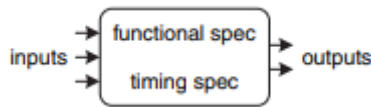


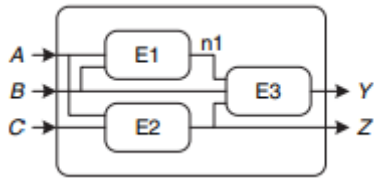
BİRLEŞİK MANTIK

Herhangi sayısal elektronik devre, ayırık değişkenleri işleyen bir sistem olarak düşünülebilir.

- bir veya daha fazla ayırık değerli giriş terminali
- bir veya daha fazla ayırık değerli çıkış terminali
- girdiler ve çıktılar arasındaki ilişkiyi açıklayan işlevsel bir özellik
- girişlerin değişmesi ve yanıt veren çıktılar arasındaki gecikmeyi açıklayan bir zamanlama özelliği.



Kara kutunun içine bakılırsa devreler, düğümlerden ve elemanlardan oluştuğu görülebilir. Bir elemanın kendisi, girişler, çıkışlar ve belli özelliklere sahip bir devredir. Bir düğüm, voltajı ayırık bir değişken taşıyan bir teldir. Düğümler giriş, çıkış veya dahili olarak sınıflandırılır. Girişler dış dünyadan değerler alır. Çıktılar, dış dünyaya değerler verir. Giriş veya çıkış olmayan kablolarla dahili düğümler denir. Aşağıdaki şekil, üç elemanlı, E1, E2 ve E3 ve altı düğümlü bir devreyi göstermektedir. A, B ve C düğümleri girişlerdir. Y ve Z çıktılarıdır. n1, E1 ve E3 arasındaki dahili bir düğümdür.

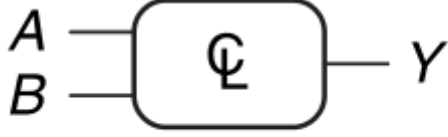


Dijital devreler, birleşik veya sıralı olarak sınıflandırılır. Bir birleşik devrenin çıktıları, yalnızca girişlerin mevcut değerlerine bağlıdır; başka bir deyişle, çıktıyı hesaplamak için mevcut giriş değerlerini birleştirir. Örneğin, bir mantık kapısı, birleşik bir devredir. Sıralı bir devrenin çıktıları, girişlerin hem mevcut hem de önceki değerlerine bağlıdır; başka bir deyişle, giriş sırasına bağlıdır. Bir birleşik devre hafızasızdır, ancak sıralı bir devre hafızaya sahiptir. Bu bölüm birleşik devrelere odaklanır ve sonraki bölüm sıralı devreleri inceler.

Bir birleşimsel devrenin işlevsel özelliği, çıkış değerlerini akım giriş değerleri cinsinden ifade eder. Bir birleşimsel devrenin zamanlama özelliği, girişten çıkışa kadar olan gecikmenin alt ve üst sınırlarından oluşur. Öncelikle fonksiyonel spesifikasyona odaklanacağız, ardından bu bölümün ilerleyen kısımlarında zamanlama spesifikasyonuna döneceğiz.

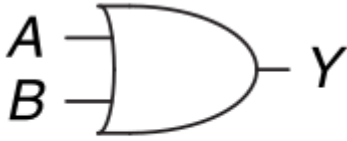
Aşağıdaki şekil, iki girişli ve bir çıkışlı bir birleşimsel devreyi göstermektedir. Şeklin solunda A ve B girişleri ve sağında Y çıkışı vardır. Kutunun içindeki CL sembolü,

yalnızca kombinasyonel mantık kullanılarak uygulandığını gösterir. Bu örnekte, F işlevi OR olarak belirtilmiştir: $Y = F(A, B) = A + B$. Diğer bir deyişle, Y çıkışının, A ve B olmak üzere iki girişin, yani $Y = A \text{ VEYA } B$ 'nin bir fonksiyonu olduğunu söylüyoruz.

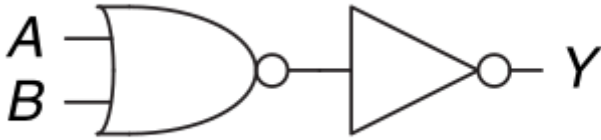


$$Y = F(A, B) = A + B$$

Aşağıdaki şekil, yukarıdaki şekilde gösterilen kombinasyonel mantık devresi için iki olası uygulamayı göstermektedir. Kitap boyunca tekrar tekrar göreceğimiz gibi, genellikle tek bir işlev için birçok uygulama vardır. Elinizdeki yapı taşları ve tasarım kısıtlamalarınız için hangisinin kullanılacağını siz seçersiniz. Bu kısıtlamalar genellikle alanı, hızı, gücü ve tasarım süresini içerir.

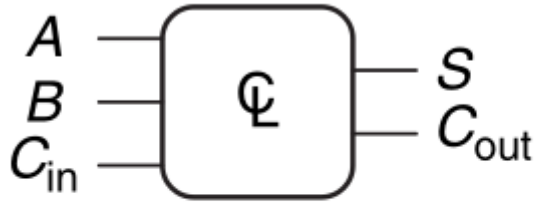


(a)



(b)

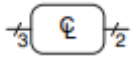
Aşağıdaki şekil, çoklu çıkışlara sahip bir birleşimsel devreyi göstermektedir. Bu belirli kombinasyonel devreye tam toplayıcı adı verilir ve onu Bölüm 5.2.1'de yeniden inceleyeceğiz. İki denklem, girişler, A, B ve Cin cinsinden çıkışların, S ve Cout'un işlevini belirtir.



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

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

Çizimleri basitleştirmek için, genellikle bir çizgi ve yanında bir sayı olan tek bir çizgi, birden çok sinyalden oluşan bir yolu belirtmek için kullanılır. Sayı, veriyolunda kaç sinyal olduğunu belirtir. Örneğin, Aşağıdaki şekil, üç giriş ve iki çıkış içeren bir birleşimsel mantık bloğunu temsil eder.



Bit sayısı önemsizse veya bağlamdan anlaşılırsa, eğik çizgi sayı olmadan gösterilebilir. Aşağıdaki şekil, bir bloktan ikinci bloğa girişler olarak hizmet eden keyfi sayıda çıktıya sahip iki kombinasyonel mantık bloğunu gösterir.

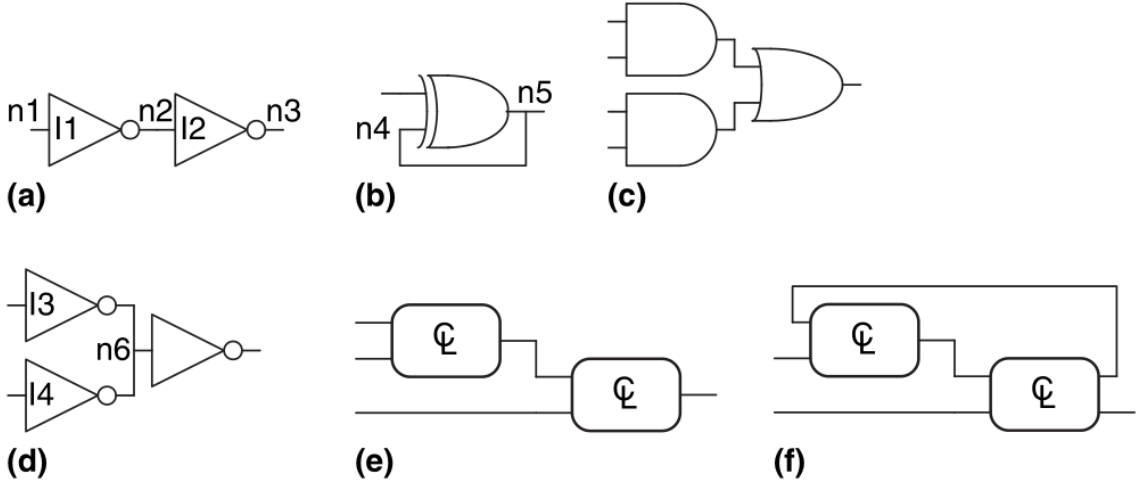


Kombinasyon bileşiminin kuralları, bize daha küçük birleşimsel devre elemanlarından nasıl büyük bir birleşimsel devre oluşturabileceğimizi söylemektedir.

Bir devre, birbirine bağlı devre elemanlarından oluşuyorsa birleşimseldir, öyle ki

- Her devre elemanının kendisi birleşimlidir.
- Devrenin her düğümü ya devreye bir giriş olarak atanır ya da bir devre elemanının tam olarak bir çıkış terminaline bağlanır.
- Devre, döngüsel yollar içermez: devre boyunca her yol, her devre düğümünü en fazla bir kez ziyaret eder.

Örnek 2.1 Aşağıdaki şekilde devrelerden hangisi, kombinasyonel kompozisyon kurallarına göre kombinasyonel devrelerdir?



Çözüm: Devre (a) kombinasyoneldir. İki kombinasyonel devre elemanından (inverterler I1 ve I2) yapılmıştır. Üç düğümü vardır: n1, n2 ve n3. n1, devreye ve I1'e bir girer; n2, I1'in çıkışı ve I2'nin girişi olan dahili bir düğümdür; n3, devrenin ve I2'nin çıkışıdır. (b) kombinasyonel değildir, çünkü döngüsel bir yol vardır: XOR'un çıkışı, girdilerinden birine geri beslenir. Dolayısıyla, n4'te başlayan döngüsel bir yol XOR'dan n5'e geçer ve n4'e geri döner. (c) kombinasyoneldir. (d) kombinasyonel değildir, çünkü düğüm n6, hem I3 hem de I4'ün çıkış terminallerine bağlanır. (e) birleşimseldir, daha büyük bir birleşimsel devre oluşturmak için bağlanan iki birleşimsel devreyi gösterir. (f) iki element arasında döngüsel bir yola sahip olduğu için birleşimsel kompozisyon kurallarına uymaz. Elemanların işlevlerine bağlı olarak, bir birleşimsel devre olabilir veya olmayabilir.

Mikroişlemciler gibi büyük devreler çok karmaşık olabilir, bu nedenle karmaşıklığı yönetmek için Bölüm 1'deki ilkeleri kullanıyoruz. Bir devreyi iyi tanımlanmış bir arayüze ve işleve sahip bir kara kutu olarak görmek, bir soyutlama ve modülerlik uygulamasıdır. Devreyi daha küçük devre elemanlarından oluşturmak bir hiyerarşi uygulamasıdır. Kombinasyonel kompozisyon kuralları, bir disiplin uygulamasıdır.

Bir birleşimsel devrenin işlevsel özelliği genellikle bir doğruluk tablosu veya bir Boole denklemi olarak ifade edilir. Sonraki bölümlerde, herhangi bir doğruluk tablosundan bir Boole denkleminin nasıl çıkarılacağını ve denklemleri basitleştirmek için Boole cebri ve Karnaugh haritalarının nasıl kullanılacağını açıklayacağız. Mantık kapılarını kullanarak bu denklemlerin nasıl uygulanacağını ve bu devrelerin hızının nasıl analiz edileceğini gösteriyoruz.

NOT: Kombinasyonel kompozisyon kuralları yeterlidir ancak kesinlikle gerekli değildir. Bu kurallara uymayan bazı devreler, çıkışlar yalnızca girişlerin mevcut değerlerine bağlı olduğu sürece hala kombinasyoneldir. Bununla birlikte, garip devrelerin kombinasyonel olup olmadığını belirlemek daha zordur, bu nedenle genellikle kombinasyonel devreler inşa etmenin bir yolu olarak kendimizi kombinasyonel kompozisyonla sınırlayacağız.

2.2 Boolean Denklemleri

Boole denklemleri, DOĞRU veya YANLIŞ olan değişkenlerle ilgilenir, bu nedenle dijital mantığı tanımlamak için mükemmeldir. Bu bölüm, Boole denklemlerinde yaygın olarak kullanılan bazı terminolojiyi tanımlar, ardından doğruluk tablosu verilen herhangi bir mantık işlevi için bir Boole denkleminin nasıl yazılacağını gösterir.

2.2.1 Terminoloji

A değişkeninin tamamlayıcısı, ters A'dır. Değişken veya onun tamamlayıcısı, "literal" olarak adlandırılır. Örneğin, A, A', B ve B' literal değerlerdir. A'ya değişkenin gerçek biçimi ve A' 'ya tamamlayıcı biçim diyoruz; "Gerçek biçim", A'nın DOĞRU olduğu anlamına gelmez, yalnızca A'nın üzerinde bir çizgi olmadığı anlamına gelir.

Bir veya daha fazla literal değer AND işlemine tabi tutulmasına product veya implicant denir. A'B, AB'C' ve B, üç değişkenli bir fonksiyonun tüm sonuçlarıdır. Bir minterm, işlevin tüm girdilerini içeren bir product işlemidir. A'B'C', A, B ve C üç değişkeninin bir fonksiyonu için bir mintermdir, ancak AB', C'yi içermediği için değildir. Benzer şekilde, bir veya daha fazla değişimin VEYA'sına sum denir. Bir maxterm, fonksiyonun tüm girdilerini içeren bir sum işlemidir. A + B' + C, A, B ve C olmak üzere üç değişkenli bir fonksiyon için bir maxterm leridir.

