}$: 0000 0011

Örnek: 4 bitlik -7_{10} : 1001 8 bitlik -7_{10} : 1111 1001

Örnek: 4 bitlik -1_{10} : 1111 8 bitlik -1_{10} : 1111 1111

16 Tabanın Kullanılması

Sayısal devrelerin yapıları 2'li sayıların kullanılmasını zorunlu hale getirmiştir.

Ancak 2'li sayıların yazılması ve okunması uzunlukları nedeniyle zor olmaktadır.

Bu nedenle kağıt üstündeki gösterimlerde kolaylık sağladığı için 16 tabanında (*hexadecima*) sayılar kullanılmaktadır.

2'li - 16'lı Dönüşüm:

- 2'li sayı 4 bitlik gruplar halinde yazılır,
- Her dörtlü için 16'lık karşılığı yazılır.

Örnek:

$01011101_2 = 0101 \ 1101$ (İkili - Binary)
 $= 5 \ D$ (Onaltılı - Hexadecima)

10 tabanına dönüşüm :

$5D_{16} = (5 \times 16) + 13 = 93$

Sonuç: $01011101_2 = 5D_{16} = 93_{10}$

16 tabanındaki sayıları göstermek için genellikle **\$** ve **h** simgeleri kullanılır.

Örnek: \$5D veya 5Dh.

Sayısal Sistemlerde Toplama ve Çıkarma İşlemleri

Bilgisayarlarda tamsayı aritmetik işlemleri Aritmetik/Lojik Birim (ALB) tarafından yapılır (*Arithmetic Logic Unit - ALU*).

Tamsayı toplama ve çıkarma işlemleri işaretli ve işaretli sayılar üzerinde aynı şekilde yapılır.

Ancak çıkan sonucun yorumlanması işaretli ve işaretli sayılarda farklı olmaktadır.

Farklı uzunluklarda sayılar ile işlem yaparken kısa sayının uzatılması gerekir.

Uzatma işlemi işaretli ve işaretli sayılarda farklı olur. (Bkz. 1.13)

Toplama:

İşaretsiz Tamsayılar:

n bitlik iki işaretli sayının toplanması sonucu n+1 bitlik bir sayı oluşabilir. (n+1). bit **elde** (*carry*) adını alır.

Örnek: 8 bitlik işaretli sayıların toplanması

$$\begin{array}{r} 01110101: 117 \\ + 01100011: 99 \\ \hline 11011000: 216 \end{array}$$

Elde oluşmadı.

$$\begin{array}{r} 11111111: 255 \\ + 00000001: 1 \\ \hline 100000000: 256 \end{array}$$

Elde oluştu.

a	b	Elde	Sonuç
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Toplama:

İşaretli Tamsayılar:

İşlem işaretli sayılarda olduğu gibi yapılır. Ancak sonucun yorumlanması farklıdır.

n bitlik işaretli iki sayının toplanması sonucu (n+1). bit oluşursa bu bit göz ardı edilir.

Örnek: 8 bitlik işaretli sayıların toplanması

$$\begin{array}{r} 11111111: -1 \\ + 00000001: +1 \\ \hline 100000000: 0 \end{array}$$

Göz ardı edilir. İşaret (+)

$$\begin{array}{r} 11111111: -1 \\ + 11111111: -1 \\ \hline 111111110: -2 \end{array}$$

Göz ardı edilir. İşaret (-)

Dikkat:

Eğer n bitlik sayılarla çalışılıyorsa işaret her zaman (sağdan sola doğru sayıldığında) n. bittir (n+1. değil).

Taşma (Overflow):

İşaretsiz sayılarda toplama sonucu oluşan değer n bit ile gösterilemeyebilir. Örneğin 8 bit ile gösterilebilecek sayılar -128, +127 arasındadır. Oluşan sonuç bu aralığın dışına çıkıyorsa **taşma** oluşur.

Toplama sonucunda taşma olduğu toplanan sayıların ve sonucun işaretinden anlaşılır.

Toplamada iki durumda taşma oluşabilir:

poz + poz → neg ve neg + neg → poz

Örnek:

01111111: +127	10000000: -128
+ 00000010: +2	+ 11111111: -1
10000001: Gösterilemiyor	10111111: Gösterilemiyor
İki pozitif sayı toplandı.	İki negatif sayı toplandı.
Sonuç negatif çıktı.	Sonuç pozitif çıktı.
Taşma vardır.	Taşma vardır.
Not: n+1. bit oluşmadı.	Not: n+1. bit oluştu.
Bu bit göz ardı edilir.	Bu bit göz ardı edilir.

Çıkarma:

Bilgisayarlar, çıkarma işlemi için çıkartılacak olan sayının işaretini değiştirip (2'ye tümleyip) diğer sayı ile toplarlar.

Böylece tek bir toplama devresi ile hem toplama hem çıkarma yapmak mümkün olur.

İşaretsiz Tamsayılar:

n bitlik iki işaretsiz sayı arasında 2'ye tümleyen yöntemine göre çıkarma yapıldığında n+1. bit oluşursa sonuç geçerlidir, borç (*borrow*) yoktur.

Eğer n+1. bit oluşmazsa (sıfırsa) birinci sayı ikinciden küçüktür, **borç** vardır.

Örnek: 8 bitlik işaretsiz sayılar ile çıkarma

00000101: 5	2'ye tümleme	00000101: 5
- 00000001: 1	→	+ 11111111: -1
		100000100: 4
		Elde oluştu: Borç Yok
00000001: 1	2'ye tümleme	00000001: 1
- 00000101: 5	→	+ 11111011: -5
		11111100: Borç (sonuç negatif)
		Elde oluşmadı: Borç Var

Çıkarma:**İşaretili Tamsayılar:**

İşaretili tamsayılar arasındaki çıkarma da işaretsizlerde olduğu gibi 2'ye tümleyen yöntemine göre yapılır.

Elde biti göz ardı edilir.

Toplama da olduğu gibi işaretili sayıların çıkarılmasında da taşma olabilir.

Çıkarmada iki durumda taşma oluşabilir:

poz - neg → neg ve neg - poz → poz

Örnek: 8 bitlik işaretili sayılar ile çıkarma

$$\begin{array}{r} 00000101: 5 \\ - 00001100: 12 \\ \hline \end{array} \xrightarrow{2'ye\ tümleme} \begin{array}{r} 00000101: 5 \\ + 11110100: -12 \\ \hline 11111001: 2'ye\ tüm.: 00000111: -7 \\ \text{İşaret } 1 \text{ sonuç } \text{negatif} \end{array}$$

$$\begin{array}{r} 11111101: -3 \\ - 01111111: 127 \\ \hline \end{array} \xrightarrow{2'ye\ tümleme} \begin{array}{r} 11111101: -3 \\ + 10000001: -127 \\ \hline 10111110: \\ \text{İşaret biti } 0, \text{ Sonuç } \text{pozitif}. \end{array}$$

Neg - poz = poz. Taşma vardır.

Elde (Carry), Borç (Borrow), Taşma (Overflow) Kavramlarının Özeti

Elde: İşaretsiz sayıların toplanmasında oluşabilir. Sonucun n bite sığmadığını (n+1). bitin gerekli olduğunu gösterir.

Borç: İşaretsiz sayıların çıkartılmasında oluşabilir. Birinci sayının ikinciden küçük olduğunu, sonucun negatif çıktığını gösterir.

2'ye tümleyen yöntemine göre yapılan çıkarmada n+1. bit oluşursa borç yoktur.

Taşma: Sadece işaretili sayılar üzerinde yapılan toplama ve çıkarma işlemlerinde oluşur. Sonucun, ayrılan bit sayısı ile ifade edilemediğini gösterir.

Taşma olduğu, işleme giren sayıların ve sonucun işareti incelenerek anlaşılır.

Aşağıdaki durumlar oluştuğunda taşma var demektir:

$$\begin{array}{ll} \text{poz} + \text{poz} \rightarrow \text{neg} & \text{poz} - \text{neg} \rightarrow \text{neg} \\ \text{neg} + \text{neg} \rightarrow \text{poz} & \text{neg} - \text{poz} \rightarrow \text{poz} \end{array}$$

Boole Cebri

- $B=\{0,1\}$ kümesi üzerinde tanımlı
- İkili işlemler: VEYA, VE $\{ + , \cdot \}$
- Birli işlem: Tümeleme $\{ ' \}$

▪ Aksiyomlar:

1. Kapalılık: $a + b \in B$ $a \cdot b \in B$
2. Değişme: $a + b = b + a$ $a \cdot b = b \cdot a$
3. Birleşme: $a + (b + c) = (a + b) + c$ $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
4. Etkisiz eleman: $a + 0 = a$ $a \cdot 1 = a$
5. Dağılma: $a + (b \cdot c) = (a + b) \cdot (a + c)$ $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
6. Tümeleme: $a + a' = 1$ $a \cdot a' = 0$

		$a \cdot b$	
b	a	0	1
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	1

		$a + b$	
b	a	0	1
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

		Tümeleme	
a	a'	0	1
0	1	0	1
1	0	1	0

Özellikler ve Teoremler:

Burada gösterilen tüm özellikler ve teoremler Boole cebrinin tanımında yer alan işlemler ve aksiyomlar ile kanıtlanabilirler.

1. Yutma: $a + 1 = 1$ $a \cdot 0 = 0$
 2. Dönüşme (*Involution*): $(a')' = a$
 3. Sabit kuvvet (*Idempotency*): $a + a + \dots + a = a$ $a \cdot a \cdot a \cdot \dots \cdot a = a$
 4. Soğurma (*Absorption*): $a + a \cdot b = a$ $a \cdot (a + b) = a$
 5. De Morgan Teoremi: $(a + b + \dots)' = a' \cdot b' \cdot \dots$ $(a \cdot b \cdot \dots)' = a' + b' + \dots$
- Genel De Morgan Teoremi: $f'(X1, X2, \dots, Xn, 0, 1, +, \cdot) = f(X1', X2', \dots, Xn', 1, 0, \cdot, +)$
 - İkili işlemler arasında ilişki sağlar: \cdot ve $+$ arasında

6. D alite

Bir lojik ifadenin d ali, \cdot yerine $+$, $+$ yerine \cdot , 0 yerine 1, 1 yerine 0, koyarak ve deęişkenler deęiştirilmeden elde edilir.

Kanıtlanan her teorem d ali iin de geerlidir.

$$a + b + \dots \Leftrightarrow a \cdot b \cdot \dots$$

Genelleştirilmiş d alite:

$$f(X_1, X_2, \dots, X_n, 0, 1, +, \cdot) \Leftrightarrow f(X_1, X_2, \dots, X_n, 1, 0, \cdot, +)$$

- De Morgan Teoreminden farklıdır
 - Teoremlerin kanıtları arasında ilişki sağlar
 - Lojik ifadelerin d n st r lmesini saęlayan bir y ntem deęildir.

İşlemler Arası  ncelik:

Bir lojik ifade deęerlendirilirken işlemler arasındaki  ncelik y ksekte  ncelikten bařlayarak şöyledir:

1. Parantez,
2. T mleme,
3. VE,
4. VEYA

Teoremlerin Kanıtlanması: a) Aksiyomlar ile

Teorem:	$X \cdot Y + X \cdot Y' = X$
Daęılma	$X \cdot Y + X \cdot Y' = X \cdot (Y + Y')$
T�mleme	$X \cdot (Y + Y') = X \cdot (1)$
Etkisiz	$X \cdot (1) = X \checkmark$
Teorem:	$X + X \cdot Y = X$
Etkisiz	$X + X \cdot Y = X \cdot 1 + X \cdot Y$
Daęılma	$X \cdot 1 + X \cdot Y = X \cdot (1 + Y)$
Yutma	$X \cdot (1 + Y) = X \cdot (1)$
Etkisiz	$X \cdot (1) = X \checkmark$

Teoremlerin Kanıtlanması: b) Doğruluk Tablosu

De Morgan:

$$(X + Y)' = X' \cdot Y'$$

X	Y	X'	Y'	(X+Y)'	X' · Y'
0	0	1	1	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	0	0

$$(X \cdot Y)' = X' + Y'$$

X	Y	X'	Y'	(X · Y)'	X' + Y'
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

Teoremler lojik ifadelerin sadeleştirilmesinde kullanılabilir.

Örnek:

$$\begin{aligned}
 Z &= A'BC + AB'C + ABC' + ABC \\
 &= A'BC + AB'C + ABC' + ABC + ABC \\
 &= A'BC + ABC + AB'C + ABC' + ABC \\
 &= (A' + A)BC + AB'C + ABC' + ABC \\
 &= (1)BC + AB'C + ABC' + ABC \\
 &= BC + AB'C + ABC' + ABC + ABC \\
 &= BC + AB'C + ABC + ABC' + ABC \\
 &= BC + A(B' + B)C + ABC' + ABC \\
 &= BC + A(1)C + ABC' + ABC \\
 &= BC + AC + AB(C' + C) \\
 &= BC + AC + AB(1) \\
 &= BC + AC + AB
 \end{aligned}$$

Lojik İfadeler (Expressions)

Lojik ifade, değişkenlerin, sabitlerin ve işlemlerin kurallara uygun şekilde yazılmış sonlu kombinezonudur.

$X = (x_1, x_2, \dots, x_n)$, Her $x_i \in \{0,1\}$ olmak üzere $E(X)$ şeklinde gösterilir.

E_1 ve E_2 lojik ifade ise, E_1' , E_2' , $E_1 + E_2$, $E_1 \cdot E_2$ gibi tüm kombinezonlar da birer lojik ifadedir.

Lojik İfadelerin Yapıları:

Monoform ifadelerde değişkenlerin sadece kendileri ya da sadece tümleyenleri bulunur.

Biform ifadeler belli bir x değişkenine göre tanımlanırlar. x' e göre biform bir ifadede hem x hem de tümleyeni bulunur.

Çarpım ifadeleri, değişkenlerin sadece lojik çarpımlarından oluşurlar.

Örnek: $ab'cd$ Çarpım (*product*) yerine **monom** sözcüğü de kullanılır.

Toplam ifadeleri, değişkenlerin sadece lojik toplamlarından oluşurlar.

Örnek: $a'+b'+c+d$ Toplam (*sum*) yerine **monal** sözcüğü de kullanılır.

Çarpım böleni, bir çarpımdan (monomdan) bir ya da daha fazla değişken kaldırıldığında elde edilen çarpım ifadesidir.

Örnek: $ab'cd$ nin bazı böleneri: a , b' , c , d , ab' , $b'c$, acd , $b'd$

İfadelerin yazılma şekilleri:

- $\Sigma\Pi$: Lojik çarpımların lojik toplamı ya da monomların "veya"lanması $bc'+ad+a'b$ gibi
- $\Pi\Sigma$: Lojik toplamların lojik çarpımı ya da monalların "ve"lenmesi $(a+b+c')(a+d)(a'+b)$ gibi

Bir lojik ifadenin değeri:

$E(X)$ ifadesi $X=(x_1, \dots, x_n)$

vektörünün her değeri için

$B=\{0,1\}$ kümesinden bir değer üretir.

Bu değerler ifadenin doğruluk tablosunu oluşturur.

Örnek: $E(X) = x_1x_2+x_3$

ifadesinin doğruluk tablosu

x_1	x_2	x_3	$E(X)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Sıra bağıntısı:

B'nin elemanları arasında şu sıra bağıntısı tanımlanır: $0 < 1$
0, 1'den "önce gelir" ya da "küçüktür" diye okunur.

Buna göre X vektörleri arasında da bir sıra bağıntısı tanımlanabilir.

Eğer X1 vektörünün tüm elemanları X2 vektörünün aynı sıradaki elemanlarından yukarıda tanımlandığı anlamda "küçük"se (önce geliyorsa) ya da eşitse $X1 \leq X2$ sıralaması geçerlidir.

Örnek:

$X1=1001$, $X2 = 1101$ ise

$X1 \leq X2$ dir.

İki vektör arasında sıra bağıntısı olmayabilir.

Örneğin, $X1=0011$, $X2 = 1001$ ise

$X1$ ile $X2$ arasında sıra bağıntısı yoktur.

İfadeler üzerinde sıra bağıntısı:

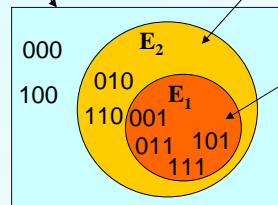
$E_1(X) \leq E_2(X)$ yazılışı, X'in tüm kombinezonları için E_1 'in alacağı değerlerin E_2 'nin alacağı değerlere eşit ya da küçük olduğunu belirtir.

Örnek :

x_1	x_2	x_3	$E_1(X)$	$E_2(X)$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1

Tüm giriş kombinezonları (X) uzayı

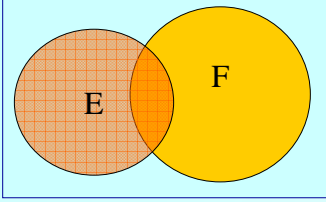
$E_2(X)$ 'nin '1' değeri ürettiği (örttüğü) kombinezonlar



$E_1(X)$ 'nin '1' değeri ürettiği (örttüğü) kombinezonlar

$E_1(X)$ 'in 1 değerini aldığı her giriş kombinezonu için $E_2(X)$ de 1 değerini alır.

$E_1(X) \leq E_2(X)$ ise $E_1(X)$, $E_2(X)$ 'yi gerektirir, $E_1(X) \Rightarrow E_2(X)$, $E_2(X)$, $E_1(X)$ 'i örter.



E ve F lojik ifadeler olmak üzere,
 $E \cdot F \leq E \leq E + F$ ve $E \cdot F \leq F \leq E + F$
 eşitsizlikleri her zaman geçerlidir.

Yutma özellikleri:

$$E + E \cdot F = E$$

ve düali

$$E(E + F) = E$$

$$\text{Kanıt: } E(E + F) = EE + EF = E + EF = E(1 + F) = E$$

$$E + E' \cdot F = E + F$$

ve düali

$$E(E' + F) = E \cdot F$$

$$\text{Kanıt: } E + E' \cdot F = (E + E')(E + F) = 1(E + F) = E + F$$

Biform kareler:

E_1 ve E_2 içinde x_1 olmayan iki ifade olsun: $E_1(x_2, \dots, x_n)$ ve $E_2(x_2, \dots, x_m)$

$E = x_1 E_1 + x_1' E_2$ ve düali $E^D = (x_1 + E_1^D)(x_1' + E_2^D)$
 ifadeleri x_1 in biform kareleridir.

Örnek: $x_1(x_2 + x_3') + x_1'(x_3 + x_4)$, $(x_1 + x_2 + x_3')(x_1' + x_3 + x_4)$ ve
 $(x_1 + x_2 x_3')(x_1' + x_3 x_4)$ x_1 ' in biform karelerine dair örneklerdir.

Konsensüs:

- Çarpımların toplamı şeklinde yazılmış olan $x E_1 + x' E_2$ biform karesinde $E_1 E_2$ kesişimine **konsensüs** adı verilir.
- Toplamların çarpımı şeklinde yazılmış olan $(x + E_1)(x' + E_2)$ biform karesinde $E_1 + E_2$ toplamı **konsensüstür**.

Teorem: Biform kareler konsensüslerini yutarlar.

$$x E_1 + x' E_2 + E_1 E_2 = x E_1 + x' E_2$$

$$(x + E_1)(x' + E_2)(E_1 + E_2) = (x + E_1)(x' + E_2)$$

Teorem: Biform kareler arasında dönüşme özelliği vardır.

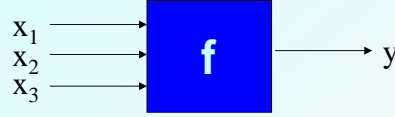
$$x E_1 + x' E_2 = (x + E_2)(x' + E_1)$$

Lojik Fonksiyonlar

Lojik fonksiyonlar B^n kümesi (n elemanlı 2'li kodların kümesi) üzerinde tanımlanırlar ve üçe ayrılırlar:

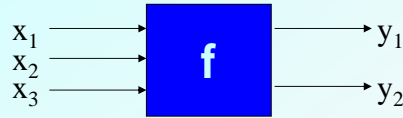
Yalın fonksiyonlar: Çok girişli bir çıkışlı

x_1	x_2	x_3	y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



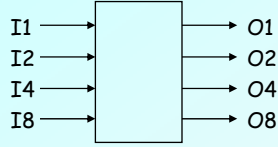
Genel fonksiyonlar: Çok girişli, çok çıkışlı

x_1	x_2	x_3	y_1	y_2
0	0	0	1	1
0	0	1	1	0
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0



Tümüyle tanımlanmamış fonksiyonlar: Bazı giriş kombinezonları için fonksiyonun alacağı değer belirsizdir.

Örnek BCD sayıları 1 arttıran fonksiyon:



I8	I4	I2	I1	O8	O4	O2	O1
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	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Bu girişler için devrenin (fonksiyonun) çıkışlarının alacağı değer belirsizdir. Belirsiz değerleri göstermek için X yerine Φ sembolü de kullanılır.

Yalın Lojik Fonksiyonlar:

$$\forall X^0 \in B^n ; \exists! y^0 \in B ; y = f(X)$$

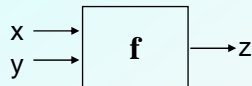
B^n kümesinden değer alan X^0 kombinezonuna f fonksiyonu uygulandığında B kümesinden değer alan bir y^0 değeri elde edilir ve bu değer tektir.

Yalın lojik fonksiyonlar B^n kümesinin kombinezonlarını 2 sınıfa ayırırlar:

- Doğru kombinezonlar sınıfı. $f(X)$ in 1 değeri karşı düşürdüğü kombinezonlardan oluşur.
- Yanlış kombinezonlar sınıfı. $f(X)$ in 0 değeri karşı düşürdüğü kombinezonlardan oluşur.

n girişli 2^{2^n} adet yalın lojik fonksiyon vardır.

İki girişli 16 adet yalın lojik fonksiyon vardır:



2 girişli 16 adet yalın lojik fonksiyon (F0–F15)

X	Y	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

0 X Y $X \text{ YA DA } Y$ $X = Y$ Y' X' 1
 $X \text{ VE } Y$ $X \text{ VEYA } Y$ $X \text{ TVEYA } Y$ $(X \text{ VEYA } Y)'$ $X \text{ TVE } Y$ $(X \text{ VE } Y)'$

Lojik Fonksiyonların Gösterilişi

Aynı lojik fonksiyon farklı yöntemler ile gösterilebilir. Bu fonksiyona ilişkin devre tasarlanırken bu gösterilimlerden uygun olanı kullanılır.

Doğruluk Tablosu İle Gösterilim

Tüm giriş kombinezonları için çıkışın (veya çıkışların) alacağı değerler tablo halinde yazılır.

Sayısal Gösterilim

Giriş kombinezonları 2'li sayılarla kodlandığına göre her kombinezona 10 tabanında bir numara verilir. Fonksiyon hangi giriş kombinezonları için lojik "1" değeri (ya da lojik "0") üretiyorsa o kombinezonların numaraları listelenir.

Örnek: Tümüyle tanımlanmış, yalın bir fonksiyonun gösterilimi:

No	x_1	x_2	y
0	0	0	1
1	0	1	0
2	1	0	1
3	1	1	0

$y=f(x_1,x_2)=U_1(0,2)$

Aynı fonksiyon lojik 0 üreten kombinezonlar ile de gösterilebilir.

$y=f(x_1,x_2)=U_0(1,3)$

Örnek: Tümüyle tanımlanmış, genel bir fonksiyonun gösterilimi:

Her çıkış için yukarıdaki gösterilim uygulanır.

No	x_1	x_2	y_1	y_2
0	0	0	1	1
1	0	1	0	1
2	1	0	1	0
3	1	1	0	0

$y_1=f(x_1,x_2)=U_1(0,2)$

$y_2=f(x_1,x_2)=U_1(0,1)$

Aynı fonksiyon lojik 0 üreten kombinezonlar ile de gösterilebilir.

$y_1=f(x_1,x_2)=U_0(1,3)$

$y_2=f(x_1,x_2)=U_0(2,3)$

Örnek: Tümüyle tanımlanmamış, genel bir fonksiyonun gösterilimi:
Bu durumda sadece lojik "1" veya lojik "0" üreten çıkışları göstermek yeterli değildir.

No	x_1	x_2	y_1	y_2
0	0	0	1	1
1	0	1	0	Φ
2	1	0	Φ	0
3	1	1	0	Φ

$y_1 = f(x_1,x_2) = U_1(0) + U_0(1,3)$

veya $y_1 = f(x_1,x_2) = U_1(0) + U_\Phi(2)$

veya $y_1 = f(x_1,x_2) = U_0(1,3) + U_\Phi(2)$

$y_2 = f(x_1,x_2) = U_1(0) + U_0(2)$

veya $y_2 = f(x_1,x_2) = U_1(0) + U_\Phi(1,3)$

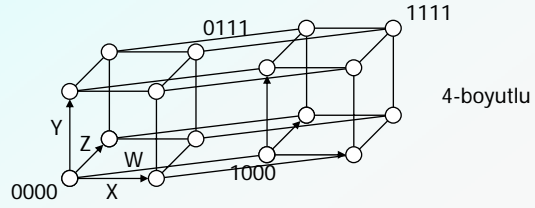
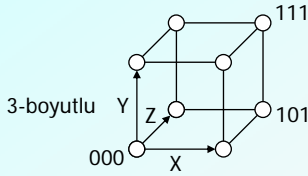
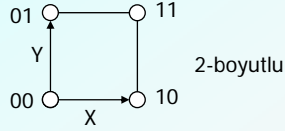
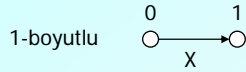
veya $y_2 = f(x_1,x_2) = U_0(2) + U_\Phi(1,3)$

Grafik Gösterim

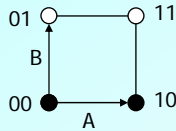
Giriş kombinasyonları B^n kümesinin elemanları olduklarına göre n boyutlu uzaydaki bir hiperküpün köşelerini oluştururlar.

Fonksiyonun doğru noktalarını (lojik 1) üreten kombinasyonlar küp üzerinde işaretlenir. Fonksiyonun giriş sayısı küpün boyutunu belirler.

