EEM212 - SAYISAL DEVRE TASARIMI DERS NOTLARI

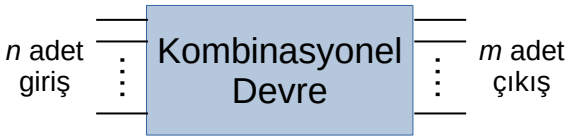
DERS NOTU 4: KOMBİNASYONEL DEVRE TASARIMI

Dr. İsmail Öztürk *

<ismail.ozturk@amasya.edu.tr>

İçindekiler

*Amasya Üniversitesi Teknoloji Fakültesi EEM Bölümü
Daha fazla bilgi için: https://iozturk.com



Şekil 1: Kombinasyonel (Bileşimsel) devrelerin blok diyagramı.

1 Kombinasyonel Devreler

Dijital devreler

1. Kombinasyonel Devreler [1]
2. Ardışıl Devreler [2]

olmak üzere iki gruba ayrılır.

Kombinasyonel devrelerin blok diyagramı Şekil 1'de görüldüğü gibidir. Şekilden görebileceğiniz üzere bir kombinasyonel devrenin genel olarak n-bitlik girişi ve m-bitlik çıkışı vardır. Giriş ve çıkışlar arasında Tanım 1.1 'daki gibi bir ilişki vardır:

Tanım 1.1: Kombinasyonel Devre

Kombinasyonel devreler, çıkışları herhangi bir anda sadece girişlerin mevcut durumlarının kombinasyonlarına bağlı olan dijital devrelerdir.

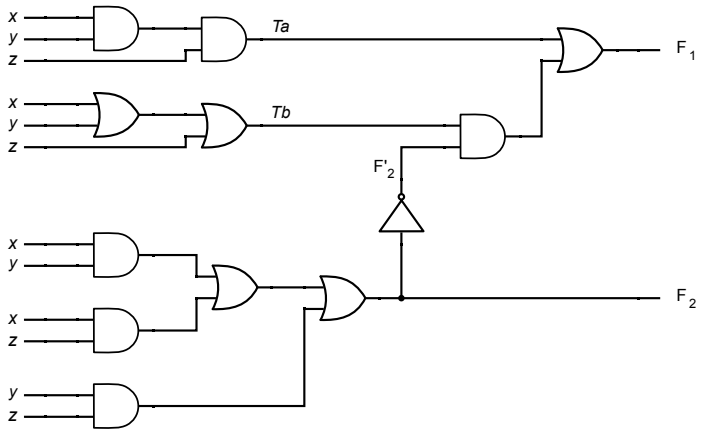
Kombinasyonel devrenin çıkışlarının sadece girişin mevcut değerlerine bağlı olması demek, devrede herhangi bir hafıza elemanı kullanmadığımız anlamına gelir. Yani çıkışın değerlerini belirlemek istediğimizde girişin o anki değerlerine bakmamız gerekir. Giriş değerlerinde yapılacak bir değişiklik (teorik olarak) çıkışa anında yansıyacaktır.

Teorik olarak böyle kabul edilse de pratikte girişte yapılacak bir değişikliğin çıkışa yansıması için bir süre gerekir. Buna propagasyon (yayılma) gecikmesi adı verilir.

m-bitlik çıkışlardan her biri n-bitlik girişlerin Boole işlemleri kullanılarak elde edilen kombinasyonlarına göre belirlenir. Bu bakımdan, her bir çıkışı bir Boole fonksiyonu ile ifade etmek mümkündür. Daha önce Boole fonksiyonlarının nasıl sadeleştirileceğini ve nasıl dijital devre olarak gerçekleştirilebileceklerini gördüğümüz için kombinasyonel devreleri en az eleman kullanarak tasarlayıp kurmamız son derece kolay olacaktır.

Daha önce incelediğimiz tüm Boole fonksiyonları ve kurmuş olduğumuz dijital devreler, tek çıkışlı kombinasyonel devre örnekleridir. Kombinasyonel devreler hafıza gerektirmeyen tüm dijital uygulamalarda kullanılmaktadır. Aritmetik işlemleri yapmada ve veri aktarımında sıklıkla kombinasyonel devreler kullanılır. Toplayıcı, çarpıcı, karşılaştırıcı devreler aritmetik işlemlerde kullanılan kombinasyonel devre örnekleridir. Veri aktarımında ise kodlayıcı (encoder), kod çözücü (decoder), veri seçiciler (multiplexer) ve veri dağıtıcılar (demultiplexer) adı verilen kombinasyonel devreler sıklıkla kullanılmaktadır.

[1] ing. *Combinational Circuits*, tr. *Bileşimsel Devreler* olarak da adlandırılır.
[2] ing. *Sequential Circuits*



Şekil 2: Kombinasyonel Devre Örneği

1.1 Kombinasyonel Devre Analizi

Kombinasyonel devreler herhangi bir hafıza elemanı içermedikleri için çıkışlar ve girişler arasında herhangi bir geribesleme bağlantısı da bulunmaz. Bu nedenle, verilen bir kombinasyonel devrenin analizini yapmak çok kolaydır. Tek yapmanız gereken her bir çıkışa karşılık gelen Boole fonksiyonunu bulmak; sonra da bu fonksiyonlara karşılık gelen doğruluk tablolarını belirlemektir. Bunun için devre üzerinde istediğiniz yerleri etiketleyerek Boole ilişkilerini bu etiketler üzerinden hesaplamak işinizi kolaylaştıracaktır. Örnek olarak Şekil 2'deki devreyi inceleyelim.

Bu devreye baktığımızda herhangi bir hafıza elemanı olmamasından devrenin kombinasyonel devre olduğunu anlarız. Girişlere baktığımızda x, y, z ile etiketlenen üç giriş olduğunu görürüz. Çıkışlara baktığımızda ise F_1 ve F_2 ile etiketlenen iki tane çıkış olduğu görülür. Buna göre, **devre üç girişli ve iki çıkışlı bir kombinasyonel devredir**. Devre analizini kolaylaştırmak adına T_a, T_b ve F_2' etiketleri şekilde görüldüğü gibi kullanılmıştır. Etiketleri kullandığımızda F_1 çıkışını hesaplamak çok kolaydır:

$$F_1 = T_a + T_b F_2'$$

F_1çıkışının asıl değerini bulmak için etiketlerin değerini bulmamız lazım. Bu ilişkileri kurmak gayet kolaydır: $T_a = xyz$ (burada üç girişli bir VE kapısı da kullanılabilirdi), $T_b = x + y + z$ (burada üç girişli bir VEYA kapısı da kullanılabilirdi). F_2 değerini bulmak için arada bir etiket daha kullanabilirdi. Fakat, etiket kullanmadan da

$$F_2 = xy + xz + yz$$

olduğunu görmek kolaydır. $F_2' = (xy + xz + yz)'$ olacağına göre, F_1 çıkışının değeri aşağıdaki gibi bulunur:

$$F_1 = xyz + (x + y + z)(xy + xz + yz)'$$

Bu denklemden görüldüğü üzere devre çarpımların toplamı veya toplamların çarpımı şeklinde standart bir formda kurulmamıştır (Ders Notu 2: Devre Gerçekleştirimi kısmına bakınız). Artık bulduğumuz fonksiyonlara karşılık gelen doğruluk tablolarını doldurabiliriz. Doğruluk tablolarını doldururken VEYA işleminde sadece tek bir 1 değerinin sonucu 1 yapmaya yeteceği; VE işleminde ise sadece tek bir 0 değerinin sonucu 0 yapmaya yeteceği gibi özelliklere dikkat ederseniz doğruluk tablosunu hızlıca doldurabilirsiniz. Mesela, $F_2 = xy + xz + yz$ ifadesinin 1 olması için xy, xz, yz terimlerinden en az biri 1 olmalı; bunun içinse x, y, z değişkenlerinden en az iki tanesinin 1 olması gerekir.