Boole denklemlerini yorumlarken işlem sırası önemlidir. $Y=A+BC$, $Y=(A \text{ OR } B) \text{ VE } C$ veya $Y = A \text{ OR } (B \text{ AND } C)$ bu iki ifadeden hangisinin yerine geçer? Boole denklemlerinde, NOT en yüksek önceliğe sahiptir, ardından AND ve ardından OR gelir. Tıpkı sıradan denklemlerde olduğu gibi, çarpımlar toplamlardan önce gerçekleştirilir. Bu nedenle denklem $Y = A \text{ OR } (B \text{ AND } C)$ olarak okunur. Denklem 2.1, işlemlerin sırasına ilişkin başka bir örnek verir.

$$A'B + BCD' = ((A')B) + ((BC)D')$$

2.2.2 Sum-of-Products Form

N girdiden oluşan doğruluk tablosu, girişlerin her olası değeri için bir tane olmak üzere 2^N satır içerir. Bir doğruluk tablosundaki her satır, o satır için DOĞRU olan bir minterm ile ilişkilendirilir. Aşağıdaki şekil, A ve B olmak üzere iki girişin doğruluk tablosunu göstermektedir. Her satır, ilgili mintermini gösterir. Örneğin, ilk satır için minterm A'B' 'dir çünkü A = 0, B = 0 olduğunda A'B' DOĞRUDUR. Mintermler 0 ile başlayarak numaralandırılır; üst sıra minterm 0, m0'a, sonraki satır minterm 1'e, m1'e vb. karşılık gelir.

A	B	Y	minterm	minterm name
0	0	0	$\bar{A} \bar{B}$	m_0
0	1	1	$\bar{A} B$	m_1
1	0	0	$A \bar{B}$	m_2
1	1	0	$A B$	m_3

Çıktı Y'nin DOĞRU olduğu mintermlerin her birini toplayarak herhangi bir doğruluk tablosu için bir Boole denklemi yazabiliriz. Örneğin, üstteki Şekide, mavi daire içinde gösterilen Y çıktısının TRUE olduğu yalnızca bir satır (veya minterm) vardır. Böylece, $Y = A'B$. Altteki Şekil, çıktının TRUE olduğu birden fazla satıra sahip bir doğruluk tablosunu göstermektedir. Daire içindeki mintermlerin her birinin toplamını almak $Y = A'B + AB$ verir:

A	B	Y	minterm	minterm name
0	0	0	$\bar{A} \bar{B}$	m_0
0	1	1	$\bar{A} B$	m_1
1	0	0	$A \bar{B}$	m_2
1	1	1	$A B$	m_3

Bu, çarpımların toplamı (OR) olduğu için (minterm oluşturan AND'ler) bir fonksiyonun toplam çarpımlar kanonik biçimi(sum-of-products canonical form) olarak adlandırılır. Aynı işlevi yazmanın $Y = B'A + BA$ gibi birçok yolu olmasına rağmen, mintermleri doğruluk tablosunda göründükleri sırayla sıralayacağız, böylece aynı doğruluk tablosu için her zaman aynı Boole ifadesini yazacağız.

Çarpımların toplamı kanonik formu, Σ toplama sembolü kullanılarak sigma gösterimi ile de yazılabilir. Bu gösterimle, Yukarıdaki Şekildeki işlev şu şekilde yazılacaktır: $F(A,B) = \Sigma(m1, m3)$ $F(A,B) = \Sigma(1, 3)$

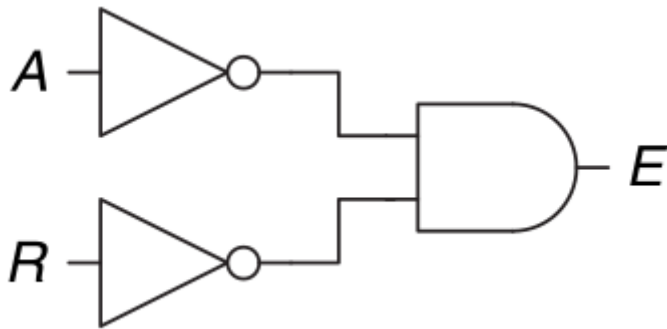
Örnek 2.2 Ben Bitdiddle piknik yapıyor. Yağmur yağarsa ya da karıncalar olursa bundan zevk almaz. Sadece Ben piknikten hoşlanırsa TRUE çıkışı verecek bir devre tasarlayın.

Çözüm: Önce girişleri ve çıkışları tanımlayın. Girişler, karıncalar ve yağmur parametreleri; A ve R'dir. Karıncalar olduğunda A DOĞRU ve karıncalar olmadığında YANLIŞ'dır. Aynı şekilde yağmur yağdığında R DOĞRU ve güneş Ben'in üzerine

gölümsediğinde YANLIŞ'dır. Çıktı, E, Ben'in piknik keyfi. Ben piknikten hoşlanıyorsa E DOĞRU, acı çekerse YANLIŞ. Aşağıdaki şekil, Ben'in piknik deneyimi için doğruluk tablosunu göstermektedir.

<i>A</i>	<i>R</i>	<i>E</i>
0	0	1
0	1	0
1	0	0
1	1	0

Toplam çarpım formunu kullanarak denklemi şu şekilde yazıyoruz: $E = A' R'$ veya $E = \Sigma(0)$. Denklemi, Aşağıdaki şekilde gösterilen iki invertör ve iki girişli bir AND geçidi kullanarak oluşturabiliriz. Bu doğruluk tablosunu NOR işlevi olarak tanıyabilirsiniz. $E = A \text{ NOR } R = (A + R)'$



Aşağıdaki şekil NOR gerçekleştirimi göstermektedir.



Ürünlerin toplamı formu, herhangi bir sayıda değişken içeren herhangi bir doğruluk tablosu için bir Boole denklemi sağlar. Aşağıdaki şekil rastgele üç girişli bir doğruluk tablosunu göstermektedir. Mantık fonksiyonunun toplam-çarpım biçimi denklemlerdeki gibidir. Ne yazık ki, çarpımların toplamı formu mutlaka en basit denklemi oluşturamaz. Bölüm 2.3'te aynı işlevi daha az terim kullanarak nasıl yazacağımızı gösteriyoruz.

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

Product-of-Sum Form

Boole işlevlerini ifade etmenin alternatif bir yolu, toplamların çarpımı kanonik biçimidir. Bir doğruluk tablosunun her satırı, o satır için YANLIŞ olan bir maksimum terime karşılık gelir. Örneğin, iki girişli bir doğruluk tablosunun ilk satırı için maxterm $(A + B)$ 'dir çünkü $(A + B)$, $A = 0$, $B = 0$ olduğunda YANLIŞ olur. Herhangi bir devre için bir Boole denklemini, çıktının FALSE olduğu makstermlerin her birinin AND'si olarak doğrudan doğruluk tablosundan yazabiliriz. Toplamların çarpımı kanonik biçimi, ürün sembolü Π kullanılarak pi gösterimi ile de yazılabilir.

Örnek 2.3 Aşağıdaki Şekilde doğruluk tablosu için toplamların çarpımı formunda bir denklem yazın.

A	B	Y	maxterm	maxterm name
0	0	0	$A + B$	M_0
0	1	1	$A + \overline{B}$	M_1
1	0	0	$\overline{A} + B$	M_2
1	1	1	$\overline{A} + \overline{B}$	M_3

Çözüm: Doğruluk tablosunda çıktının YANLIŞ olduğu iki satır vardır. Dolayısıyla fonksiyon, toplamların çarpımı formunda $Y = (A + B)(A' + B)$ as veya pi gösterimi kullanılarak $Y = \Pi(M_0, M_2)$ veya $Y = \Pi(0, 2)$ şeklinde yazılabilir. İlk maxterm $(A + B)$, herhangi bir AND 0 değeri 0 olduğu için $A = 0$, $B = 0$ için $Y = 0$ olduğunu garanti

eder. Benzer şekilde, ikinci makterm, $(A' + B)$, $A = 1$, $B = 0$ için $Y = 0$ olmasını garanti eder.

Benzer şekilde, Şekildeki Ben'in pikniği için bir Boole denklemi, 0'ların üç satırını daire içine alarak, toplamların çarpımı biçiminde yazılabilir. $E = (A' + R) (A + R') (A' + R')$ veya $E = \prod (1, 2, 3)$. Bu, çarpımların toplamı denkleminde daha çirkin, $E = A' R'$, ancak iki denklem mantıksal olarak eşdeğerdir. Çarpımların toplamı, bir doğruluk tablosunun yalnızca birkaç satırında çıktı TRUE olduğunda daha kısa bir denklem üretir; toplamların çarpımı, bir doğruluk tablosunun yalnızca birkaç satırında çıktı FALSE olduğunda daha basittir.

2.3. BOOLEAN CEBRİ

Önceki bölümde, bir doğruluk tablosu verilen bir Boole ifadesinin nasıl yazılacağını öğrendik. Ancak, bu ifade mutlaka en basit mantık kapıları kümesine götürmez. Matematiksel denklemleri basitleştirmek için cebiri kullandığınız gibi, Boolean denklemlerini basitleştirmek için Boole cebirini kullanabilirsiniz. Boole cebirinin kuralları, sıradan cebir kurallarına çok benzer, ancak bazı durumlarda daha basittir, çünkü değişkenlerin yalnızca iki olası değeri vardır: 0 veya 1.

Boole cebri, doğru olduğunu varsaydığımız bir dizi aksiyoma dayanır. Aksiyomlar, bir tanımın kanıtlanamaması anlamında kanıtlanamaz. Bu aksiyomlardan, Boole cebirinin tüm teoremlerini kanıtlıyoruz. Bu teoremlerin pratik önemi büyüktür, çünkü bize daha küçük ve daha az maliyetli devreler üretmek için mantığı nasıl basitleştireceğimizi öğretirler.

Boole cebirinin aksiyomları ve teoremleri dualite ilkesine uyar. 0 ve 1 sembolleri ve \bullet (VE) ve $+$ (VEYA) operatörleri değiştirilirse, ifade yine de doğru olacaktır. Bir ifadenin ikilisini belirtmek için asal sembolü ($'$) kullanıyoruz.

2.3.1 Aksiyomlar

Aşağıdaki tablo, Boole cebirinin aksiyomlarını belirtir. Bu beş aksiyom ve ikilileri Boole değişkenlerini ve NOT, AND ve OR'nin anlamlarını tanımlar. Aksiyom A1, bir Boolean değişkeni B'nin 1 değilse 0 olduğunu belirtir. Aksiyomun değili A1', değişkenin 0 değilse 1 olduğunu belirtir. A1 ve A1' birlikte, bize 0 ve 1'lerin Boole veya ikili alanında çalıştığımızı söyler. Aksiyomlar A2 ve A2' - NOT işlemini tanımlar. A3 ile A5 aksiyomları AND'yi tanımlar; onların değilleri, A3' ile A5' OR'yi tanımlar.

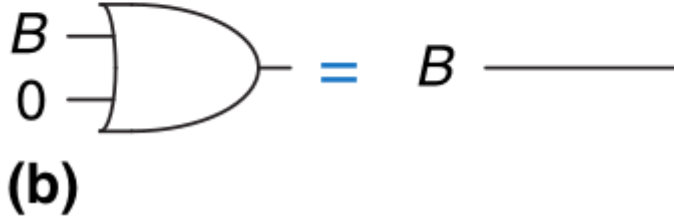
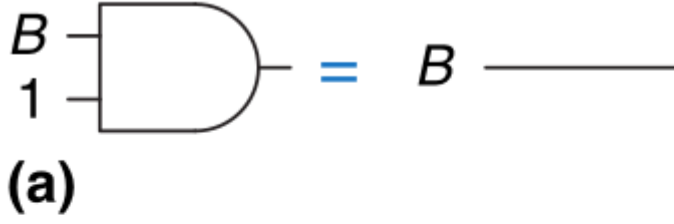
Axiom		Dual		Name
A1	$B = 0 \text{ if } B \neq 1$	A1'	$B = 1 \text{ if } B \neq 0$	Binary field
A2	$\bar{0} = 1$	A2'	$\bar{1} = 0$	NOT
A3	$0 \bullet 0 = 0$	A3'	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	A4'	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5'	$1 + 0 = 0 + 1 = 1$	AND/OR

Bir Degiskene ait Teoremler

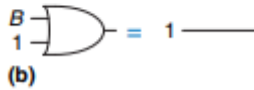
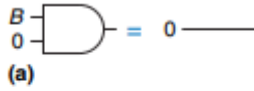
Aşağıdaki tablodaki T1 ila T5 teoremleri, bir değişken içeren denklemlerin nasıl basitleştirileceğini açıklamaktadır.

Theorem		Dual		Name
T1	$B \bullet 1 = B$	T1'	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2'	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3'	$B + B = B$	Idempotency
T4		$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \bar{B} = 0$	T5'	$B + \bar{B} = 1$	Complements

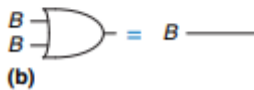
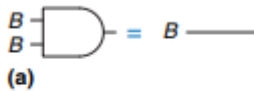
Özdeşlik teoremi T1, herhangi bir Boole değişkeni için B , $B \text{ VE } 1 = B$ olduğunu belirtir. Aynı zamanda değil durumu, $B \text{ OR } 0 = B$ olduğunu belirtir. Donanımda, aşağıdaki şekilde gösterildiği gibi, T1, iki girişli bir AND geçidinin bir girişi her zaman 1 ise, AND geçidini kaldırıp onu değişken girişe (B) bağlı bir kabloyla değiştirebileceğimiz anlamına gelir. Benzer şekilde, T1', iki girişli bir OR geçidinin bir girişi her zaman 0 ise, OR geçidini B 'ye bağlı bir tel ile değiştirebileceğimiz anlamına gelir. Genel olarak, kapılar para, güç ve gecikmeye mal olur, bu nedenle bir kapıyı kabloyla değiştirmek faydalıdır.



Boş eleman teoremi, T2, B AND 0'ın her zaman 0'a eşit olduğunu söyler. Bu nedenle, 0, diğer herhangi bir girdinin etkisini geçersiz kıldığından, AND işlemi için boş öge olarak adlandırılır. Dual, B OR 1'in her zaman 1'e eşit olduğunu belirtir. Bu nedenle, 1, OR işlemi için boş ögedir. Donanımda, aşağıdaki şekil gösterildiği gibi, bir AND geçidinin bir girişi 0 ise, AND geçidini LOW (0'a) bağlı bir tel ile değiştirebiliriz. Benzer şekilde, bir OR geçidinin bir girişi 1 ise, OR geçidini HIGH (1'e) bağlı bir tel ile değiştirebiliriz.