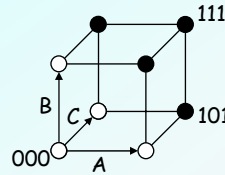
n giriş \rightarrow n boyutlu küp

Boole Küpleri:**Örnek:**

A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

**Örnek:**

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

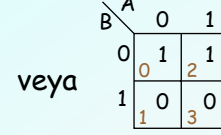
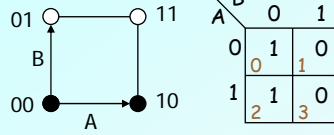


Giriş sayısı arttıkça çizimin zorlaşması nedeniyle, Boole küpleri lojik fonksiyonların gösterilmesi için pratikte kullanılan bir yöntem değildir. Grafik gösterim lojik fonksiyonların anlaşılması ve bundan sonraki konuların anlatılması açısından yararlıdır.

Karnaugh Diyagramları (Karnaugh Map)

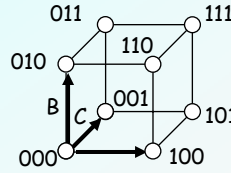
Boole küplerinin düzlem üzerindeki iz düşümleri olarak düşünülebilir.

No	A	B	F
0	0	0	1
1	0	1	0
2	1	0	1
3	1	1	0



Tabloların gözleri Gray koduna göre düzenlenir. Yan yana gözlerle ait kombinezonların bitişik olması sağlanır.

A	BC	00	01	11	10
0		0	1	3	2
1		4	5	7	6



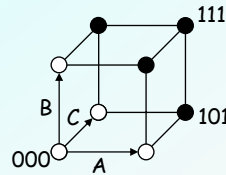
A	BC	000	001	011	010

4 girişli bir fonksiyona ilişkin Karnaugh diyagramı:

AB	CD	00	01	11	10
00		0	1	3	2
01		4	5	7	6
11		12	13	15	14
10		8	9	11	10

Örnek:

No	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



A	BC	00	01	11	10
0		0	0	3	2
1		4	5	7	6

Cebirsel Gösterilim

Çarpımların toplamı ($\Sigma\Pi$) şeklindeki cebirsel gösterilime doğruluk tablosundan **1. kanonik açılım** ile geçilir.

Toplamların çarpımı ($\Pi\Sigma$) şeklindeki cebirsel gösterilime doğruluk tablosundan **2. kanonik açılım** ile geçilir.

1. Kanonik Açılım: Çarpımların Toplamları

$\Sigma\Pi$ şeklindeki gösterim fonksiyonun "doğru" noktalarına ilişkin çarpımların (monomların) toplamından oluşur. Bu çarpımlara **minterim** denir.

Bir minterimde fonksiyonun tüm giriş değişkenleri bulunur ve her minterim sadece bir doğru kombinezonla karşı gelir.

"Doğru" değer üreten kombinezonlar: $F = 001 \quad 011 \quad 101 \quad 110 \quad 111$
Minterimlerin Toplamı: $F = A'B'C + A'BC + AB'C + ABC' + ABC$

A	B	C	F	F'
0	0	0	0	1
0	0	1	1	0
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	0

Fonksiyonun tümleyeni de benzer şekilde "yanlış" noktalardan hareket edilerek yazılır:

$$F' = A'B'C' + A'BC' + AB'C'$$

Minterimlerde her değişkenin ya kendisi ya da tümleyeni yer alır (İkisi aynı anda olamaz).

Kanonik açılım fonksiyonun en basit cebirsel ifadesi değildir.

Çoğunlukla kanonik açılımlar yalınlaştırılabilir (basitleştirilebilir).

A	B	C	minterimler
0	0	0	$A'B'C'$ m0
0	0	1	$A'B'C$ m1
0	1	0	$A'BC'$ m2
0	1	1	$A'BC$ m3
1	0	0	$AB'C'$ m4
1	0	1	$AB'C$ m5
1	1	0	ABC' m6
1	1	1	ABC m7

F nin Kanonik açılımı:

$$F(A, B, C) = \Sigma m(1,3,5,6,7)$$

$$= m1 + m3 + m5 + m6 + m7$$

$$= A'B'C + A'BC + AB'C + ABC' + ABC$$

$$F = \Sigma_{A,B,C} (1,3,5,6,7) \text{ şeklinde de yazılabilir.}$$

Kanonik açılımın basitleştirilmesi

$$F(A, B, C) = A'B'C + A'BC + AB'C + ABC + ABC'$$

$$= (A'B' + A'B + AB' + AB)C + ABC'$$

$$= ((A' + A)(B' + B))C + ABC'$$

$$= C + ABC'$$

$$= ABC' + C$$

$$= AB + C$$

3 değişkenli minterimlerin simgesel gösterilimi

2. Kanonik Açılım: Toplamların Çarpımı

$\Pi\Sigma$ şeklindeki gösterilim fonksiyonun "yanlış" noktalarına ilişkin **maksterim**lerin çarpımından oluşur. Maksterimler tüm değişkenleri içeren toplamlardır (monallardır).

Maksterimler oluşturulurken, giriş kombinasyonunda 0 değerine karşı düşen değişkenlerin kendisi, 1 değeri karşı düşenlerin tümleyeni alınır (Minterim oluşturmanın tersi).

"Yanlış" değer üreten kombinasyonlar: $F =$ $\begin{matrix} 000 & 010 & 100 \\ \text{Maksterimlerin Çarpımı: } F = (A + B + C) & (A + B' + C) & (A' + B + C) \end{matrix}$

A	B	C	F	F'
0	0	0	0	1
0	0	1	1	0
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	0

Fonksiyonun tümleyeninin 2.kanonik açılımı benzer şekilde "doğru" noktalardan hareket edilerek yazılır:

$$F' = (A + B + C') (A + B' + C') (A' + B + C') (A' + B' + C') (A' + B' + C')$$

Maksterimlerde her değişkenin ya kendisi ya da tümleyeni yer alır (İkisi aynı anda olamaz).

Kanonik açılım fonksiyonun en basit cebirsel ifadesi değildir. Çoğunlukla kanonik açılımlar indirgenebilir (basitleştirilebilir).

A	B	C	maksterimler
0	0	0	$A+B+C$ M0
0	0	1	$A+B+C'$ M1
0	1	0	$A+B'+C$ M2
0	1	1	$A+B'+C'$ M3
1	0	0	$A'+B+C$ M4
1	0	1	$A'+B+C'$ M5
1	1	0	$A'+B'+C$ M6
1	1	1	$A'+B'+C'$ M7

3 değişkenli maksterimlerin simgesel gösterilimi

F nin kanonik açılımı:

$$\begin{aligned} F(A, B, C) &= \prod M(0,2,4) \\ &= M0 \cdot M2 \cdot M4 \\ &= (A + B + C) (A + B' + C) (A' + B + C) \end{aligned}$$

$F = \prod_{A,B,C}(0,2,4)$ şeklinde de yazılabilir.

İndirgeme

$$\begin{aligned} F(A, B, C) &= (A + B + C) (A + B' + C) (A' + B + C) \\ &= (A + B + C) (A + B' + C) \\ &= (A + B + C) (A' + B + C) \\ &= (A + C) (B + C) \end{aligned}$$

Kanonik Açılımların Dönüştürülmesi

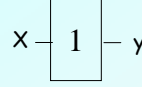
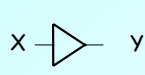
- **Minterim'den maksterime dönüşüm**
 - 1. kanonik açılımda yer almayan minterimlerin indisleri maksterim olarak seçilir
 - $F(A,B,C) = \Sigma m(1,3,5,6,7) = \Pi M(0,2,4)$
- **Maksterim'den minterime dönüşüm**
 - 2. kanonik açılımda yer almayan maksterimlerin indisleri minterim olarak seçilir
 - $F(A,B,C) = \Pi M(0,2,4) = \Sigma m(1,3,5,6,7)$
- **Minterimlerle tümleyen ifadenin bulunması**
 - Açılımda yer almayan minterimler seçilir
 - $F(A,B,C) = \Sigma m(1,3,5,6,7) \quad F'(A,B,C) = \Sigma m(0,2,4)$
- **Maksterimlerle tümleyen ifadenin bulunması**
 - Açılımda yer almayan maksterimler seçilir
 - $F(A,B,C) = \Pi M(0,2,4) \quad F'(A,B,C) = \Pi M(1,3,5,6,7)$

Kanonik Açılımlar ve De Morgan Teoremi

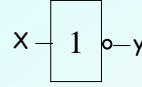
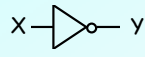
- **Çarpımların Toplamı (Fonksiyonun tümleyeni)**
 - $F' = A'B'C' + A'BC' + AB'C'$
- **De Morgan**
 - $(F')' = (A'B'C' + A'BC' + AB'C')$
 - $F = (A + B + C)(A + B' + C)(A' + B + C) \quad 2. \text{ kanonik açılım elde edildi}$
- **Toplamların Çarpımı (Fonksiyonun tümleyeni)**
 - $F' = (A + B + C')(A + B' + C')(A' + B + C')(A' + B' + C)(A' + B' + C')$
- **De Morgan**
 - $(F')' = ((A + B + C')(A + B' + C')(A' + B + C')(A' + B' + C)(A' + B' + C'))'$
 - $F = A'B'C + A'BC + AB'C + ABC' + ABC \quad 1. \text{ kanonik açılım elde edildi}$

Lojik Bağlaçlar (Logic Gates)

ANSI/IEEE-1973 ANSI/IEEE-1984

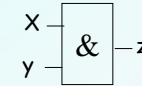
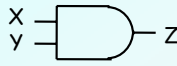
SÜRÜCÜ (BUFFER) $Y=X$ 

X	Y
0	0
1	1

TÜMLEME (NOT) X' 

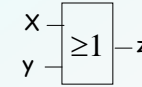
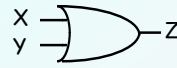
X	Y
0	1
1	0

VE (AND)

 $X \cdot Y$ 

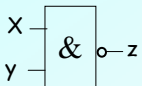
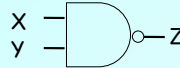
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

VEYA (OR)

 $X + Y$ 

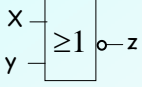
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

TVE (NAND)

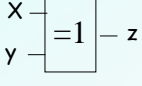
 $(xy)'$ 

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

TVEYA (NOR)

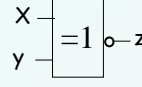
 $(x+y)'$ 

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

YA DA (XOR)
 $xy' + x'y$ $X \oplus Y$ 

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

EŞDEĞER (XNOR)

 $X \odot Y$ 

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

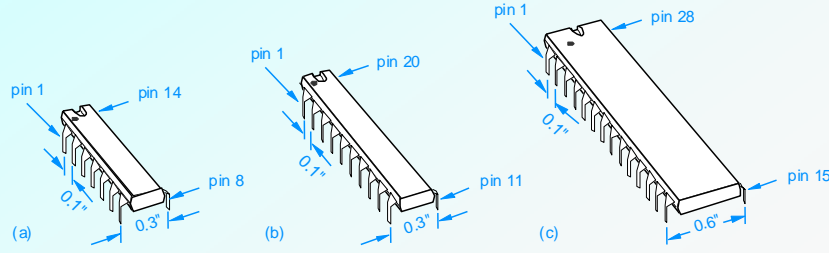
Tümdevreler (*Integrated Circuits - IC*)

Lojik bağlaçlar, tümdevrelerin içinde yer alacak şekilde üretilir ve pazarlanırlar. Bir tümdevrede, büyüklüğüne ve bağlacın giriş sayısına bağlı olarak birden fazla lojik bağlaç yer alır.

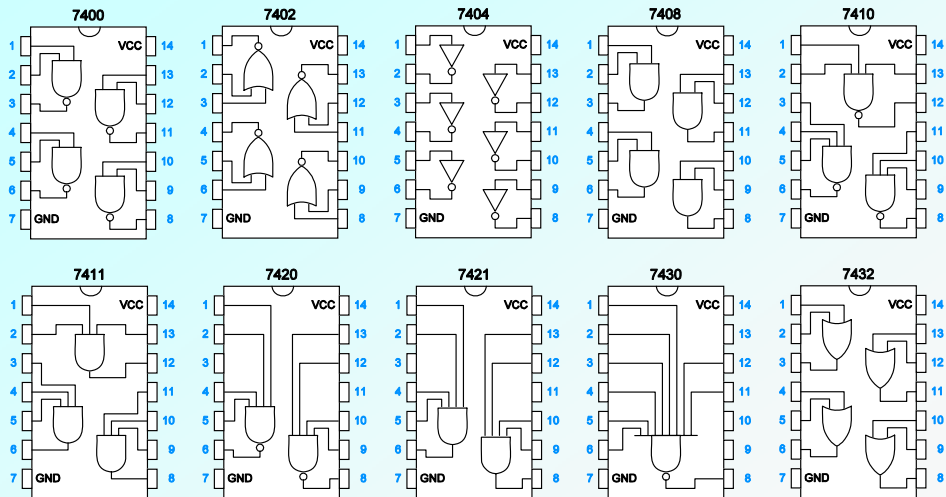
Tümdevreler, farklı şekillerde üretilirler. Laboratuvar ortamında en çok karşılaşılabileceğiniz tip, dikdörtgen şeklinde olan, bacakları iki sıra halinde kenarlarda yer alan tümdevrelerdir.

Bu tümdevreler "dual in-line pin (DIP)" olarak adlandırılırlar.

Dual in-line Pin (DIP) Tümdevreler



74xx Serisi Tümdevrelere Örnekler



Tümdevreler ile ilgili bilgiler tümdevre kataloglarında yer alırlar.

Pozitif ve Negatif Lojik

Sıfır ve 1 değerini alan girişler ve çıkışlar, genel olarak, fiziksel bir büyüklüğün 2 farklı seviyesine karşı düşer: Gerilim, akım, basınç v.b. Yüksek seviyeye 1, alçak seviyeye 0 karşı düşürülüyorsa buna **pozitif lojik**, aksi halde **negatif lojik** denir.

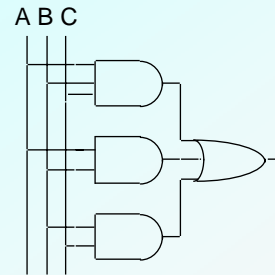
L (*Low*) düşük seviye, H (*High*) yüksek seviye olmak üzere, 2 girişli 1 çıkışlı bir kapının giriş-çıkış ilişkisi aşağıda gösterilmiştir. Pozitif lojik kullanıldığı takdirde fiziksel devre bir VE kapısı, negatif lojik kullanıldığı takdirde de bir VEYA kapısı gerçekleştirmektedir. Bir lojik devrenin tümünde ya pozitif ya da negatif lojik kullanılır.

Fiziksel Devre			Pozitif Lojik			Negatif Lojik		
Girişler:		Çıkış:	Girişler:		Çıkış:	Girişler:		Çıkış:
x1	x2	z	x1	x2	z	x1	x2	z
L	L	L	0	0	0	1	1	1
L	H	L	0	1	0	1	0	1
H	L	L	1	0	0	0	1	1
H	H	H	1	1	1	0	0	0

Lojik Fonksiyonların Bağlaçlar İle Gerçeklenmesi

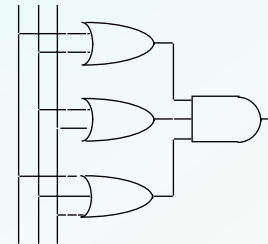
- Çarpımların Toplamı

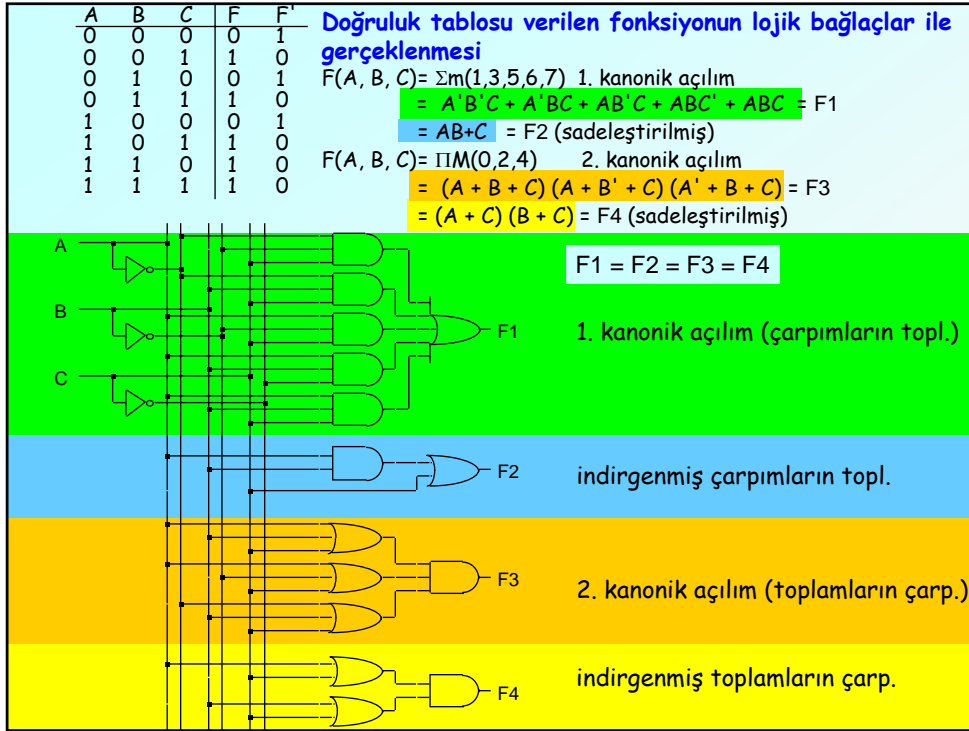
- VE (AND) kapıları çarpımları gerçekleştirir
- VEYA (OR) kapısı toplamayı gerçekleştirir



- Toplamların Çarpımı

- VEYA (OR) kapıları toplamaları gerçekleştirir
- VE (AND) kapısı çarpımı gerçekleştirir





Sayısal Devreler (Lojik Devreleri)

Bir lojik ifade farklı şekillerde lojik bağlaçlar kullanılarak gerçekleştirilebilir.

Örnek: $Z = A' \cdot B' \cdot (C + D) = (A' \cdot (B' \cdot (C + D)))$

3 girişli kapı
 Sadece 2 girişli kapılar

Elinizde var olan fiziksel kapılara göre lojik ifadeyi düzenlemek gerekir.

<http://www.buzluca.info/sayisal> ©2000-2006 Yrd.Doç.Dr. Feza BUZLUCA 2.8

Yetkin İşlemler

VE, VEYA, TÜMLEME işlemleri ile tüm lojik fonksiyonları gerçeklemek mümkündür (Boole cebirinin tanımından). Bu nedenle bu işlemler **yetkin bir işlem kümesi** oluştururlar.

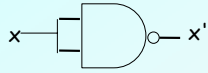
Bu işlemlerin dışında TVE (NAND) işlemi de tek başına yetkin bir işlemdir. Benzer şekilde TVEYA (NOR) da yetkin bir işlemdir.

VE, VEYA, TÜMLEME işlemlerinin her birini sadece TVE veya TVEYA kapıları kullanarak gerçekleştirmek mümkündür.

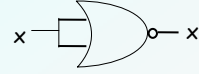
↓ simgesi TVE işlemini, / simgesi ise TVEYA'yı göstermek için kullanılmıştır.

Buna göre aşağıdaki eşitlikler yazılabilir.

$$\begin{aligned} x' &= x \downarrow x \\ &= (x \cdot x)' \\ &= x' \end{aligned}$$



$$x' = x/x$$



$$x \cdot y = (x \downarrow y)'$$

$$x \cdot y = (x' / y')$$
 de Morgan

$$x + y = (x' \downarrow y')$$
 de Morgan

$$x + y = (x / y)'$$

TVE - TVEYA Arasındaki İlişki

- TVE - TVEYA Dönüşümleri

- de Morgan:

$$(A + B)' = A' \cdot B'$$

$$(A \cdot B)' = A' + B'$$

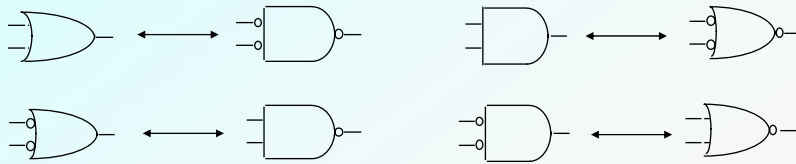
- diğer bir yazım şekli:

$$A + B = (A' \cdot B')'$$

$$(A \cdot B) = (A' + B)'$$

- Buna göre:

- Girişleri tümlenmiş TVE kapısı, VEYA kapısının eşdeğeridir.
- Girişleri tümlenmiş TVEYA kapısı, VE kapısının eşdeğeridir.
- Girişleri tümlenmiş VEYA kapısı, TVE kapısının eşdeğeridir.
- Girişleri tümlenmiş VE kapısı, TVEYA kapısının eşdeğeridir.



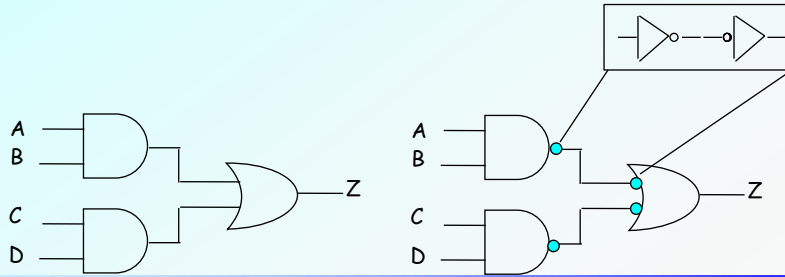
Lojik fonksiyonların TVE veya TVEYA bağlaçları ile gerçekleştirilmesi

TVE yetkin bir işlem olduğundan tüm lojik fonksiyonlar sadece TVE bağlaçları kullanılarak gerçekleştirilebilir. Aynı durum TVEYA bağlaçları için de geçerlidir.

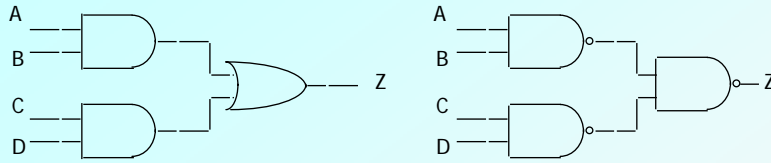
Çarpımların toplamı (VElerin VEYAsı) şeklindeki fonksiyonların TVE ile gerçekleştirilmesi:

Bu tür devrelerde tüm VE kapıları ve VEYA kapılarının yerine TVE kapıları yerleştirilebilir. Bu değişiklik devrenin çıkış fonksiyonunu etkilemez.

Aşağıda gösterildiği gibi VE kapılarının çıkışları, VEYA kapılarının da girişlerine tümlene elemanı yerleştirilirse TVE kapıları elde edilir. Bir hatta peş peşe iki tümlene elemanı yerleştirilmesi herhangi bir değişikliğe neden olmaz.

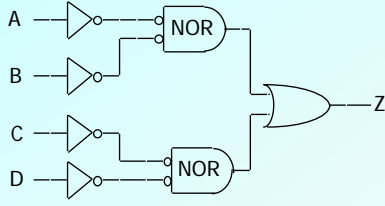
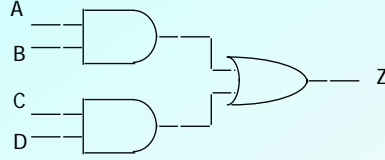


Cebirsel olarak sınıma:

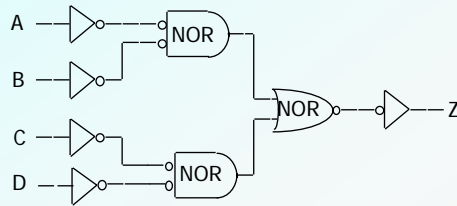


$$\begin{aligned}
 Z &= [(A \cdot B)' \cdot (C \cdot D)']' \\
 &= [(A' + B') \cdot (C' + D')]' \\
 &= [(A' + B')' + (C' + D')'] \\
 &= (A \cdot B) + (C \cdot D) \checkmark
 \end{aligned}$$

VE lerin VEYA lanması şeklinde devreler sadece TVEYA kullanılarak da gerçekleştirilebilir. Bu durumda girişlere ve çıkışa tümeleme elemanları yerleştirmek gerekir.



1. Adım



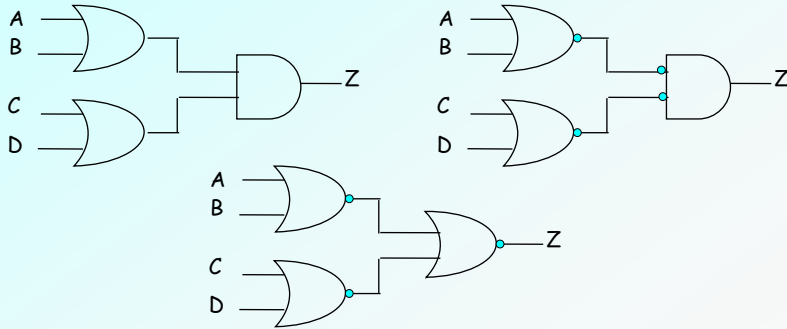
2. Adım

Hatırlatma: Tümeleme bağlaçları TVE bağlaçları ile gerçekleştirilebilir.

Toplamların çarpımı (VEYA'ların VE'si) şeklindeki fonksiyonların TVEYA ile gerçekleştirilmesi:

Bu tür devrelerde tüm VEYA kapıları ve VE kapılarının yerine TVEYA kapıları yerleştirilebilir. Bu değişiklik devrenin çıkış fonksiyonunu etkilemez.

Aşağıda gösterildiği gibi VEYA kapılarının çıkışlarına, VE kapılarının da girişlerine tümeleme elemanı yerleştirilirse TVEYA kapıları elde edilir. Bir hatta peş peşe iki tümeleme elemanı yerleştirilmesi herhangi bir değişikliğe neden olmaz.



Lojik Fonksiyonların Yalınlaştırılması (İndirgenmesi)

Bir lojik fonksiyonun birçok cebirsel ifadesi vardır. Yalınlaştırmada amaç, belli bir maliyet kriterine göre bu cebirsel ifadeler içinden en uygun olanını seçmektir.

Maliyet kriteri uygulamaya göre değişebilir. Örneğin tasarım aşamasında istenen özellikler şunlar olabilir: İfadenin az sayıda çarpım (ya da toplam) içermesi, her çarpımda az sayıda değişken olması, devrenin aynı tip bağlaçlar (örneğin TVE) ile tasarlanabilmesi, elde var olan bağlaçların kullanılabilmesi gibi.

Yalınlaştırma İle İlgili Tanımlar

Asal Çarpım (Temel İçeren) "Prime Implicant":

Bir fonksiyonun 1. kanonik açılımını oluşturan çarpımlar (minterimler) bu fonksiyon tarafından örtülürler (içerilirler).