Bunun dışında, $T_a = xyz$, $T_b = x+y+z$, $F_2 = xy+xz+yz$, $F_1 = T_a+T_bF_2'$ gibi basit ilişkileri kullanarak tabloyu doldurmamız daha kolay olacaktır. Buna göre, verilen devreye ait doğruluk tablosu aşağıdaki gibi elde edilir:

x	y	z	F_2	F_2'	T_a	T_b	F_1
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	1	0	1	1
0	1	1	1	0	0	1	0
1	0	0	0	1	0	1	1
1	0	1	1	0	0	1	0
1	1	0	1	0	0	1	0
1	1	1	1	0	1	1	1

Tabi doğruluk tablosu oluşturmak için ara işlemleri doğruluk tablosunda kullanmak şart değildir. Doğrudan $F_1 = xyz + (x + y + z)(xy + xz + yz)'$ ve $F_2 = xy + xz + yz$ denklemlerini kullanarak da doğruluk tablosunu aşağıdaki gibi oluşturabilirsiniz:

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

1.2 Kombinasyonel Devre Tasarımı

Kombinasyonel bir devre tasarlamak için takip etmemiz gereken adımlar aşağıdaki gibi sıralanmıştır:

1. Kaç tane giriş, kaç tane çıkışa ihtiyacımız olduğunu belirleriz. Ardından bu giriş ve çıkışlara semboller atarız.
2. Girişlere karşılık çıkışların ne olması gerektiğini doğruluk tablosu oluşturarak belirleriz.
3. Her bir çıkışa karşılık gelen Boole fonksiyonlarını en sade halleriyle belirleriz.
4. Boole fonksiyonlarına karşılık gelen devreyi kurar ve devrenin doğruluğunu simülasyon yaparak kontrol ederiz.

Şu ana kadar edinmiş olduğumuz bilgi birikimi ile kombinasyonel devreleri tasarlayıp kurmamız son derece kolay olacaktır. Örnek olarak aşağıdaki devreyi tasarlayıp kuralım:

Örnek 1.1:

Bir trafik lambası doğru çalışmadığı zaman uyarı veren dijital devreyi tasarlayınız.

Öncelikle giriş-çıkış sayılarını ve sembollerini belirlememiz lazım. Trafik lambasında üç ışığın doğru çalışıp çalışmadığını kontrol ettiğimiz için devre üç girişli olmalıdır. Kırmızı, sarı ve yeşil ışığı sırasıyla K, S ve Y değişkenleri ile ifade edelim. Işıkların yanma durumu 1, yanmama durumu da 0 olsun. Hatalı yanma kombinasyonlarına karşılık çıkışta tek bir uyarı sinyali göndereceğimiz için devre tek çıkışlı olmalıdır. Bu çıkışı H ile etiketlersek (hata anlamında), çıkışımız $H(K, S, Y)$ şeklinde bir fonksiyon olacaktır. Hata varsa $H = 1$, hata yoksa $H = 0$ olsun.

$H(K, S, Y)$ fonksiyonunun daha önce gördüğümüz $F(x, y, z)$ fonksiyonlarından hiçbir farkı yok. Sadece kullanılan semboller farklı!

Trafik lambalarının normal çalışmasında aşağıdaki durumlar ortaya çıkabilir:

- Sadece yeşil ışık yanabilir ($K = 0$, $S = 0$, $Y = 1$)
- Sadece sarı ışık yanabilir ($K = 0$, $S = 1$, $Y = 0$)
- Sadece kırmızı ışık yanabilir ($K = 1$, $S = 0$, $Y = 0$)
- Hem kırmızı hem sarı ışık yanabilir ($K = 1$, $S = 1$, $Y = 0$)

Bu durumlar için devre herhangi bir uyarı vermemelidir ($H = 0$). Bu listelenenler dışındaki durumlar için devre hata uyarısı vermelidir ($H = 1$). Buna göre, tasarlayacağımız devrenin doğruluk tablosu aşağıdaki gibi olacaktır:

K	S	Y	H
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Buradan devrenin kanonik formunu

$$H(K,S,Y) = \sum(0,3,5,7) = \prod(1,2,4,6)$$

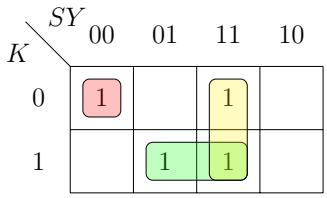
olarak buluruz. Bu aşamadan sonra devreyi nasıl kurmak istediğimizi kararlaştırmamız gerekir. Devreyi miniterimleri kullanarak çarpımların toplamı formunda kurabileceğimiz gibi; maksiterimleri kullanarak toplamların çarpımı formunda da kurabiliriz. Örnek olması açısından burada devreyi her iki formda da oluşturacağız.

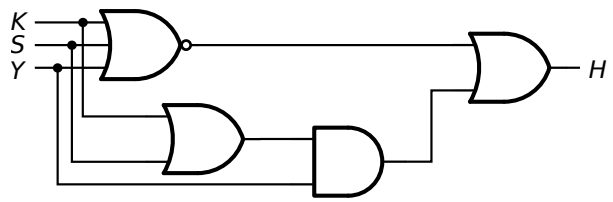
Burada unutmamanız gereken husus aynı trafik uyarı devresini doğrudan kanonik formları kullanarak da kurabileceğimizdir. Mesela, $\sum(0,3,5,7)$ kanonik ifadesi

$$\sum(0,3,5,7) = K'S'Y' + K'SY + KS'Y + KSY$$

şeklinde ifade edilir ve mantık kapıları kullanarak bu devreyi nasıl kurabileceğimizi daha önce gördük. Ancak, bu devreyi doğrudan kurmak çok fazla sayıda mantık kapısı kullanmamıza neden olacaktır. Bu ise maliyeti ve güç tüketimini arttıracaktır. Dolayısıyla, kanonik formu doğrudan kurmak yerine, Karnaugh haritası ile yukarıdaki Boole ifadesini en sade haline getirdikten sonra elde ettiğimiz ifadeyi kurmamız gerekir.

Öncelikle, miniterimleri kullanarak devreyi çarpımların toplamı formunda kuralım. Bunun için $H(K,S,Y) = \sum(0,3,5,7)$ eşitliğini kullanarak Karnaugh haritaları ile sadeleştirme yaparız. Bu eşitlik için Karnaugh haritası aşağıdaki gibi kurulur:





Şekil 3: Örnek 1.1 'e ait devrenin miniterim sadeleştirmesi ile gerçekleştirimi.

Yukarıda görmüş olduğunuz seçimler için H fonksiyonunun en sade halini

$$H(K,S,Y) = K'S'Y' + KY + SY$$

olarak elde ederiz. Bu devreyi "Ders Notu 2: Devre Gerçekleştirimi" kısmında gösterdiğimiz üzere çarpımların toplamı formunda kurabiliriz. Fakat, yine aynı notlarda anlattığımız üzere VE, VEYA, DEĞİL kapıları dışındaki mantık kapılarını kullanmak istediğimizde devreyi serbest formda da ifade edip kurabiliriz. Mesela, De Morgan kuralını uygularsak yukarıdaki devreyi VEYADEĞİL kapısı ile aşağıdaki serbest formda da kurabiliriz:

$$H(K,S,Y) = (K+S+Y)' + Y(K+S)$$

Bu ikinci form için devre gerçekleştirimi Şekil 3'deki gibi olacaktır.