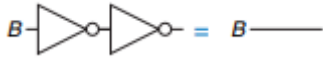


Idempotency, T3, bir değişkenin VE kendisinin sadece kendisine eşit olduğunu söyler. Benzer şekilde, bir değişken VEYA'nın kendisi kendisine eşittir. Teorem adını Latin köklerinden alır: idem (aynı) ve güçlü (güç). İşlemler, onlara koyduğunuz şeyin aynısını döndürür. Aşağıdaki şekil idempotency'nin bir kapının bir kabloyla değiştirilmesine yine izin verdiğini göstermektedir.

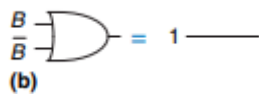
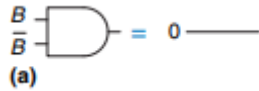


İnvölüsyon, T4, bir değişkeni iki kez tamamlamanın orijinal değişkenle sonuçlandığını söylemenin süslü bir yoludur. Dijital elektronikte iki yanlış bir doğru yapar. Seri

haldeki iki invertör mantıksal olarak birbirini iptal eder ve aşağıdaki şekilde gösterildiği gibi mantıksal olarak bir kabloya eşdeğerdir. T4'ün ikilisi kendisidir.



Tamamlayıcı teoremi, T5 (aşağıdaki şekil), bir değişkenin VE onun tamamlayıcısının 0 olduğunu belirtir (çünkü bunlardan birinin 0 olması gerekir). Ve dualite ile, bir değişken VEYA onun tamamlayıcısı 1'dir (çünkü bunlardan birinin 1 olması gerekir).



Birden çok Değişkene ait Teoremler

Aşağıdaki tabloda T6 ila T12 teoremleri, birden fazla Boole değişkenini içeren denklemlerin nasıl basitleştirileceğini açıklar.

Theorem	Dual	Name
T6 $B \cdot C = C \cdot B$	T6' $B + C = C + B$	Commutativity
T7 $(B \cdot C) \cdot D = B \cdot (C \cdot D)$	T7' $(B + C) + D = B + (C + D)$	Associativity
T8 $(B \cdot C) + (B \cdot D) = B \cdot (C + D)$	T8' $(B + C) \cdot (B + D) = B + (C \cdot D)$	Distributivity
T9 $B \cdot (B + C) = B$	T9' $B + (B \cdot C) = B$	Covering
T10 $(B \cdot C) + (B \cdot \bar{C}) = B$	T10' $(B + C) \cdot (B + \bar{C}) = B$	Combining
T11 $(B \cdot C) + (\bar{B} \cdot D) + (C \cdot D) = B \cdot C + \bar{B} \cdot D$	T11' $(B + C) \cdot (\bar{B} + D) \cdot (C + D) = (B + C) \cdot (\bar{B} + D)$	Consensus
T12 $\overline{B_0 \cdot B_1 \cdot B_2 \dots} = (\bar{B}_0 + \bar{B}_1 + \bar{B}_2 \dots)$	T12' $\overline{B_0 + B_1 + B_2 \dots} = (\bar{B}_0 \cdot \bar{B}_1 \cdot \bar{B}_2 \dots)$	De Morgan's Theorem

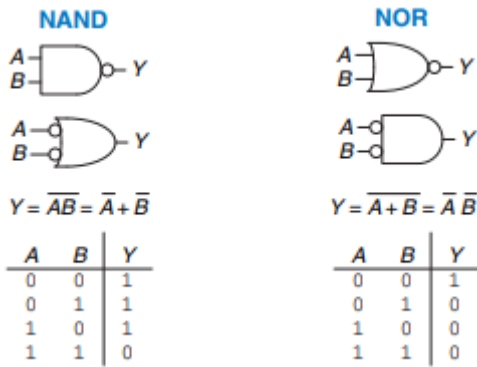
Değişim ve birleşimlilik, T6 ve T7, geleneksel cebirde olduğu gibi çalışır. Komütatiflik ile, bir VE veya VEYA işlevi için girişlerin sırası, çıktının değerini etkilemez. İlişkilendirme ile, belirli girdi grupları, çıktının değerini etkilemez.

Dağılım teoremi, T8, geleneksel cebirdeki ile aynıdır, ancak ikili, T8 ' değildir. T8'e göre, AND, OR üzerinden dağıtır ve T8 ' ile OR, AND üzerinden dağıtır. Geleneksel cebirde, çarpma toplamaya dağıtılır ancak toplama, çarpma üzerinden dağıtılmaz, böylece $(B + C) \times (B + D) \neq B + (C \times D)$.

Örtme, birleştirme ve fikir birliği teoremleri, T9 ila T11, gereksiz değişkenleri ortadan kaldırmamıza izin verir. Biraz düşünerek, kendinizi bu teoremlerin doğru olduğuna ikna edebilirsiniz.

De Morgan'ın Teoremi, T12, dijital tasarımda özellikle güçlü bir araçtır. Teorem, tüm terimlerin çarpımının tamamlamasının, her terimin tamamlayıcısının toplamına eşit olduğunu açıklar. Benzer şekilde, tüm terimlerin toplamının tamamlayıcısı, her terimin tamamlayıcısının ürününe eşittir.

De Morgan'ın teoremine göre, bir NAND geçidi, ters girdilere sahip bir OR geçidine eşdeğerdir. Benzer şekilde, bir NOR geçidi, ters girdilere sahip bir AND geçidine eşdeğerdir. Aşağıdaki şekil, NAND ve NOR kapıları için bu De Morgan eşdeğer kapılarını göstermektedir. Her bir işlem için gösterilen iki sembole ikili denir. Mantıksal olarak eşdeğerdirler ve birbirlerinin yerine kullanılabilirler.



Ters çevirme çemberi bir balon olarak adlandırılır. Sezgisel olarak, bir baloncuğun kapıdan "itilmesinin" diğer taraftan çıkmasına ve kapının gövdesini AND'den OR'a veya tersi yönde çevirmesine neden olduğunu hayal edebilirsiniz. Örneğin, yukarıdaki şekil'deki NAND geçidi, çıkışta bir balon bulunan bir AND gövdesinden oluşur. Kabarcığı sola itmek, girişlerde kabarcıklar bulunan bir OR gövdesi ile sonuçlanır. Kabarcık itmenin temel kuralları şunlardır:

- Baloncukları geriye (çıkıştan) veya ileriye (girişlerden) itmek, geçidin gövdesini AND'den OR'a veya tersi yönde değiştirir.
- Çıkıştan girişlere bir balonun itilmesi, tüm geçit girişlerine baloncuklar yerleştirir.
- Tüm kapı girişlerindeki baloncukları çıkışa doğru ileri itmek, çıkışta bir balon oluşturur.

Örnek 2.4: Allataki şekil, bir Boole fonksiyonu Y ve onun tamamlayıcısı Y' için doğruluk tablosunu gösterir: De Morgan Teoremini kullanarak, Y'nin product-of-sums kanonik biçimini Y' 'nin sum-of-products biçiminden türetin:

A	B	Y	\overline{Y}
0	0	0	1
0	1	0	1
1	0	1	0
1	1	1	0

Çözüm: Aşağıdaki şekil , Y' 'nin içerdiği mintermleri (daire içine alınmış) göstermektedir: Y' 'nin sum-of-products kanonik biçimin; $Y' = A'B' + A'B$

A	B	Y	\bar{Y}	minterm
0	0	0	1	$\bar{A} \bar{B}$
0	1	0	1	$\bar{A} B$
1	0	1	0	$A \bar{B}$
1	1	1	0	$A B$

Her iki tarafın tümleyicisini alarak ve De Morgan Teoremini iki kez uygulayarak şunu elde ederiz: $Y'' = Y = (A' B' + A'B)' = (A'B')'(A'B)' = (A + B)(A + B')$

2.3.5 Denklem indirgeme

Boole cebirinin teoremleri Boole denklemlerini basitleştirmemize yardımcı olur. Örneğin, Aşağıdaki şekilde doğruluk tablosundaki sum-of-products ifadesini düşünün: $Y = A'B' + AB'$: Teorem T10'a göre, denklem $Y = B'$ 'ye sadeleştirilir: Bu, doğruluk tablosuna bakıldığında açıkça görülmüş olabilir. Genel olarak, daha karmaşık denklemleri basitleştirmek için birden çok adım gerekli olabilir.

A	B	Y	minterm	minterm name
0	0	0	$\bar{A} \bar{B}$	m_0
0	1	1	$\bar{A} B$	m_1
1	0	0	$A \bar{B}$	m_2
1	1	1	$A B$	m_3

sum-of-products denklemlerini basitleştirmenin temel ilkesi, $PA + PA' = P$ ilişkisini kullanarak terimleri birleştirmektir, burada P herhangi bir dolaylı olabilir. Bir denklem ne kadar basitleştirilebilir? Mümkün olan en az etkiyi kullanırsa en aza indirilecek sum-of-products biçiminde bir denklem tanımlarız. Aynı sayıda etkiye sahip birkaç denklem varsa, en az olan en az değişmez değere sahip olmalıdır.

Bir implicant, daha az değişmez değerle yeni bir implikant oluşturmak için denklemdeki diğer herhangi bir implicant ile birleştirilemezse, birincil implikant olarak adlandırılır. Minimal bir denklemdeki sonuçların tümü birincil çıkarımlar(implikant) olmalıdır. Aksi takdirde, değişmez değerlerin sayısını azaltmak için birleştirilebilirler.

Örnek 2.6 $A'B'C' + AB'C' + AB'C$ ifadesini sadeleştiriniz.

Çözüm: Orijinal denklemlerle başlıyoruz ve aşağıdaki tabloda gösterildiği gibi Boole teoremlerini adım adım uyguluyoruz. Bu noktada denklemi tamamen basitleştirdik mi? Hadi daha yakından bakalım.

Step	Equation	Justification
	$\overline{A} \overline{B} \overline{C} + \overline{A} B \overline{C} + \overline{A} B C$	
1	$\overline{B} \overline{C} (\overline{A} + A) + \overline{A} B C$	T8: Distributivity
2	$\overline{B} \overline{C} (1) + \overline{A} B C$	T5: Complements
3	$\overline{B} \overline{C} + \overline{A} B C$	T1: Identity

Orijinal denklemden, $A'B'C'$ ve $AB'C'$ mintermleri yalnızca A değişkeninde farklılık gösterir. Böylece mintermleri birleştirerek $B'C'$ 'yi oluşturduk. Ancak, orijinal denkleme bakarsak, son iki minterm $AB'C'$ ve $AB'C$ 'nin de tek bir değişmez değerle (C ve C') farklılık gösterdiğini not ederiz. Böylece, aynı yöntemi kullanarak, bu iki mintermi birleştirerek minterm AB' 'yi oluşturabiliriz. Etkilenen $B'C'$ ve AB' 'nin minterm $AB'C'$ 'yi paylaştığını gösteririz.

Öyleyse, minterm çiftlerinden yalnızca birini sadeleştirmek zorunda mıyız yoksa ikisini de basitleştirebilir miyiz? İdempotens teoremini kullanarak, terimleri istediğimiz kadar çoğaltabiliriz: $B = B + B + B + B \dots$ Bu prensibi kullanarak, aşağıdaki tabloda gösterildiği gibi denklemi iki asal çarpımı olan $B \overline{C} + AB'$ 'ye tamamen sadeleştiriyoruz.

Step	Equation	Justification
	$\overline{A} \overline{B} \overline{C} + \overline{A} B \overline{C} + \overline{A} B C$	
1	$\overline{A} \overline{B} \overline{C} + \overline{A} B \overline{C} + \overline{A} B \overline{C} + \overline{A} B C$	T3: Idempotency
2	$\overline{B} \overline{C} (\overline{A} + A) + \overline{A} B (\overline{C} + C)$	T8: Distributivity
3	$\overline{B} \overline{C} (1) + \overline{A} B (1)$	T5: Complements
4	$\overline{B} \overline{C} + \overline{A} B$	T1: Identity

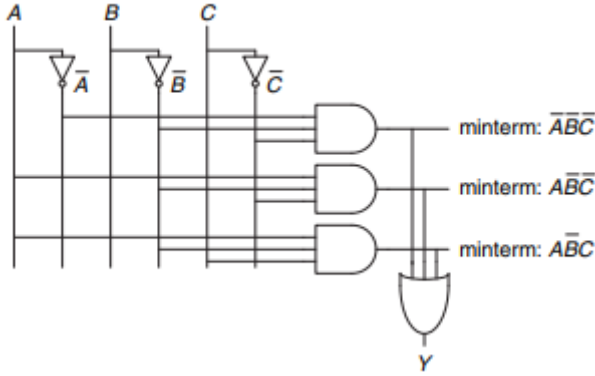
Biraz mantığa aykırı olsa da, bir etkiyi genişletmek (örneğin, AB' 'yi $ABC + ABC'$ 'ye dönüştürmek) bazen denklemleri en aza indirmede yararlıdır. Bunu yaparak, başka bir minterm ile birleştirilmek (paylaşılmak) için genişletilmiş mintermlerden birini tekrarlayabilirsiniz.

Bir Boole denklemini Boole cebri teoremleriyle tamamen basitleştirmenin biraz deneme yanılma gerektirebileceğini fark etmiş olabilirsiniz. Bölüm 2.7, süreci kolaylaştıran Karnaugh haritaları adı verilen yöntemsel bir tekniği açıklar.

Mantıksal olarak eşdeğer kalmaya devam ederse neden bir Boole denklemini basitleştirmeye çalışalım? Basitleştirme işlevi fiziksel olarak uygulamak için kullanılan kapıların sayısını azaltır, böylece daha küçük, daha ucuz ve muhtemelen daha hızlı hale getirir. Bir sonraki bölümde mantık geçitleriyle Boole denklemlerinin nasıl uygulanacağı açıklanmaktadır.