Buradaki her çarpım sadece bir "doğru" noktaya karşı gelir. Bu çarpımlardan bazılarının bölenleri de o fonksiyon tarafından örtülürler.

Buna göre 1. kanonik açılımda yer alan bazı çarpımlar birleştirilerek daha az değişken içeren ve birden fazla "doğru" noktaya karşı gelen yeni çarpımlar elde edilebilir.

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

$F(A, B, C) = \Sigma m(1,3,5,6,7)$ 1. kanonik açılım
 $= A'B'C + A'BC + AB'C + ABC' + ABC$

Bu çarpımlar, asal çarpım (temel içeren) değildir, çünkü onlardan daha az değişkene sahip olan bölenleri de bu fonksiyonun içinde yer almaktadır. Bu durum basitleştirme sonucu görülmüştü ve fonksiyon için aşağıdaki ifade elde edilmişti.

$F = AB + C$

Buna göre **asal çarpım (temel içeren)** kendi bölenleri fonksiyonda yer almayan çarpımlardır.

Örneğin yukarıdaki örnekte ABC' bir asal çarpım değildir, çünkü onun böleni olan AB de fonksiyon tarafından örtülmektedir.

AB monomu ise bir asal çarpımdır, çünkü onun bölenleri A ve B fonksiyon tarafından örtülmez (daha fazla 1 üretiliyorlar, fonksiyonun ifadesinde yer alamazlar).

Yalınlaştırma işlemi 2 aşamadan oluşmaktadır:

1. Tüm asal çarpımlar kümesinin (Tüm temel içerenlerin) bulunması
2. Fonksiyonun tüm "doğru" noktalarını örtecek şekilde, asal çarpımlardan en uygun olanların seçilmesi.

Asal Çarpımların Bulunması:

Çarpım terimlerini (monomları) birleştirmek için Boole cebirinden yararlanılır. Bu işlemi özellikle büyük fonksiyonlar için elle kağıt üstünde yapmak zor olur. Bu işlemler bilgisayar programları ile yapılır.

Konuyu anlayabilmek için kağıt üzerinde uygulanabilecek bir yöntem ise doğruluk tablosunda "1" üreten kombinezonları inceleyerek, bir veya daha fazla değişkenin sabit kaldığı kombinezonları birleştirmektir. Değeri sabit kalan değişkenler çarpımda kalır, değişkenler çarpımdan çıkarılır.

Örnek:

$$F = A'B' + AB' = (A'+A)B' = B'$$

A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

B sabit. Her ikisinde de B=0.
B değişkeni yeni çarpımda yer alacak.

A'nın değeri değişiyor.
A yeni çarpımda olmayacak.

B=0 olduğu için yeni çarpım: B'

Yapılan işlemin Boole küpünde gösterilmesi:

A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

Boyutu 0 olan iki nokta birleştirilerek boyutu 1 olan bir çizgi elde edildi. Bu çizgi B=0'ı yani B'nin tümleyenini temsil etmektedir.

Yapılan işlemin Karnaugh diyagramında gösterilmesi:

		B	0	1
A	0	0	1	0
	1	1	1	0

Bu tür gruplamaları Karnaugh diyagramları ile yapmak daha kolaydır. Bitişiklik özelliğinden yararlanılarak komşu noktalar gruplanabilir. Yukarıda gruplamanın yapıldığı sütunda B=0 (sabit), A ise değişkendir. Bu sütun B'nin tümleyenini temsil etmektedir.

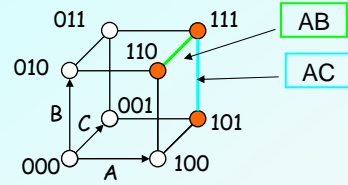
- Aynı anda birden fazla değişken sabit kalıyorsa gruplama sonucu bu değişkenlerin çarpımı oluşur.

Örnek:

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

A=1, C=1 ve sabit. B ise değişiyor. Bu gruplama sonucu AC çarpımı oluşur

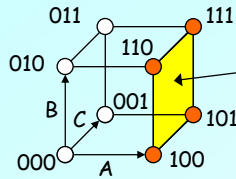
A=1, B=1 ve sabit. C ise değişiyor. Bu gruplama sonucu AB çarpımı oluşur



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

- Gruplamalarda 2'den daha fazla nokta da birleştirilebilir.

Örnek: $F(A,B,C) = \Sigma(4,5,6,7)$



A=1 ve sabit. B ve C ise değişiyor. Küpün bu yüzü A'yı temsil ediyor.

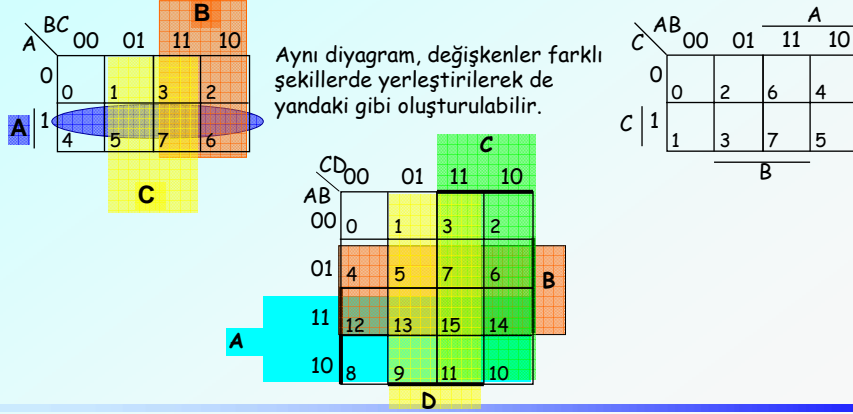
A	BC		B	
	00	01	11	10
0	0	1	0	2
1	4	5	7	6

Asal Çarpımların Karnaugh Diyagramları İle Bulunması:

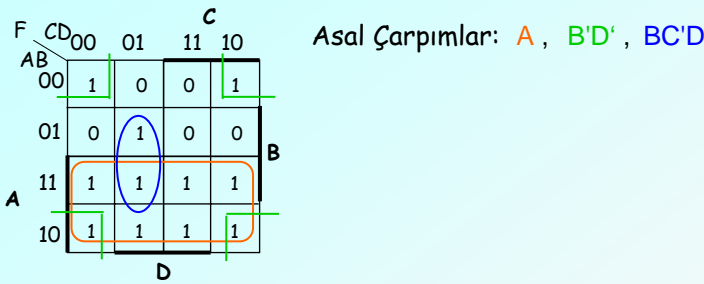
Karnaugh diyagramlarındaki bitişiklik ve çevrimsellik özelliği nedeniyle komşu gözler arasındaki geçişlerde sadece 1 değişken (giriş) değer değiştirir, diğerleri sabit kalır.

Girişlerin sabit kaldığı komşu gözlerdeki "doğru" noktaları 2'li, 4'lü, 8'li ... gruplarda toplamak mümkündür.

Aşağıda 3 ve 4 değişkenli Karnaugh diyagramları için girişlerin sabit kaldıkları alanlar gösterilmiştir



Örnek: Aşağıda verilen fonksiyonun asal çarpımlarının bulunması
 $F(A,B,C,D) = \Sigma(0,2,5,8,9,10,11,12,13,14,15)$



Asal çarpımlar bulunurken fonksiyonun "doğru" noktaları mümkün olan en büyük gruplara yerleştirilirler.

Bir grupta yer alan "doğru" nokta daha küçük bir gruba yerleştirilmez.

Örneğin ayrı ayrı 4'lü gruplarda bulunan iki nokta birleştirilerek 2'li yeni bir grup oluşturmaya gerek yoktur.

Ancak noktalardan biri daha büyük bir gruba ait değilse (yukarıdaki 0101 gibi) o nokta gruptaki başka bir nokta ile kümelenebilir.

Tüm Asal Çarpımlar Kümesinin (Temel İçeren Tabanının) Bulunması:

Lojik devre tasarımında yalınlaştırma işlemi o fonksiyonun bütün asal çarpımlarının bulunmasıyla başlar. Bütün asal çarpımların oluşturduğu kümeye **tüm asal çarpımlar kümesi (tüm temel içeren tabanı)** denir. İndirgemenin 2. aşamasında fonksiyonun bütün doğru noktalarını örtecek şekilde, tüm asal çarpımlar kümesinden en uygun asal çarpımlar seçilir.

Fonksiyonun bütün doğru noktalarını örten asal çarpımların oluşturduğu kümeye **yeterli taban** denir. Yeterli tabandan bir asal çarpım kaldırılırsa fonksiyonun tüm doğru noktaları örtülmemiş olur.

Buna göre bir fonksiyonu yalınlaştırma işlemi en uygun (ucuz) yeterli tabanı bulmak demektir.

Örnek: Aşağıdaki fonksiyonun tüm asal çarpımlar kümesini bulunuz.

Asal Çarpımlar:
BC' , A'B , A'C , AB' , B'C , AC'

		B			
		00	01	11	10
A	0		1	1	1
	1	1	1		1
		C			

Aynı fonksiyonun bir çok yeterli tabanı olabilir.

		B			
		00	01	11	10
A	0		1	1	1
	1	1	1		1
		C			

$F(A,B,C) = A'B + B'C + AC'$

		B			
		00	01	11	10
A	0		1	1	1
	1	1	1		1
		C			

$F(A,B,C) = A'B + BC' + B'C + AB'$

		B			
		00	01	11	10
A	0		1	1	1
	1	1	1		1
		C			

$F(A,B,C) = BC' + A'C + AB'$

		B			
		00	01	11	10
A	0		1	1	1
	1	1	1		1
		C			

$F(A,B,C) = BC' + A'C + B'C + AC'$

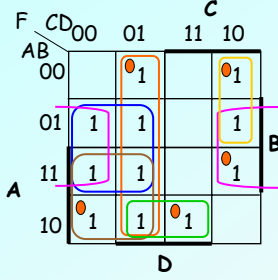
Yeterli tabandan bir asal çarpım kaldırıldığında tüm doğru noktalar kapsanmamış olur.

Başlıca Nokta ve Gerekli Asal çarpım:

Bazı fonksiyonlarda bazı doğru noktalar sadece bir asal çarpım tarafından örtülürler. Bu noktalara **başlıca nokta** denir. Bu noktaları örten asal çarpımlara da **gerekli asal çarpım** denir.

Gerekli asal çarpımlar fonksiyonun yeterli tabanında mutlaka yer alırlar. Çünkü başlıca noktaların başka asal çarpımlar tarafından örtülmesi mümkün değildir.

Örnek:

**Tüm Asal Çarpımlar Kümesi**

$C'D$, BC' , AC' , BD' , $A'CD'$, $AB'D$

Başlıca Noktalar

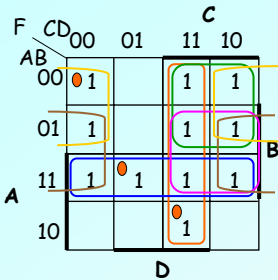
0001
0010
1000
1110
1011

Gerekli çarpımlar

$C'D$
 $A'CD'$
 AC'
 BD'
 $AB'D$

Buradaki gerekli asal çarpımlar fonksiyonun tüm doğru noktalarını örtmektedir. $F = C'D + A'CD' + AC' + BD' + AB'D$

Örnek: Bir fonksiyonun tüm asal çarpımlar kümesinin, başlıca noktalarının ve gerekli çarpımların bulunması.

**Tüm Asal Çarpımlar Kümesi:**

CD , AB , $A'C$, BC , $A'D'$, BD'

Başlıca Noktalar

0000
1101
1011

Gerekli çarpımlar

$A'D'$
 AB
 CD

Yalınlaştırma: Uygun Asal Çarpımların Seçilmesi

Tüm asal çarpımlar kümesi bulunduğundan sonra, fonksiyonun tüm doğru noktalarını örtecek şekilde en uygun (ucuz) asal çarpımların seçilmesi gerekir. Bu seçimi yapmak için kullanılan yöntemlerden biri seçenekler tablosu yöntemidir.

Seçenekler Tablosu:

- Fonksiyonun asal çarpımları bulunduğundan sonra bu çarpımlara isimler verilir. Örneğin A, B, C, .. gibi.
- Verilen bir maliyet kriterine göre her asal çarpımın maliyeti hesaplanır.

Seçenekler tablosu bir matris şeklinde hazırlanır.

- Tablonun satırlarında, fonksiyonun asal çarpımlarının isimleri yer alır. Sütunlarda ise o fonksiyonun doğru noktalarının numaraları bulunur.
- En son sütuna asal çarpımların maliyetleri yazılır.
- Bir asal çarpım bir noktayı örtüyorsa matrisin ilgili gözüne X konur.

Örnek: Verilen fonksiyonun tüm asal çarpımlar kümesini bulunuz ve seçenekler tablosunu oluşturunuz.

$$f(x_1, x_2, x_3, x_4) = \sum m(2, 4, 6, 8, 9, 10, 12, 13, 15)$$

Maliyet hesabında her değişken 2 birim, her tümlleme işlemi 1 birim maliyete sahip olacaktır.

f		x_3, x_4			
		00	01	11	10
x_1, x_2	00				1
	01	1			1
x_1	11	1	1	1	
	10	1	1		1
		x_4			

Tüm Asal Çarpımlar Kümesi:

$x_1 x_3'$	$x_2 x_3' x_4'$	$x_1' x_2 x_4'$	$x_1 x_2 x_4$	$x_1' x_3 x_4'$	$x_2' x_3 x_4'$	$x_1 x_2' x_4'$
------------	-----------------	-----------------	---------------	-----------------	-----------------	-----------------

Semboller: A B C D E F G

Maliyetler: 5 8 8 6 8 8 8

Örttüğü Noktalar: 8,9,12,13 4,12 4, 6 13, 15 2, 6 2, 10 8, 10

Tüm Asal Çarpımlar Kümesi:

$x_1 x_3'$	$x_2 x_3' x_4'$	$x_1' x_2 x_4'$	$x_1 x_2 x_4$	$x_1' x_3 x_4'$	$x_2' x_3 x_4'$	$x_1 x_2' x_4'$
------------	-----------------	-----------------	---------------	-----------------	-----------------	-----------------

Semboller:	A	B	C	D	E	F	G
Maliyetler:	5	8	8	6	8	8	8
Örttüğü Noktalar:	8,9,12,13	4,12	4, 6	13, 15	2, 6	2, 10	8, 10

Fonksiyonun "doğru" noktaları

	2	4	6	8	9	10	12	13	15	Maliyet
A				X	X		X	X		5
B		X					X			8
C		X	X							8
D							X	X		6
E	X		X							8
F	X				X					8
G				X	X					8

Seçenekler Tablosunun İndirgenmesi

1. Başlıca noktalar belirlenir. Bir sütunda sadece bir tane X varsa o sütundaki nokta başlıca noktadır.

Başlıca noktayı örten asal çarpım (gerekli asal çarpım) mutlaka fonksiyonun ifadesinde yer alacağından seçilir. Bu asal çarpıma ait satır ve onun örttüğü noktalara ait sütunlar tablodan kaldırılır.

2. Tabloda j. satırın X olan her gözünde i. satırda da X varsa i. satır, j. satırı örtüyor denir. Yani j. satırın örttüğü bütün noktaları i. satır da örtüyordur.

Eğer i. satır j. satırı örtüyorsa ve i. satırdaki maliyet j. satırdaki maliyetten küçükse veya ona eşitse j. satır (örtülen satır) tablodan kaldırılır.

3. Bir sütun başka bir sütunu örtüyorsa örten sütun (daha fazla X'e sahip olan) tablodan silinir.

Bu kurallar peş peşe uygulanarak fonksiyonun doğru noktaları toplam maliyet en az olacak şekilde örtülmeye çalışılır.

Örnek: Aşağıda verilen fonksiyona ait seçenekler tablosunu indirgenmesi.
 $f(x_1, x_2, x_3, x_4) = \sum m(2, 4, 6, 8, 9, 10, 12, 13, 15)$

Fonksiyonun "doğru" noktaları

	2	4	6	8	9	10	12	13	15	Maliyet
$\checkmark x_1 x_3$				x	x		x	x		5
$x_2 x_3 x_4$		x					x			8
$x_1 x_2 x_4$		x	x							8
$\checkmark x_1 x_2 x_4$								x	x	6
$x_1 x_3 x_4$	x		x							8
$x_2 x_3 x_4$	x					x				8
$x_1 x_2 x_4$				x		x				8

1. Adım: Bu tabloda 9 ve 15 başlıca noktalardır. A ve D gerekli çarpımlar oldukları için onlara ait satır ve örttükleri sütunlar tablodan kaldırılır. Bu çarpımlar daha sonra sonucu oluştururken kullanılmak üzere işaretlenir.

	2	4	6	10	Maliyet
$x_2 x_3 x_4$		x			8
$x_1 x_2 x_4$		x	x		8
$x_1 x_3 x_4$	x		x		8
$x_2 x_3 x_4$	x			x	8
$x_1 x_2 x_4$				x	8

2. Adım: Bu tabloda C, B'yi örter. Maliyetleri aynı olduğu için örtülen satır(B) tablodan silinir. Benzer şekilde F, G'yi örter ve maliyetleri aynıdır. Bu nedenle G satırı tablodan silinir. Bu çarpımlar sonuç ifadeye yer almayacaktır.

	2	4	6	10	Maliyet
$\checkmark x_1 x_2 x_4$		x		x	8
$x_1 x_3 x_4$	x		x		8
$\checkmark x_2 x_3 x_4$	x			x	8

3. Adım: Bu tabloda 4 ve 10 başlıca noktalardır. Bu nedenle C ve F çarpımlarını almak gerekir. Bu iki asal çarpım seçildiğinde tüm noktalar örtülmüş olur.

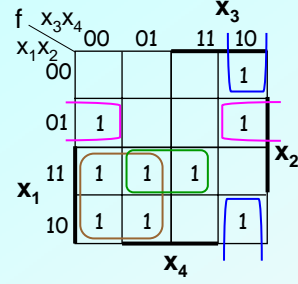
Sonuç: İşaretlenmiş olan asal çarpımlar fonksiyonun en ucuz ifadesini oluştururlar.

Seçilen asal çarpımlar: $A + D + C + F$

Toplam Maliyet= $5 + 6 + 8 + 8 = 27$

$$f(x_1, x_2, x_3, x_4) = x_1 x_3' + x_1 x_2 x_4 + x_1' x_2 x_4' + x_2' x_3 x_4'$$

Karnaugh diyagramı ile hangi asal çarpımların seçildiğini görebiliriz.



Bu seçimde tüm 1'ler örtülmeli ve bir fazlalık olmamalı.

Seçilmiş olan asal çarpımlar bir yeterli taban oluşturmalı. Yani çarpımlardan biri kaldırıldığında tüm noktalar örtülememeli.

$$x_1 x_3'$$

$$x_1' x_2 x_4'$$

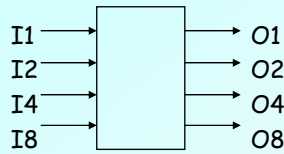
$$x_1 x_2 x_4$$

$$x_2' x_3 x_4'$$

Tümüyle Tanımlanmamış Fonksiyonların Yalınlaştırılması

Hatırlatma: Tümüyle tanımlanmamış fonksiyonlarda, bazı giriş kombinezonları için fonksiyonun alacağı değer belirsizdir (önemli değildir). Çünkü bu giriş kombinezonları ilgili devrede fiziksel olarak oluşamazlar ya da tasarımcı tarafından yasaklanmışlardır.

Örnek BCD sayıları 1 arttıran devre:



	I8	I4	I2	I1	O8	O4	O2	O1
	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	1	0	1	0	0
	0	1	0	0	0	1	0	1
	0	1	0	1	0	1	1	0
	0	1	1	0	0	1	1	1
	0	1	1	1	1	0	0	0
	1	0	0	0	1	0	0	1
	1	0	0	1	0	0	0	0
	1	0	1	0	X	X	X	X
	1	0	1	1	X	X	X	X
	1	1	0	0	X	X	X	X
	1	1	0	1	X	X	X	X
	1	1	1	0	X	X	X	X
	1	1	1	1	X	X	X	X

Bu girişler için devrenin (fonksiyonun) çıkışlarının alacağı değer belirsizdir.

Belirsiz değerleri göstermek için

X yerine Φ sembolü de kullanılır.

Yalınlaştırma işleminde, belirsiz değerler (Φ) en ucuz ifadeyi elde edecek şekilde gerektiğinde lojik 0 gerektiğinde lojik 1 olarak seçilebilirler.

Tüm asal çarpımlar kümesi bulunurken daha basit çarpımlar elde etmek için (Karnaugh diyagramında daha büyük gruplamalar yapabilmek için) Φ ler 1 olarak seçilir.

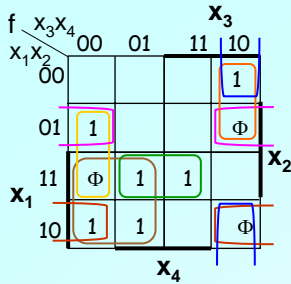
• Seçenekler tablosunda kapsanması gereken noktalar yazılırken Φ ler 0 olarak seçilir. Çünkü bu noktaların çarpımlar tarafından örtülmesine gerek yoktur.

Örnek: Aşağıda verilen tümüyle tanımlanmamış fonksiyonu en düşük maliyetle tasarlayınız.

$$f(x_1, x_2, x_3, x_4) = \sum_m(2, 4, 8, 9, 13, 15) + \sum_\Phi(6, 10, 12)$$

(Not: $f(x_1, x_2, x_3, x_4) = \cup_1(2, 4, 8, 9, 13, 15) + \cup_\Phi(6, 10, 12)$ şeklinde de yazılabilir.)

Maliyet hesabında her değişken 2 birim, her tümlleme işlemi 1 birim maliyete sahip olacaktır.



Asal çarpımlar bulunurken Φ 'ler 1 olarak seçilir.

Tüm Asal Çarpımlar Kümesi:

$x_1 x_3'$	$x_2 x_3' x_4'$	$x_1' x_2 x_4'$	$x_1 x_2 x_4$	$x_1' x_3 x_4'$	$x_2' x_3 x_4'$	$x_1 x_2' x_4'$
------------	-----------------	-----------------	---------------	-----------------	-----------------	-----------------

Semboller:	A	B	C	D	E	F	G
Maliyetler:	5	8	8	6	8	8	8
Örttüğü Noktalar:	8,9,13	4	4	13,15	2	2	8

Tüm Asal Çarpımlar Kümesi:

$x_1 x_3'$	$x_2 x_3' x_4'$	$x_1' x_2 x_4'$	$x_1 x_2 x_4$	$x_1' x_3 x_4'$	$x_2' x_3 x_4'$	$x_1 x_2' x_4'$
------------	-----------------	-----------------	---------------	-----------------	-----------------	-----------------

Semboller:	A	B	C	D	E	F	G
Maliyetler:	5	8	8	6	8	8	8
Örttüğü Noktalar:	8,9,13	4	4	13,15	2	2	8

Fonksiyonun "doğru" noktaları

	2	4	8	9	13	15	Maliyet
A			X	X	X		5
B		X					8
C		X					8
D					X	X	6
E	X						8
F	X						8
G			X				8

Tablo oluşturulurken Φ 'ler 0 olarak seçilir. Bu noktaların örtülmesine gerek olmadığından Φ 'ler seçenekler tablosunda yer almazlar.

Fonksiyonun "doğru" noktaları

	2	4	8	9	13	15	Maliyet
✓	A		X	X	X		5
	B		X				8
	C		X				8
✓	D				X	X	6
	E	X					8
	F	X					8
	G		X				8

1. Adım: Bu tabloda 9 ve 15 başlıca noktalarıdır. A ve D gerekli çarpımlar oldukları için onlara ait satır ve örttüğü sütunlar tablodan kaldırılır. Bu çarpımlar daha sonra sonucu oluştururken kullanılmak üzere işaretlenir.

	2	4	Maliyet
$x_2 x_3' x_4'$	B	x	8
$x_1' x_2 x_4'$	C	x	8
$x_1' x_3 x_4'$	E	x	8
$x_2' x_3 x_4'$	F	x	8

2. Adım: B ve C aynı noktaları örtmektedir ve maliyetleri eşittir. Bu nedenle bu iki çarpım arasında bir seçim yapmak mümkün değildir. Verilen maliyet kriterine göre herhangi biri seçilebilir.

Aynı durum E ve F çarpımları için de geçerlidir.

Buna göre fonksiyon aşağıdaki ifadelerden herhangi biri kullanılarak gerçekleştirilebilir:

$$f = A + D + B + E = x_1 x_3' + x_1 x_2 x_4 + x_2 x_3' x_4' + x_1' x_3 x_4'$$

$$f = A + D + B + F = x_1 x_3' + x_1 x_2 x_4 + x_2 x_3' x_4' + x_2' x_3 x_4'$$

$$f = A + D + C + E = x_1 x_3' + x_1 x_2 x_4 + x_1' x_2 x_4' + x_1' x_3 x_4'$$

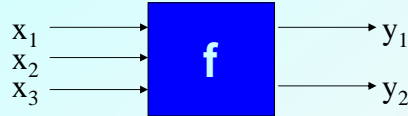
$$f = A + D + C + F = x_1 x_3' + x_1 x_2 x_4 + x_1' x_2 x_4' + x_2' x_3 x_4'$$

Tüm tasarımların maliyeti eşittir (27).

Genel Fonksiyonların Yalınlaştırılması

Hatırlatma: Genel fonksiyonların birden fazla çıkışı vardır.

$x_1 x_2 x_3$	y_1	y_2
0 0 0	1	1
0 0 1	1	Φ
0 1 0	0	0
0 1 1	Φ	0
1 0 0	1	Φ
1 0 1	0	1
1 1 0	0	1
1 1 1	Φ	0



$$y_1 = f_1(x_1, x_2, x_3)$$

$$y_2 = f_2(x_1, x_2, x_3)$$

Genel fonksiyonlar yalınlaştırılırken her çıkışa ait fonksiyon için ayrı ayrı tüm asal çarpımlar kümesi bulunur ve bunların içinden seçim yapılır. Burada dikkat edilmesi gereken nokta her iki çıkış için ortak çarpımların kullanılmaya çalışılmasıdır.

Tüm Asal Çarpımlar Kümesinin Tablo Yöntemiyle (Quine-McCluskey) Bulunması

Karnaugh diyagramları görsel özellikleri nedeniyle az değişkenli fonksiyonlarla ilgili çalışmalarda kolaylık sağlarlar. Ancak değişken sayısı 5 ve daha fazla olduğunda Karnaugh diyagramlarını çizmek ve bitişiklik özelliğini kullanmak zorlaşır.