***

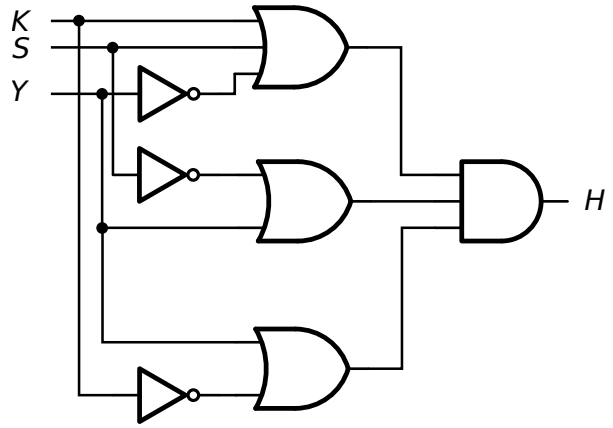
Şimdi de aynı devreyi maksiterimleri kullanarak toplamların çarpımı formunda kuralım. Bunun için, $H(K,S,Y) = \prod(1,2,4,6)$ eşitliğini kullanarak Karnaugh haritaları ile sadeleştirme yaparız. Maksiterimler için Karnaugh haritası aşağıdaki gibi elde edilir:

K \ SY	00	01	11	10
0		0		0
1	0			0

Yukarıda görmüş olduğunuz seçimler için H fonksiyonunun en sade halini

$$H(K,S,Y) = (K+S+Y')(S'+Y)(K'+Y)$$

olarak elde ederiz. Bu devreyi "Ders Notu 2: Devre Gerçekleştirimi" kısmında gösterdiğimiz üzere çarpımların toplamı formunda kurduğumuzda Şekil 4 'de görülen devreyi elde ederiz. Bu şekilden görülebileceği üzere dijital devre gerçekleştirimi standart toplamların çarpımı gerçekleştirimidir (VEYA katını takip eden VE katı).



Şekil 4: Örnek 1.1 'e ait devrenin maksiterim sadeleştirmesi ile gerçekleştirimi.

***

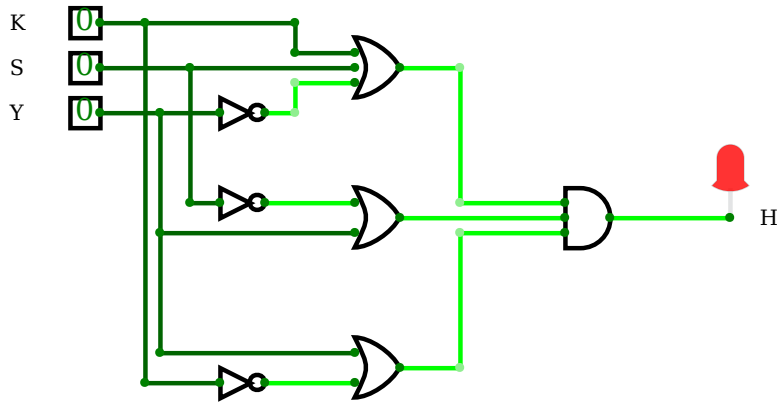
Yukarıdaki devreleri pratik olarak kurmadan önce doğru devreyi tasarlayıp tasarlamadığınızı belirlemeniz gerekir. Bunun için devrelerin simülasyonunu yapmamız gerekir. Proteus, Multisim gibi programlar aşina olduğunuz devre simülasyon araçlarıdır. Simülasyon için bu tür programları kullanabilirsiniz. Fakat günümüzde web tarayıcınız üzerinde "online" çalıştırabileceğiniz devre simülasyonu araçları yaygınlık kazanmaktadır. Bu tür online uygulamalar ile simülasyonları çok hızlı bir şekilde yapabilmeniz mümkün olmaktadır. Kuruluma gerek olmaksızın herhangi bir bilgisayardan erişilebilmeleri ve ücretsiz olmaları bu tür online simülatörlerin diğer avantajlarıdır.

CircuitVerse [3] uygulaması dijital devrelerin simülasyonunu yapabileceğiniz bu türden bir online simülatördür. CircuitVerse ile Şekil 4'deki devrenin simülasyonu 2-3 dk. içerisinde yapılabilmektedir. Yapılan simülasyonun ekran görüntüsü Şekil'deki gibidir. Tasarlamış olduğunuz dijital devreleri hızlı bir şekilde denemek istediğinizde CircuitVerse gibi online araçları kullanabilirsiniz.

1.3 Kombinasyonel Devre Çeşitleri

Giriş kısmında da bahsetmiş olduğumuz üzere kodlayıcı (encoder), kod çözücü (decoder), veri seçici (multiplexer) ve veri dağıtıcı (demultiplexer) adı verilen devreler dijital devre tasarımında sıklıkla kullanılan kombinasyonel devrelerdir. Bu kısımda bu devrelerin nasıl çalıştıklarını ve bu devrelerle diğer kombinasyonel devrelerin nasıl tasarlanabileceğini inceleyeceğiz.

[3] https://circuitverse.org/simulator

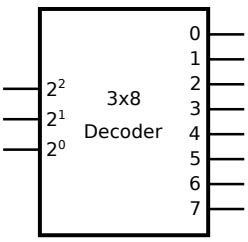


Şekil 5: Şekil 4'deki devrenin CircuitVerse simülasyonu.

1.3.1 Kod Çözücü (Decoder)

Daha önce görmüş olduğumuz üzere n-bitlik bir veri toplamda 2^n farklı durumu temsil etmek için kullanılabilir. Bunu "n-bitlik veri 2^n farklı durumu kodlar" şeklinde de ifade edebiliriz. Bu tanımdan, kod çözücü devrenin n bitlik bir girişi (kod) alarak, bu giriş kombinasyonuna (koda) karşılık gelen 2^n durumdan birini aktive eden (yani giriş kodunu çözen) bir kombinasyonel devre olacağını tahmin etmeniz zor olmayacaktır.

Genel olarak, kod çözücüler n-girişli ve m-çıkışlı olarak adlandırılır. Burada, m değeri 2^n olmak zorunda değildir. Çünkü kod çözücü devreler 2^n olası durumun hepsini kullanmayabilir. Buna göre, $m \leq 2^n$ olmak şartıyla $n \times m$ kod çözücü (decoder) dediğimizde girişin n-bit, çıkışın ise m-bit olduğunu anlarız. Mesela, 2×4, 3×5 ve 4×10 şeklinde kod çözücüler oluşturmak mümkündür. Fakat, 2×5, 3×10, 4×20 gibi kod çözücüleri çıkışlar 2^n değerinden daha büyük olduğu için oluşturmak mümkün değildir. Örnek olarak 3×8 bir kod çözücünün blok diyagramı aşağıdaki gibi olacaktır:

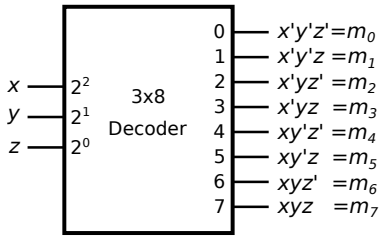


Burada 2^2 giriş biti MSB anlamına gelmekte; 2^0 biti ise LSB anlamına gelmektedir. Diğer 8 çıkış biti ise girişteki 3-bitlik giriş kombinasyonuna göre aktive edilecek bit-

lerdir. **Bu 8-bitlik çıkış girişteki 3-bitlik verinin (kodun) alabileceği $2^3 = 8$ durumdan her birini temsil ettiği için, aynı anda çıkış bitlerinden sadece bir tanesi aktif olabilir**. Örneğin, giriş $(000)_2$ ise sadece 0 çıkışı aktif olmalı; eğer giriş $(111)_2$ ise sadece 7 çıkışı aktif olmalı. Girişleri x, y, z ile (x MSB ve z LSB olmak üzere); çıkışları ise D_0, D_1, ..., D_7 ile etiketlersek, yukarıdaki 3×8 kod çözücünün doğruluk tablosu aşağıdaki gibi elde edilecektir:

x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Biz bu doğruluk tablosunu daha önce görmüştük. Kanonik miniterimlerin toplamı formunu hatırlarsanız, bu formdayken giriş kombinasyonlarına karşılık miniterimlerden sadece bir tanesi 1 olurken diğerleri 0 oluyordu. Üç değişken için miniterimlerin doğruluk tablosu yukarıdaki 3×8 kod çözücünün doğruluk tablosu ile birebir aynıdır. Yani kod çözücünün her bir çıkışı aslında bir miniterime karşılık gelmektedir ($D_i = m_i$). Bunu aşağıdaki gibi ifade edebiliriz:



Buna göre, 3×8 kod çözücü devreyi mantık kapılarıyla oluşturmak istediğimizde de yapmamız gereken aşikardır. Çünkü her bir çıkışın Boole ifadesini artık biliyoruz: $D_0 = x'y'z'$, $D_1 = x'y'z$, ..., $D_7 = xyz$.