2.4 Lojik ifadelerden Lojik Kapılara (sematik)

Şematik, elemanları ve bunları birbirine bağlayan telleri gösteren bir dijital devrenin diyagramıdır. Örneğin, aşağıdaki şekilde, favori mantık denklemin ($Y = A'B'C' + AB'C' + AB'C$) olası bir donanım uygulamasını göstermektedir.

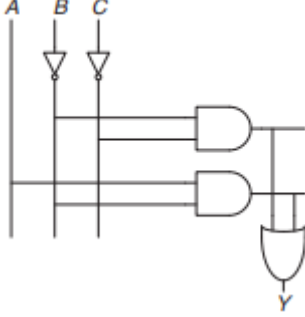


Şemaları tutarlı bir şekilde çizerek, bunların okunmasını ve hata ayıklamasını kolaylaştırıyoruz. Genel olarak aşağıdaki kurallara uyacağız:

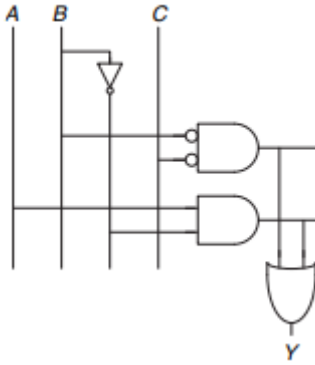
- Girişler, şematiğin sol (veya üst) tarafındadır.
- Çıkışlar, bir şemanın sağ (veya alt) tarafındadır.
- Mümkün olduğunda kapılar soldan sağa akmalıdır.
- Düz kablolar, çok köşeli kablolardan daha iyidir (karmaşık teller, devrenin ne yaptığını düşünmek yerine teli takip etmek gibi zihinsel çabayı gerektirir).
- Kablolar her zaman bir T bağlantısında bağlanır.
- Tellerin kesiştiği nokta, teller arasındaki bir bağlantıyı gösterir.
- Noktasız kesişen kablolar bağlantı yapmaz.
Son üç kısıt aşağıdaki şekilde görülmektedir.

Çarpımların toplamı formundaki herhangi bir Boole denklemi, 2.23 şekile benzer bir sistematik yolla şematik olarak çizilebilir. İlk olarak, girdiler için sütunlar çizin. Gerekirse tamamlayıcı girişler sağlamak için inverterleri bitişik sütunlara yerleştirin. Mintermlerin her biri için sıra sıra AND kapıları çizin. Ardından, her çıktı için, o çıktıyla ilgili mintermlere bağlı bir OR geçidi çizin. Bu stile programlanabilir mantık dizisi (PLA) adı verilir çünkü invertörler, AND geçitleri ve OR kapıları sistematik bir şekilde dizilmiştir.

Aşağıdaki şekil, geçen son derste Boole cebri kullanarak bulduğumuz basitleştirilmiş denklemin bir uygulamasını göstermektedir. Basitleştirilmiş devrenin bir 2.23 şekilden önemli ölçüde daha az donanıma sahip olduğuna dikkat edin. Daha az girişli kapılar kullandığı için daha hızlı da olabilir.



Kapıların ters çevrilmesinden yararlanarak kapı sayısını daha da azaltabiliriz (tek bir invertörle de olsa). $B' C'$ 'nin ters girdilere sahip bir AND olduğunu gözlemleyin. aşağıdaki şekil, C'deki inverteri ortadan kaldırmak için bu optimizasyonu kullanan bir şematik gösterir. De Morgan teoremine göre, ters girdilerle AND'nin bir NOR geçidine eşdeğer olduğunu hatırlayın. Uygulama teknolojisine bağlı olarak, en az sayıda geçidi kullanmak veya diğerlerine göre tercihe göre belirli kapı tiplerini kullanmak daha ucuz olabilir. Örneğin, CMOS uygulamalarında AND'ler ve OR'ler yerine NAND'ler ve NOR'lar tercih edilir.



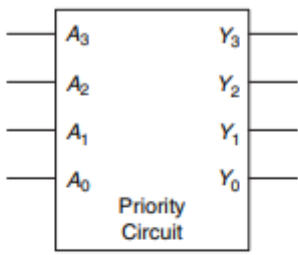
Birçok devrenin, her biri girişlerin ayrı bir Boole işlevini hesaplayan birden fazla çıkışı vardır. Her çıktı için ayrı bir doğruluk tablosu yazabiliriz, ancak tüm çıktıları tek bir doğruluk tablosuna yazmak ve tüm çıktılarla bir şematik çizim yapmak genellikle uygundur.

Örnek 2.7: Dekan, bölüm başkanı, öğretim görevlisi ve yurt müdürü bazen konferans salonunu kullanmaktadır. Ne yazık ki, ara sıra aynı anda kullanmak istemektedirler. Bu durum Dekan'ın huysuz mütevellilerle bağış toplama toplantısı yurdun BTB partisiyle aynı anda gerçekleşmesi gibi felaketlere yol açıyorlar. Bu durumu çözmek için Alyssa P. Hacker, bir oda rezervasyon sistemi tasarlaması için çağrıldı.

Sistemin dört girişi, A_3, \dots, A_0 ve dört çıkışı, Y_3, \dots, Y_0 vardır. Bu sinyaller $A_3:0$ ve $Y_3:0$ olarak da yazılabilir. Her kullanıcı, ertesi gün için konferans salonunu talep ettiğinde bildirir. Sistem, konferans salonunu en yüksek öncelikli kullanıcıya vererek en fazla bir çıktı ileri sürer. Sistemin bedelini ödeyen dekan en yüksek önceliği talep eder. Bölüm başkanı, öğretim görevlisi ve yurt müdürü azalan önceliğe sahiptir.

Sistem için bir doğruluk tablosu ve Boole denklemleri yazın. Bu işlevi yerine getiren bir devre çizin.

Çözüm: Bu işleve dört girişli öncelik devresi denir. Sembolü ve doğruluk tablosu gösterilmektedir.

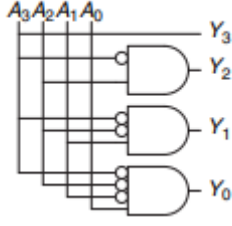


A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	0	0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

Her çıktıyı çarpımların toplamı biçiminde yazabilir ve denklemleri Boole cebri kullanarak azaltabiliriz. Bununla birlikte, basitleştirilmiş denklemler, fonksiyonel açıklamadan (ve doğruluk tablosundan) incelenerek kolayca çıkarılabilir.

A_3 bir ise Y_3 DOĞRUDUR, yani $Y_3 = A_3$. A_2 bir ise ve A_3 sıfır ise Y_2 DOĞRU'dur, bu nedenle $Y_2 = A_3' A_2$. A_1 bir ise ve yüksek öncelikli girişlerden hiçbiri bir değilse Y_1 DOĞRUDUR: $Y_1 = A_3' A_2' A_1$. A_0 ve başka hiçbir giriş bir olmadığında Y_0 DOĞRUDUR iddia edildi: $Y_0 = A_3' A_2' A_1' A_0$

Şematik, şekil 2.28 de gösterilmiştir. Deneyimli bir tasarımcı, genellikle inceleme yoluyla bir mantık devresi uygulayabilir. Açık bir özellik verildiğinde, kelimeleri denklemlere ve denklemleri kapılara çevirin.



Öncelikli devrede A3 bir ise, çıkışların diğer girişlerin ne olduğunu umursamadığına dikkat edin. Çıktının önemsemediği girdileri tanımlamak için X sembolünü kullanıyoruz. Şekil 2.29, dört-girişli öncelikli devre doğruluk tablosunun önemsemeyen çok daha küçük hale geldiğini göstermektedir. Bu doğruluk tablosundan, X'lerin girişlerini göz ardı ederek Boole denklemlerini çarpımların toplamı formunda kolayca okuyabiliriz.

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

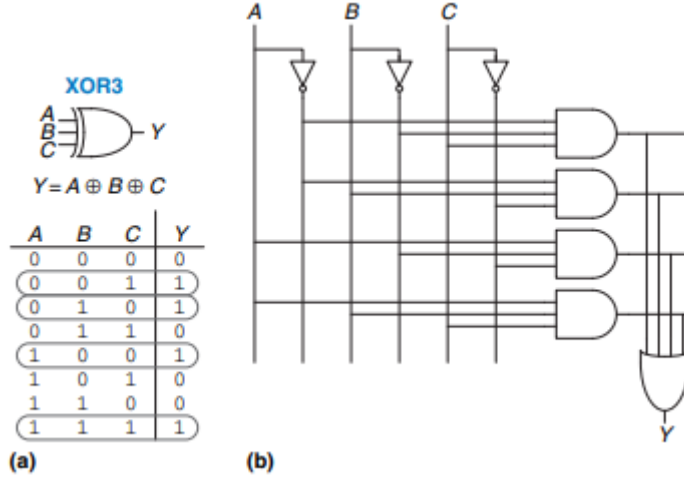
ÇOK SEVİYELİ BİRLESİK MANTIK

Çarpımların toplamı formundaki mantık, iki seviyeli mantık olarak adlandırılır, çünkü bir VEYA kapıları seviyesine bağlı bir AND kapıları seviyesine bağlı değişmez değerlerden oluşur. Tasarımcılar genellikle ikiden fazla mantık kapısı seviyesi olan devreler inşa ederler. Bu çok seviyeli kombinasyonel devreler, iki seviyeli muadillerinden daha az donanım kullanabilir. Kabarcık itme, çok seviyeli devrelerin analizinde ve tasarlanmasında özellikle yararlıdır.

2.5.1 Donanım Azaltma

Bazı mantık işlevleri, iki seviyeli mantık kullanılarak oluşturulduklarında çok büyük miktarda donanım gerektirir. Dikkate değer bir örnek, çoklu değişkenlerin XOR fonksiyonudur. Örneğin, şimdiye kadar incelediğimiz iki seviyeli teknikleri kullanarak üç girişli bir XOR oluşturmayı düşünün.

Tek sayıda giriş TRUE olduğunda bir N-girişli XOR'un bir TRUE çıktı oluşturduğunu hatırlayın. Şekil 2.20, TRUE çıktılar üreten satırlar daire içine alınmış üç girişli bir XOR için doğruluk tablosunu göstermektedir. Doğruluk tablosundan, $Y = A'B'C + A'B'C' + AB'C' + ABC$ deklemini çarpımların toplamı biçiminde bir Boole denklemi okuyoruz. Ne yazık ki, bu denklemi daha az etkiye indirgemenin bir yolu yoktur.



Öte yandan, $A \oplus B \oplus C = (A \oplus B) \oplus C$ (şüpheniz varsa bunu mükemmel tümevarım ile kendinize kanıtlayın). Bu nedenle, üç girişli XOR, Şekil 2.31'de gösterildiği gibi, iki girişli XOR'un bir kademesinden oluşturulabilir.

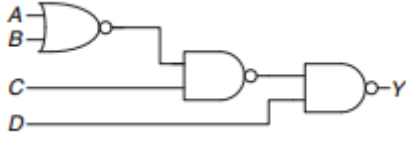
Benzer şekilde, sekiz girişli bir XOR, iki seviyeli toplam ürün uygulaması için 128 sekiz girişli AND geçidi ve bir 128 girişli OR geçidi gerektirir. Çok daha iyi bir seçenek, Şekil 2.32'de gösterildiği gibi, iki girişli XOR geçitlerinden oluşan bir ağaç kullanmaktır.

Belirli bir mantık işlevinin en iyi çok düzeyli uygulamasını seçmek basit bir süreç değildir. Dahası, "en iyi" birçok anlama sahiptir: en az kapı, en hızlı, en kısa tasarım süresi, en az maliyet, en az güç tüketimi. Bölüm 5'te, bir teknolojiye "en iyi" devrenin diğerinde mutlaka en iyi olmadığını göreceksiniz. Örneğin, AND'leri ve OR'leri kullanıyoruz, ancak CMOS'ta NAND'ler ve NOR'lar daha verimlidir. Biraz deneyimle, çoğu devre için inceleme yaparak iyi bir çok düzeyli tasarım oluşturabileceğinizi göreceksiniz. Bu kitabın geri kalanında devre örneklerini incelerken bu deneyimin bir kısmını geliştireceksiniz. Öğrenirken, çeşitli tasarım seçeneklerini keşfedin ve değiş tokuşları düşünün. Bilgisayar destekli tasarım (CAD) araçları, olası çok düzeyli tasarımların geniş bir alanında arama yapmak ve mevcut yapı taşları göz önüne alındığında kısıtlamalarınıza en uygun olanı aramak için de mevcuttur.

2.5.2 Tersleyici itme

Bölüm 1.7.6'dan CMOS devrelerinin AND'ler ve OR'ler yerine NAND'ları ve NOR'ları tercih ettiğini hatırlayabilirsiniz. Ancak denklemi NAND'lar ve NOR'lar ile çok düzeyli bir devreden inceleyerek okumak oldukça zor olabilir. Şekil 2.33, işlevi incelemeye hemen netleşmeyen çok düzeyli bir devreyi göstermektedir. Kabarcık itme, bu

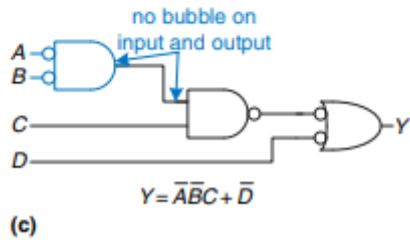
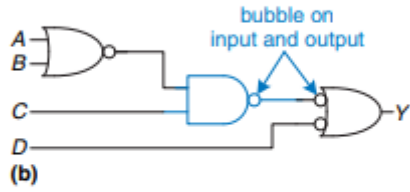
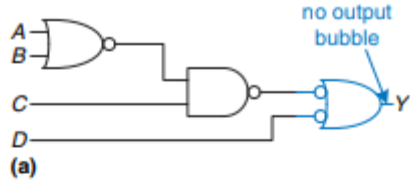
devreleri yeniden çizmenin yararlı bir yoludur, böylece kabarcıklar birbirini götürür ve işlev daha kolay belirlenebilir.