Tablo yöntemi (Quine-McCluskey) ise sistematik bazı işlemlerin peş peşe tekrarlanmasından oluşmaktadır. Bu işlemleri elle yapmak fazla zaman alabilir, ancak söz konusu işlemleri bilgisayar programı ile gerçekleştirmek kolaydır.

Tablo Yöntemi:

Hatırlanacağı gibi, asal çarpımları bulmak için "1" değeri üreten ve bitişik olan giriş kombinezonları (minterimler) gruplanmaya çalışılıyordu. Sadece bir değişkenin değiştiği (bitişik) olan kombinezonlar aynı gruba alınıyordu.

Tablo yönteminde "1" değeri olan her kombinezon (minterim) diğer minterimler ile karşılaştırılır.

Eğer iki kombinezon arasında sadece bir giriş (değişken) farklıysa o iki kombinezon gruplanır.

Farklı olan değişken silinerek yeni terim elde edilir.

Bu durum hiç gruplama yapılamayana kadar devam eder.

Hiç bir gruba girmeyen terimler asal çarpımlardır.

Yöntem:

- Karşılaştırma kolaylığı sağlamak için içindeki 1'lerin sayısına göre kombinezonları kümeleyin.
- Komşu kümlerdeki kombinezonları karşılaştırın. Tek girişin farklı olduğu kombinezonları gruplayıp yeni kombinezonlar oluşturun.
- Yeni kombinezonlarda değeri değişen giriş yer almayacaktır.
- Bir gruba girmiş olan kombinezonları işaretleyin.
- Yeni oluşan kombinezonlar üzerinde de aynı gruplama işlemlerini yeni gruplar oluşmayıncaya kadar sürdürün.
- Hiç bir gruba girmemiş olan kombinezonlar (işaretsizler) tüm asal çarpımlar kümesini oluştururlar.

Quine-McCluskey yöntemi sadece tüm asal çarpımlar kümesini (tüm temel içeren tabanını) bulmamızı sağlar. Yalınlaştırma işlemi için yine seçenekler tablosunu kullanmamız gerekecektir.

Örnek:

Aşağıda verilen fonksiyonun tüm asal çarpımlar kümesini Quine-McCluskey yöntemiyle bulunuz.

$$f(x_1, x_2, x_3, x_4) = \sum_m(0, 1, 2, 8, 10, 11, 14, 15)$$

K.No	x_1	x_2	x_3	x_4	✓	K.No	x_1	x_2	x_3	x_4	✓	K.No	x_1	x_2	x_3	x_4
0	0	0	0	0	✓	0,1	0	0	0	-	✓	0,2,8,10	-	0	-	0
1	0	0	0	1	✓	0,2	0	0	-	0	✓	0,8,2,10	-	0	-	0
2	0	0	1	0	✓	0,8	-	0	0	0	✓	10,11,14,15	1	-	1	-
8	1	0	0	0	✓	2,10	-	0	1	0	✓	10,11,14,15	1	-	1	-
10	1	0	1	0	✓	8,10	1	0	-	0	✓	10,11,14,15	1	-	1	-
11	1	0	1	1	✓	10,11	1	0	1	-	✓					
14	1	1	1	0	✓	10,14	1	-	1	0	✓					
15	1	1	1	1	✓	11,15	1	-	1	1	✓					
						14,15	1	1	1	-	✓					

Aynı olanları yazmaya gerek yok

Tüm asal çarpımlar kümesi (İşaretsiz olanlar): $x_1' x_2' x_3'$, $x_2' x_4'$, $x_1 x_3$

En ucuz çözümü elde etmek için bu aşamadan sonra seçenekler tablosu oluşturulur ve en ucuz yeterli taban bulunur.

Tümleştirilmiş Kombinezonal Devre Elemanları

Sayısal sistemlerin gerçekleştirilmesinde çokça kullanılan lojik devreler, klasik bağlaçların bir araya getirilmesiyle tümleştirilmiş devre olarak üretilirler ve satılırlar. Bağlaçlar yerine bu devrelerin kullanılması tasarımları kolaylaştırır. Tümdevreler içerdikleri kapı sayısına göre çeşitli gruplara ayrılırlar.

Tümleştirme düzeylerine göre gruplama:

- **Küçük Ölçekli Tümleştirme (Small-Scale Integration SSI):** Bu gruptaki tümdevreler 10 taneden az lojik kapı içerirler. Örneğin 7400 4 adet TVE kapısı içerir.
- **Orta Ölçekli Tümleştirme (Medium-Scale Integration MSI):** Bu gruptaki tümdevreler 10 ile 1000 tane arasında lojik kapı içerirler. Toplayıcı, veri seçici, kod çözücü elemanlar bu gruba girer.
- **Büyük Ölçekli Tümleştirme (Large-Scale Integration LSI):** Bu gruptaki tümdevreler binler mertebesinde lojik kapı içerirler. Mikroişlemciler, bellekler bu grupta yer alırlar.
- **Çok Büyük Ölçekli Tümleştirme (Veri Large-Scale Integration VLSI):** Bu gruptaki tümdevreler yüzbinlerce ve daha fazla sayıda lojik kapı içerirler. Örnek: Gelişmiş mikroişlemciler ve büyük bellek tümdevreleri.

Yarım Toplayıcı (Half Adder):

İki adet birer bitlik sayıyı toplayan bir devredir.



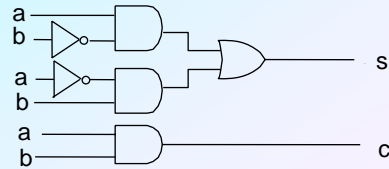
a: Birinci Sayı
b: İkinci Sayı
s: Sonuç
c: Elde Çıkışı

a	b	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Doğruluk tablosundan devrenin ifadesi elde edilir.

$$s = ab' + a'b$$

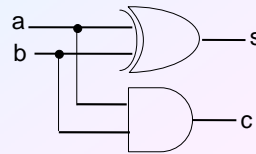
$$c = ab$$



Bu devre yanda gösterildiği gibi YA DA (DARVEYA) bağlacı kullanılarak da gerçekleştirilebilir.

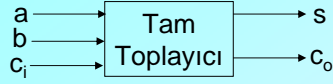
$$s = a \oplus b$$

$$c = ab$$



Tam Toplayıcı (Full Adder)::

İki adet birer bitlik sayıyı eldeli olarak toplayan devredir.



a: Birinci Sayı
 b: İkinci Sayı
 c_i: Elde Girişi
 s: Sonuç
 c_o: Elde Çıkışı

a	b	c _i	c _o	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

s	a	bc _i		b	
		00	01	11	10
0	0	0	1	0	1
1	1	1	0	1	0

c_i

c _o	a	bc _i		b	
		00	01	11	10
0	0	0	0	1	0
1	1	0	1	1	1

c_i

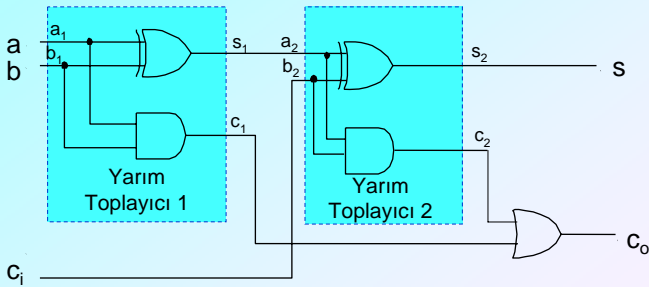
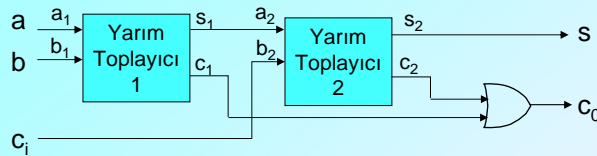
$$s = a'b'c_i + a'bc_i + ab'c_i + abc_i$$

$$s = a \oplus (b \oplus c_i)$$

$$s = a \oplus b \oplus c_i$$

$$c_o = ac_i + bc_i + ab$$

İki adet yarım toplayıcı ve bir adet VEYA kapısı kullanarak bir tam toplayıcı gerçekleştirilebilir:



$$s = (a \oplus b) \oplus c_i$$

$$s = a \oplus b \oplus c_i$$

$$c_o = (a \oplus b)c_i + ab$$

$$c_o = (ab' + a'b)c_i + ab$$

$$c_o = ab'c_i + a'bc_i + ab$$

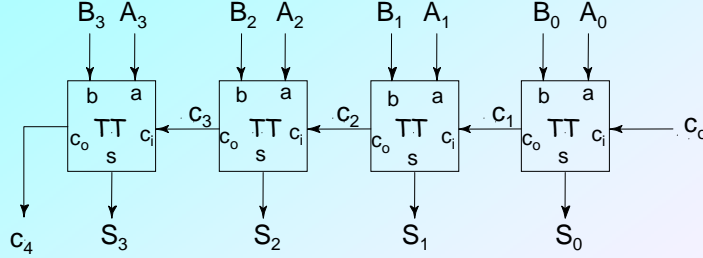
$$c_o = ac_i + bc_i + ab$$

n-Bitlik İkili Paralel Toplayıcı:

İki adet n bitlik 2'li sayıyı toplayan devredir.

Toplanmak istenen sayıların basamak sayısına bağlı olarak bir bitlik tam toplayıcılar peş peşe bağlanarak ikili paralel toplayıcılar gerçekleştirilebilir.

Aşağıda 4 bitlik bir ikili toplayıcı gösterilmiştir.



1. Sayı: $A_3A_2A_1A_0$
 2. Sayı: $B_3B_2B_1B_0$
 Sonuç: $S_3S_2S_1S_0$
 Elde Girişi: c_0
 Elde Çıkışı: c_4

Örnek:

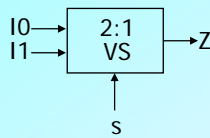
1. Sayı: 0110
 2. Sayı: 1100
 Sonuç: 0010
 Elde : 1

7483 tümdevresi 4 bitlik bir ikili toplayıcıdır. Bu tümdevre MSI tipindedir.

Veri Seçiciler (Multiplexer):

- 2^n adet veri girişi, n adet seçme (denetim) girişi, 1 adet çıkışı vardır.
- Seçme girişlerine gelen değere göre, veri girişlerinden birindeki değer çıkışa aktarılır. Seçme girişlerindeki n bitlik ikili sayı hangi veri girişinin seçileceğini belirler.
- Veri seçiciler giriş sayılarına göre m:1 olarak adlandırılır. Burada m veri girişlerinin sayısını gösterir.

Örnek: 2:1 Veri seçici ("İkiye bir veri seçici" olarak okunur)



İşlev Tablosu:

s	Z
0	I_0
1	I_1

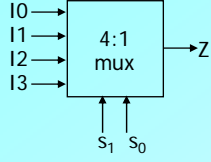
Doğruluk Tablosu:

I_1	I_0	s	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Lojik ifade:

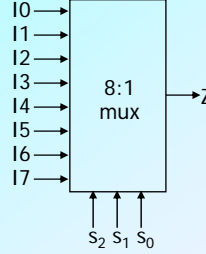
$$Z = s' I_0 + s I_1$$

Diğer Veri Seçici (MUX) Örnekleri:



İşlev Tablosu:

$s_1 s_0$	Z
0 0	I_0
0 1	I_1
1 0	I_2
1 1	I_3



İşlev Tablosu:

$s_2 s_1 s_0$	Z
0 0 0	I_0
0 0 1	I_1
0 1 0	I_2
0 1 1	I_3
1 0 0	I_4
1 0 1	I_5
1 1 0	I_6
1 1 1	I_7

Lojik İfadeler:

$$2:1 \text{ mux: } Z = s' I_0 + s I_1$$

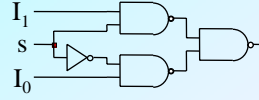
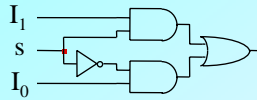
$$4:1 \text{ mux: } Z = s_1' s_0' I_0 + s_1' s_0 I_1 + s_1 s_0' I_2 + s_1 s_0 I_3$$

$$8:1 \text{ mux: } Z = s_2' s_1' s_0' I_0 + s_2' s_1' s_0 I_1 + s_2' s_1 s_0' I_2 + s_2' s_1 s_0 I_3 + s_2 s_1' s_0' I_4 + s_2 s_1' s_0 I_5 + s_2 s_1 s_0' I_6 + s_2 s_1 s_0 I_7$$

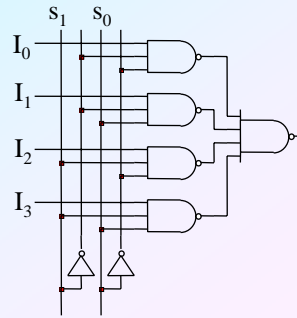
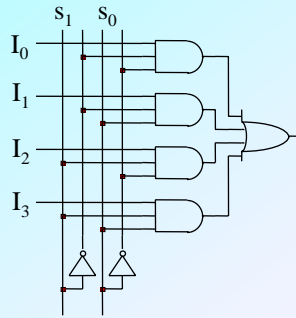
$$\text{Genel İfade (k:1 Mux): } Z = \sum_{j=0}^{k-1} (m_j I_j) \quad k=2^n, m_j = j. \text{ minterim}$$

Veri Seçiciler lojik bağlaçlar kullanılarak aşağıdaki gibi gerçekleştirilebilirler.

2:1 mux

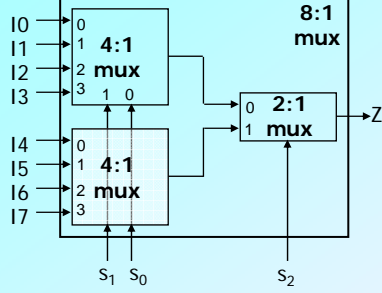


4:1 mux



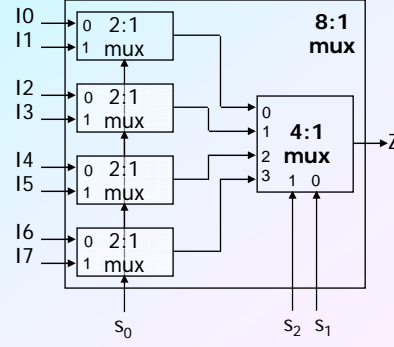
Büyük boyutlardaki veri seçiciler, daha küçüklerin uygun şekilde bağlanmasıyla gerçekleştirilebilir.
Aşağıda 8:1 veri seçicinin 2 farklı şekilde gerçekleştirilmesi gösterilmiştir.

1. Yöntem

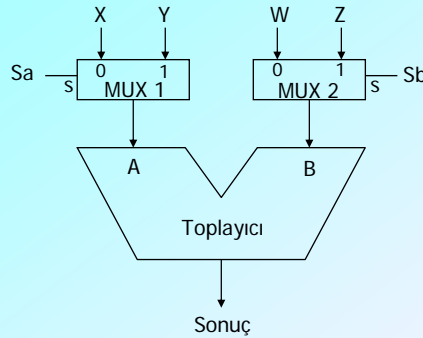


Burada s_0 ve s_1 seçme girişleri 4:1 veri seçicileri için ortaktır. İki veri seçicinin de aynı girişi seçilir. Hangi veri seçicinin çıkışının seçileceğini ise s_2 belirler.

2. Yöntem

**Veri seçicilerin kullanımına bir örnek:**

Bir toplayıcının girişine isteğe bağlı olarak farklı sayılar uygulanabilir.



Sa	Sb	Sonuç
0	0	X+W
0	1	X+Z
1	0	Y+W
1	1	Y+Z

Veri Seçiciler ile Genel Amaçlı Lojik Devre Tasarımı 1:

$2^n:1$ boyutlu bir adet veri seçici kullanılarak n girişli herhangi bir lojik devre **başka bir bağlaç kullanmadan** gerçekleştirilebilir.

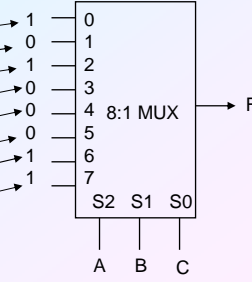
Yöntem:

- Tasarlanacak olan fonksiyonun değişkenleri (devrenin girişleri) veri seçicinin seçme uçlarına bağlanır.
- Her seçme değeri bir giriş kombinasyonuna karşı düştüğüne göre, tasarlanmak istenen fonksiyonun doğruluk tablosuna göre veri seçicinin veri girişlerine lojik "0" veya "1" sabitleri bağlanır.

Örnek:

$$F(A,B,C) = m_0 + m_2 + m_6 + m_7 = \Sigma_1(0,2,6,7)$$

No.	A	B	C	F
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	1

**Veri Seçiciler ile Genel Amaçlı Lojik Devre Tasarımı 2:**

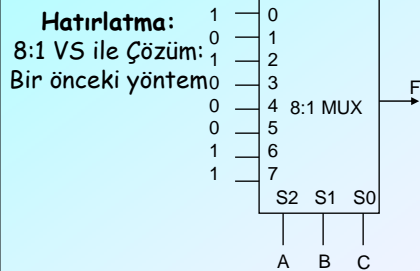
$2^{n-1}:1$ boyutlu bir adet veri seçici kullanılarak n girişli herhangi bir lojik devre ek olarak **sadece bir adet tümlenme bağlacı kullanılarak** gerçekleştirilebilir.

Yöntem:

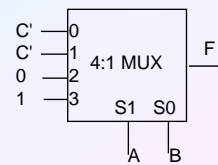
- Tasarlanacak olan fonksiyonun değişkenlerinden $n-1$ tanesi veri seçicinin seçme uçlarına bağlanır.
- Arta kalan değişkenin kendisi ya da tümleyeni, doğruluk tablosuna göre veri seçicinin veri girişlerine bağlanır.

Örnek:

$$F(A,B,C) = m_0 + m_2 + m_6 + m_7 = \Sigma_1(0,2,6,7)$$



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

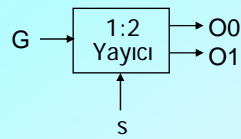
4:1 VS ile Çözüm:

Burada her iki c' değeri de aynı tümlenme kapısından elde edilebilir.

Yayıcı Makas (Demultiplexer):

- 1 adet veri girişi, n adet seçme (denetim) girişi, 2^n adet çıkışı vardır.
- Seçme girişlerine gelen değere göre, veri girişindeki değer çıkışlardan birine aktarılır. Diğer çıkışlar "0" değerini alır. Seçme girişlerindeki n bitlik ikili sayı girişteki değer hangi çıkışa aktarılacağını belirler.
- Yayıcılar çıkış sayılarına göre 1:m olarak adlandırılır. Burada m çıkış sayısını gösterir.

Örnek: 1:2 Yayıcı Makas ("Bire iki yayıcı" olarak okunur)

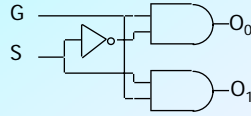


İşlev Tablosu:

s	O ₁	O ₀
0	0	G
1	G	0

Doğruluk Tablosu:

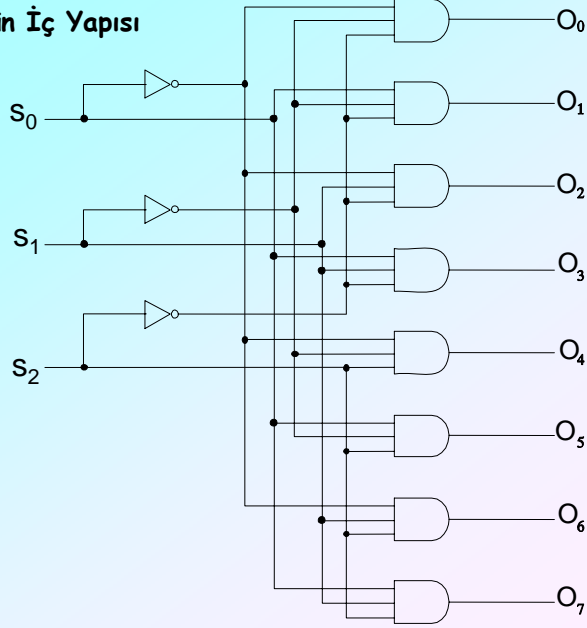
s	G	O ₁	O ₀
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

**Kod Çözücüler (Decoder):**

- n adet seçme (denetim) girişi, 2^n adet çıkışı vardır.
- Seçme girişlerine gelen değere göre, çıkışlardan bir tanesi "1" değerini diğerleri "0" değerini alır. Seçme girişlerindeki n bitlik ikili sayı hangi çıkışın "1" değerini alacağını belirler.
- Kod çözücü, girişine sabit "1" değeri verilmiş bir yayıcı makas gibi düşünülebilir.
- Kod çözücüler seçme girişi ve çıkış sayılarına göre $n:2^n$ olarak adlandırılır. Burada n seçme girişi sayısı, 2^n çıkış sayısıdır.

Örnek: 3:8 Kod Çözücü

	S ₂	S ₁	S ₀	O ₇	O ₆	O ₅	O ₄	O ₃	O ₂	O ₁	O ₀
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	1	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	1	0	0	0	0	0
5	1	0	1	0	0	1	0	0	0	0	0
6	1	1	0	0	1	0	0	0	0	0	0
7	1	1	1	1	0	0	0	0	0	0	0

3:8 Kod Çözücünün İç Yapısı**Kod Çözücüler ile Genel Amaçlı Lojik Devre Tasarımı:**

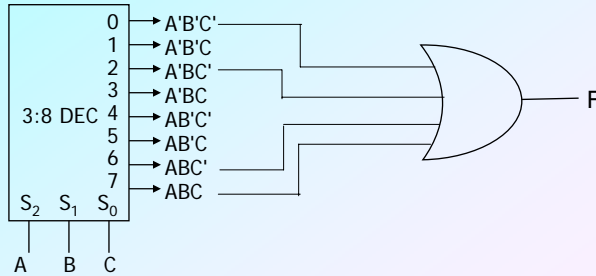
$n:2^n$ boyutlu bir kod çözücü kullanılarak n girişli m çıkışlı herhangi bir genel fonksiyon ek olarak **VEYA bağlaçları** kullanılarak gerçekleştirilebilir.

Yöntem:

- Tasarlanacak olan fonksiyonun değişkenleri (devrenin girişleri) kod çözücünün seçme uçlarına bağlanır.
- Kod çözücünün her çıkışı bir minterime karşı düşer. Gerçeklenecek olan fonksiyonu oluşturan minterimlere ilişkin çıkışlar VEYA kapıları ile toplanır.

Örnek:

$$F(A,B,C) = m_0 + m_2 + m_6 + m_7 = \Sigma_1(0,2,6,7)$$

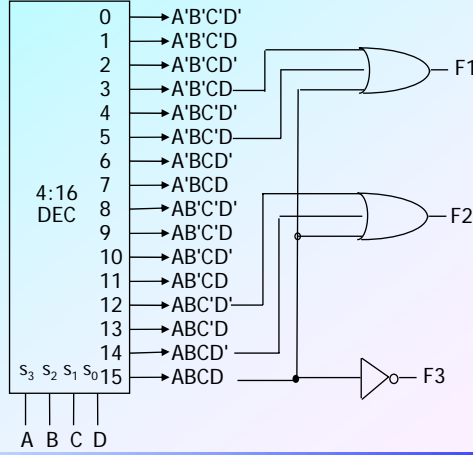


Örnek: 4 girişli 3 çıkışlı genel fonksiyon tasarımı

$$F1(A,B,C,D) = A' B C' D + A' B' C D + A B C D$$

$$F2(A,B,C,D) = A B C' D' + A B C$$

$$F3(A,B,C,D) = (A' + B' + C' + D')$$

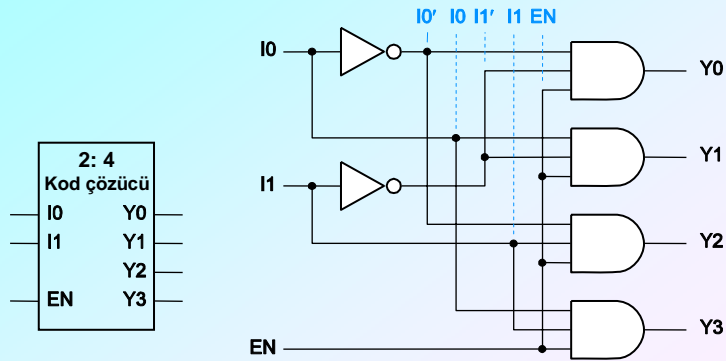
**İzin Girişli (EN) Kod Çözücü:**

Kod çözücülerde izin girişi (*Enable* -EN) olabilir.

EN girişi lojik "1" olduğunda kod çözücü normal işlevini görür.

EN girişi lojik "0" olduğunda kod çözücünün tüm çıkışları "0" olur.

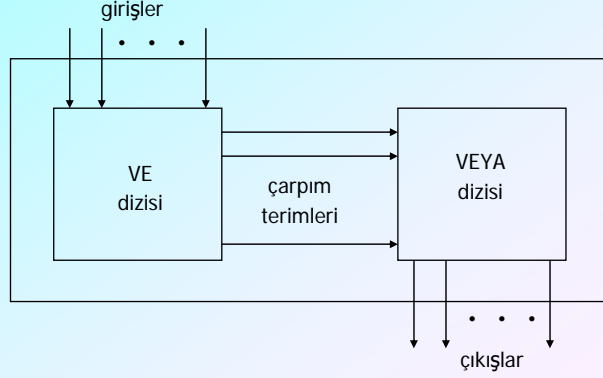
Aşağıda izin girişli bir 2:4 kod çözücü gösterilmiştir:



Programlanabilir Lojik Elemanlar (*Programmable Logic Device- PLD*)

Lojik devre gerçeğlemenin pratik yollarından biri de programlanabilir lojik elemanlar kullanmaktır. Bu elemanlar, içinde çok sayıda TÜMLEME, VE, VEYA bağlacı bulunduran tümdevrelerdir.

Tasarımcı bir "programlama" cihazı kullanarak bu bağlaçların arasında belli sınırlar içinde istediği bağlantıları gerçekleştirir. Böylece sadece tek bir tümdevre kullanılarak karmaşık lojik devreler gerçekleştirilebilir.