Bu özellikleri nedeniyle kod çözücüler "programlanabilir lojik eleman" olarak kullanılır. Yani aynı kod çözücü ile farklı farklı dijital devreleri hızlıca gerçekleştirebiliriz.

Kod çözücülerin çıkışlarının her birinin bir miniterime karşılık gelmesi nedeniyle, istediğimiz herhangi bir kombinasyonel devreyi kod çözücü kullanarak da oluşturabiliriz: Tüm kombinasyonel devrelerin çıkışları bir Boole fonksiyonuna karşılık gelir ve Boole fonksiyonları da kanonik miniterimlerin toplamı formunda ifade edilebilir. Miniterimlerin toplamı demek miniterimleri VEYA işlemine sokmak demektir. Dolayısıyla, kod çözücünün çıkışlarını VEYA işlemine alarak da o fonksiyonları gerçekleştirmiş oluruz.

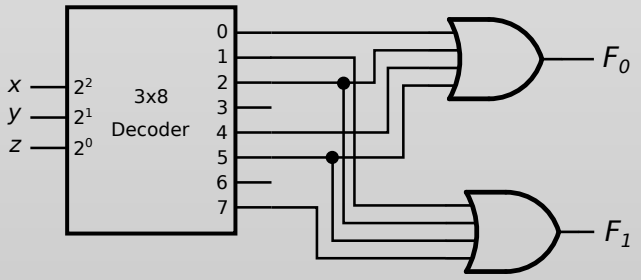
Örnek 1.2:

Aşağıda doğruluk tablosu verilmiş olan üç girişli ve iki çıkışlı kombinasyonel devreyi kod çözücü kullanarak gerçekleştiriniz.

x	y	z	F_0	F_1
0	0	0	1	0
0	0	1	0	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	0	1

—

Doğruluk tablosundan çıkışların Boole karşılıkları $F_0 = \sum(0, 2, 4, 5)$ ve $F_1 = \sum(1, 2, 5, 7)$ olarak elde edilir. Bu devreyi kod çözücü ile kurabilmek için yapmamız gereken tek şey bu miniterimlere karşılık gelen kod çözücü çıkışlarını VEYA işlemine sokmaktır. Buna göre, devre aşağıdaki gibi kurulacaktır:



1.3.2 Kodlayıcı (Encoder)

Kod çözücü (kodlayıcı) devrenin tam tersi şekilde çalışır. Yani 2^n girişe karşılık n-bitlik çıkış verir. Buna göre, 4×2, 8×3, 16×4 şeklinde kodlayıcı devreler kurmak mümkündür. Kod çözücü örneğinin simetriği olmasından dolayı 8×3 kodlayıcıyı inceleyelim:

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Görüldüğü üzere artık girişler D_0, D_1, ..., D_7 olurken çıkışlar x, y, z olmaktadır. Yukarıdaki doğruluk tablosuna bakarak kodlayıcı devreyi mantık kapıları ile gerçekleştirmek son derece kolaydır. Tabloya baktığımızda x çıkışı sadece D_4, D_5, D_6 veya D_7 girişleri 1 olduğunda 1 olmaktadır. Buna göre, $x = D_4 + D_5 + D_6 + D_7$ olmalıdır. Diğer çıkışlar da benzer olarak bulunduğunda çıkış ilişkileri aşağıdaki gibi topluca yazılabilir:

$$x = D_4 + D_5 + D_6 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$z = D_1 + D_3 + D_5 + D_7$$

Bu ilişkileri kullanarak 8×3 kodlayıcıyı mantık kapıları ile kolaylıkla kurabilirsiniz.

Kodlayıcıları kullanırken göz önünde bulundurulması gereken en önemli husus, girişlerin aynı anda sadece bir tanesinin 1 olması gerekliliğidir. Eğer tüm girişler sıfır olursa veya birden fazla giriş aynı anda 1 olursa, kodlayıcı doğru çalışmayacaktır. Dijital bir sistem içerisinde doğru çalışıp çalışmadığı belirsiz olan devreler kullanmak sistemde beklenmedik hatalara veya davranışlara neden olabileceği için mühendisler **öncelikli kodlayıcı** [4] adı verilen bir kodlayıcı devresi geliştirmişlerdir.

Bu yeni tür kodlayıcı devresine girişin geçerli olup olmadığını belirten ve V ile etiketlenen (ing. *valid* anlamında) bir çıkış biti eklenmiştir. Eğer girişteki tüm değerler 0 olursa V değeri 0 olmakta; diğer durumlarda ise V değeri 1 olmaktadır. V çıkışının 0 olması giriş değerlerinin uygun olmadığını bize belirtir. Dolayısıyla, bu durumda devrenin ne yapması gerektiğini belirleyebiliriz. Bu tüm girişlerin 0 olması problemini çözer.

Öncelikli kodlayıcı adı buradan gelmektedir.

Birden fazla giriş değerinin 1 olması problemi ise bitlerin öncelik sırası belirlenerek halledilmiştir . Buna göre, 8 girişli bir öncelikli kodlayıcıda D_7 girişi en yüksek öncelikli bit olarak belirlenirken; D_0 girişi en düşük öncelikli bit olarak kullanılır. En yüksek öncelikten başlayarak, eğer D_7 değeri 1 olursa diğer daha düşük öncelikli bitlerin (D_6, D_5, ..., D_0) değerine bakılmaksızın çıkış $(111)_2$ olur. Eğer D_7 değeri 0 ise aynı şekilde D_6 değeri kontrol edilir. D_6 değeri 1 ise diğer düşük öncelikli bitlerin

[4] ing. *priority encoder*

(D_5, D_4, ..., D_0) değerine bakılmaksızın çıkış $(110)_2$ olur. Bu kontrol D_0 girişine kadar benzer şekilde uygulanır. Bu sayede, kodlayıcıda birden fazla giriş 1 olsa bile devrenin istenilen şekilde çalışacağı garanti altına alınmış olur.

Örnek 1.3:

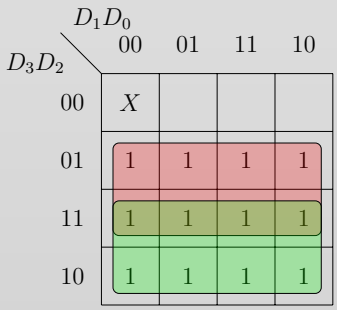
4×2 öncelikli kodlayıcıyı mantık kapıları kullanarak tasarlayınız.

—

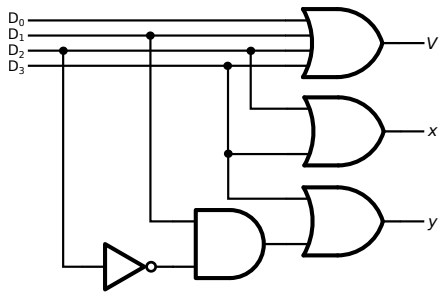
Kodlayıcının girişleri D_0, D_1, D_2, D_3 iken çıkışları x (MSB), y (LSB) ve V olsun. Yukarıda bahsettiğimiz üzere V çıkışı sadece tüm girişler 0 iken 0 olmalı; diğer tüm durumlarda 1 olmalıdır. $D_3 = 1$ iken diğer girişlerin değeri ne olursa olsun $xy = (11)_2$; $D_3 = 0$, $D_2 = 1$ iken diğer girişlerin değeri ne olursa olsun $xy = (10)_2$; $D_3 = 0$, $D_2 = 0$, $D_1 = 1$ iken diğer girişlerin değeri ne olursa olsun $xy = (01)_2$ olmalıdır. Son olarak, sadece $D_0 = 1$ ise çıkış $xy = (00)_2$ olmalıdır. Doğruluk tablosunda "ne olursa olsun" dediğimiz durumları X ile (fark etmez durumu) ifade edersek aşağıdaki tabloyu elde ederiz:

D_3	D_2	D_1	D_0	x	y	V
1	X	X	X	1	1	1
0	1	X	X	1	0	1
0	0	1	X	0	1	1
0	0	0	1	0	0	1
0	0	0	0	X	X	0

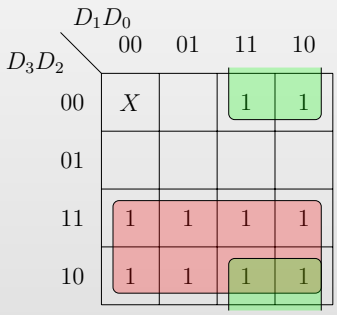
Bu tabloya baktığımızda, V değerinin sadece tüm girişler 0 iken 0 olduğu; diğer tüm durumlarda 1 olduğu görüldüğünden Karnaugh haritasına ihtiyaç duymaksızın $V = D_3 + D_2 + D_1 + D_0$ olması gerektiğini biliriz. Diğer çıkışlar için Karnaugh haritası oluşturmaya ihtiyacımız vardır. Buna göre, x çıkışı için Karnaugh haritası aşağıdaki gibi elde edilir:



Bu tablo doldurulurken girişteki X değerlerinin hem 0 hem 1 için tabloya doldurulduğuna dikkat ediniz. Mesela, $D_3D_2D_1D_0 = (01XX)_2$ girişleri için x çıkışı 1 olduğundan, tabloda $D_3D_2 = (01)_2$ ile başlayan tüm hücrelere 1 yazarız. Buna göre, yukarıdaki iki seçim bize $x = D_2 + D_3$ değerini verir.
Aynı işlemi y çıkışı için tekrarladığımızda ise aşağıdaki Karnaugh haritasını elde ederiz:



Şekil 6: Örnek 1.3'e ait 4×2 öncelikli kodlayıcı gerçekleştirimi.



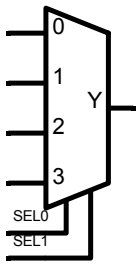
Yukarıdaki iki seçim bize $y = D_3 + D_1 D_2'$ sadeleştirmesini verir. Buna göre, 4×2 öncelikli kodlayıcı devresi aşağıdaki gibi olacaktır:

$$x = D_2 + D_3$$
$$y = D_3 + D_1 D_2'$$
$$V = D_3 + D_2 + D_1 + D_0$$

Bu devreyi mantık kapıları ile kurarsak Şekil 6 'deki devreyi elde ederiz.

1.3.3 Multiplexer (Mux)

Multiplexer (tr. veri seçici) 2^n-bitlik girişten birini n-bitlik seçim bitleriyle seçerek çıkışa aktaran bir kombinasyonel devredir. Giriş sayısına göre 2×1, 4×1, 8×1, 16×1 vb. düzenlemeleri mevcuttur. Her mux düzenlemesi için sadece tek bir çıkış vardır. 4×1 bir mux'a ait blok diyagram aşağıdaki gibidir:



Burada SEL0 ve SEL1 seçim bitleri olup kısaca S_0 ve S_1 ile etiketlenir. 0, 1, 2, 3 ile gösterilen girişler ise sırasıyla I_0, I_1, I_2, I_3 ile etiketlenir. Eğer $S_1S_0 = (00)_2$ ise $Y = I_0$; $S_1S_0 = (01)_2$ ise $Y = I_1$; $S_1S_0 = (10)_2$ ise $Y = I_2$; ve son olarak $S_1S_0 = (11)_2$ ise $Y = I_3$ olur. **Yani, S_0 LSB olmak üzere, seçim bitlerinin ondalık değerine karşılık gelen giriş biti çıkışa aktarılır**. Buna göre 4×1 mux'un doğruluk tablosu aşağıdaki gibi olacaktır:

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Bu tür bir doğruluk tablosunu nasıl gerçekleştireceğiniz konusunda biraz zorlanabilirsiniz. Teorik olarak I_0, I_1, I_2, I_3 girişlerini de sola atıp bunlara da 0 ve 1 şeklinde değer atayarak 6-değişkenli bir doğruluk tablosu oluşturup bunu çözmeye çalışabilirsiniz. Fakat bu hiç pratik olmaz, çünkü yukarıdaki doğruluk tablosunu gerçekleştirmek aslında çok basit. Bunu görebilmek için bir kez daha miniterimlerin özelliğini hatırlamanız lazım. Neydi? Bir giriş kombinasyonuna karşılık miniterimlerden sadece bir tanesi 1 olurken diğer tüm miniterimler 0 olmak zorundaydı. Buna göre, S_1 ve S_0 için oluşturulacak miniterimler $S_1'S_0'$, $S_1'S_0$, S_1S_0' ve S_1S_0 olacaktır. Y çıkışı tüm bu miniterimlerin VEYA'sı alınarak elde edilsin:

$$Y = S_1'S_0' + S_1'S_0 + S_1S_0' + S_1S_0$$

Bu durumda, $S_1S_0 = (00)_2$ olursa $S_1'S_0'$ miniterimi 1 olurken diğer tüm miniterimler 0 olacak. S_1S_0'ın diğer kombinasyonlarında da sadece tek bir miniterim 1 olurken diğerleri 0 olacak. Bu neredeyse mux gerçekleştiriminde istediğimiz şey. Bizim istediğimiz tam olarak $S_1S_0 = (00)_2$ için $Y = I_0$ olması. Dikkat ederseniz, eğer $S_1S_0 = (00)_2$ için $S_1'S_0'$ değeri 1 oluyorsa, aynı kombinasyon için $S_1'S_0'I_0$ değeri I_0 olacaktır. Dolayısıyla, her bir m_i miniterimini I_i giriş değeri ile çarparsak (VE'sini alırsak), tam istediğimiz şeyi elde ederiz:

$$Y = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3 \tag{1}$$

Eşitlik 1 ile ifade edilen 4×1 mux fonksiyonunu devre olarak tasarlayıp simülasyonunu yapın.

Bu durumda, $S_1S_0 = (00)_2$ için sadece $S_1'S_0'$ miniterimi 1 olup diğerleri 0 olacağından sonuçta $Y = I_0$ elde ederiz. Benzer şekilde, $S_1S_0 = (01)_2$ için $Y = I_1$; $S_1S_0 = (10)_2$ için $Y = I_2$; $S_1S_0 = (11)_2$ için $Y = I_3$ elde ederiz. Bu ise tam olarak yukarıdaki doğruluk tablosuna ve 4×1 mux'un çalışmasına karşılık gelir. Bu nedenle, 4×1 mux'un Boole karşılığı Eşitlik 1 'deki gibi olmalıdır. Diğer mux'ların Boole karşılıkları da benzer şekilde miniterimleri kullanarak kolaylıkla belirlenebilir ve mux'lar bulunan Boole ifadeleri kullanılarak mantık kapıları ile de kurulabilir.

Mux'lar piyasada entegre devre olarak bulunmaktadır. Elinizde mux entegresi olmadığı zaman diğer kapıları kullanarak yukarıda gördüğümüz gibi mux'ları gerçekleştirebiliriz.

Örnek 1.4:

Kurmak istediğiniz bir devrede 2×1 mux gerekiyor olsun; fakat elinizde bu mux entegresi olmasın. Diğer kapıları kullanarak bu mux devresini oluşturun.

—

2×1 mux için tek bir S seçim biti 2 bitlik I_0 ve I_1 girişlerini seçmek için yeterli olacaktır. Buna göre, doğruluk tablosu aşağıdaki gibi olmalıdır:

S	Y
0	I_0
1	I_1

S için miniterimler S' ve S olduğu için I_0 ve I_1 girişlerini sırasıyla bu miniterimler ile çarparak elde ettiğimiz terimleri toplarsak, aşağıdaki Boole ifadesini elde ederiz:

$$Y = S'I_0 + SI_1$$

Aynı sonucu, aşağıdaki doğruluk tablosunu sadeleştirerek de elde edebilirsiniz (deneyin):

S	I_1	I_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

***

Mux devreleri de tıpkı kod çözücü devreler gibi programlanabilir lojik eleman olarak kullanılabilir. Yani mux kullanarak da diğer kombinasyonel devreleri gerçekleştirebilirsiniz. Genel olarak n girişli tek çıkışlı bir F kombinasyonel devresini kurmak için kontrol biti sayısı $n-1$ olan $2^{n-1} \times 1$ mux kullanmanız gerekir. Kombinasyonel devrenin $n-1$ değişkeni doğrudan mux'un $n-1$ bitlik kontrol kısmına bağlanırken, son değişken mux'un giriş değerlerine doğruluk tablosu kullanılarak öyle bir şekilde bağlanır ki her bir $n-1$ bitlik seçim kombinasyonu mux çıkışında F fonksiyonu değerini döndürür.