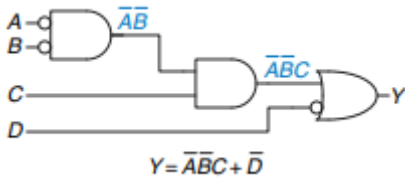
Bölüm 2.3.3'teki ilkelere dayalı olarak, balon itme kuralları aşağıdaki gibidir:

- Devrenin çıkışından başlayın ve girişlere doğru ilerleyin.
- Son çıktındaki kabarcıkları girişlere doğru geri itin, böylelikle (Y') çıktısının tümleyicisi yerine çıktı cinsinden bir denklem (örneğin, Y) okuyabilirsiniz.
- Geriye doğru hareket ederek, her bir kapıyı kabarcıklar birbirini götürecek formda çizin. Mevcut geçitte bir giriş balonu varsa, önceki kapıyı bir çıkış balonuyla çizin. Mevcut geçitte bir giriş balonu yoksa, önceki kapıyı bir çıkış balonu olmadan çizin.

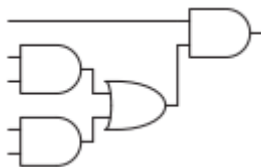
Şekil 2.34, Şekil 2.33 balon itme kurallarına göre nasıl yeniden çizileceğini gösterir. Y çıkışından başlayarak, NAND geçidinin çıkışta ortadan kaldırmak istediğimiz bir balonu vardır. Çıkış balonunu, Şekil 2.34(a) gösterildiği gibi, ters girdilerle bir OR oluşturmak için geri itiyoruz. Sola doğru çalışırken, en sağdaki kapı, orta NAND geçidinin çıkış baloncuğunu iptal eden bir giriş balonuna sahiptir, bu nedenle Şekil 2.34(b) gösterildiği gibi hiçbir değişiklik gerekmez. Orta kapının giriş balonu yoktur, bu nedenle Şekil 2.34(c) gösterildiği gibi en soldaki kapıyı çıkış balonu olmayacak şekilde dönüştürürüz. Şimdi, girişler haricinde devredeki tüm kabarcıklar birbirini götürür, böylece fonksiyon, gerçek veya tamamlayıcı girişlerin AND'leri ve OR'leri açısından incelenerek okunabilir: $Y = A'B'C + D'$



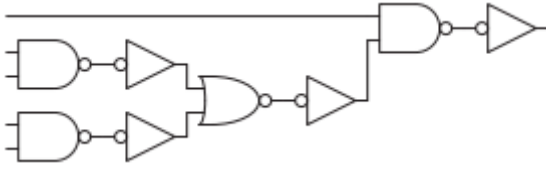
Bu son noktanın vurgulanması için, Şekil 2.35, Şekil 2.34 mantıksal olarak eşdeğer bir devreyi göstermektedir. Dahili düğümlerin işlevleri mavi ile etiketlenmiştir. Serideki kabarcıklar birbirini götürdüğü için, Şekil 2.35 mantıksal olarak eşdeğer devreyi üretmek için orta kapının çıkışındaki ve en sağdaki kapının bir girişindeki kabarcıkları yok sayabiliriz.



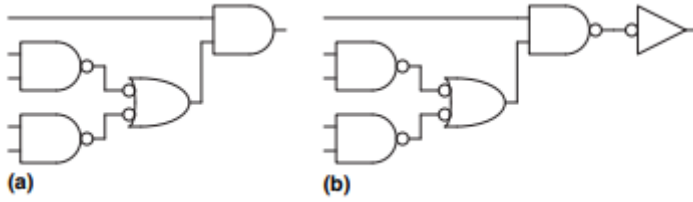
Örnek: Çoğu tasarımcı AND ve OR kapıları açısından düşünür, ancak NAND ve NOR kapılarını destekleyen CMOS mantığında Şekil 2.36 devreyi uygulamak istediğinizi varsayalım. Devreyi NAND'lara, NOR'lara ve invertörlere dönüştürmek için kabarcık itme özelliğini kullanın.



Çözüm: Kaba kuvvet çözümü, Şekil 2.37 gösterildiği gibi, her AND geçidini bir NAND ve bir invertörle ve her OR geçidini bir NOR ve bir invertörle değiştirmektir. Bu sekiz kapı gerektirir. Kabarcığın önceki ters çevirme geçidi ile nasıl iptal edilebileceğini vurgulamak için, eviricinin arkadan değil öndeki balonla çizildiğine dikkat edin.

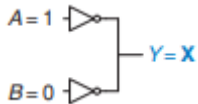


Daha iyi bir çözüm için, Şekil 2.38(a) gösterildiği gibi, kabarcıkların işlevi değiştirmeden bir geçidin çıkışına ve sonraki geçidin girişine eklenebileceğini gözlemleyin. Nihai AND, Şekil 2.38(b) gösterildiği gibi bir NAND ve bir invertöre dönüştürülür. Bu çözüm yalnızca beş kapı gerektirir.



2.6.1 illegal X Degeri

X sembolü, devre düğümünün bilinmeyen veya geçersiz bir değere sahip olduğunu gösterir. Bu genellikle aynı anda hem 0 hem de 1'e sürülüyorsa olur. Şekil 2.39, Y düğümünün hem YÜKSEK hem de DÜŞÜK sürüldüğü bir durumu gösterir. Çekişme adı verilen bu durum bir hata olarak değerlendirilir ve kaçınılması gerekir. Çekişmeli bir düğümdeki gerçek voltaj, YÜKSEK ve DÜŞÜK süren kapıların göreceli güçlerine bağlı olarak 0 ile VDD arasında bir yerde olabilir. Çoğu zaman, ama her zaman değil, yasak bölgededir. Çatışma ayrıca savaş kapıları arasında büyük miktarda güç akmasına neden olarak devrenin ısınmasına ve muhtemelen hasar görmesine neden olabilir.



X değerleri ayrıca bazen devre simülatörleri tarafından başlatılmamış bir değeri belirtmek için kullanılır. Örneğin, bir girdinin değerini belirtmeyi unutursanız, simülatör sizi soruna karşı uyarmak için bunun bir X olduğunu varsayabilir.

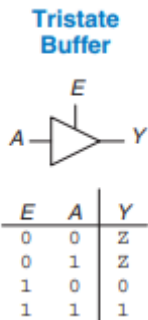
Bölüm 2.4'te belirtildiği gibi, dijital tasarımcılar, doğruluk tablolarında "umursamıyorum" değerlerini belirtmek için X sembolünü de kullanırlar. İki anlamı karıştırmamaya dikkat edin. X bir doğruluk tablosunda görüldüğünde, doğruluk tablosundaki değişkenin değerinin önemsiz olduğunu gösterir (0 veya 1 olabilir). X bir devrede görüldüğünde, devre düğümünün bilinmeyen veya geçersiz bir değeri olduğu anlamına gelir.

Kayan Z Degeri

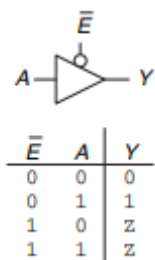
Z sembolü, bir düğümün YÜKSEK veya DÜŞÜK sürülmediğini gösterir. Düğümün yüzen, yüksek empedans veya yüksek Z olduğu söyleniyor. Tipik bir yanılğı, kayan veya yönlendirilmemiş bir düğümün mantık 0 ile aynı olmasıdır. Gerçekte, sistem geçmişine bağlı olarak, yüzen bir düğüm 0 olabilir, 1 olabilir veya arada bir voltajda olabilir. Başka bir devre elemanı, düğümün değeri devre çalışmasıyla ilgili olduğunda düğümü geçerli bir mantık düzeyine sürdüğü sürece, bir yüzer düğüm, devrede bir hata olduğu anlamına gelmez.

Kayan bir düğüm oluşturma yaygın bir yolu, bir devre girişine bir voltaj bağlamayı unutmak veya bağlantısız bir girişin 0 değerine sahip bir girişle aynı olduğunu varsaymaktır. Bu hata, kayan giriş rastgele 0'dan 1'e değiştiğinden devrenin düzensiz davranmasına neden olabilir. Nitekim vücuttan statik elektrikle değişimi tetiklemek için devreye dokunmak yeterli olabilir. Sadece öğrencinin parmağını çipin üzerine basılı tuttuğu sürece doğru çalışan devreler gördük.

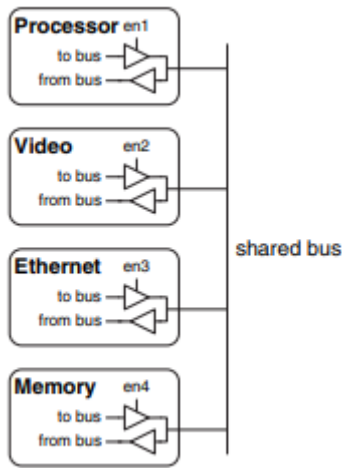
Şekil 2.40 gösterilen üç durumlu tampon, üç olası çıkış durumuna sahiptir: YÜKSEK (1), DÜŞÜK (0) ve kayan (Z). Üç durumlu tamponun bir A girişi, Y çıkışı ve E'yi etkinleştirmesi vardır. Etkinleştirme TRUE olduğunda, üç durumlu tampon, giriş değerini çıkışa aktararak basit bir tampon görevi görür. Etkinleştirme YANLIŞ olduğunda, çıkışın yüzmesine izin verilir (Z).



Şekilde 2.40 üç durumlu tampon aktif bir yüksek yetkiye sahiptir. Yani, etkinleştirme YÜKSEK (1) olduğunda, arabellek etkinleştirilir. Şekil 2.41, aktif düşük etkinleştirmeli bir üç durumlu tamponu göstermektedir. Etkinleştirme DÜŞÜK (0) olduğunda, arabellek etkinleştirilir. Giriş teline bir balon koyarak sinyalin düşük aktif olduğunu gösteriyoruz. Genellikle, adı E' üzerine bir çubuk çizerek veya Eb veya Ebar adından sonra "b" veya "bar" harflerini ekleyerek aktif bir düşük girişi belirtiriz.



Üç durumlu tamponlar genellikle birden fazla çipi bağlayan veri yollarında kullanılır. Örneğin, bir mikroişlemci, bir video denetleyici ve bir Ethernet denetleyicisi, bir kişisel bilgisayardaki bellek sistemiyle iletişim kurmaya ihtiyaç duyabilir. Her yonga, Şekil 2.42 gösterildiği gibi, üç durumlu tamponları kullanarak paylaşılan bir bellek veri yoluna bağlanabilir. Veriyoluna bir değer sürmek için bir seferde yalnızca bir çipin etkinleştirme sinyalinin ileri sürmesine izin verilir. Diğer yongalar, bellekle konuşan yonga ile çekişmeye neden olmaması için kayan çıktılar üretmelidir. Herhangi bir çip, herhangi bir zamanda paylaşılan veriyolundaki bilgileri okuyabilir. Bu tür üç durumlu otobüsler bir zamanlar yaygındı. Bununla birlikte, modern bilgisayarlarda, çiplerin paylaşılan bir veri yolu yerine doğrudan birbirine bağlandığı noktadan noktaya bağlantılarla daha yüksek hızlar mümkündür.



2.7 Harita Yöntemi ile Sadeleştirme(Karnaugh Map Minimization)

Boole cebirini kullanarak Boole denklemlerinin birkaç minimizasyonu üzerinde çalıştıktan sonra, dikkatli olmazsanız, bazen basitleştirilmiş bir denklem yerine tamamen farklı bir denklemle karşılaşacağınızı fark edeceksiniz. Karnaugh haritaları (K-haritaları), Boole denklemlerini basitleştirmek için grafiksel bir yöntemdir. 1953 yılında Bell Labs'ta bir telekomünikasyon mühendisi olan Maurice Karnaugh tarafından icat edildi. K-haritaları, dört değişkene kadar olan problemlerde iyi çalışır. Daha da önemlisi, Boole denklemlerini manipüle etme konusunda fikir verirler.

Mantık küçültmenin terimleri birleştirmeyi içerdiğini hatırlayın. Dolaylı bir P içeren iki terim ve bazı değişken A'nın gerçek ve tamamlayıcı formları, A'yı ortadan kaldırmak

için birleştirilir: $PA + PA' = P$ Karnaugh haritaları, bu birleştirilebilir terimleri bir ızgarada yan yana koyarak görmeyi kolaylaştırır.

Şekil 2.43, üç girişli bir fonksiyon için doğruluk tablosunu ve K-haritasını gösterir. K haritasının üst satırı, A ve B girişleri için dört olası değeri verir. Sol sütun, C girişi için iki olası değeri verir. K-haritasındaki her kare, doğruluk tablosundaki bir satıra karşılık gelir ve o satır için Y çıktısının değerini içerir. Örneğin, sol üst kare, doğruluk tablosundaki ilk satıra karşılık gelir ve $ABC = 000$ olduğunda çıktı değeri $Y = 1$ olduğunu belirtir. Tıpkı doğruluk tablosundaki her satır gibi, bir K-haritasındaki her kare tek bir mintermi temsil eder. Açıklama amacıyla, Şekil 2.43(c), K-haritasındaki her kareye karşılık gelen mintermi göstermektedir.