Programlanabilir Lojik Dizi (*Programmable Logic Array - PLA*)

PLD'ler iki gruba ayrılırlar:

1. *Programmable Logic Array - PLA* , 2. *Programmable Array Logic - PAL*

PLA'lar VE, VEYA gruplarının esnek olarak programlanabildiği elemanlardır. Bu elemanların sınırlarını belirleyen parametreleri şunlardır:

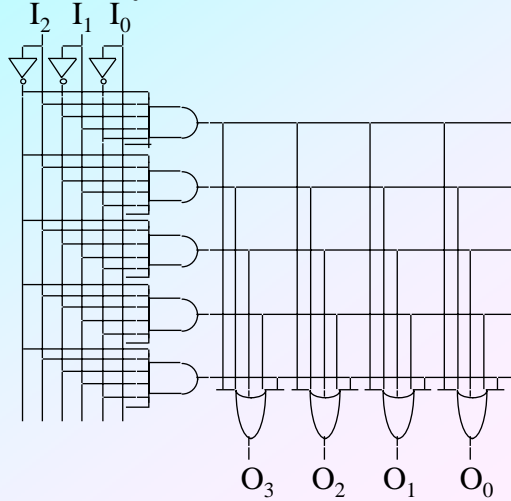
Giriş sayısı: n

Çıkış sayısı: m

VE kapısı sayısı: p

Bu tür bir eleman, " p çarpımlı $n \times m$ PLA" olarak adlandırılır.

Yandaki şekilde 5 çarpımlı 3×4 bir PLA gösterilmiştir.



Programlama:

Bağlaçların girişlerinde "sigortalar" (*fuse*) bulunur.

PLA'ların türlerine göre iki türlü programlama yapılır:

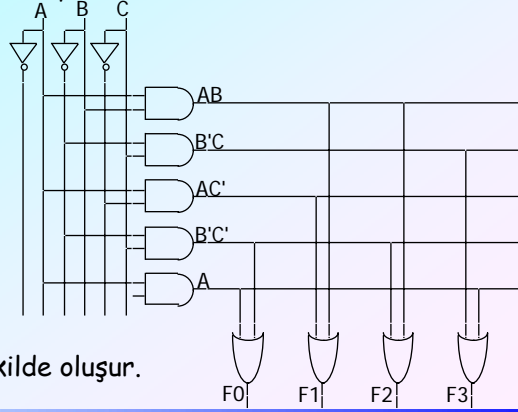
a) Normalde (programlamadan önce) tüm bağlantılar vardır. İstenmeyen bağlantıları koparmak için ilgili sigortalar devre dışı bırakılır.

b) Normalde (programlamadan önce) hiç bir bağlantı yoktur. İstenen bağlantıları gerçekleştirmek için ilgili sigortalar devreye sokulur.

Bu işlemleri gerçekleştirmek için özel yazılımlar ve cihazlar vardır.

Örnek:

$$\begin{aligned} F0 &= A + B' C' \\ F1 &= A C' + A B \\ F2 &= B' C' + A B \\ F3 &= B' C + A \end{aligned}$$

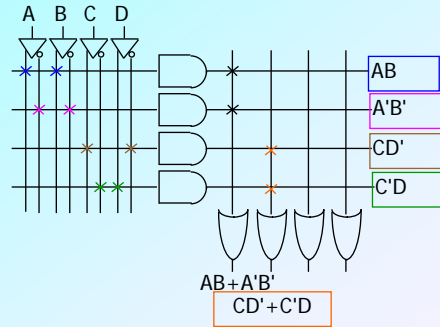


3x4PLA'nın iç bağlantıları, programlamadan sonra bu şekilde oluşur.

Basit Gösterim: Çizimleri karmaşık hale getirmemek için PLA çizimlerinde tüm hatlar gösterilmez. Onun yerine ilgili kapının girişine hangi hatlar bağlanacaksa o hattın üstüne X konur.

Örnek:

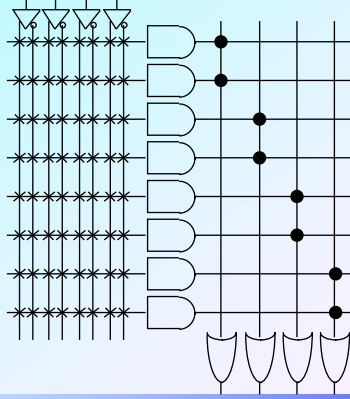
$$\begin{aligned} F0 &= A B + A' B' \\ F1 &= C D' + C' D \end{aligned}$$



Programlanabilir Dizi Lojii (Programmable Array Logic - PAL)

VE Baęlaçlarının girişleri PLA' larda olduęu gibi esnek bir biçimde programlanabilir. Ancak VEYA baęlaçlarının girişleri esnek deęildir. Her VEYA baęlacının girişine sadece belli VE baęlaçlarının çıkışları baęlıdır. Örneęin ilk VEYA baęlacının girişine sadece ilk iki VE baęlacının çıkışları gelebilir.

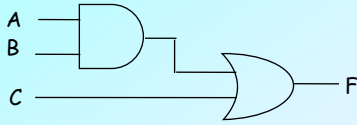
PAL' ler daha kolay programlanabilirler, daha ucuzdurlar, daha çok eleman içerebilirler.



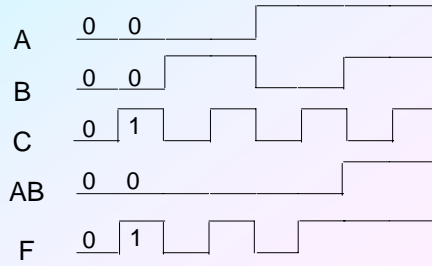
Zaman Diyagramları (Timing Diagrams)

- Lojik devrelerin zaman içindeki davranışlarını (giriş/çıkış ilişkisini) gösteren diyagramlardır.
- x ekseninde zaman, y ekseninde ise girişlerin ve çıkışların lojik değerleri (0/1 veya L/H) yer alır. Daha ayrıntılı zaman diyagramlarında y ekseninde elektriksel büyüklükler de (gerilim veya akım) yazılır.
- Fiziksel elemanların elektriksel özelliklerinden dolayı ortaya çıkan bazı durumların doğruluk tablosu ile gösterilmesi mümkün değildir. Böyle durumlarda devrelerin zaman diyagramlarını incelemek gerekir.

Örnek:

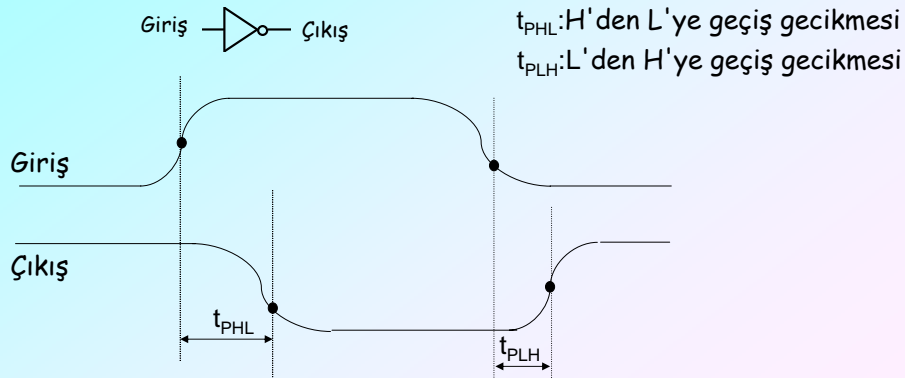


Yandaki diyagramda devrenin sadece lojik davranışı gösterilmiş, daha sonra anlatılacak olan gecikmeler dikkate alınmamıştır.



Propagasyon Gecikmesi (Propagation Delay)

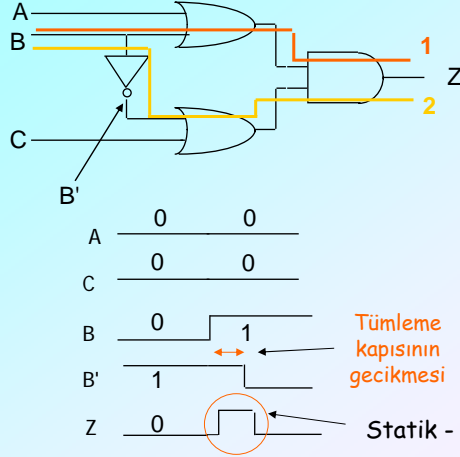
Lojik elemanları oluşturan elektronik devrelerin fiziksel yapılarından dolayı bir lojik elemanın girişine uygulanan işaret (lojik değer) ancak belli bir süre geçtikten sonra o elemanın çıkışında etkili olur. Giriş işaretinin elemanın içinde yol alarak çıkışı etkilemesi için geçen zaman o elemanın **propagasyon gecikmesini** belirler. Propagasyon gecikmesi lojik devrenin çalışma hızını belirler.



Gecikmeler nedeniyle oluşan problemler: **Kaza (Hazard)**

Bir giriş değerinin, farklı bir kaç yoldan çıkışı etkilemesi nedeniyle çıkışta beklenmedik değer değişiklikleri (kazalar) oluşur.

Örneğin aşağıdaki devrede B girişinin değeri Z çıkışına iki farklı yoldan etki eder.



Bu devrenin doğruluk tablosu incelendiğinde $A=0$, $B=0$, $C=0$ girişi için $Z=0$ olduğu görülür. Bu durumdayken $B=1$ olursa devrenin çıkışının $Z=0$ olarak kalması gerekir. Ancak 1. yol, 2. yola göre gecikmeler açısından daha "kısa" olduğundan Z çıkışında anlık bir değişim (kaza) oluşur.

Üç tür kaza (*hazard*) vardır:

a) **Statik 0:** Çıkış lojik 0'da kalması gerekirken kısa bir süre "1" olup tekrar 0'a iner.

Statik 0 kaza, toplamların çarpımı şeklinde gerçekleşen devrelerde oluşur.

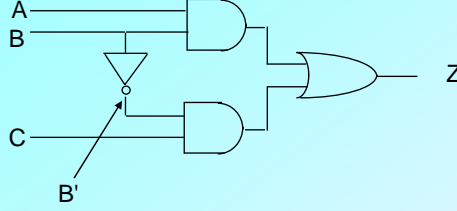
b) **Statik 1:** Çıkış lojik 1'de kalması gerekirken kısa bir süre "0" olup tekrar 1'e çıkar.

Statik 1 kaza, çarpımların toplamı şeklinde gerçekleşen devrelerde oluşur.

b) **Dinamik:** Çıkış bir kez değer değiştirmesi gerekirken bir kaç defa değer değiştirir.



Kazaların önlenmesi:



Çarpımların toplamı şeklinde gerçekleştirilen yandaki devrenin doğruluk tablosu incelendiğinde $A=1, B=1, C=1$ girişi için $Z=1$ olduğu görülür. Bu durumdayken $B=1$ 'den 0 'a inerse devrenin çıkışının $Z=1$ olarak kalması gerekir. Ancak Z çıkışında anlık bir değişim (statik 1 kaza) oluşur.

Bir devrede kaza tehlikesi olup olmadığı Karnaugh diyagramından da anlaşılabilir.

		B			
		00	01	11	10
A	0		1		
	1		1	1	1

$Z = AB + B'C$

B 'deki değişim bir asal çarpımdan diğerine geçilmesine neden olmaktadır. Böyle geçişler gecikmelerden dolayı kazalara neden olurlar. Eğer kazalar kesinlikle önlenmek isteniyorsa devrenin maliyeti arttırılarak, aralarında geçiş olan iki monomun konsansüsü de tasarıma eklenir.

		B			
		00	01	11	10
A	0		1		
	1		1	1	1

$Z = AB + B'C + AC$

ARDIŞIL DEVRELER (Sequential Circuits)

- Dersin ilk bölümünde **kombinezonsal devreleri** inceledik. Bu tür devrelerde çıkışın değeri o andaki girişlerin değerlerine bağlıdır.
- **Ardışıl (sequential) devrelerde** ise çıkış değeri, hem girişlerden gelen değerlere hem de devrenin bir önceki "durumuna" bağlıdır. Durum bilgisini tutmak için bu devrelerde bellek elemanları bulunur.
- Ardışıl devrelere örnek olarak bozuk parayla çalışan meşrubat makinelerindeki lojik devreler gösterilebilir. Böyle bir lojik devre, ürünü vermek için sadece o anda atılan parayı değil, daha önce atılmış olan parayı da dikkate almalıdır.
- Ardışıl devreler "**sonlu durumlu makine**" (*Finite State Machine- FSM*) modeli kullanılarak tasarlanırlar.

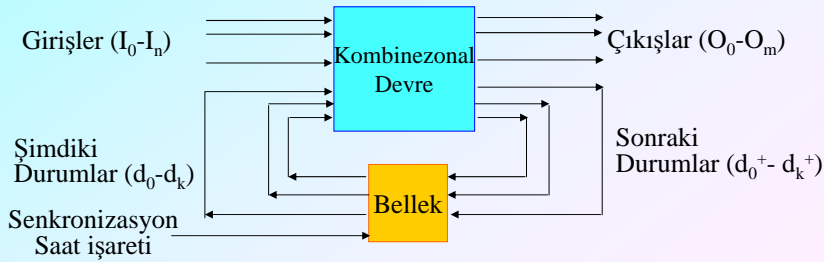
Sonlu Durumlu Makine (Finite State Machine- FSM) Modeli

Bu **modelleme yöntemi** sadece lojik devrelerde değil, bir çok başka sistemin tasarımında da kullanılır.

- Böyle bir makine ilk çalışmaya başladığında belli bir durumda bulunur.
- Gelen giriş değerine göre ve içinde bulunduğu duruma göre makine bir çıkış üretir.
- Gelen giriş değerine göre ve içinde bulunduğu duruma göre yeni bir duruma geçer.

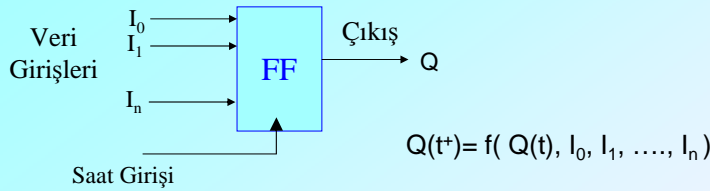
Sonlu durumlu makineler, lojik devre olarak olarak gerçekleştirilirken iki kısımdan oluşturulurlar:

- Lojik işlemleri yapan kombinezonel devre,
- Durum bilgisini tutan bellek elemanları.



Veri Saklama (Bellek) Elemanları

'Flip-flop': Bir bitlik bellek elemanıdır. Çok girişli, bir çıkışlı lojik bir devre olarak tasarlanırlar.



Q çıkışı flip-flopun o anda içindeki ikili değeri (0,1) dışarı yansıtır. Bu değer aynı zamanda flip-flopun durum bilgisidir. Q çıkışının alacağı yeni değer $Q(t^+)$, veri girişlerinin ve o andaki durumun $Q(t)$ bir fonksiyonu olarak belirlenir.

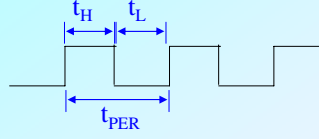
Saat işareti, veri girişlerindeki değerlerin ne zaman değerlendirileceğini, yani flip-flop'un ne zaman değer değiştireceğini belirten işarettir.

Sadece saat işaretinin etkin olduğu anlarda flip-flop'un içeriği yukarıdaki fonksiyona göre belirlenerek değiştirilir.

Saat işareti etkin değilse, veri girişleri değişse bile flip-flop bir önceki içeriğini korur.

Saat (Clock) İşareti:

Sayısal sistemlerdeki elemanların eş zamanlı (senkronize) çalışmasını sağlayan dikdörtgen dalga şeklinde bir işarettir.

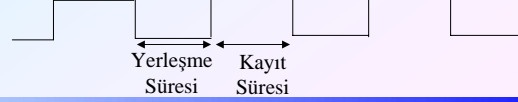


Saat işaretiyle denetlenen elemanlar (örneğin flip-flop) sadece saat işareti etkin olunca işlem yaparlar. Onun dışında eski durumlarını korurlar. Saat işaretinin kullanılması açısından elemanlar ikiye ayrılır.

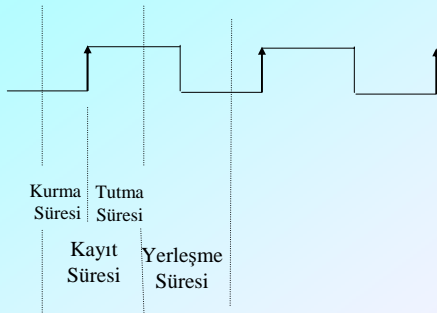
a) Düzey tetiklemeli elemanlar, b) Kenar tetiklemeli elemanlar

Düzey tetiklemeli elemanlar: Saat işaretinin bir düzeyini (pozitif lojikte "1" düzeyini) etkin düzey olarak kabul ederler. Bu elemanlar saat işareti "1" düzeyindeyken işlem yaparak durumlarını ve çıkışlarını değiştirirler; saat işareti "0" düzeyindeyken eski durumlarını korurlar.

Saat işaretinin "1" düzeyindeyken girişler işleme sokulduğundan, bu süre boyunca giriş değerleri sabit tutulmalıdır. Aksi durumda ardışıl elemanın çıkışının alacağı değer belirsiz olur. Bu süreye **kayıt süresi** denir. Saat işaretinin "0" olduğu sürede ise girişler değiştirilebilir. Bu süreye **yerleşme süresi** denir.



Kenar tetiklemeli elemanlar: Saat işaretinin bir kenarını (pozitif lojikte çıkan kenar) etkin kenar olarak kabul ederler. Bu elemanlar saat işareti 0→1 geçişi yapınca (çıkan kenar) işlem yaparak durumlarını ve çıkışlarını değiştirirler; saat işareti geçiş yapmazsa eski durumlarını korurlar. Negatif lojikte ise işlemler 1→0 geçişinde (inen kenar) yapılır. Saat işaretinin 0→1 geçişi yaparken girişler işleme sokulduğundan, bu kenardan belli bir süre önce ve sonra giriş değerleri sabit tutulmalıdır. Aksi durumda ardışıl elemanın çıkışının alacağı değer belirsiz olur.

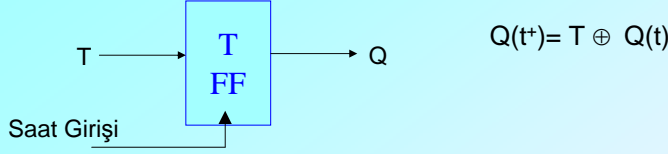


Kurma süresi "*Set-up time*"

Tutma süresi "*Hold time*"

Kayıt süresi, kurma ve tutma sürelerinin toplamından oluşur.

Ardışıl devrenin sağlıklı çalışması için bu süre boyunca girişlerin sabit kalması gerekir.

Örnek: Kenar tetiklemeli T Flip-Flop (Toggle Flip-flop)

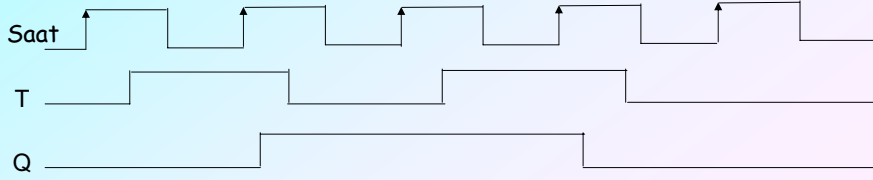
T flip-flopunun çıkışının (içeriği) alacağı değer, o andaki değer ile girişinin YA DA işlemine sokulmasıyla bulunur.

Buna göre girişine T=0 uygulanırsa flip-flopun içeriği değişmez.

Çünkü: $0 \oplus x = x$

Flip-flopun girişine T=1 uygulanırsa flip-flopun içeriği tümlenir.

Çünkü: $1 \oplus x = x'$

**İki Kararlı (Bistable) Devre**

Bellek elemanlarını açıklamadan önce, onların çalışmasını anlamakta yardımcı olacak iki kararlı elemandan söz edilecektir.

İki kararlı eleman, iki adet tümlenme kapısının **geri beslemeli** (*feedback*) olarak bağlanmasıyla oluşturulan, girişi olmayan, 2 tane çıkışı olan bir lojik devredir.

Bu devre iki durumdan birinde bulunur.

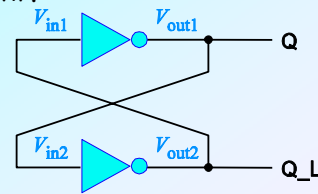
Üstteki tümlenme kapısının çıkışı (Q) sıfırsa, alttaki tümlenme kapısının girişi '0', çıkışı (Q_L) '1' olur. Bu da zaten Q'nun '0' olmasını gerektirdiğinden bu kararlı bir durumdur.

Üstteki tümlenme kapısının çıkışı (Q) '1' ise, alttaki tümlenme kapısının girişi '1', çıkışı (Q_L) '0' olur. Bu da zaten Q'nun '1' olmasını gerektirdiğinden bu da kararlı bir durumdur.

Bu elemanın iki kararlı durumu vardır. Q=0 ve Q=1

Girişi olmadığından elemanın durumunu dışarıdan denetlemek (değiştirmek) mümkün değildir.

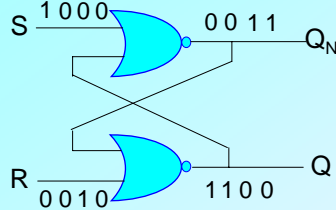
İlk gerilim verildiğinde eleman rasgele bir duruma geçer.



S-R (Set-Reset) Bilgi Saklama Elemanı

İki adet TVEYA veya iki adet TVE bağlacı ile oluşturulabilen bir bitlik saklama elemanıdır. Tüm flip-floplar, bu temel saklama elemanına yapılan eklemeler ile oluşturulabilir.

TVEYA ile oluşturulan S-R Saklama Elemanı:



S: Set (Birleme)
R: Reset (Sıfırlama)
Q: Çıkış (Durum)
QN: Tümlenmiş Çıkış (Q')

Hatırlatma: Bir TVEYA bağlacının bir girişi "1" olduğunda çıkışı mutlaka "0" olur

S	R	Q	QN
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	0	0

S=1, R=0'dan sonra
S=0, R=1'den sonra
Yasaklı girişler

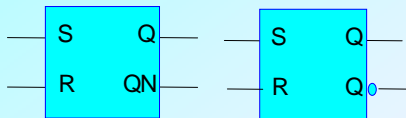
S girişi saklama elemanına "1" yazmak için, R girişi de "0" yazmak için kullanılır. Her iki giriş de "0" olduğunda SR elemanı bir önceki durumunu korur. Girişlerin her ikisine birden "1" verilmez.

Q çıkışını bir sonraki değeri $Q(t+1)$, girişlere ve saklama elemanının o anki durumuna $Q(t)$ bağlıdır. Buna göre S-R saklama elemanının doğruluk tablosu ve lojik ifadesi aşağıdaki gibi yazılabilir.

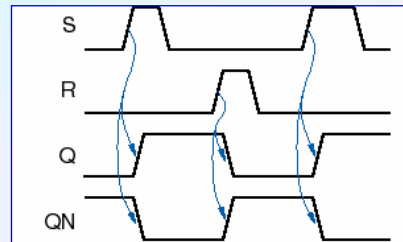
Q(t)	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Yasak
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Yasak

Q(t)	SR		S	
	00	01	11	10
0			Φ	1
1	1		Φ	1

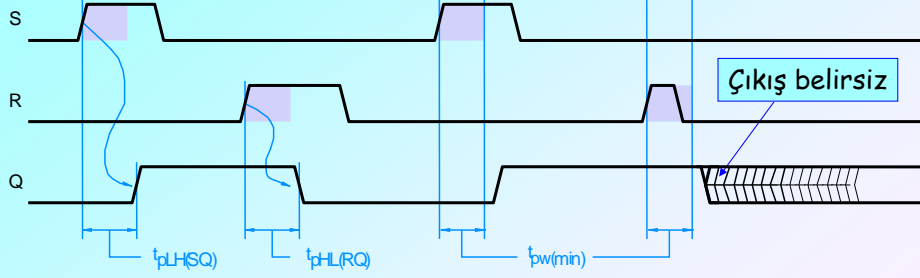
$$Q(t+1) = S + Q(t)R', \quad SR=0$$



Saat işareti ile tetiklenmeyen bu elemana **tutucu (latch)** denir. Flip-flop adı saat işareti ile tetiklenen bellek elemanlarına verilir.



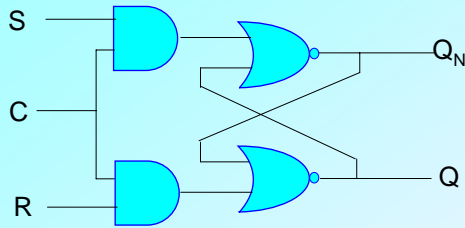
Elemanın içindeki propagasyon gecikmesinden dolayı S veya R girişlerindeki değişimlerin etkisi belli bir süre geçtikten sonra çıkışta etkili olur. Bu süre boyunca girişler sabit kalmalıdır. Aksi durumda çıkışın alacağı değer belirsiz olur.



- $t_{pLH(SQ)}$: S değiştiğinde çıkışın 0-1 değişim yapması için geçen süre.
 $t_{pHL(RQ)}$: R değiştiğinde çıkışın 1-0 değişim yapması için geçen süre.
 $t_{pw(min)}$: Girişlerin sabit kalması gereken en küçük süre.

İzin Girişli S-R Bilgi Saklama Elemanı

S ve R girişlerinin sadece istenen (izin verilen) zamanlarda etkili olabilmesi için bu girişlere VE kapıları bağlanır.



- S: Set (Birleme)
R: Reset (Sıfırlama)
Q: Çıkış (Durum)
 Q_N : Tümlenmiş Çıkış (Q')
C: İzin girişi

Ancak C=1 olduğunda elemanın içeriği değiştirilebilir. C=0 olduğunda elemanın içeriği korunur.



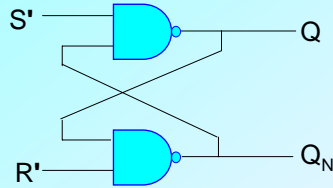
Tutucu (Latch), Flip-flop Farkı:

Buraya kadar tanıtilen S-R saklama elemanın bir saat işareti ile tetiklenmesi söz konusu değildir. İzin girişi etkin olduğu sürece bu elemanın içeriği değiştirilebilir. Bu tip elemanlara **tutucu (latch)** denir.

Saat işareti ile tetiklenen saklama elemanlarına ise **flip-flop** denir.

TVE Bağlı S' -R' Tutucu (Latch)

S-R veri saklama elemanları TVEYA kapıları yerine TVE kapıları kullanılarak da tasarlanabilir.