Bunu bir örnekle göstermek daha rahat anlamanızı sağlayacaktır. Örnek olarak,

$$F(x, y, z) = \sum(1, 2, 6, 7)$$

fonksiyonunu mux ile gerçekleştirelim. F üç değişkenli olduğu için ($n = 3$), 4×1 mux kullanmamız gerekir. x ve y değişkenleri doğrudan mux'un kontrol bitlerine bağlanarak mux girişlerini çıkışlara aktarmada kullanılacaktır. Bu durumda, mux girişlerine z, z', 0 veya 1 değerlerini bağlayabilme ihtimalimiz vardır. Hangi girişe neyi bağlayacağımızı F fonksiyonunun doğruluk tablosuna bakarak belirleriz:

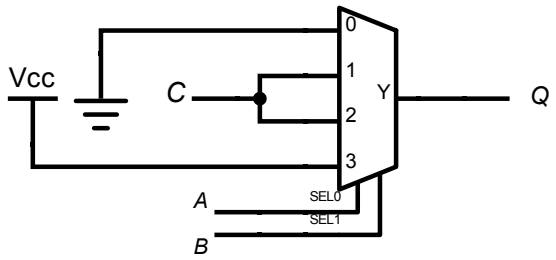
x	y	z	F	
0	0	0	0	$I_0 = F = z$
0	0	1	1	
0	1	0	1	$I_1 = F = z'$
0	1	1	0	
1	0	0	0	$I_2 = F = 0$
1	0	1	0	
1	1	0	1	$I_3 = F = 1$
1	1	1	1	

Yukarıda doğruluk tablosunu 4 gruba ayırdığımızı fark etmişsinizdir. Her bir grupta sırasıyla $xy = (00)_2$, $xy = (01)_2$, $xy = (10)_2$ ve $xy = (11)_2$'dir. Dolayısıyla, bu gruplarda mux çıkışı sırasıyla I_0, I_1, I_2 ve I_3 olacaktır. Biz mux çıkışında F fomksiyonunun değerini görmek istiyorsak yapmamız gereken I_0, I_1, I_2 ve I_3 değerlerini F cinsinden belirlemektir. İlk gruba baktığımızda, $xy = (00)_2$ için F ile z değişkeni arasında $F = z$ ilişkisi olduğunu görürüz. Dolayısıyla, $xy = (00)_2$ için çıkışta F değeri elde etmek için I_0 girişine z değeri atamamız gerekir. Benzer şekilde,

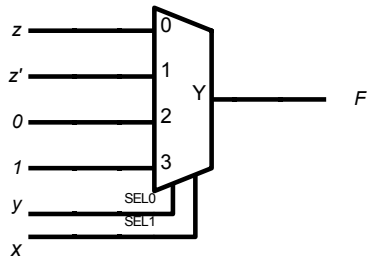
- $xy = (01)_2$ için F ile z arasında $F = z'$ ilişkisi olduğundan $I_1 = z'$ olmalı;
- $xy = (10)_2$ için $F = 0$ olduğundan $I_2 = 0$ olmalı;
- $xy = (11)_2$ için $F = 1$ olduğundan $I_3 = 1$ olmalı.

x MSB olduğu için S_1 olarak kullanılmalı. Seçim bitlerine neyi atadığınıza dikkat etmeniz gerekir!

Buna göre, F fonksiyonunun mux ile devre gerçekleştirimi aşağıdaki gibi olacaktır:



Şekil 7: Örnek 1.5'deki devrenin gerçekleştirimi.



Örnek 1.5:

3-bitlik girişi olan bir devrede, girişlerden 2 veya 3 tanesi 1 olduğunda çıkışı 1 olan devreyi mux ile tasarlayınız (*Sayısal Devre Tasarımı Laboratuvarı 1. Deney sorusu*).

—

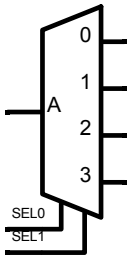
3-bit giriş için 4×1 mux kullanmalıyız. Değişkenler A, B, C ve çıkış Q olsun. A ve B mux'un girişlerine bağlanacaktır. Buna göre, doğruluk tablosu aşağıdaki gibi olmalıdır:

A	B	C	Q	
0	0	0	0	$I_0 = Q = 0$
0	0	1	0	
0	1	0	0	$I_1 = Q = C$
0	1	1	1	
1	0	0	0	$I_2 = Q = C$
1	0	1	1	
1	1	0	1	$I_3 = Q = 1$
1	1	1	1	

Buradan da $Q(A, B, C)$ devresini Şekil 7 'deki gibi elde ederiz.

1.3.4 Demultiplexer (Demux)

Demultiplexer (tr. dağıtıcı) mux'un tam tersi bir şekilde çalışır. n seçim biti olan bir demux, tek bir giriş bitini alarak kontrol bitinin değerine göre bu girişi 2^n adet çıkıştan birine aktarır. Giriş sayısına göre 1×2, 1×4, 1×8, 1×16 vb. düzenlemeleri mevcuttur. Her demux düzenlemesi için (kontrol bitleri dışında) sadece tek bir giriş vardır. 1×4 bir demux'a ait blok diyagram aşağıdaki gibidir:



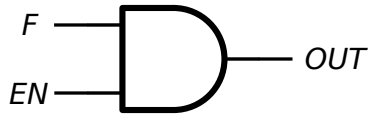
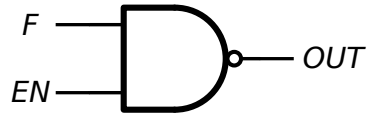
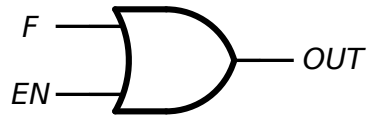
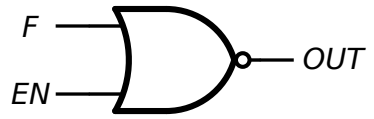
Burada SEL0 ve SEL1 seçim bitleri olup kısaca S_0 ve S_1 ile etiketlenir. 0, 1, 2, 3 ile gösterilen çıkışlar ise sırasıyla D_0, D_1, D_2, D_3 ile etiketlenir. Eğer $S_1S_0 = (00)_2$ ise $D_0 = A$; $S_1S_0 = (01)_2$ ise $D_1 = A$; $S_1S_0 = (10)_2$ ise $D_2 = A$; ve son olarak $S_1S_0 = (11)_2$ ise $D_3 = A$ olur. **Yani, S_0 LSB olmak üzere, giriş değeri seçim bitlerinin ondalık değerine karşılık gelen çıkış bitine aktarılır**. Bir çıkış biti seçiliyken diğer çıkış bitleri 0 durumundadır. Buna göre 1×4 demux'un doğruluk tablosu aşağıdaki gibi olacaktır:

S_1	S_0	D_0	D_1	D_2	D_3
0	0	A	0	0	0
0	1	0	A	0	0
1	0	0	0	A	0
1	1	0	0	0	A

Bu doğruluk tablosu 2×4 kod çözücünün doğruluk tablosuna çok benzerdir (A yerine 1 yazarsanız kod çözücünün doğruluk tablosu elde edilir). Bu nedenle, demux ile kod çözücü (decoder) arasında yakın bir ilişki vardır. Hatırlarsanız kod çözücünün çıkışları miniterimlerdi. Burada ise miniterimlerin 1 olduğu yerlerde A elde etmemiz için miniterimleri A ile çarpmamız (VE'sini almamız) gerekir. Bu nedenle, demux çıkışları aşağıdaki gibi olmalıdır:

$$D_0 = m_0A = S_1'S_0'A$$
$$D_1 = m_1A = S_1'S_0A$$
$$D_2 = m_2A = S_1S_0'A$$
$$D_3 = m_3A = S_1S_0A$$

Bu durum diğer demux çeşitleri için de benzerdir (yapmanız gereken tek şey çıkışa karşılık gelen miniterim ile A girişini çarpmaktır). Elde edilen bu tür Boole ilişkileri ile demux mantık kapıları ile kolaylıkla kurulabilir.