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

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

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

Her kare veya minterm, tek bir değişkendeki değişikliklerle bitişik kareden farklıdır. Bu, bitişik karelerin, biri dışında, bir karede gerçek biçimde ve diğerinde tamamlayıcı biçimde görünen tüm değişmez değerleri paylaştığı anlamına gelir. Örneğin, $A'B'C'$ ve $A'B'C$ mintermlerini temsil eden kareler bitişiktir ve yalnızca C değişkeninde farklılık gösterir. Üst sıradaki A ve B kombinasyonlarının tuhaf bir sırada olduğunu fark etmiş olabilirsiniz: 00, 01, 11, 10. Bu düzene Gri kod denir. Sıradan ikili düzenden (00, 01, 10, 11) farklıdır, çünkü bitişik girişler yalnızca tek bir değişkende farklılık gösterir. Örneğin, 01: 11 yalnızca A'yı 0'dan 1'e değiştirirken, 01: 10, A'yı 1'den 0'a ve B'yi 0'dan 1'e değiştirir. Dolayısıyla, kombinasyonları ikili sırayla yazmak, bitişik kareler için istediğimiz özelliği üretmezdi. sadece bir değişkende farklılık gösterir.

K-haritası da "etrafını sarar". En sağdaki kareler, en soldaki karelere etkin bir şekilde bitişiktir, çünkü bunlar yalnızca bir değişken olan A'da farklılık gösterirler. Diğer bir deyişle, haritayı alıp bir silindire yuvarlayabilir, sonra da karelerin uçlarını birleştirebilirsiniz. bir simit (yani bir halka) oluşturmak için silindir ve yine de bitişik karelerin yalnızca bir değişkende farklılık göstereceğini garanti eder.

2.7.1 Döngüsel Düşünme

Şekil 2.43, K-haritasında, sol sütundaki 1'lerle gösterildiği gibi, denklemde yalnızca iki minterm vardır, $A'B'C'$ ve $A'B'C$. Mintermleri K-haritasından okumak, çarpımların toplamı formundaki denklemleri doğrudan doğruluk tablosundan okumaya tam olarak eşdeğerdir.

Daha önce olduğu gibi, çarpımların toplamı formundaki denklemleri en aza indirmek için Boole cebirini kullanabiliriz.

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

K haritaları, Şekil 2.44 gösterildiği gibi, 1'leri bitişik karelerde daire içine alarak bu basitleştirmeyi grafiksel olarak yapmamıza yardımcı olur. Her daire için karşılık gelen implikantı yazıyoruz. Bölüm 2.2'den bir dolaylı ifadenin bir veya daha fazla değişmezin ürünü olduğunu unutmayın. Hem gerçek hem de tamamlayıcı formları daire içinde olan değişkenler, imanın dışında bırakılır. Bu durumda, C değişkeni hem gerçek formuna (1) hem de çemberde tamamlayıcı formuna (0) sahiptir, bu nedenle onu implicant'a dahil etmiyoruz. Başka bir deyişle, C'den bağımsız olarak $A = B = 0$ olduğunda Y DOĞRUDUR. Dolayısıyla, sonuç $A'B'$ 'dir. K-haritası Boole cebri kullanarak ulaştığımız aynı cevabı verir.

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

2.7.2 K-Maps ile Mantık Minimizasyonu

K haritaları, mantığı en aza indirmek için kolay bir görsel yol sağlar. Mümkün olan en az sayıda daire kullanarak haritadaki 1'lerin tüm dikdörtgen bloklarını daire içine almanız yeterlidir. Her daire mümkün olduğu kadar geniş olmalıdır. Daha sonra daire içine alınmış sonuçları okuyun.

Daha resmi olarak, bir Boole denkleminin en aza indirildiğini hatırlayın. en az sayıda asal çarpımın toplamı olarak yazılır. K-haritasındaki her daire bir içeriği temsil eder. Olası en büyük çemberler birincil çıkarımlardır.

Örneğin, Şekil 2.44 K-haritasında, $A'B'C'$ ve $A'B'C$, implikantlardır, ancak asal çıkarımlar değildir. Sadece $A'B'$, bu K-haritasında en önemli etkidir. K-haritasından küçültülmüş denklem bulma kuralları aşağıdaki gibidir:

- 1'leri kaplamak için gereken en az daireyi kullanın.
- Her dairedeki tüm kareler 1'leri içermelidir.
- Her daire, her yönde 2 (yani 1, 2 veya 4) karenin üssü olan dikdörtgen bir bloğu kapsamalıdır.
- Her daire mümkün olduğu kadar büyük olmalıdır.
- K-haritasının kenarlarını bir daire sarabilir.

- K-haritasındaki bir 1, daha az dairenin kullanılmasına izin veriyorsa, birden çok kez daire içine alınabilir

Örnek 2.9 Şekil 2.45 K-haritasıyla $Y = F(A, B, C)$ fonksiyonumuz olduğunu varsayalım. K-haritasını kullanarak denklemi en aza indirin.

		AB			
		00	01	11	10
C	0	1	0	1	1
	1	1	0	0	1

Çözüm: Şekil 2.46 gösterildiği gibi, mümkün olduğunca az daire kullanarak K-haritasındaki 1'leri daire içine alın. K-haritasındaki her daire bir asal anlamı temsil eder ve her dairenin boyutu ikinin bir kuvvetidir (2×1 ve 2×2). Çemberde sadece doğru olarak veya sadece tamamlayıcı biçimde görünen değişkenleri yazarak, her çember için birincil anlamı oluştururuz.

Örneğin, 2×1 çemberde, B'nin gerçek ve tamamlayıcı biçimleri çembere dahil edilmiştir, bu nedenle B'yi asal implicant'a dahil etmiyoruz. Bununla birlikte, yalnızca A (A) 'nın gerçek formu ve C (C') nin tamamlayıcı formu bu çemberdedir, bu nedenle bu değişkenleri asal dolaylı AC' 'ye dahil ederiz. Benzer şekilde, 2×2 çemberi $B = 0$ olan tüm kareleri kapsar, yani asıl önemli olan B' 'dir.

		AB			
		00	01	11	10
C	0	1	0	1	1
	1	1	0	0	1

$Y = AC' + \bar{B}$

Sağ üstteki karenin (minterm) iki kez nasıl kaplandığına dikkat edin ve birincil dolaylı daireleri olabildiğince geniş yapın. Boole cebri tekniklerinde gördüğümüz gibi, bu, implikantın boyutunu küçültmek için bir mintermi paylaşmaya eşdeğerdir. Ayrıca, dört kareyi kaplayan dairenin K-haritasının kenarlarını nasıl çevrelediğine dikkat edin.

Örnek 2.10 Yedi bölümlü bir ekran kod çözücü, 4 bitlik bir veri girişi D3:0 alır ve 0'dan 9'a kadar bir rakam görüntülemek için ışık yayan diyotları kontrol etmek için yedi çıkış üretir. Yedi çıkışa genellikle a'dan g'ye kadar olan bölümler veya Sa-Sg denir , Şekil 2.47 tanımlandığı gibi. Rakamlar Şekil 2.48 gösterilmektedir. Çıktılar için bir doğruluk tablosu yazın ve Sa ve Sb çıktıları için Boole denklemlerini bulmak için K-haritalarını kullanın. Geçersiz giriş değerlerinin (10-15) boş bir okuma ürettiğini varsayın.

Çözüm: Doğruluk tablosu Tablo 2.6 verilmiştir. Örneğin, 0000 girişi S_g hariç tüm segmentleri açmalıdır.

Table 2.6 Seven-segment display decoder truth table

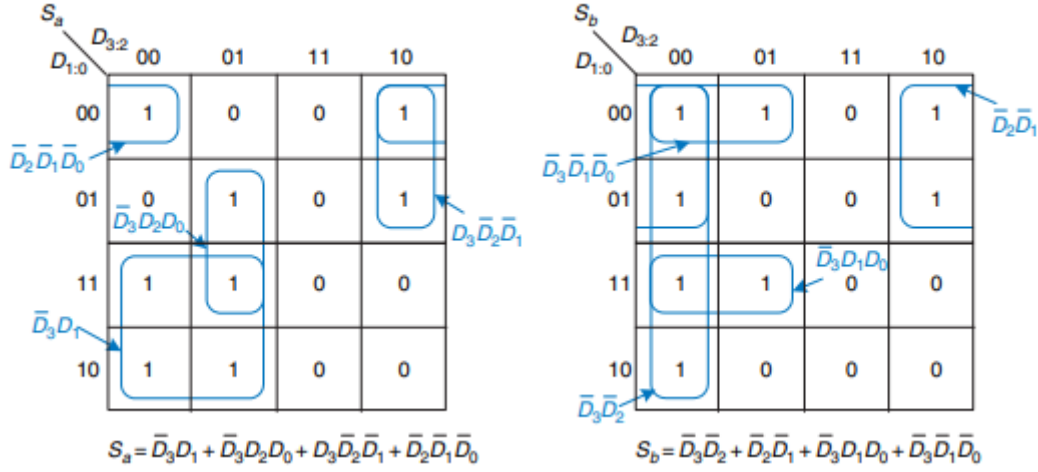
$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
others	0	0	0	0	0	0	0

Yedi çıktının her biri, dört değişkenli bağımsız bir işlevdir. S_a ve S_b çıktıları için K-haritaları Şekil 2.49 gösterilmektedir. Bitişik karelerin yalnızca tek bir değişkende farklılık gösterebileceğini unutmayın, bu nedenle satırları ve sütunları Gray kodu sırasına göre etiketleriz: 00, 01, 11, 10. Çıktı değerlerini karelere girerken bu sıralamayı da hatırlamaya dikkat edin.

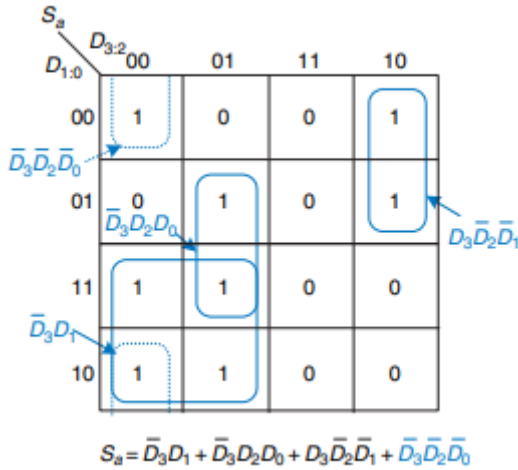
S_a	$D_{3:2}$	00	01	11	10
$D_{1:0}$	00	1	0	0	1
	01	0	1	0	1
	11	1	1	0	0
	10	1	1	0	0

S_b	$D_{3:2}$	00	01	11	10
$D_{1:0}$	00	1	1	0	1
	01	1	0	0	1
	11	1	1	0	0
	10	1	0	0	0

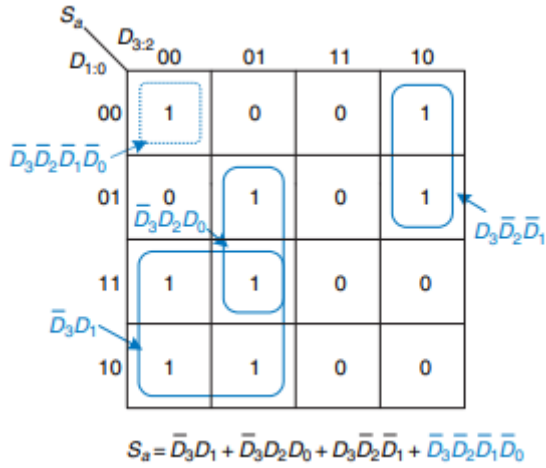
Sonra, asal sonuçları daire içine alın. Tüm 1'leri kapsayacak en az sayıda daire kullanın. Bir daire kenarların (dikey ve yatay) etrafını sarabilir ve 1 birden fazla daire içine alınabilir. Şekil 2.50, asal çarpımları ve basitleştirilmiş Boole denklemlerini gösterir.



Asal ifadelerin minimal kümesinin benzersiz olmadığına dikkat edin. Örneğin, S_a K-haritasındaki 0000 girişi, $D'_2D'_1D'_0$ mintermini üretmek için 1000 girişiyle birlikte daire içine alındı. Daire, bunun yerine 0010 girişini içerebilir ve Şekil 2.51'de kesikli çizgilerle gösterildiği gibi bir $D_3D_2D_0$ minterm üretebilirdi.



Şekil 2.52 (bkz. sayfa 82), sol üst köşedeki 1'i kapatmak için asal olmayan bir dolaylının seçildiği yaygın bir hatayı göstermektedir. Bu minterm, $D'_3D'_2D'_1D'_0$, minimal olmayan bir çarpımlar toplamı denklemini verir. Minterm, önceki iki şekilde yapıldığı gibi, daha büyük bir daire oluşturmak için bitişik olanlardan herhangi biriyle birleştirilebilirdi.



2.7.3 Önemsenmeyen Durumlar

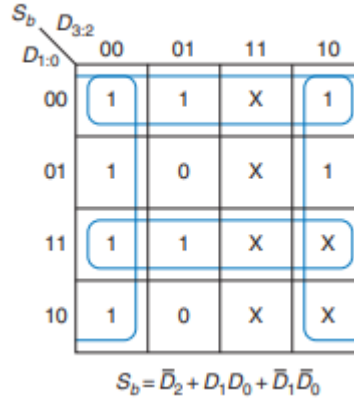
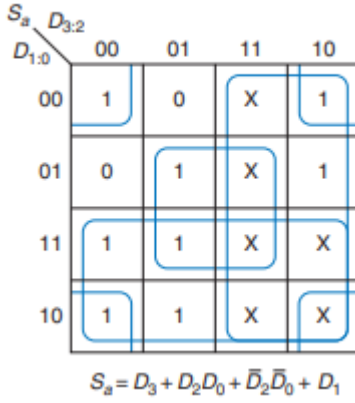
Doğruluk tablosu girdileri için "umursamıyorum" girişlerinin, bazı değişkenler çıktıyı etkilemediğinde tablodaki satır sayısını azaltmak için Bölüm 2.4'te tanıtıldığını hatırlayın. X simgesiyle gösterilirler, bu da girişin 0 veya 1 olabileceği anlamına gelir.