S': Set (Birleme) Tümlenyeni

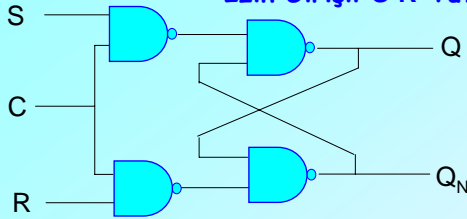
R': Reset (Sıfırlama) Tümlenyeni

Q: Çıkış (Durum)

Q_N: Tümlen Çıkış (Q')

S'	R'	Q	Q _N	
0	1	1	0	
1	1	1	0	S'=0, R'=1'den sonra
1	0	0	1	
1	1	0	1	S'=1, R'=0'dan sonra
0	0	1	1	Yasaklı girişler

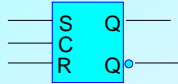
Hatırlatma: Bir TVE bağlacının bir girişi "0" olduğunda çıkışı mutlaka "1" olur

İzin Girişli S-R Tutucu

Yansı 5.16'da, TVEYA ve VE kapıları kullanılarak gerçekleştirilmiş olan izin girişli S-R tutucu sadece TVE bağlacıları kullanılarak yandaki şekilde gerçekleştirilebilir.

Tutucunu girişine yasaklı değerler (SR=11) uygulanırsa Q ve Q' çıkışlarının ikisi de 1 olur. Bu durumdayken izin kaldırılırsa tutucunun değeri belirsiz olur.

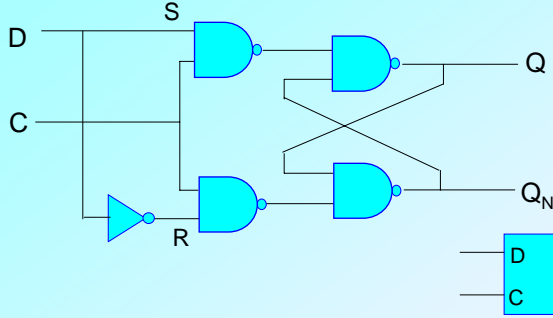
C	S	R	Q(t+1)
0	X	X	Q(t)
1	1	0	1
1	0	1	0
1	0	0	Q(t)
1	1	1	Yasak



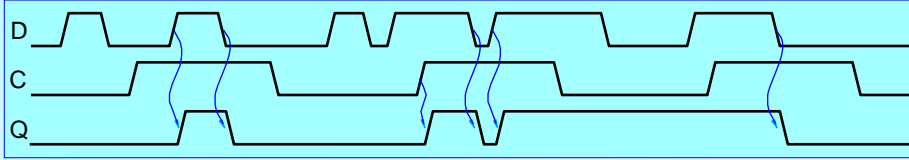
D tipi Tutucu (Delay Latch)

S-R tutucunun yapısına bazı eklemeler yaparak değişik fonksiyonlara sahip başka tipte tutucular elde edilebilir.

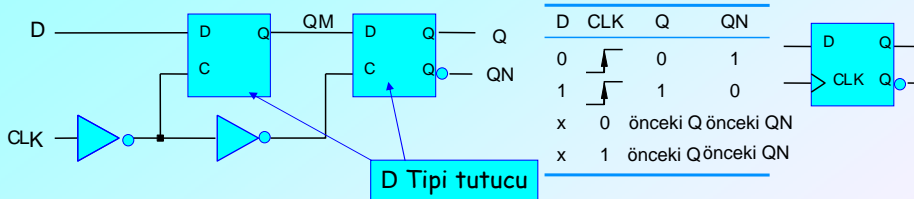
C=1 olduğu sürece D'den gelen değer tutucuya yazılır. C=0 olduğu sürece tutucu bir önceki değerini korur.



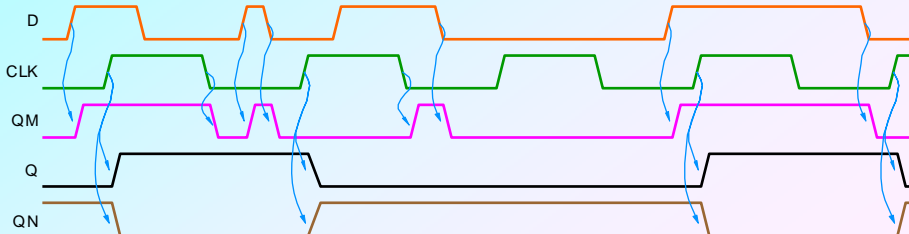
C	D	Q(t ⁺)	Q _N (t ⁺)
1	0	0	1
1	1	1	0
0	X	Q(t)	Q _N (t)

**Pozitif (çıkan) kenar tetiklemeli D tipi Flip-flop**

Tutucular izin girişleri etkin olduğu sürece veri girişlerindeki değerlere göre içeriklerini değiştirirler. Flip-floplar ise ancak bir saat işareti etkin olduğunda veri girişlerindeki değerlerden etkilenirler.



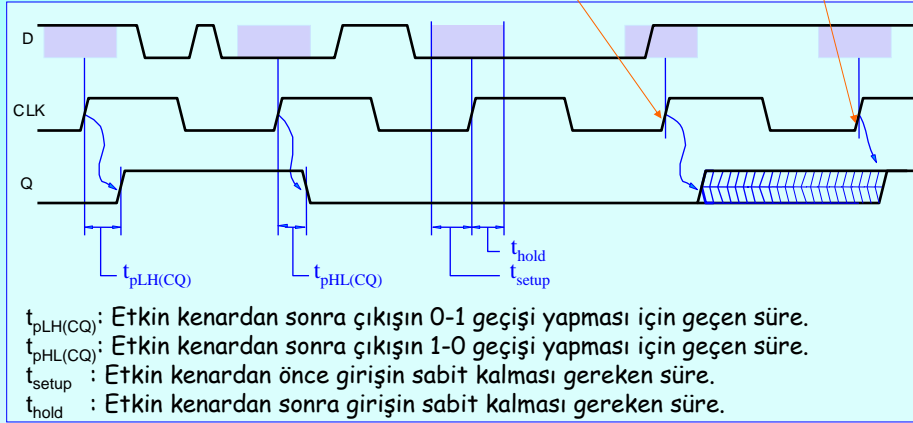
D	CLK	Q	Q _N
0	↑	0	1
1	↑	1	0
x	0	önceki Q	önceki Q _N
x	1	önceki Q	önceki Q _N



Pozitif kenar tetiklemeli D tipi flip-flopunun zamanlama özellikleri

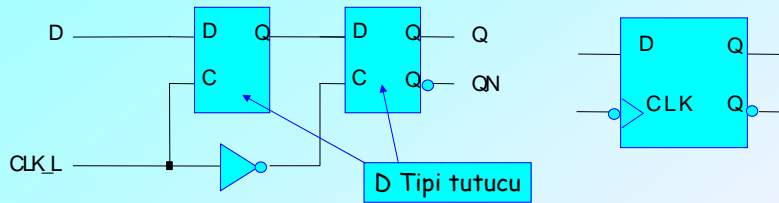
Kurma süresine (*setup time*) uyulmadığı için çıkış belirsizdir.

Çıkış tekrar belirli bir değer (örnekte 1) alır.



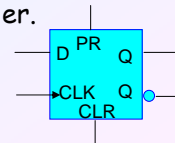
Negatif (inen) kenar tetiklemeli D tipi Flip-flop

Saat işaretinin inen kenarlarında D girişindeki veri flip-flopa yazılır.



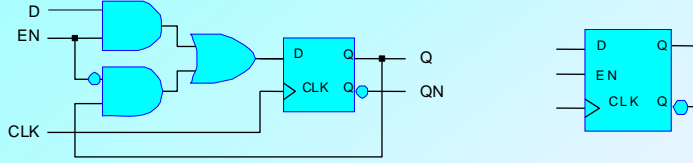
D	CLK_L	Q	QN
0	↓	0	1
1	↓	1	0
x	0	eski Q	eski QN
x	1	eski Q	eski QN

Flip-floparda, özellikle başlangıç değeri yazabilmek için saat işaretinden bağımsız olarak (**asen kron**) çalışan girişler de bulunabilir. Flip-flopa 1 yazmak için PR (*Preset*), 0 yazmak için CLR (*Clear*) girişi kullanılır. Asenkron girişler, saat işareti etkin olmasa da flip-flopu etkilerler.



Kenar tetiklemeli ve izin girişli D tipi Flip-flop

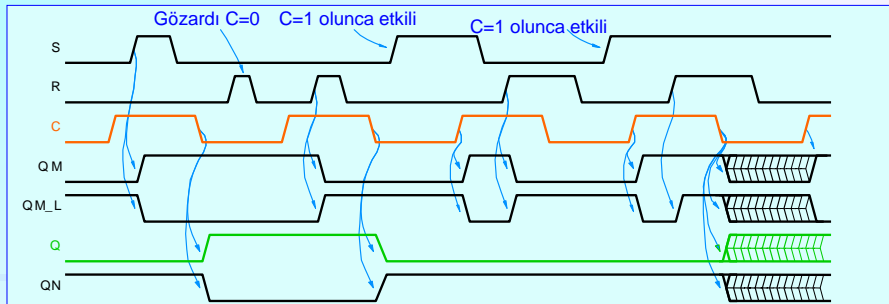
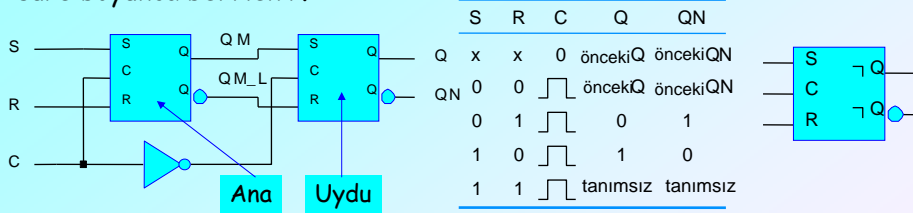
Flip-floplarda da izin girişi (*enable*) bulunabilir. Flip-flopun içeriğinin değiştirilebilmesi için izin girişi etkin olmalıdır. Aksi durumda flip-flopun içeriği korunur.



D	EN	CLK	Q	QN
0	1		0	1
1	1		1	0
x	0		eski Q	eski QN
x	x	0	eski Q	eski QN
x	x	1	eski Q	eski QN

Ana/Uydu (Master/Slave) tipi SR Flip-flop

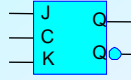
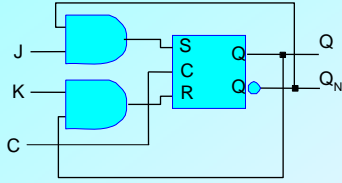
Ana/uydu tipi flip-floplar darbe tetiklemeli (*pulse-triggered*) türden elemanlardır. Bu tip flip-flopun içeriği (çıkışı) sadece saat işaretinin inen kenarında değişir. Ancak flip-flopun alacağı değer saat işaretinin 1 olduğu süre boyunca belirlenir.



JK Tutucu

SR flip-floplarındaki yasaklı giriş ($S=1, R=1$) problemi JK tipi saklama elemanları ile çözülmüştür.

Bu elemanlar SR elemanları gibi çalışır. J girişi birleme, K girişi ise sıfırlama işlemi yapar. $J=1, K=1$ girişi uygulanması durumunda elemanın içeriği tümlenmiş olur.

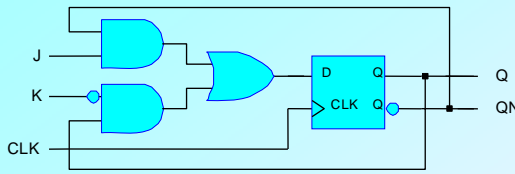


J	K	C	Q	QN
x	x	0	eski Q	eski QN
0	0	1	eski Q	eski QN
0	1	1	0	1
1	0	1	1	0
1	1	1	eski QN	eski Q

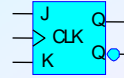
$$Q(t+1) = J \cdot Q(t)' + K' \cdot Q(t)$$

Kenar Tetiklemeli JK Flip-flopu

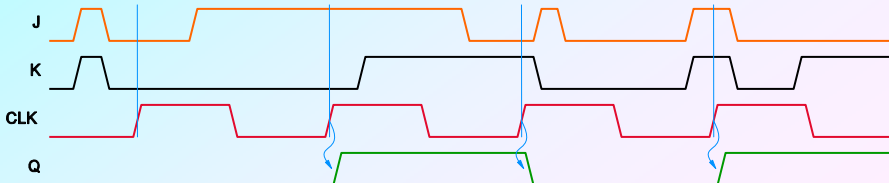
Kenar tetiklemeli bir D flip-flopu ve lojik bağlaçlar kullanılarak kenar tetiklemeli bir JK flip-flopu tasarlanabilir. Bu flip-flopta JK girişleri sadece saat işaretinin etkin geçişlerinde (kenarlarında) değerlendirilir.

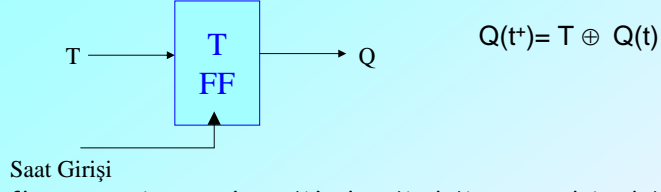


J	K	CLK	Q	QN
x	x	0	eski Q	eski QN
x	x	1	eski Q	eski QN
0	0	↑	eski Q	eski QN
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	eski QN	eski Q



$$Q(t+1) = J \cdot Q(t)' + K' \cdot Q(t)$$



Kenar tetiklemeli T Flip-Flop (Toggle Flip-flop)

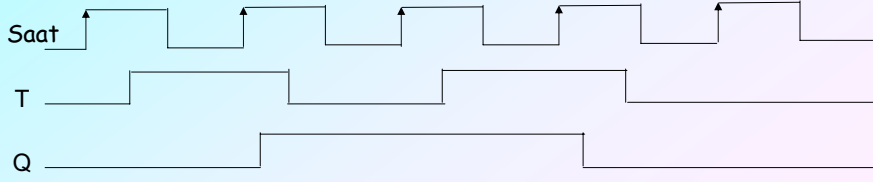
T flip-flopunun çıkışının (içeriği) alacağı değer, o andaki değer ile girişinin YA DA işlemine sokulmasıyla bulunur.

Buna göre girişine $T=0$ uygulanırsa flip-flopun içeriği değişmez.

Çünkü: $0 \oplus x = x$

Flip-flopun girişine $T=1$ uygulanırsa flip-flopun içeriği tümlenir.

Çünkü: $1 \oplus x = x'$



Senkron Ardışıl Devreler (Synchronous Sequential Circuits)

Ardışıl (*sequential*) devrelerde çıkış değeri, hem girişlerden gelen değerlere hem de devrenin "durumuna" bağlıdır.

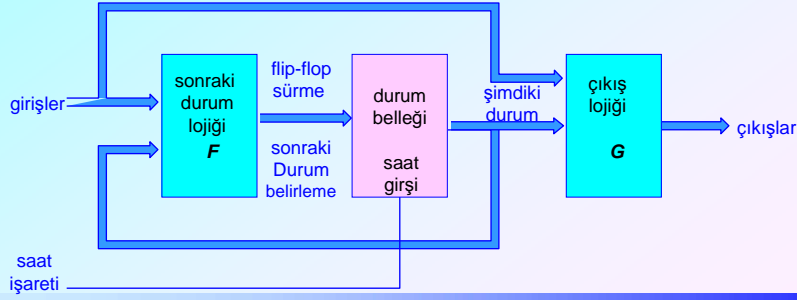
Sonlu durumlu makine modeline göre oluşturulan bu devrelerde durum bilgileri flip-floplarda tutulur.

Tüm flip-floplar aynı saat işareti ile tetiklenir (senkron) . Buna göre makine sadece saat işaretinin etkin geçişlerinde durum değiştirebilir.

Senkron ardışıl devreler iki farklı modele göre tasarlanabilir.

a) Mealy Modeli

Bu modelde çıkışlar hem o andaki girişlerin hem de o andaki durumun fonksiyonudur.

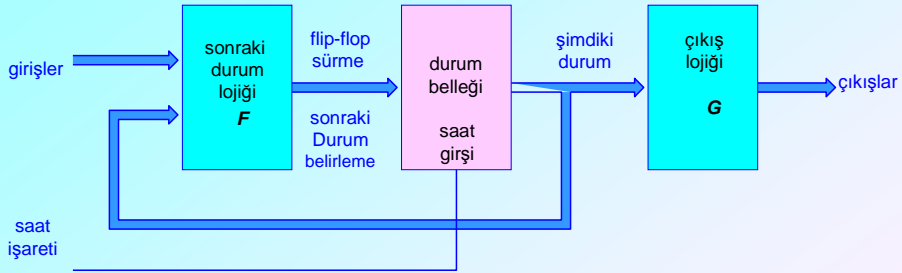


b) Moore Modeli

Bu modelde çıkışlar sadece durumların bir fonksiyonudur.

Girişlerdeki değerler sadece sonraki durumu belirler.

Durum bilgisi de çıkıştaki değeri belirler.



Bir modele göre tasarlanmış olan bir devreyi diğer modele dönüştürmek mümkündür.

Bir çok devreyi hem Mealy hem de Moore modeline göre tasarlamak mümkündür.

Senkron ardışıl devrelerin çözümlenmesi (Analiz)

Bir ardışıl devrenin tasarlanıp gerçekleşmesi konusunda önce tasarlanmış olan bir devrenin nasıl çözümleneceği incelenecektir.

Hatırlatma: Mealy modelinde bir devrenin gerçekleşmesi aşağıda gösterilmiş olan F ve G fonksiyonlarının gerçekleşmesi anlamına gelmektedir.

$$\begin{array}{lll} S^+ = F(S,I) & S: \text{Şimdiki durumlar (State),} & S^+: \text{Sonraki durumlar} \\ O = G(S,I) & I: \text{Girişler (Input),} & O : \text{Çıkışlar (Output)} \end{array}$$

Bir ardışıl devrenin çözümlenmesi (analiz edilmesi) demek, F ve G fonksiyonları şeklinde verilmiş bir devrenin "ne yaptığının" belirlenmesi demektir.

Çözümleme 3 adımdan oluşur:

1. Devrenin çiziminden F ve G fonksiyonlarının ifadeleri bulunur.
2. F ve G fonksiyonları kullanılarak olası tüm girişler ve durumlar için makinenin hangi durumlara geçeceği ve hangi çıkışları üreteceği bir tablo halinde yazılır. Bu tabloya **durum/çıkış tablosu** denir.
3. Makinenin işlevini daha iyi görebilmek için durum geçişlerini ve çıkışları grafik olarak gösteren **durum geçiş diyagramı** çizilir.

F fonksiyonu, flip-flopların girişlerine gelecek olan sürücü değerleri belirler. Bu değerler ise bir saat darbesi sonra flip-flopun hangi değeri alacağını (sonraki durumu) belirler.

Gelen girişlere göre Flip-flopun içeriğinin nasıl değişeceğini hesaplamak için flip-flopların karakteristik fonksiyonlarını bilmek gerekir.

Flip-flopların karakteristik fonksiyonları:

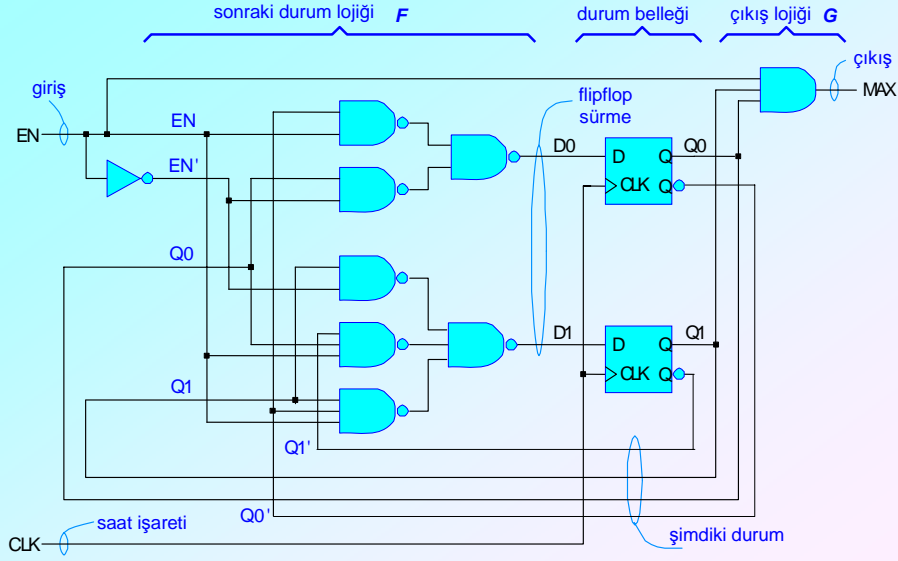
$$\text{SR FF: } Q(t+1) = S + R' \cdot Q(t), \quad \text{SR}=0$$

$$\text{JK FF: } Q(t+1) = J \cdot Q(t)' + K' \cdot Q(t)$$

$$\text{D FF: } Q(t+1) = D$$

$$\text{T FF: } Q(t+1) = T \oplus Q(t)$$

Örnek: Aşağıda çizimi verilmiş olan senkron ardışıl devreyi çözümlünüz.



$$S = \{Q_0(t), Q_1(t)\} \quad S^+ = \{Q_0(t^+), Q_1(t^+)\} = ff(D_0, D_1) \quad I = \{EN\} \quad O = \{MAX\}$$

1. F fonksiyonunun ifadesi belirlenir.

$$D_0 = Q_0 \cdot EN' + Q_0' \cdot EN$$

$$D_1 = Q_1 \cdot EN' + Q_1' \cdot Q_0 \cdot EN + Q_1 \cdot Q_0' \cdot EN$$

2. Sonraki durumlar $S^+ = \{Q_0(t^+), Q_1(t^+)\}$ hesaplanır

$$Q_0^+ = D_0 \quad (\text{D tipi FF karakteristik fonksiyonu})$$

$$Q_1^+ = D_1 \quad (\text{D tipi FF karakteristik fonksiyonu})$$

$$Q_0^+ = Q_0 \cdot EN' + Q_0' \cdot EN$$

$$Q_1^+ = Q_1 \cdot EN' + Q_1' \cdot Q_0 \cdot EN + Q_1 \cdot Q_0' \cdot EN$$

3. Durum geçiş tablosu (*State transition table*) oluşturulur.

Q1 ⁺ Q0 ⁺		EN	
		0	1
Q1Q0	00	00	01
	01	01	10
	10	10	11
	11	11	00

Tabloyu daha anlaşılır hale getirmek için durum kodlarına simgeler karşı düşürülür.

00: A

01: B

10: C

11: D

S ⁺		EN	
		0	1
S	A	A	B
	B	B	C
	C	C	D
	D	D	A

S: Şimdiki durum S⁺: Sonraki durum Q1 ve Q0: Durum değişkenleri

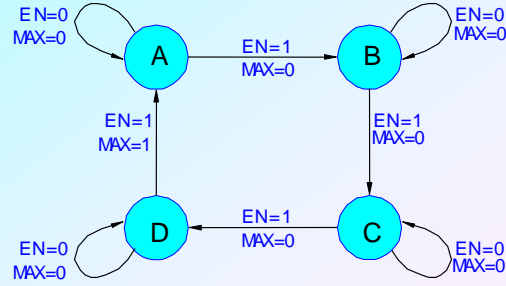
4. Çıkış fonksiyonunu G'nin ifadesi belirlenir.

$$MAX = Q1 \cdot Q0 \cdot EN$$

5. Durum/Çıkış tablosu oluşturulur.

S ⁺ , MAX	EN	
	0	1
A	A,0	B,0
B	B,0	C,0
C	C,0	D,0
D	D,0	A,1

Bu tablo sonlu durumlu makinenin davranışını göstermektedir. Bu davranış görsel olarak durum diyagramları ile de gösterilebilir.

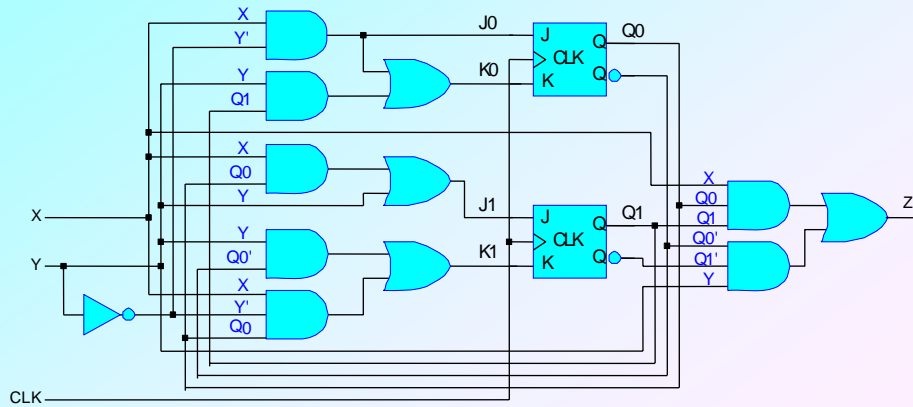


Makinenin davranışının sözle ifade edilmesi: A durumu başlangıç durumu olarak ele alınıp diyagram incelendiğinde, bu devrenin girişine 4'ün katları kadar "1" geldiğinde devrenin çıkışının "1" olduğu, aksi durumlarda ise "0" olduğu görülür.