(i)		$EN = 1 \Rightarrow OUT = F$ $EN = 0 \Rightarrow OUT = 0$	EN girişi: **AKTİF YÜKSEK**
(ii)		$EN = 1 \Rightarrow OUT = \overline{F}$ $EN = 0 \Rightarrow OUT = 1$	EN girişi: **AKTİF YÜKSEK**
(iii)		$EN = 1 \Rightarrow OUT = 1$ $EN = 0 \Rightarrow OUT = F$	EN girişi: **AKTİF DÜŞÜK**
(iv)		$EN = 1 \Rightarrow OUT = 0$ $EN = 0 \Rightarrow OUT = \overline{F}$	EN girişi: **AKTİF DÜŞÜK**

Tablo 1: Mantık kapıları ile Enable (EN) girişi oluşturma.

1.4 Enable (Etkinleştirme) Girişi

Çoğu zaman tasarladığınız kombinasyonel devrelerin sadece belli anlarda aktif olmasını isteriz. Bu ise **enable** adı verilen girişlerle sağlanır. Bir kombinasyonel devreyi etkinleştirdiğimiz zaman devre normal çalışmasını sergiler. Enable girişleri ile deaktive edildikleri zaman ise aşağıdaki davranışlardan birini sergilerler:

1. Çıkışların tümü 0 yapılır.
2. Çıkışların tümü 1 yapılır.
3. Çıkışlar yüksek empedans gösterir (Hi-Z durumu [5]).

1.4.1 Çıkışların 0 veya 1 Olduğu Durumlar

İlk iki davranışı elde etmek için daha önce görmüş olduğumuz mantık kapılarını kullanırız. Enable girişi olmayan bir devreye enable girişi eklemek için devrenin çıkışı F ile yeni oluşturduğumuz bir EN girişini Tablo 1'deki gibi işleme sokarız.

Sinyallerin Yüksek (High) ve Düşük (Low) olması için Ders Notu 2'ye bakınız.

Tablo 1'deki (i) durumunda $EN = 0$ olursa çıkış 0 yapılarak devre deaktive edilir. $EN = 1$ olduğunda ise çıkış devrenin normal çıkışı olan F olur. Yani devre $EN = 1$

[5] High-Z kısaltması. Z empedans sembolü.

olduğunda etkinleştirilir. Bu durumda, EN girişi "Yüksek" (H) sinyal seviyesi ile etkinleştirildiği için EN girişi **aktif yüksek** [6] olarak adlandırılır.

(ii) durumuna baktığımızda ise $EN = 0$ olduğunda çıkışın 1 yapılarak devrenin deaktive edildiğini görürürüz. $EN = 1$ içinse F'nin tersi şeklinde olsa da devrenin normal çalışması çıkışa aktarılır. Yani, devre yine $EN = 1$ olduğunda etkinleştirilmektedir. Bu durumda, EN girişi yine "Yüksek" (H) sinyal seviyesi ile etkinleştirildiğinden EN girişi aktif yüksek olacaktır.

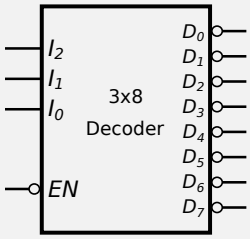
(iii) durumunda ise $EN = 1$ olduğunda çıkış deaktive edilip 1 yapılırken, devre normal çalışmasını $EN = 0$ iken göstermektedir. Yani devre $EN = 0$ olduğunda çalışmaktadır. Bu durumda, EN girişi "Düşük" (L) sinyal seviyesi ile etkinleştirildiği için EN girişi **aktif düşük** [7] olarak adlandırılır.

(iv) durumunda ise şekilde $EN = 1$ olduğunda çıkış yine deaktive edilir fakat bu sefer çıkış 0 olur. $EN = 0$ içinse F'nin tersi şeklinde olsa da devrenin normal çalışması çıkışa aktarılır. Yine devre $EN = 0$ durumunda çalışır. Bu nedenle de EN girişi yine aktif düşük olarak adlandırılır.

Tanım 1.2: Aktif Yüksek / Aktif Düşük

Dijital devrelerde yapması gereken işlevi "Yüksek" (ing. High veya H) sinyal seviyesi için gerçekleştiren girişler **aktif yüksek** olarak adlandırılır.

Tam tersi, yapması gereken işlevi "Düşük" (ing. Low veya L) sinyal seviyesi için gerçekleştiren girişler ise **aktif düşük** olarak adlandırılırlar. Aktif düşük girişler datasheet'lerin bağlantı diyagramlarında DEĞİL sembolü ile gösterilir. Örnek: $\overline{EN}$, $\overline{RESET}$ vb. Şematik gösterimde ise aktif düşük girişlerden önce $\circ$ sembolü kullanılır. Mesela, aktif düşük enable girişine sahip bir 3×8 kod çözücünün blok diyagramı aşağıdaki gibidir. Enable girişi aktif düşük olduğundan, bu kod çözücü $EN = 0$ durumunda aktif olacaktır.



Benzer şekilde, bir dijital devrede çıkış normalde yüksek (H) sinyal seviyesinde kalıyor ve düşük sinyal seviyesi (L) için aktif kabul ediliyorsa, bu tür çıkış da **aktif düşük** olarak adlandırılır. Şematik gösterimde bu tür çıkışların önüne de $\circ$ sembolü konulur. Datasheetler'in bağlantı diyagramlarında ise bu aktif düşük çıkışlar $\overline{D_i}$ ile gösterilir.
Örnek olarak, yukarıdaki 3×8 kod çözücünün çıkışları aktif düşük olduğu için

[6] ing. *active high*
[7] ing. *active low*

çıkışlardan önce ∘ sembolü kullanılmaktadır. Bu çıkışlar düşük sinyal seviyesi için aktif kabul edildiğinden, yukarıdaki kod çözücünün çıkışları normal kod çözücü çıkışlarının tam tersi olacaktır (yani normalde aynı anda çıkışlardan sadece biri aktive edilip 1 yapılırken, yukarıdaki kod çözücüde **tam tersi** aktive edilen çıkış 0 yapılırken diğer çıkışlar 1 durumunda kalacaktır). Buna göre, yukarıdaki kod çözücünün doğruluk tablosu aşağıdaki gibi olur:

EN	I_2	I_1	I_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
1	X	X	X	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1
0	1	1	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	0

Aktif düşük çıkışlara sahip bir kod çözücünün çıkışlarının maksiterimlere karşılık geldiğini fark ettiniz mi? Ettiyseniz maksiterimlerin çarpımı kanonik formundaki Boole fonksiyonlarını bu tür kod çözüler ile VE kapıları kullanarak gerçekleştirebileceğinizi de görebilmiş olmanız lazım.

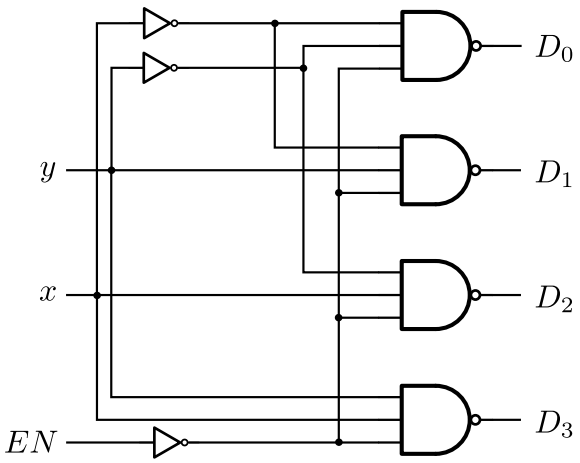
Örnek 1.6:

Aktif düşük enable girişine ve aktif düşük çıkışlara sahip bir 2×4 *kod çözücüyü DEĞİL ile VEDEĞİL kapıları kullanarak tasarlayınız.*

—

VEDEĞİL kapıları ile aktif düşük çıkışlar elde etmek için Tablo 1'de durum (ii)'deki gibi bir tasarım yapmamız lazım. Fakat bu durum için enable girişi aktif yüksek olduğundan, bu girişi aktif düşük yapmak adına enable girişi DEĞİL kapısı ile terslendirilmelidir.