Önemsemeyin, çıktı değerinin önemsiz olduğu veya karşılık gelen girdi kombinasyonunun asla gerçekleşmeyeceği doğruluk tablosu çıktılarında da görünür. Bu tür çıktılar, tasarımcının takdirine bağlı olarak 0'lar veya 1'ler olarak ele alınabilir.

Bir K-haritasında, X'ler daha fazla mantık küçültmeye izin verir. 1'leri daha az veya daha büyük dairelerle kapatmaya yardımcı olurlarsa daire içine alınabilir, ancak yardımcı olmadıklarında daire içine alınmaları gerekmez.

Örnek 2.11 10 ila 15 arasındaki geçersiz girdi değerleri için çıktı değerleri umurumuzda değilse Örnek 2.10'u tekrarlayın.

Çözüm: K-haritası, Şekilde 2.53 umursamadığını temsil ettiği şekilde gösterilmiştir. Önemsizler 0 veya 1 olabileceğinden, 1'leri daha az veya daha büyük dairelerle kaplamamıza izin verip vermediğini umursamıyoruz. Daire içine alınmış umursamazlar 1'ler olarak kabul edilirken, daire içine alınmış olmayanlar 0'lar olarak kabul edilir. Sa segmenti için dört köşenin etrafına 2×2 kare sarmanın nasıl daire içine alındığını gözlemleyin. Önemsizlerin kullanılması mantığı büyük ölçüde basitleştirir.



2.7.4 Büyük Resim

Boole cebri ve Karnaugh haritaları, mantık sadeleştirmenin iki yöntemidir. Nihayetinde amaç, belirli bir mantık işlevini uygulamak için düşük maliyetli bir yöntem bulmaktır.

Modern mühendislik uygulamalarında, mantık sentezleyicileri olarak adlandırılan bilgisayar programları, Bölüm 4'te göreceğimiz gibi, mantık işlevinin bir tanımından basitleştirilmiş devreler üretir. Büyük problemler için, mantık sentezleyicileri insanlardan çok daha etkilidir. Küçük sorunlar için, biraz deneyime sahip bir insan teftişle iyi bir çözüm bulabilir. Yazarların hiçbirisi gerçek hayatta pratik bir problemi çözmek için bir Karnaugh haritasını kullanmadı. Ancak Karnaugh haritalarının altında yatan ilkelerden elde edilen içgörü değerlidir. Ve Karnaugh haritaları genellikle iş görüşmelerinde görünür!

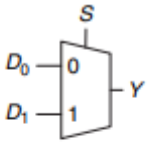
Birlesik Lojik Blokları

Birleşik mantık, daha karmaşık sistemler oluşturmak için genellikle daha büyük yapı blokları halinde gruplandırılır. Bu, yapı bloğunun işlevini vurgulamak için gereksiz kapı seviyesi ayrıntılarını gizleyen soyutlama ilkesinin bir uygulamasıdır. Bu tür üç yapı bloğunu zaten inceledik: tam toplayıcılar (Bölüm 2.1'den), öncelik devreleri (Bölüm 2.4'ten) ve yedi bölümlü ekran kod çözücüler (Bölüm 2.7'den). Bu bölüm daha yaygın olarak kullanılan iki yapı taşını tanıtır: çoklayıcılar ve kod çözücüler. Bölüm 5, diğer kombinasyonel yapı bloklarını kapsar.

2.8.1 Çoklayıcılar

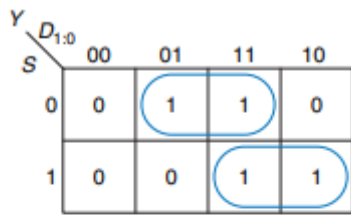
Çoklayıcılar en yaygın kullanılan birleşik mantık devrelerindendir. Seçim sinyalinin değerine bağlı olarak birkaç olası giriş arasından bir çıkış seçerler. Bir çoklayıcıya bazen kısaca mux denir.

2:1 Çoklayıcı Şekil 2.54, iki veri girişi D0 ve D1, bir seçim girişi S ve bir çıkış Y olan 2:1 çoklayıcı için şematik ve doğruluk tablosunu göstermektedir. Çoklayıcı, seçime göre iki veri girişi arasında seçim yapar: S = 0 ise, Y = D0 ve S = 1 ise Y = D1 S, çoklayıcının ne yaptığını kontrol ettiği için kontrol sinyali olarak da adlandırılır.



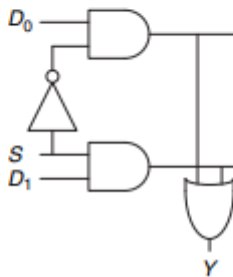
S	D1	D0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Şekilde 2.55 gösterildiği gibi 2:1 çoklayıcı, çarpımların toplamı mantığından oluşturulabilir. Çoklayıcı için Boole denklemleri bir Karnaugh haritası ile türetilir veya inceleme ile okunabilir (S=0 VE D0=1 ise Y=1'dir VEYA S=1 VE D1=1 Y=1 dir).

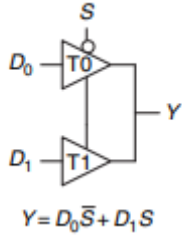


S \ D1:0	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$Y = D_0 \bar{S} + D_1 S$



Alternatif olarak, çoklayıcılar, Şekilde 2.56 gösterildiği gibi üç durumlu tamponlardan oluşturulabilir. Üç durumlu etkinleştirmeler, her zaman tam olarak bir üç durumlu tampon aktif olacak şekilde düzenlenmiştir. S = 0 olduğunda, üç durumlu T0 etkinleştirilir ve D0'in Y'ye akmasına izin verilir. S = 1 olduğunda, üç durumlu T1 etkinleştirilir ve D1'in Y'ye akmasına izin verilir.



Daha Geniş Çoklayıcılar Şekil 2.57 gösterildiği gibi 4: 1 çoklayıcıda dört veri girişi ve bir çıkış vardır. Dört veri girişi arasından seçim yapmak için iki seçim sinyali gereklidir. 4: 1 çoklayıcı, Şekil 2.58 gösterildiği gibi, çarpımların toplamı mantığı, üç durumlu veya birden çok 2: 1 çoklayıcı kullanılarak oluşturulabilir.

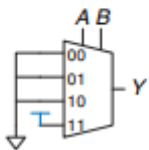
Üçdurumluları mümkün kılan çarpım terimleri, AND geçitleri ve invertörler kullanılarak oluşturulabilir. Ayrıca Bölüm 2.8.2'de tanıtacağımız bir kod çözücü kullanılarak da oluşturulabilirler.

8: 1 ve 16: 1 çoklayıcılar gibi daha geniş çoklayıcılar, Şekil 2.58 gösterilen yöntemler genişletilerek oluşturulabilir. Genel olarak, bir N: 1 çoklayıcı, $\log_2 N$ seçme hatlarına ihtiyaç duyar. Yine, en iyi uygulama seçimi, hedef teknolojiye bağlıdır.

Çoklayıcı Mantığı

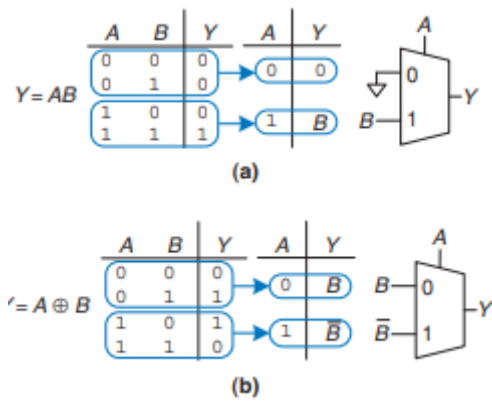
Çoklayıcılar, mantık işlevlerini gerçekleştirmek için bakma tabloları olarak kullanılabilir. Şekil 2.59, iki girişli bir AND geçidi uygulamak için kullanılan 4: 1 çoklayıcıyı göstermektedir. Girişler, A ve B, seçme hatları olarak hizmet eder. Çoklayıcı veri girişleri, doğruluk tablosunun karşılık gelen satırına göre 0 veya 1'e bağlanır. Genel olarak, 2^N girişli bir çoklayıcı, uygun veri girişlerine 0'lar ve 1'ler uygulayarak herhangi bir N girişli mantık işlevini gerçekleştirmek üzere programlanabilir. Aslında, veri girişlerini değiştirerek, çoklayıcı farklı bir işlevi gerçekleştirmek için yeniden programlanabilir.

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

$$Y = AB$$


Biraz akıllıca, herhangi bir N girişli mantık işlevini gerçekleştirmek için yalnızca 2^{N-1} girişli bir çoklayıcı kullanarak çoklayıcı boyutunu ikiye bölebiliriz. Strateji, çoklayıcı veri girişlerine 0 ve 1'lerin yanı sıra değişmez değerlerden birini sağlamaktır.

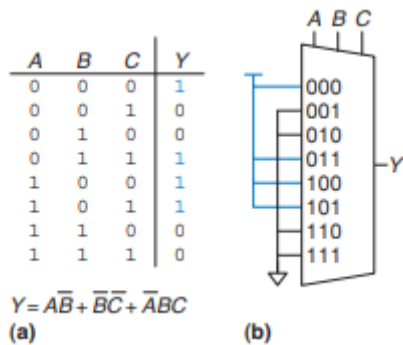
Bu prensibi göstermek için, Şekil 2.60 iki girişli VE ve 2: 1 çoklayıcı ile uygulanan XOR fonksiyonları. Sıradan bir doğruluk tablosu ile başlıyoruz ve sonra en sağdaki girdi değişkenini ortadan kaldırmak için çıktıyı bu değişken cinsinden ifade ederek satır çiftlerini birleştiriyoruz. Örneğin, VE durumunda, $A = 0, Y = 0$, B'den bağımsız olarak, $A = 1$ olduğunda, $Y = 0$, $B = 0$ ve $Y = 1$, $B = 1$ ise, $Y = B$. Çoklayıcıyı yeni, daha küçük doğruluk tablosuna göre bir arama tablosu olarak kullanın.



Örnek 2.12 MULTİPLEKSER İLE MANTIK

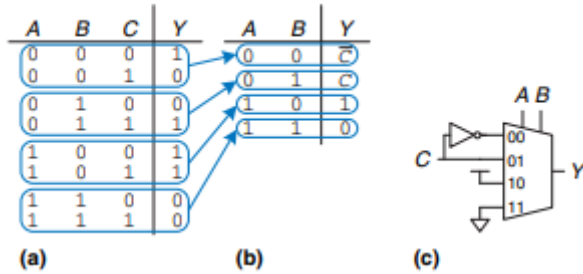
Alyssa P. Hacker'ın kıdemli projesini bitirmek için $Y = AB' + B'C' + A'BC$ işlevini uygulaması gerekiyor, ancak laboratuvar kitine baktığında kalan tek parçası 8: 1 çoklayıcı. İşlevi nasıl uyguluyor?

Çözüm: Şekil 2.61, Alyssa'nın tek bir 8: 1 çoklayıcı kullanan uygulamasını göstermektedir. Çoklayıcı, doğruluk tablosundaki her satırın bir çoklayıcı girişine karşılık geldiği bir arama tablosu görevi görür.



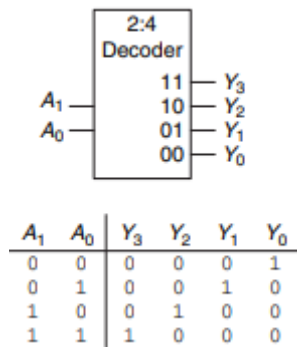
Örnek 2.13 Alyssa, son sunumdan önce devresini bir kez daha açar ve 8: 1 çoklayıcıyı havaya uçurur. (Bütün gece uyumadıktan sonra yanlışlıkla 5 V yerine 20 V ile çalıştırdı.) Arkadaşlarına yedek parça için yalvarıyor ve ona 4: 1 çoklayıcı ve invertör veriyorlar. Devresini sadece bu parçalarla kurabilir mi?

Çözüm: Alyssa, çıktının C'ye bağlı olmasına izin vererek doğruluk tablosunu dört satıra indiriyor (Çıktının A veya B'ye bağlı olmasına izin vermek için doğruluk tablosunun sütunlarını yeniden düzenlemeyi de seçebilirdi) Şekil 2.62 yeni tasarımı göstermektedir.



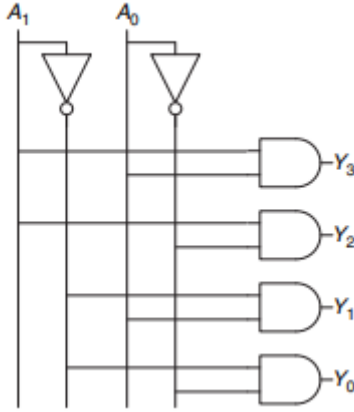
Decoder

Bir kod çözücünün N girişi ve 2^N çıkışı vardır. Giriş kombinasyonuna bağlı olarak çıktılarından tam olarak birini belirtir. Şekil 2.63, 2: 4 kod çözücüyü göstermektedir. $A_{1:0} = 00$ olduğunda, Y_0 1'dir. $A_{1:0} = 01$ olduğunda, Y_1 1'dir. Ve bu böyle devam eder. Çıkışlara tekli olarak adlandırılır çünkü belirli bir zamanda tam olarak biri "sıcak" (YÜKSEK) olur.



Örnek 2.14 AND, OR ve NOT kapıları ile 2: 4 kod çözücü uygulayın.