JK Flip-flopları ile tasarlanmış bir senkron ardışıl devrelerin çözümlenmesi

Bir önceki örnekten farklı olarak, burada sonraki durum değerleri Q_i^+ belirlenirken JK flip-flopunun karakteristik fonksiyonu kullanılacaktır. Hatırlatma: $Q^+ = J \cdot Q' + K' \cdot Q$

Örnek:



1. Flip-flopların girişlerini süren F fonksiyonu:

$$J0 = X \cdot Y'$$

$$K0 = X \cdot Y' + Y \cdot Q1$$

$$J1 = X \cdot Q0 + Y$$

$$K1 = Y \cdot Q0' + X \cdot Y' \cdot Q0$$

2. Sonraki durumlar $S^+ = \{Q0(t^+), Q1(t^+)\}$ hesaplanır

$$Q0^+ = J0 \cdot Q0' + K0 \cdot Q0 \quad (\text{JK tipi FF karakteristik fonksiyonu})$$

$$Q0^+ = X \cdot Y' \cdot Q0' + (X \cdot Y' + Y \cdot Q1)' \cdot Q0$$

$$Q0^+ = X \cdot Y' \cdot Q0' + X' \cdot Y' \cdot Q0 + X' \cdot Q1' \cdot Q0 + Y \cdot Q1' \cdot Q0$$

$$Q0^+ = X \cdot Y' \cdot Q0' + X' \cdot Y' \cdot Q0 + Y \cdot Q1' \cdot Q0 \quad (\text{sadeleştirme})$$

$$Q1^+ = J1 \cdot Q1' + K1 \cdot Q1 \quad (\text{JK tipi FF karakteristik fonksiyonu})$$

$$Q1^+ = (X \cdot Q0 + Y) \cdot Q1' + (Y \cdot Q0' + X \cdot Y' \cdot Q0)' \cdot Q1$$

$$Q1^+ = X \cdot Q1' \cdot Q0 + Y \cdot Q1' + X' \cdot Y' \cdot Q1 + Y' \cdot Q1 \cdot Q0' + X' \cdot Q1 \cdot Q0 + Y \cdot Q1 \cdot Q0$$

$$Q1^+ = X \cdot Q1' \cdot Q0 + Y \cdot Q1' + Y \cdot Q0 + Y' \cdot Q1 \cdot Q0 + Y' \cdot Q1 \cdot Q0' \quad (\text{sadeleştirme})$$

3. Çıkış fonksiyonunun ifadesi belirlenir

$$Z = X \cdot Q1 \cdot Q0 + Y \cdot Q1' \cdot Q0'$$

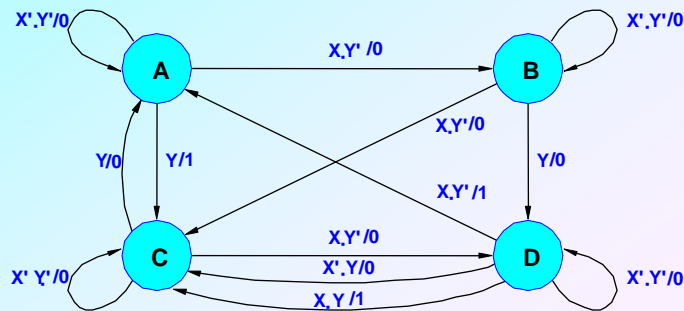
Durum/Çıkış Tablosu:

 $Q1^+, Q0^+, Z$

Q1Q0	XY			
	00	01	10	11
00	00,0	10,1	01,0	10,1
01	01,0	11,0	10,0	11,0
10	10,0	00,0	11,0	00,0
11	11,0	10,0	00,1	10,1

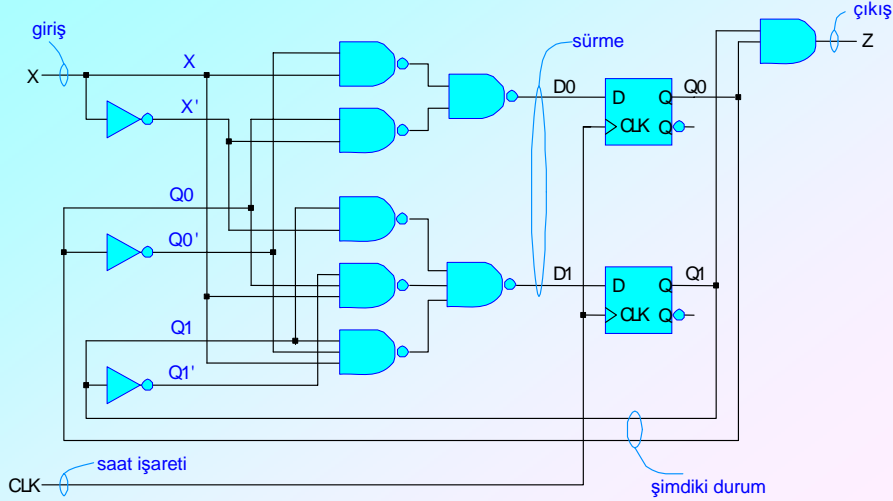
 S^+, Z

S	XY			
	00	01	10	11
A	A,0	C,1	B,0	C,1
B	B,0	D,0	C,0	D,0
C	C,0	A,0	D,0	A,0
D	D,0	C,0	A,1	C,1



Moore modeline göre tasarlanmış bir ardışıl devrenin çözülmesi:

Önceki örneklerde çözümlenen devreler Mealy modeline göre tasarlanmıştı. Aşağıdaki örnekte verilen devre ise Moore modeline göre tasarlanmıştır. Görüldüğü gibi Z çıkışı sadece durumlara (Q1,Q0) bağlıdır.



Moore modeline göre tasarlanmış makinelerin çözülmesi Mealy modeli ile büyük ölçüde aynıdır. Sadece çıkış tablosu ve durum geçiş diyagramının oluşturulması farklıdır.

1. Flip-flopları süren F fonksiyonunun ifadesi belirlenir.

$$D0 = Q0 \cdot X' + Q0' \cdot X$$

$$D1 = Q1 \cdot X' + Q1' \cdot Q0 \cdot X + Q1 \cdot Q0' \cdot X$$

2. Sonraki durumlar $S^+ = \{Q0(t^+), Q1(t^+)\}$ hesaplanır

$$Q0^+ = D0$$

$$Q1^+ = D1$$

$$Q0^+ = Q0 \cdot X' + Q0' \cdot X$$

$$Q1^+ = Q1 \cdot X' + Q1' \cdot Q0 \cdot X + Q1 \cdot Q0' \cdot X$$

3. Durum geçiş tablosu oluşturulur.

Q1 ⁺ Q0 ⁺		X	
		0	1
Q1Q0	00	00	01
	01	01	10
	10	10	11
	11	11	00

Durum kodlarına simgeler karşı düşürülür.

S ⁺		X	
		0	1
S	A	A	B
	B	B	C
	C	C	D
	D	D	A

S: Şimdiki durum S⁺: Sonraki durum Q1 ve Q0: Durum değişkenleri

4. Çıkış fonksiyonunun ifadesi belirlenir.

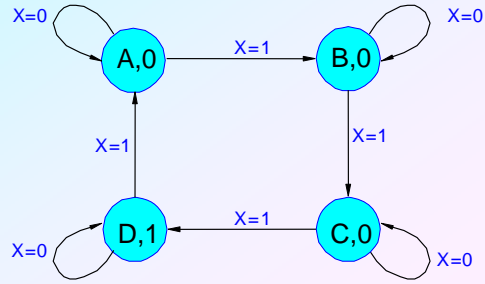
$$Z = Q1 \cdot Q0 \quad (\text{sadece durum deęişkenlerine baęlı})$$

5. Durum/Çıkış tablosu oluşturulur.

S ⁺ \ S	X=0	X=1	Z
A	A B	0	0
B	B C	0	0
C	C D	0	0
D	D A	1	1

Moore modelinde çıkış sadece durumların bir fonksiyonu olduğu için tablonun her satırına bir çıkış değeri yazılır. Bu değeri, makinenin çıkışının, o satırdaki duruma geldiğinde alacağı değerdir.

Moore modelinde durum geçiş diyagramı çizilirken çıkış değerleri durumların içine yazılır.



Mealy ve Moore Modellerinde Çıkışların Yorumlanması

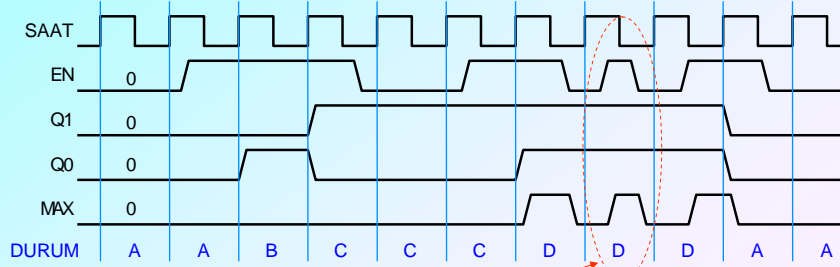
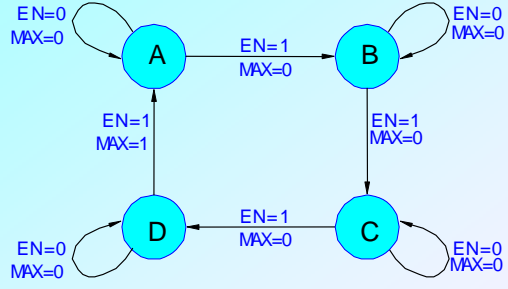
Mealy ve Moore modellerinde çıkıştaki değerin hangi anda geçerli olacağı (çıkışın ne zaman örnekleneceği) farklılık göstermektedir.

Mealy Modeli:

Mealy modelinde çıkış girişlere de baęlı olduğundan girişteki değeri deęiştirdiği anda çıkıştaki değeri de deęişir. Buna göre Mealy modeli ile tasarlanan bir makine şu şekilde çalışır:

1. Girişlere (I) değeri verilir.
2. Çıkışların değeri, giriş ve o andaki durumun bir fonksiyonu olarak belirlenir. $O = G(S, I)$
3. Saat işareti etkin olur. Örneğin çıkan kenar oluşur.
4. Yeni duruma geçilir. Yeni durum, girişin ve o andaki durumun fonksiyonu olarak belirlenir. $S^+ = F(S, I)$

Örnek: Yanda durum geçiş diyagramı verilen ve Mealy modeline göre tasarlanmış olan ardışıl devrenin zamanlama diyagramı aşağıda gösterilmiştir.



Giriş değiştiği anda çıkış da değişmektedir.

Moore Modeli:

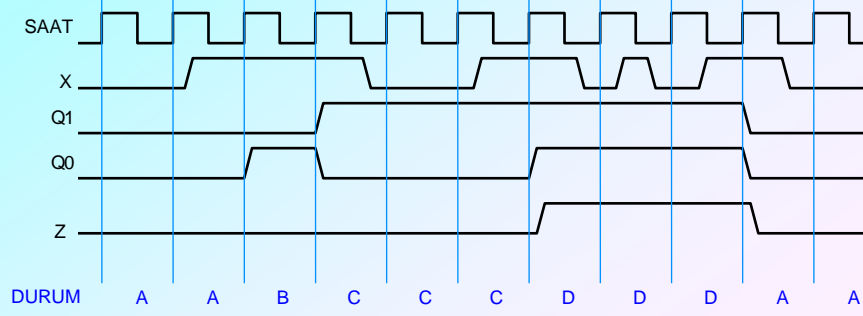
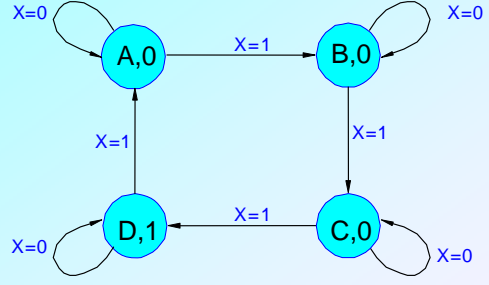
Moore modelinde çıkışın değeri sadece durum değişkenlerine bağlı olduğundan girişteki değer değişimi çıkışı hemen etkilemez. Girişteki değer çıkış üzerindeki etkisi ancak durum değiştikten sonra görülür.

Buna göre Moore modeli ile tasarlanan bir makine şu şekilde çalışır:

1. Girişlere (I) değer verilir.
2. Saat işareti etkin olur. Örneğin çıkan kenar oluşur.
3. Yeni duruma geçilir. Yeni durum, girişin ve o andaki durumun fonksiyonu olarak belirlenir. $S^+ = F(S, I)$
4. Çıkışların değeri, **yeni durumun** bir fonksiyonu olarak belirlenir. $O = G(S)$

Görüldüğü gibi bu modelde girişlerdeki değişimin çıkıştaki etkisi bir saat darbesi sonra görülür.

Örnek: Yanda durum geçiş diyagramı verilen ve Moore modeline göre tasarlanmış olan ardışıl devrenin zamanlama diyagramı aşağıda gösterilmiştir.



Senkron Ardışıl Devrelerin Tasarlanması (*Design*)

Bir ardışıl devrenin tasarlanması, çözülecek olan problemin sözle anlatımıyla başlar. Bundan sonra aşağıdaki aşamalardan geçilerek senkron ardışıl devre tasarlanarak gerçekleştirilir. Tasarım aşaması bilgisayar programı yazmaya benzer. Fiziksel dünyadaki problem ortaya konulduktan sonra uygun bir modelleme yapılarak çözüme giden yolun aranması gerekir.

Bir ardışıl devrenin tasarlanması aşağıdaki adımlardan oluşur:

1. Çözülecek problemin (devrenin yapması gereken işin) sözle anlatımı. Burada belirsizlikleri ortadan kaldırmak için zaman diyagramı da çizilebilir.
2. Devrenin hangi modele (Mealy ya da Moore) göre tasarlanmasının uygun olacağına karar verilir.
3. Seçilen modele göre devrenin durum geçiş ve çıkış tabloları oluşturulmaya çalışılır. Bu aşamaya, eğer gerekiyorsa durum geçiş diyagramı çizilerek de başlanabilir. Bu aşama program yazmaya benzer; bu nedenle sezgisel yaklaşım da gerektirir. Mümkünse durum indirgemesi yapılır. Burada amaç en az sayıda durum ile makinenin istenen işlevi yerine getirmesini sağlamaktır.

4. Durum kodlaması: Durumlara ikili kodlar karşı düşürülür. Eğer durum sayısı n ise durum değişkeni sayısı (flip-flop sayısı) m aşağıdaki gibi hesaplanır.

$$m = \lceil \log_2 n \rceil$$

Burada $\lceil x \rceil$ tavan fonksiyonudur. Örneğin $\lceil 4.1 \rceil = 5$ ve $\lceil 4.0 \rceil = 4$

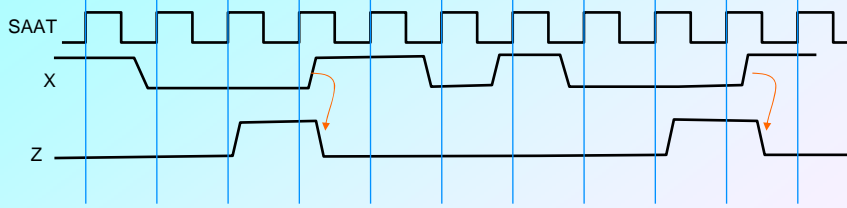
Durum geçiş ve çıkış tablosu gerçek durum değişkenleri değerleri kullanılarak oluşturulur.

5. Kullanılacak flip-flop tipine karar verilir.
6. Seçilen flip-flopların geçiş tablolarından yararlanılarak durum geçiş tablosuna uygun değerler yazılır ve flip-flopları sürme fonksiyonu (F) elde edilir.
7. Çıkış tablosundan çıkış fonksiyonu (G) elde edilir.
8. Fonksiyonlara ait kombinezon devreler dersin ilk bölümünde öğretilmiş şekilde en düşük maliyetle gerçekleştirilerek çizilir.

Senkron Devre Tasarım Örneği:

Bir girişi (X) ve bir çıkışı (Z) olan senkron ardışıl bir devre tasarlanacaktır. Devrenin girişi bir birini izleyen en az iki saat darbesi boyunca lojik 0'da kaldıktan sonra, girişten lojik 0 geldiği sürece devrenin çıkışı lojik 1 olacaktır.

Problemi daha iyi anlayabilmek için zamanlama diyagramı da çizilebilir.



Devrenin, yukarıdaki zaman diyagramına uygun olarak çalışması isteniyorsa tasarımın Mealy modeline göre yapılması gerekir. Çünkü çıkış, girişteki değişimden hemen (saat işareti gelmeden) etkilenmektedir.

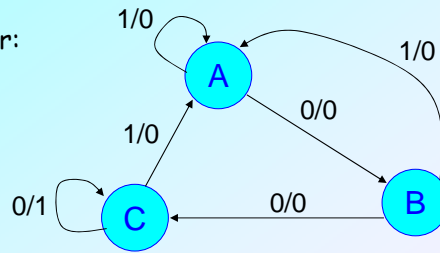
1. Sözlü anlatımdan (zamanlama diyagramından) durum diyagramının oluşturulması.

Makine üç durum ile tasarlanabilir:

A: Hiç sıfır gelmedi durumu

B: Birinci sıfır geldi

C: İkinci sıfır geldi



2. Durum geçiş tablosu

S ⁺ , Z	X	
S	0	1
A	B,0	A,0
B	C,0	A,0
C	C,1	A,0

Durum Kodlaması:

A: 00

B: 01

C: 11

Durum değişkenleri:

Q_1, Q_0

Q_1+Q_0, Z	X	
Q_1, Q_0	0	1
00	01,0	00,0
01	11,0	00,0
11	11,1	00,0
10	\emptyset, \emptyset	\emptyset, \emptyset

Durum kodlaması farklı şekilde de yapılabilirdi. Örneğin A:00, B: 10, C:01 olabilirdi. Bu durumda devrenin iç yapısı farklı olurdu. Ancak dışarıdan bakıldığında devre aynı işlevi yerine getirirdi.

3. Kullanılacak flip-floplara karar verilmesi. Bu örnekte pozitif kenar tetiklemeli D tipi flip-flopları kullanılacaktır.

Kullanılacak flip-flopun geçiş tablosundan yararlanılarak ardışıl devrenin durum tablosu düzenlenecektir.

D flip-flop geçiş tablosu:

simge	QQ ⁺	D
0	00	0
α	01	1
β	10	0
1	11	1

Bu tablo D flip-flopunun belli bir durum değişikliğini yapması için girişlerine uygulanması gereken değerleri gösterir. Değişik tipteki flip-flopların geçiş tabloları da farklıdır. Görüldüğü gibi D flip-flopunun tablosu basittir. D girişine verilmesi gereken değer sonraki durum değişkeninin değeri ile aynıdır.

Devrenin durum geçiş tablosundan yararlanılarak her durum değişkeninin (flipflopun) hangi geçişi yapacağı ayrı ayrı belirlenir.

$Q_1^+Q_0^+, Z$ $Q_1Q_0 \backslash X$	0	1
00	01,0	00,0
01	11,0	00,0
11	11,1	00,0
10	$\emptyset\emptyset, \emptyset$	$\emptyset\emptyset, \emptyset$

$Q_1Q_1^+ \backslash X$ Q_1Q_0	0	1
00	0	0
01	α	0
11	1	β
10	\emptyset	\emptyset

$Q_0Q_0^+ \backslash X$ Q_1Q_0	0	1
00	α	0
01	1	β
11	1	β
10	\emptyset	\emptyset

Durum geçiş tablolarına flip-flopun alması gereken giriş değerleri yerleştirilir.

$Q_1Q_1^+ \backslash X$ Q_1Q_0	0	1
00	0	0
01	α	0
11	1	β
10	\emptyset	\emptyset

$Q_0Q_0^+ \backslash X$ Q_1Q_0	0	1
00	α	0
01	1	β
11	1	β
10	\emptyset	\emptyset

simge	QQ ⁺	D
0	00	0
α	01	1
β	10	0
1	11	1

$D_1 \backslash X$ Q_1Q_0	0	1
00	0	0
01	1	0
11	1	0
10	\emptyset	\emptyset

$D_0 \backslash X$ Q_1Q_0	0	1
00	1	0
01	1	0
11	1	0
10	\emptyset	\emptyset

$$D_1 = X'Q_0$$

$$D_0 = X'$$

Böylece flip-flopları sürerek sonraki durumu belirleyen F fonksiyonu elde edilmiş oldu.

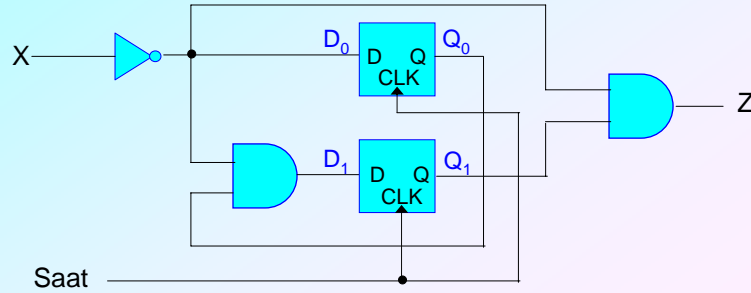
4. Çıkış tablosu kullanılarak çıkış fonksiyonu G belirlenir.

Z	X	0	1
Q ₁ Q ₀			
00		0	0
01		0	0
11		1	0
10		∅	∅

$$Z = X'Q_1$$

F ve G fonksiyonları tasarlanırken, dersin ilk bölümlerinde öğrenilen kombinezonsal devre tasarımı yöntemleri (asal çarpımlar, seçenekler tablosu) uygulanmalıdır. Bu örnekteki fonksiyonlar basit olduğundan indirgemeye gerek kalmamıştır.

5. Devrenin lojik elemanlar ile gerçekleştirilip çizilmesi.



Örnek: Aynı devrenin JK flip-flopları ile tasarlanması

Tasarım 3. maddeye kadar aynı şekilde yapılacaktır.

3. Bu örnekte pozitif kenar tetiklemeli JK tipi flip-floplar kullanılacaktır.

JK flip-flopu geçiş tablosu:

simge	QQ ⁺	J	K
0	00	0	∅
α	01	1	∅
β	10	∅	1
1	11	∅	0

D flip-flopları yerine JK flip-floplarının kullanılması genellikle daha basit lojik fonksiyonların elde edilmesini sağlar. Ancak bu örnekteki devreler zaten çok sade olduğundan daha fazla basitleşme sağlanmamaktadır.

Durum geçiş tablosundan durum değişkenlerinin geçişleri belirlenecektir.

Q ₁ ⁺ Q ₀ ⁺ Z	X	0	1
Q ₁ Q ₀			
00		01,0	00,0
01		11,0	00,0
11		11,1	00,0
10		∅∅,∅	∅∅,∅

Q ₁ Q ₁ ⁺	X	0	1
Q ₁ Q ₀			
00		0	0
01		α	0
11		1	β
10		∅	∅

Q ₀ Q ₀ ⁺	X	0	1
Q ₁ Q ₀			
00		α	0
01		1	β
11		1	β
10		∅	∅

Durum geiş tablolarına flip-flopun alması gereken giriş deęerleri yerleřtirilir.

$Q_1Q_0^+ \backslash X$	0	1
Q_1Q_0		
00	0	0
01	α	0
11	1	β
10	\emptyset	\emptyset

$Q_0Q_0^+ \backslash X$	0	1
Q_1Q_0		
00	α	0
01	1	β
11	1	β
10	\emptyset	\emptyset

JK flip-flopu geiş tablosu:

simge	QQ ⁺	J	K
0	00	0	\emptyset
α	01	1	\emptyset
β	10	\emptyset	1
1	11	\emptyset	0

$J_1 \backslash X$	0	1
Q_1Q_0		
00	0	0
01	1	0
11	\emptyset	\emptyset
10	\emptyset	\emptyset

$$J_1 = X'Q_0$$

$K_1 \backslash X$	0	1
Q_1Q_0		
00	\emptyset	\emptyset
01	\emptyset	\emptyset
11	0	1
10	\emptyset	\emptyset

$$K_1 = X$$

$J_0 \backslash X$	0	1
Q_1Q_0		
00	1	0
01	\emptyset	\emptyset
11	\emptyset	\emptyset
10	\emptyset	\emptyset

$$J_0 = X'$$

$K_0 \backslash X$	0	1
Q_1Q_0		
00	\emptyset	\emptyset
01	0	1
11	0	1
10	\emptyset	\emptyset

$$K_0 = X$$

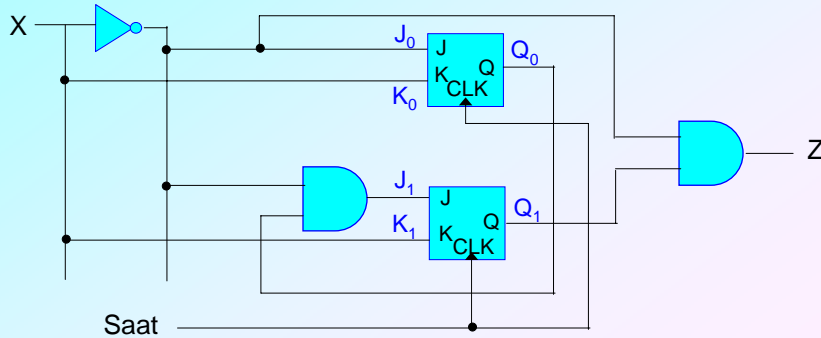
Böylece flip-flopları sürerek sonraki durumu belirleyen F fonksiyonu elde edilmiş oldu.

4. Çıkış tablosu kullanılarak çıkış fonksiyonu G belirlenir.

$Z \backslash X$	0	1
Q_1Q_0		
00	0	0
01	0	0
11	1	0
10	\emptyset	\emptyset

$$Z = X'Q_1$$

5. Devrenin lojik elemanlar ile gereklenip izilmesi.



Flip-flopların geiş tabloları:

Senkron ardışıl devre tasarımında gerekli olduğundan deęişik flip-flopların geiş tabloları ařaęıda verilmiřtir.

SR flip-flopu geiş tablosu:

simge	QQ+	S	R
0	00	0	∅
α	01	1	0
β	10	0	1
1	11	∅	0

JK flip-flopu geiş tablosu:

simge	QQ+	J	K
0	00	0	∅
α	01	1	∅
β	10	∅	1
1	11	∅	0

D flip-flopu geiş tablosu:

simge	QQ+	D
0	00	0
α	01	1
β	10	0
1	11	1

T flip-flopu geiş tablosu:

simge	QQ+	T
0	00	0
α	01	1
β	10	1
1	11	0

Senkron Devre Tasarım Örneęi 2: Moore Modeli

Moore modeline göre tasarım yapılırken de önceki örneklerde gösterilmiş olan ařamalardan geilir. Burada dikkat edilmesi gereken nokta,

- çıkışların sadece durumlara baęlı olduęu,
- bu nedenle de her duruma bir çıkış deęerinin karřı düřürüldüęüdür.

Problem:

İki giriři (X,Y) bir çıkışı (Z) olan senkron ardışıl bir devre tasarlanacaktır. Makinenin alıřmaya bařlamasından itibaren girişlerden gelen '1' deęerlerinin sayısı 4'ün katları ise devrenin çıkışı '1' deęerini alacaktır. Aksi durumda çıkış '0' olacaktır. Giriřten hiç '1' gelmemesi (sıfır tane) durumunda çıkış '1' olacaktır.