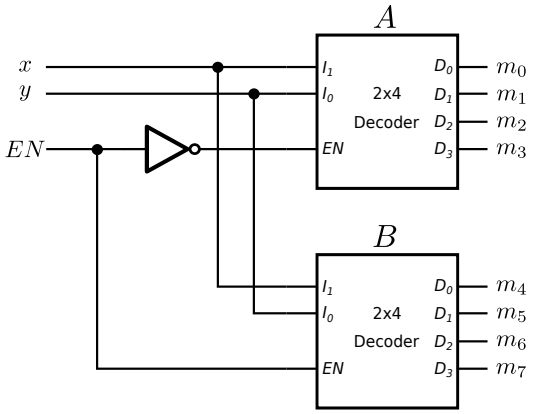
2×4 kod çözücünün 4 çıkışının iki değişkenli miniterimler olması gerektiğini biliyoruz. Bunlar $x'y'$, $x'y$, xy', xy şeklindedir. Aktif düşük çıkışlar için tek yapmamız gereken bu miniterimleri terslendirmek. VEDEĞİL kapıları bunu zaten yaptığından, tek yapmamız gereken her bir çıkış için Tablo 1 durum (ii)'deki F yerine sırasıyla bu miniterimlerden birini koymak. Bunları yaptığımızda Şekil 8 'deki devreyi elde ederiz.



Şekil 8: Aktif düşük enable girişine ve aktif düşük çıkışlara sahip olan 2×4 kod çözücünün VEDEĞİL kapıları ile gerçekleştirimi.

***

Enable girişleri dijital devreleri modüler olarak kullanmamızı sağlar. Mesela, elimizde iki tane enable girişli 2×4 kod çözücü varsa, enable girişleri ile aynı anda sadece birini aktive ederek aşağıdaki gibi 3×8'lik bir kod çözücü elde edebiliriz:



Görüldüğü üzere bu düzenlemede EN girişi ile birlikte üç bitlik giriş; kod çözücülerin beraber çıkışları ise sekiz bitlik çıkış oluşturur. Kod çözücülerin enable girişi ve çıkışları aktif yüksek olduğundan $EN = 0$ olduğunda A kod çözücüsü normal çalışmasını sergilerken B kod çözücüsü deaktive olduğundan çıkışları hep 0 olur. $EN = 1$ olduğunda ise A kod çözücü deaktive olduğundan çıkışları hep 0 olurken B kod çözücüsü normal çalışmasını sergiler. Fakat, EN girişi 3×8 kod çözücünün MSB'si olarak kabul edildiğinden $EN = 1$ için B kod çözücüsünün D_0 çıkışı m_0 miniterimi yerine m_4 miniterimine karşılık gelecektir. Bu konfigürasyonun doğruluk tablosu aşağıdaki gibidir (kırmızı bölgeler deaktif durumu ifade etmektedir):

EN	I_1	I_0	$D_0(A)$	$D_1(A)$	$D_2(A)$	$D_3(A)$	$D_0(B)$	$D_1(B)$	$D_2(B)$	$D_3(B)$
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

***

Örnek olarak 74138 entegresini inceleyiniz.

Daha önce demux ile kod çözücü arasında yakın bir ilişki olduğundan bahsetmiştik. Aslına bakarsanız enable girişi olan bir kod çözücü (decoder) aynı zamanda bir demultiplexer'dır. Bu nedenle enable girişi olan kod çözücüler **decoder-demultiplexer** olarak adlandırılır. Bu ilişkiyi görebilmek için enable girişi olan 2×4 kod çözücünün doğruluk tablosuna bakalım:

EN	I_1	I_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

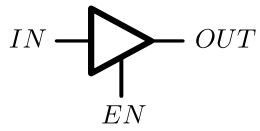
1×4 demux'un doğruluk tablosu ise A giriş biti olmak üzere aşağıdaki gibiydi:

S_1	S_0	D_0	D_1	D_2	D_3
0	0	A	0	0	0
0	1	0	A	0	0
1	0	0	0	A	0
1	1	0	0	0	A

Kod çözücü için EN girişini demux'un A girişi; I_0 ve I_1 girişlerini ise demux'un S_0 ve S_1 kontrol bitleri olarak kullandığımızda bu iki doğruluk tablosunun aslında aynı şey olduğunu görürüz ($A = 1$ ve $A = 0$ için iki tabloyu da karşılaştırınız). Dolayısıyla, EN etkinleştirme girişini bu şekilde kullanarak 2×4 kod çözücüyü aynı zamanda 1×4 demux olarak da kullanabiliriz. Diğer enable girişli kod çözücüleri de benzer şekilde demux olarak kullanabilirsiniz.

1.4.2 Çıkışların Yüksek Empedans Gösterdiği Durum

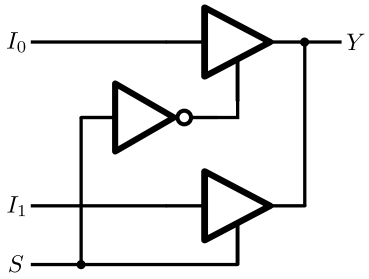
Daha önce görmüş olduğumuz mantık kapıları sadece 0 ve 1 olmak üzere iki durumu ifade edebilirler. Dijital devrelerde çıkışında üçüncü bir durum olan "yüksek empedans" (Hi-Z) durumunu gösterebilen kapı türü **üç durumlu tampon** [8] olarak adlandırılır ve aşağıdaki gibi gösterilir:



Üç durumlu tamponun normal tampon kapısından tek farkı enable (EN) girişidir. $EN = 1$ için üç durumlu tampon normal tampon gibi girişi çıkışa aktarır. $EN = 0$ olduğunda ise çıkış yüksek empedans (Hi-Z) durumu gösterir. Çıkışın yüksek empedans göstermesi çıkış hattının tıpkı açık devreymiş gibi değerlendirilmesine neden olur. Yani, bu durumda tamponun çıkışı bir yere bağlı değilmiş gibi değerlendirilir. Üç durumlu tamponun doğruluk tablosu aşağıdaki gibidir:

EN	IN	OUT
0	X	Hi-Z
1	0	0
1	1	1

Üç durumlu tampon mux tasarlamak için kullanılabilir. Örnek olarak aşağıdaki devre 2×1 mux devresine eşdeğerdir:



Çıkışları yukarıdaki gibi birbirine bağlayabilmemizin nedeni aynı anda sadece bir tamponun aktif olmasıdır. Aktif olmayan tampon devreye bağlı değilmiş gibi davranır. Üç durumlu tampon dışında diğer kapıların çıkışları bu şekilde birbirine bağlanamaz.

[8] ing. *tri-state buffer*

Daha fazla girişli mux'ları üç durumlu tampon kullanarak gerçekleştirmek için kod çözücülerden faydalanılır. Örnek olarak 4×1 mux'u 2×4 kod çözücü ve üç durumlu tamponlar ile tasarlayınız.

Üç durumlu tamponların enable girişleri ve çıkışları da aktif düşük olabilir. Bu durumda, aktif düşük giriş veya çıkışı belirtmek için yine $\circ$ sembolü kullanılır. Örneğin, aktif düşük enable girişine sahip üç durumlu tamponun sembolü aşağıdaki gibidir:

Çıkış aktif düşük olduğunda ise aslında enable girişi olan bir DEĞİL kapısı elde ederiz.