Çözüm:

Devrenin *modulo 4* iřlemine gerekleřtirmesi ve kalan 0 ise çıkışını '1' yapması istenmektedir. Bu makine 4 adet durum ile gereklenebilir:

1. Kalan 0: S0 Çıkış sadece devre bu durumdayken '1' olacaktır.
2. Kalan 1: S1
3. Kalan 2: S2
4. Kalan 3: S3

Durum/çıkış tablosu:

Anlam	S ⁺	S ⁻	XY				Z
			00	01	11	10	
Sıfır tane 1	S0	S0	S1	S2	S1	1	
Bir tane 1	S1	S1	S2	S3	S2	0	
İki tane 1	S2	S2	S3	S0	S3	0	
Üç tane 1	S3	S3	S0	S1	S0	0	

Durum Kodlaması:

S0: 00

S1: 01

S2: 11

S3: 10

Durum Değişkenleri:

Q1, Q0

Kodlanmış Durum/çıkış tablosu:

Q1+Q0 ⁺	Q1Q0	XY	XY				Z
			00	01	11	10	
00	00	00	01	11	01	1	
01	01	01	11	10	11	0	
11	11	11	10	00	10	0	
10	10	10	00	01	00	0	

D1	Q1Q0	XY	Q0·X'·Y			
			00	01	11	10
00	00	00	0	0	1	0
01	01	01	0	1	1	1
11	11	11	1	1	0	1
10	10	10	1	0	0	0

D0	Q1Q0	XY	Q1'·X·Y			
			00	01	11	10
00	00	00	0	1	1	1
01	01	01	1	1	0	1
11	11	11	1	0	0	0
10	10	10	0	0	1	0

D Flip-flopları ile tasarım yapıldığında $Q^+=D$ karakteristik fonksiyonundan yararlanılır.

$$D1 = Q0 \cdot X' \cdot Y + Q1' \cdot X \cdot Y + Q1 \cdot X' \cdot Y' + Q0 \cdot X \cdot Y'$$

$$D0 = Q1' \cdot X' \cdot Y + Q1' \cdot X \cdot Y' + Q0 \cdot X' \cdot Y' + Q0' \cdot X \cdot Y$$

$$Z = Q1' \cdot Q0'$$

Senkron Devrelerin Gerçeklenmesinde Veri Seçicilerin Kullanılması

Bir senkron ardışıl devre D flip-flopları ile tasarlanırsa, flip-flopların girişlerini süren fonksiyonun gerçekleşmesinde veri seçicilerin kullanılması daha uygun çözümlerin bulunmasını sağlayabilir.

Bu yöntemde,

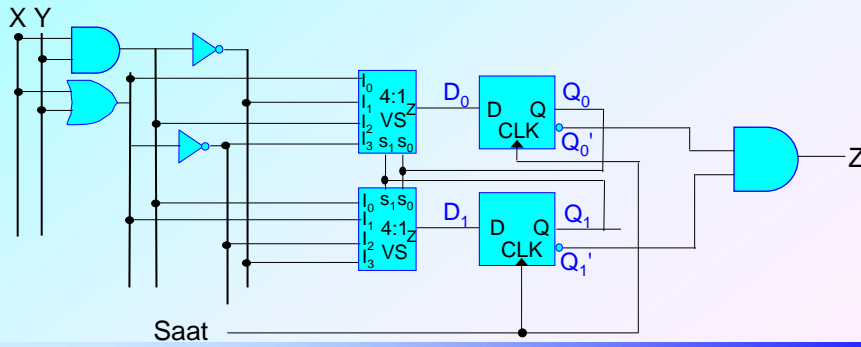
- Her D flip-flopunun girişi bir veri seçici ile sürülür.
- Veri seçicilerin seçme uçlarına, durum değişkenleri (flip-flopların çıkışları) bağlanır. Böylece bir veri seçici makinenin her durumu için girişlerinden birini seçmiş olur.
- Veri seçicinin veri girişlerine makinenin o durumdan sonra geçeceği durumun kodunu üretecek değerler bağlanır.

Veri seçicilerin veri girişlerine uygulanacak değerler durum tablosunun satırlarından yararlanılarak bulunur. Bir önceki örnekte gerçekleştirilen devre bir sonraki yansıda veri seçiciler ile yeniden gerçekleştirilmiştir.

D girişlerine gelmesi gereken değerler: (Önceki örnekten alınmıştır.)

D1	XY	Veri Seçiciye:
Q1 Q0	00 01 11 10	
00	0 0 1 0	$X \cdot Y$
01	0 1 1 1	$X + Y$
11	1 1 0 1	$(X \cdot Y)'$
10	1 0 0 0	$(X + Y)'$

D0	XY	Veri Seçiciye:
Q1 Q0	00 01 11 10	
00	0 1 1 1	$X + Y$
01	1 1 0 1	$(X \cdot Y)'$
11	1 0 0 0	$(X + Y)'$
10	0 0 1 0	$X \cdot Y$

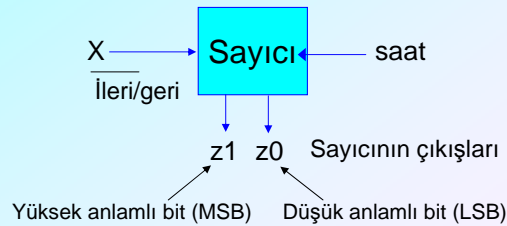


Sayıcı Tasarımı

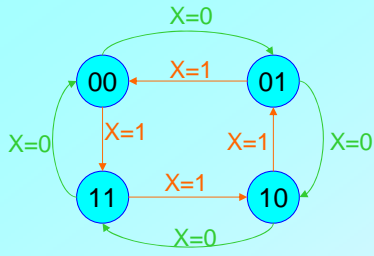
Belli bir sekansta sayım yapan sayıcılar senkron ardışıl devre olarak tasarlanırlar. Sayıcıların tasarlanmasında Moore modelinin kullanılması daha uygundur.

Sayıcının üreteceği her sayı, bir durum olarak kabul edilir. Çıktılar durum değişkenlerinden doğrudan elde edilir.

Örnek: Aşağıda blok diyagramı gösterilen, bir adet denetim girişine (X) sahip sayıcıyı tasarlayınız. Sayıcı, doğal ikili sayı sisteminde 0-1-2-3 düzeninde sayacaktır. 3'ten 0'a geri dönelecektir. X=0 olduğunda sayım ileriye doğru, X=1 olduğunda geriye doğru yapılacaktır.



Durum diyagramı:



Durum değişkenleri ile çıkışlar aynı değerlere sahiptir.

Sayıncının D flip-flopları ile tasarlanması:

Hatırlatma:
 $Q^+ = D$

D_1	$Q_1 Q_0$	X	0	1
	00		0	1
	01		1	0
	11		0	1
	10		1	0

$$D_1 = X' \cdot Q_1 \oplus Q_0 + X \cdot (Q_1 \oplus Q_0)'$$

D_0	$Q_1 Q_0$	X	0	1
	00		1	1
	01		0	0
	11		0	0
	10		1	1

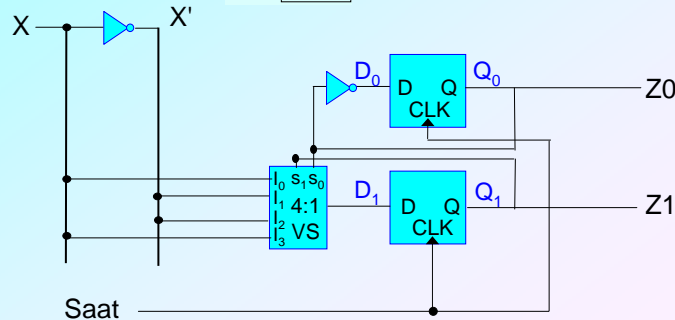
$$D_0 = Q_0'$$

Çıkışlar:
 $Z_0 = Q_0$
 $Z_1 = Q_1$

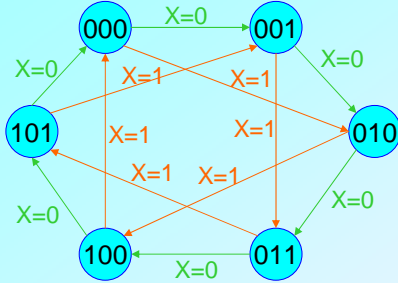
Sayıncının gerçekleştirilmesinde lojik bağlaçlar (VE, VEYA, YA DA) kullanılabileceği gibi veri seçiciler de tercih edilebilir.

Aşağıda D_0 girişi ifadesi basit olduğu için ($D_0 = Q_0'$) lojik kapı ile (bir adet tüleme) gerçekleştirilmiştir. D_1 girişini sürmek için veri seçici kullanılmıştır.

D_1	$Q_1 Q_0$	X	0	1	Veri Seçiciye:
	00		0	1	X
	01		1	0	X'
	11		0	1	X
	10		1	0	X'



Örnek: Doğal ikili sayı sisteminde 0-1-2-3-4-5 düzeninde sayan (6'ya sayıcı) ve bir adet denetim girişine (X) ait sayıcıyı tasarlayınız. X=0 olduğunda sayım birer adım ileriye doğru, X=1 olduğunda ikişer adım ileriye doğru yapılacaktır.



Karnaugh diyagramı olarak düzenlenmiş durum tablosu

Durum tablosu:

$Q_2+Q_1+Q_0$	X	
$Q_2Q_1Q_0$	0	1
000	001	010
001	010	011
010	011	100
011	100	101
100	101	000
101	000	001
110	ØØØ	ØØØ
111	ØØØ	ØØØ

$Q_2+Q_1+Q_0$	Q_0X			
Q_2Q_1	00	01	11	10
00	001	010	011	010
01	011	100	101	100
11	ØØØ	ØØØ	ØØØ	ØØØ
10	101	000	001	000

Bu örnekte tasarımı T flip-flopı kullanarak yapalım.

Hatırlatma:

T flip-flopı geçiş tablosu:

simge	QQ ⁺	T
0	00	0
α	01	1
β	10	1
1	11	0

$Q_2+Q_1+Q_0$	Q_0X			
Q_2Q_1	00	01	11	10
00	001	010	011	010
01	011	100	101	100
11	ØØØ	ØØØ	ØØØ	ØØØ
10	101	000	001	000

T ₂	Q_0X			
Q_2Q_1	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	Ø	Ø	Ø	Ø
10	0	1	1	1

T ₁	Q_0X			
Q_2Q_1	00	01	11	10
00	0	1	1	1
01	0	1	1	1
11	Ø	Ø	Ø	Ø
10	0	0	0	0

T ₀	Q_0X			
Q_2Q_1	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	Ø	Ø	Ø	Ø
10	1	0	0	1

$$T_2' = Q_0' \cdot X' + Q_2' \cdot Q_1'$$

$$T_2 = (Q_0 + X) \cdot (Q_2 + Q_1)$$

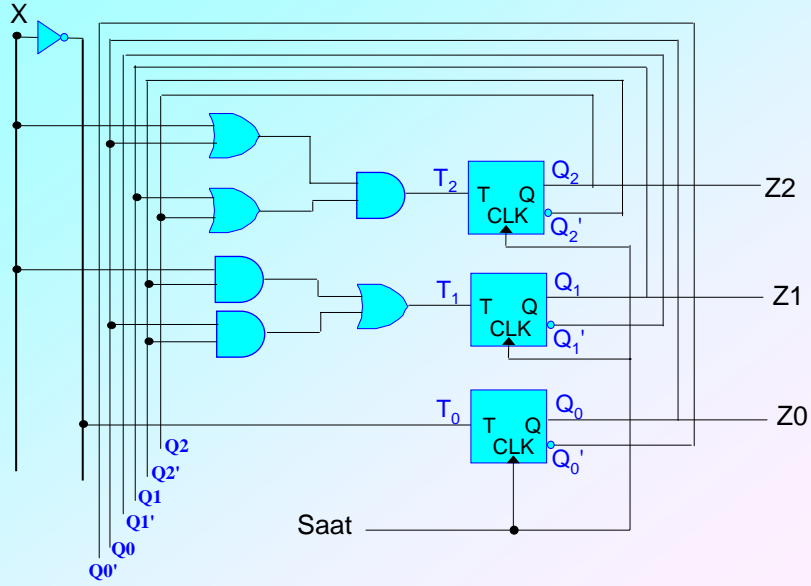
$$T_1 = Q_2' \cdot X + Q_2' \cdot Q_0$$

$$T_0 = X'$$

$$T_2 = (Q_0 + X) \cdot (Q_2 + Q_1)$$

$$T_1 = Q_2' \cdot X + Q_2' \cdot Q_0$$

$$T_0 = X'$$



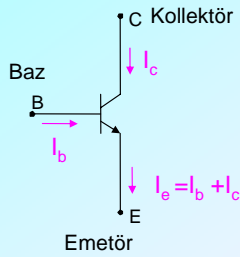
SAYISAL ELEMANLARIN İÇ YAPILARI

Sayısal tümdevrelerin gerçekleşmesinde çeşitli tipte tranzistorlar kullanılır. İlk olarak bipolar tipteki tranzistorlar tanıtılacaktır.

Bipolar Tranzistor:

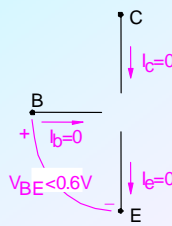
Sayısal tümdevrelerde tranzistorlar bir anahtar elemanı olarak kullanılır. Bu nedenle tranzistorlar ya iletimde (anahtar akım iletiyor) ya da kesimde (anahtar akım iletmiyor). Bipolar tranzistorun iletimde olduğu duruma tranzistor doymada denir.

Bipolar Tranzistor



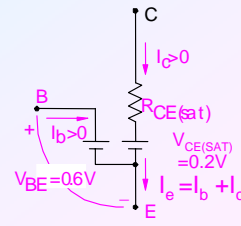
Tranzistor kesimde

$$V_{BE} < 0.6V$$

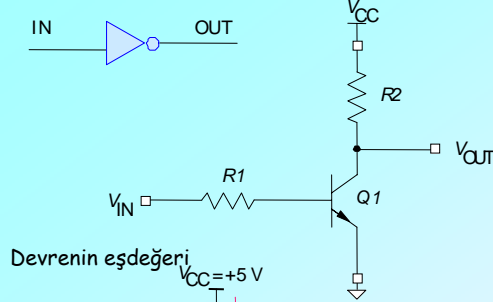
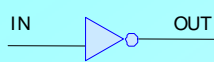


Tranzistor doymada

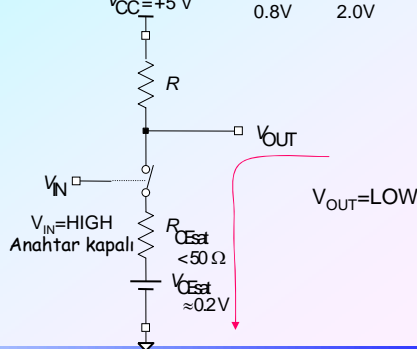
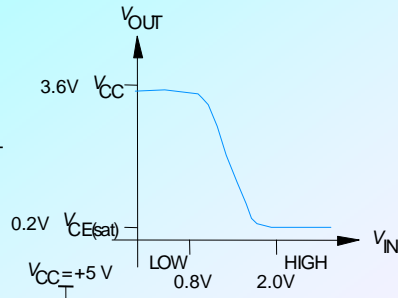
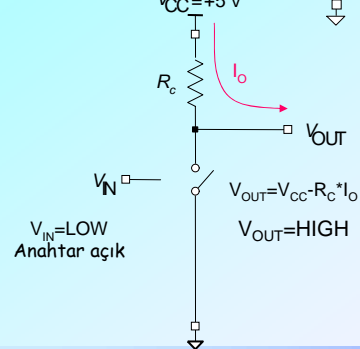
$$V_{BE} > 0.6V$$



Bir Tümlenme kapısının tranzistor ve dirençle gerçekleştirilmesi



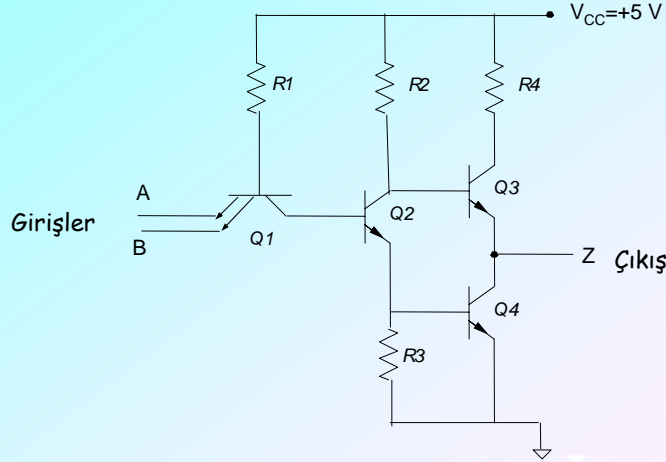
Devrenin eşdeğeri



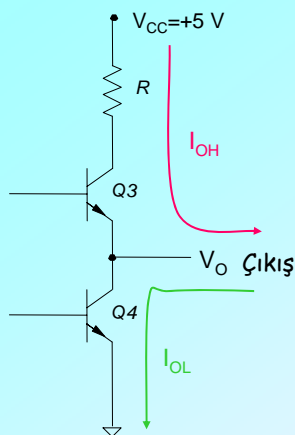
TTL (Tranzistor- Tranzistor) Lojiği Ailesi

Bipolar tranzistorlar ve dirençler kullanılır.

Örnek: İki girişli TVE bağlacı



TTL Çıkış Katının Çalışması



Çıkışın lojik 0 (LOW) olması için Q_4 iletimde, Q_3 kesimde olur. Bu durumda bağlacın çıkışından içeriye doğru I_{OL} akımı akar.

$$V_{OL} = V_{CE(Q4)} + I_{OL} \cdot R_{Q4}$$

Çıkışın lojik 1 (HIGH) olması için Q_3 iletimde, Q_4 kesimde olur. Bu durumda bağlacın çıkışından dışarıya doğru I_{OH} akımı akar.

$$V_{OH} = V_{CC} - (V_{CE(Q3)} + I_{OH} \cdot (R + R_{Q3}))$$

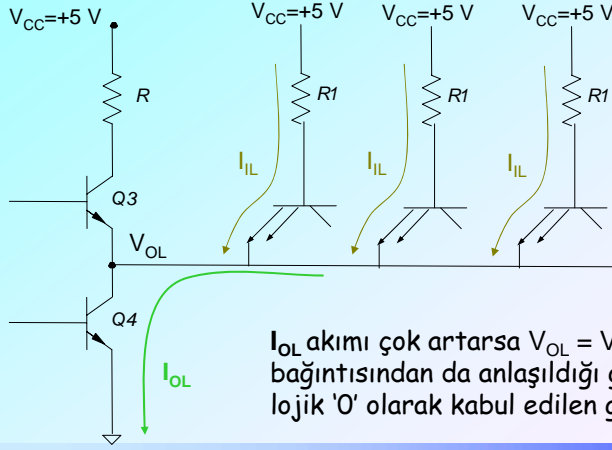
Hem Q_3 hem de Q_4 kesimde olursa **çıkış yüksek empedans (high Z) konumunda** olur. Bu durumda bağlacın çıkışından akım akmaz ve bağlaç bağlandığı hattan yalıtılmış olur.

TTL elemanlar için $V_{OL(MAX)} = 0.4V$ $V_{OH(MIN)} = 2.5V$

TTL ailesinde değişik tipte elemanlar vardır (LS, ALS, L, F gibi). Bunların her biri için akım değerleri farklıdır. Bu değerler kataloglardan öğrenilebilir.

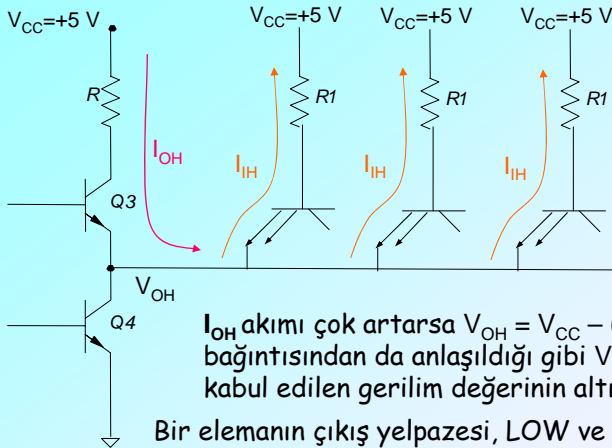
TTL Çıkış Yelpazesi (Fan Out)

Bir lojik bağlacın çıkışı diğer lojik bağlaçların girişlerine bağlanmaktadır. Akım olaylarından dolayı bir elemanın çıkışına bağlanabilecek eleman sayısı sınırlıdır. TTL elemanların girişleri tranzistorların emetörlerinden oluşmaktadır.

Çıkış LOW olduğunda:

Girişi LOW olan elemanların girişinden dışarıya doğru I_{IL} akımı akar. Bu akımların toplamı diğer elemanın çıkışı tarafından yutulmaktadır.
 $I_{OL} < \sum I_{IL}$

I_{OL} akımı çok artarsa $V_{OL} = V_{CE(Q4)} + I_{OL} * R_{Q4}$ bağıntısından da anlaşıldığı gibi V_{OL} de artar ve lojik '0' olarak kabul edilen gerilim değeri aşılır.

Çıkış HIGH olduğunda:

Girişi HIGH olan elemanların girişinden içeriye doğru I_{IH} akımı akar. Bu akımların toplamı diğer elemanın çıkışından çekilecektir.
 $I_{OH} < \sum I_{IH}$

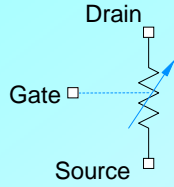
I_{OH} akımı çok artarsa $V_{OH} = V_{CC} - (V_{CE(Q3)} + I_{OH} * (R + R_{Q3}))$ bağıntısından da anlaşıldığı gibi V_{OH} azalır ve lojik '1' olarak kabul edilen gerilim değerinin altına düşer.

Bir elemanın çıkış yelpazesi, LOW ve HIGH konumları için hesaplanan değerlerden küçük olana eşittir.

TTL elemanlara ait V_{OH} , V_{OL} , V_{IH} , V_{IL} , I_{OH} , I_{OL} , I_{IH} , I_{IL} gibi değerler bu elemanların kataloglarında yer almaktadır.

CMOS (Complementary MOS) Lojîği Ailesi

MOS FET (*Metal-Oxide Semiconductor Field-Effect Transistor*) kullanılır. Lojik bağlaçlarda kullanılan MOS tranzistorlar birer ayarlı direnç gibi düşünülebilir.

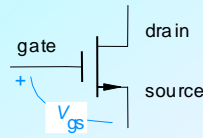


Gate-Source (V_{GS}) arasına uygulanan gerilime göre Drain Source (R_{DS}) arasındaki direnç değişir.
Tranzistor tıkamadayken $R_{DS} \geq 1M\Omega$
Tranzistor iletimdeyken $R_{DS} \leq 100\Omega$

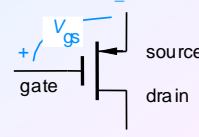
İki tip MOS tranzistor vardır.

a) n kanallı MOS: NMOS.

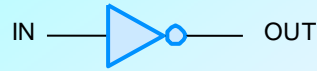
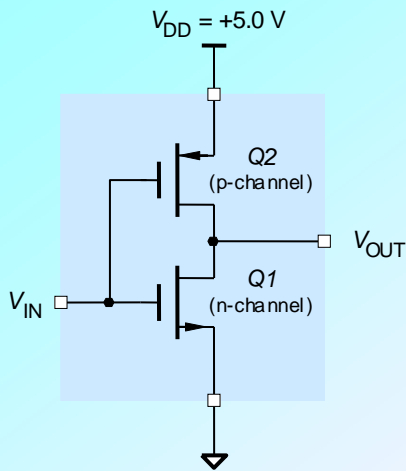
b) p kanallı MOS: PMOS.



V_{GS} arttıkça R_{DS} direnci azalır.
Normalde: $V_{GS} \geq 0V$

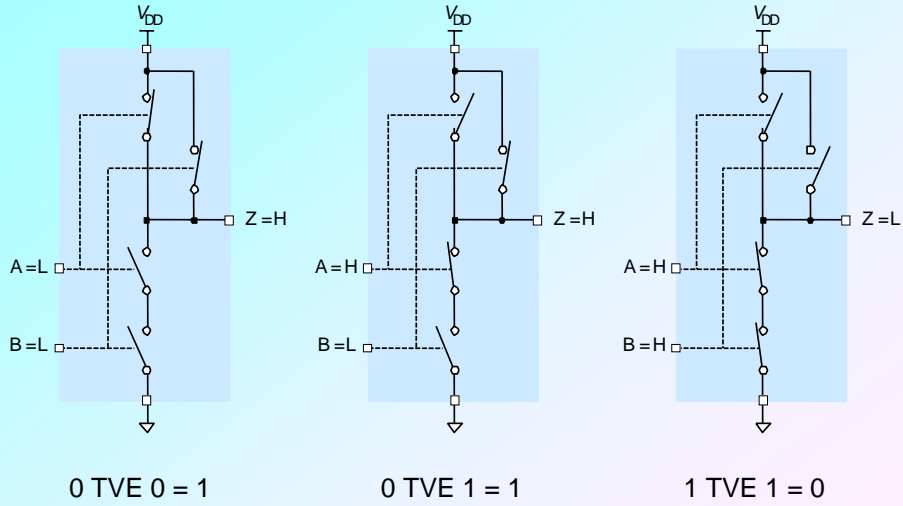


V_{GS} azaldıkça R_{DS} direnci azalır.
Normalde: $V_{GS} \leq 0V$

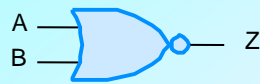
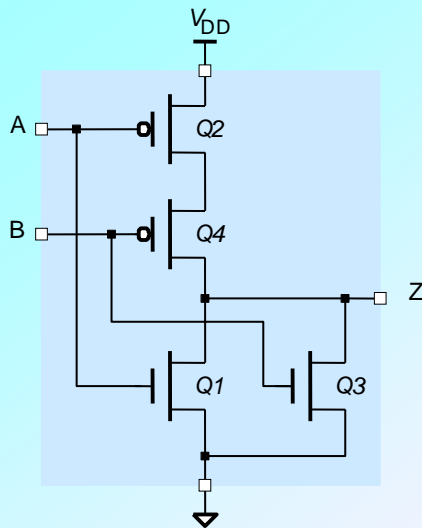
CMOS Tümeleme Bağlaç

V_{IN}	Q1	Q2	V_{OUT}
0.0 (L)	off	on	5.0 (H)
5.0 (H)	on	off	0.0 (L)

CMOS TVE (NAND) Bağlacı Anahtar Modeli



CMOS TVEYA (NOR) Bağlacı



A	B	Q1	Q2	Q3	Q4	Z
L	L	off	on	off	on	H
L	H	off	on	on	off	L
H	L	on	off	off	on	L
H	H	on	off	on	off	L