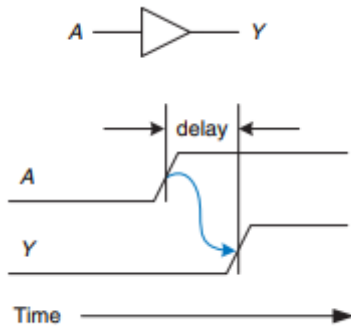
Çözüm: Şekil 2.64, dört AND geçidi kullanan 2: 4 kod çözücünün uygulamasını göstermektedir. Her kapı, her girdinin gerçek veya tamamlayıcı şekline bağlıdır. Genel olarak, bir N: 2^N kod çözücü, doğru veya tamamlayıcı girişlerin çeşitli kombinasyonlarını kabul eden 2^N N-girişli AND geçitlerinden oluşturulabilir. Bir kod çözücüdeki her çıktı, tek bir mintermi temsil eder. Örneğin, Y_0 minterm $A_1'A_0'$ ı temsil eder. Bu gerçek, diğer dijital yapı blokları ile kod çözücüler kullanırken kullanışlı olacaktır.



Zamanlama

Önceki bölümlerde, öncelikle devrenin çalışıp çalışmadığı ile ilgilenmiştik - ideal olarak, en az sayıda geçidi kullanarak. Bununla birlikte, herhangi bir tecrübeli devre tasarımcısının da onaylayacağı gibi, devre tasarımındaki en zorlu konulardan biri zamanlamadır: bir devrenin hızlı çalışmasını sağlamak.

Bir çıkışın, bir giriş değişikliğine yanıt olarak değişmesi zaman alır. Şekil 2.66, bir tampon için bir giriş değişikliği ile sonraki çıkış değişikliği arasındaki gecikmeyi göstermektedir. Şekle zamanlama diyagramı denir; bir giriş değiştiğinde tampon devresinin geçici yanıtını gösterir. DÜŞÜK'ten YÜKSEK'e geçiş yükselen kenar olarak adlandırılır. Benzer şekilde, HIGH'dan LOW'a geçiş (şekilde gösterilmemiştir) düşen kenar olarak adlandırılır. Mavi ok, Y'nin yükselen kenarının A'nın yükselen kenarından kaynaklandığını gösterir. Giriş sinyalinin % 50 noktası olan A'dan çıkış sinyalinin % 50 noktası Y'ye kadar olan gecikmeyi ölçüyoruz. % 50 noktası sinyalin geçiş sırasında LOW ve HIGH değerleri arasında yarı yolda (% 50) olduğu noktadır.

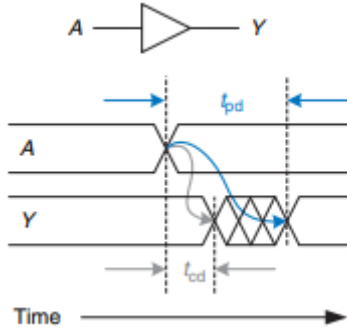


2.9.1 Yayılma ve Kirlenme Gecikmesi

Kombinasyonel mantık, yayılma gecikmesi ve kirlenme gecikmesi ile karakterize edilir. Yayılma gecikmesi t_{pd}, bir girişin değişmesinden çıkış veya çıkışların nihai değerlerine

ulaşmasına kadar geçen maksimum süredir. Kirlenme gecikmesi t_{cd} , bir girişin değiştiği andan herhangi bir çıkışın değerini değiştirmeye başladığı ana kadar geçen minimum süredir.

Şekil 2.67, sırasıyla mavi ve gri olarak bir tamponun yayılma gecikmesini ve kirlenme gecikmesini göstermektedir. Şekil, A'nın başlangıçta YÜKSEK veya DÜŞÜK olduğunu ve belirli bir zamanda diğer duruma değiştiğini gösterir; biz sadece değiştiği gerçeğiyle ilgileniyoruz, sahip olduğu değere değil. Yanıt olarak, Y bir süre sonra değişir. Yaylar, Y'nin, A geçişlerinden sonra t_{cd} 'yi değiştirmeye başlayabileceğini ve Y'nin kesinlikle t_{pd} içinde yeni değerine yerleştiğini gösterir.

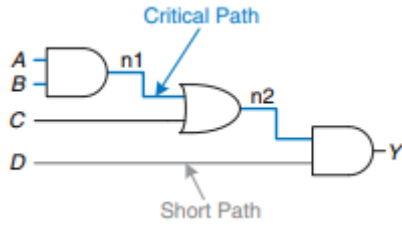


Devrelerdeki gecikmenin altında yatan nedenler, bir devrede kapasitansı şarj etmek için gereken süre ve ışık hızını içerir. t_{pd} ve t_{cd} birçok nedenden dolayı farklı olabilir.

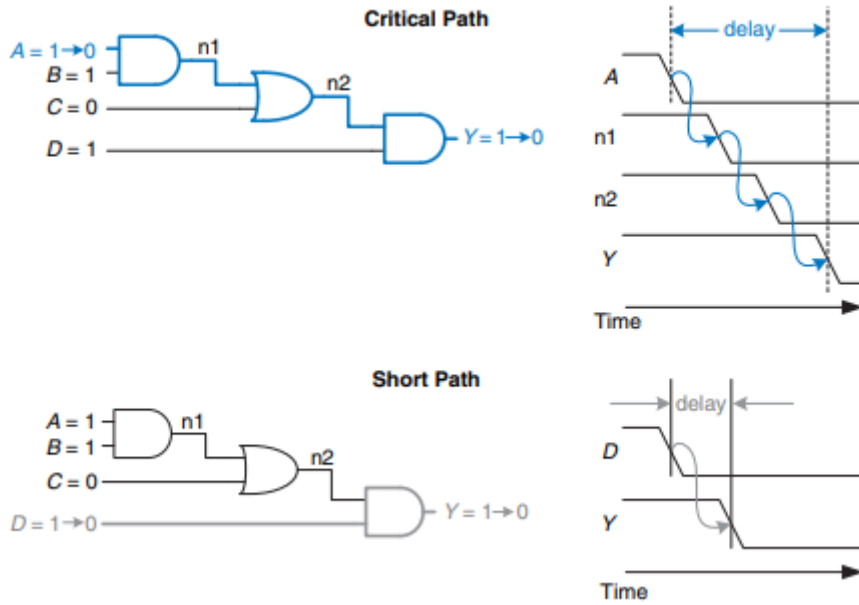
- farklı yükselme ve düşme gecikmeleri
- bazıları diğerlerinden daha hızlı olan birden fazla giriş ve çıkış
- sıcakken yavaşlayan ve soğukken hızlanan devreler

t_{pd} ve t_{cd} 'yi hesaplamak, bu kitabın kapsamının ötesinde soyutlamanın daha düşük seviyelerine inmeyi gerektirir. Bununla birlikte, üreticiler normalde her kapı için bu gecikmeleri belirten veri sayfaları sağlar.

Halihazırda listelenen faktörlerin yanı sıra, yayılma ve kirlenme gecikmeleri de bir sinyalin girişten çıkışa geçtiği yol tarafından belirlenir. Şekil 2.68, dört girişli bir mantık devresini göstermektedir. Mavi ile gösterilen kritik yol, A veya B girişinden Y çıkışına giden yoldur. Bu, en uzun ve bu nedenle en yavaş yoldur, çünkü giriş üç kapıdan çıkışa doğru ilerler. Bu yol kritiktir çünkü devrenin çalıştığı hızı sınırlar. Gri ile gösterilen devre boyunca kısa yol, D girişinden Y çıkışına kadardır. Bu, devre boyunca en kısa ve dolayısıyla en hızlı yoldur, çünkü giriş, çıkışa yalnızca tek bir kapıdan geçer.

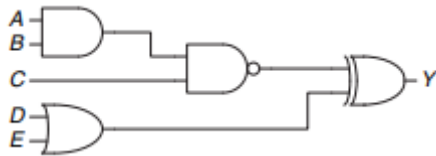


Bir birleşimsel devrenin yayılma gecikmesi, kritik yoldaki her bir elemandan geçen yayılma gecikmelerinin toplamıdır. Kirlenme gecikmesi, kısa yoldaki her bir elemanın kirlenme gecikmelerinin toplamıdır. Bu gecikmeler Şekil 2.69 gösterilmiş ve aşağıdaki denklemlerle açıklanmıştır:



$$t_{pd} = 2t_{pd_AND} + t_{pd_OR} \quad t_{cd} = t_{cd_AND}$$

Örnek 2.15 Ben Bitdiddle'ın, Şekil 2.70 gösterilen devrenin yayılma gecikmesini ve kirlenme gecikmesini bulması gerekir. Veri kitabına göre, her kapının yayılma gecikmesi 100 pikosaniye (ps) ve kirlenme gecikmesi 60 ps'dir.



Çözüm: Ben, kritik yolu ve devre boyunca en kısa yolu bularak başlar. Şekil 2.71 mavi ile vurgulanan kritik yol, A veya B girişinden üç kapıdan Y çıkışına doğrudur. Dolayısıyla, t_{pd} , tek bir geçidin yayılma gecikmesinin üç katı veya 300 ps'dir.

Şekil 2.72 gri olarak gösterilen en kısa yol, C, D veya E girişinden iki kapıdan Y çıkışına kadardır. En kısa yolda yalnızca iki kapı vardır, bu nedenle t_{cd} 120 ps'dir.

Örnek 2.16 Bölüm 2.8.1'deki Şekil 2.58 gösterilen üç dört girişli çoklayıcı tasarımının en kötü durum zamanlamasını karşılaştırın. Tablo 2.7 , bileşenler için yayılma gecikmelerini listeler. Her tasarım için kritik yol nedir? Zamanlama analizinize göre, neden bir tasarımı diğerine tercih edebilirsiniz?

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

Çözüm: Üç tasarım seçeneğinin her biri için kritik yollardan biri Şekil 2.73 ve 2.74 mavi ile vurgulanmıştır. t_{pd_sy} , S girişinden Y çıkışına yayılma gecikmesini belirtir; t_{pd_dy} , D girişinden Y çıkışına yayılma gecikmesini belirtir; t_{pd} , ikisinin en kötüsüdür: $\max(t_{pd_sy}, t_{pd_dy})$.

Şekil 2.73 hem iki seviyeli mantık hem de üç durumlu uygulamalar için kritik yol, kontrol sinyallerinden S birinden Y çıkışına kadardır: $t_{pd} = t_{pd_sy}$. Bu devreler kontrol açısından kritiktir, çünkü kritik yol kontrol sinyallerinden çıkışa kadardır. Kontrol sinyallerindeki herhangi bir ek gecikme, doğrudan en kötü durum gecikmesine eklenecektir. Şekil 2.73(b) D'den Y'ye gecikme, 125 ps'lik S'den Y'ye gecikme ile karşılaştırıldığında sadece 50 ps'dir.

Şekil 2.74, 2: 1 çoklayıcıların iki aşamasını kullanan 4: 1 çoklayıcının hiyerarşik uygulamasını göstermektedir. Kritik yol, herhangi bir D girişinden çıktıya kadardır. Bu devre veri açısından kritiktir çünkü kritik yol veri girişinden çıkışa kadardır: $t_{pd} = t_{pd_dy}$.

Veri girişleri kontrol girişlerinden çok önce gelirse, en kısa kontrolden çıkışa gecikmeli tasarımı tercih ederiz (şekil 2.74). Benzer şekilde, kontrol girişleri veri girişlerinden çok önce gelirse, veriden çıkışa en kısa gecikmeli tasarımı tercih ederiz (Şekil 2.73(b)).

En iyi seçim, yalnızca devredeki kritik yola ve giriş varış sürelerine değil, aynı zamanda parçaların gücüne, maliyetine ve kullanılabilirliğine de bağlıdır.

Glitches (Tek girisin çoklu çıkışı tetiklemesi)

Şimdiye kadar, tek bir giriş geçişinin tek bir çıktı geçişine neden olduğu durumu tartıştık. Ancak, tek bir giriş geçişinin birden çok çıkış geçişine neden olması mümkündür. Bunlara aksaklıklar veya tehlikeler denir. Aksaklıklar genellikle sorun yaratmasa da, bunların var olduğunu anlamak ve zamanlama diyagramlarına bakarken onları tanımak önemlidir. Şekil 2.75 arızalı bir devreyi ve devrenin Karnaugh haritasını göstermektedir.

Boole denklemi doğru şekilde küçültülmüştür, ancak $A = 0$, $C = 1$ ve B 1'den 0'a geçiş yaptığında ne olduğuna bakalım. Şekil 2.76 bu senaryoyu göstermektedir. Kısa yol (gri olarak gösterilmiştir) iki kapıdan, AND ve OR kapılarından geçer. Kritik yol (mavi ile gösterilmiştir) bir tersleyici ve iki kapıdan, AND ve OR kapılarından geçer.

B 1'den 0'a geçerken, n_2 (kısa yolda), n_1 'in (kritik yolda) yükselmesinden önce düşer. n_1 yükselene kadar, OR geçidinin iki girişi 0'dır ve Y çıkışı 0'a düşer. n_1 sonunda yükseldiğinde, Y 1'e döner. Şekil 2.76 zamanlama diyagramında gösterildiği gibi, Y 1'de başlar ve 1'de biter ancak anlık olarak 0'a geçer.

Çıktıya bağlı kalmadan önce yayılma gecikmesinin geçmesini beklediğimiz sürece, hatalar bir sorun değildir, çünkü çıktı sonunda doğru cevaba yerleşir.

İstersek, uygulamaya başka bir kapı ekleyerek bu aksaklıktan kaçınabiliriz. Bu, K-haritası açısından anlaşılması en kolay yoldur. Şekil 2.77, B üzerindeki $ABC = 001$ 'den $ABC = 011$ 'e bir giriş geçişinin bir asal dolaylı çemberden diğerine nasıl hareket ettiğini gösterir. K-haritasındaki iki temel etkenin sınırını aşan geçiş, olası bir aksaklığı gösterir.

Şekil 2.76 önceki'daki zamanlama diyagramından da gördüğümüz gibi, birincil implikantlardan birini uygulayan devre, diğer birincil implikantın devresi açılmadan önce kapanırsa, bir aksaklık vardır. Bunu düzeltmek için, Şekil 2.78 gösterildiği gibi, bu birincil dolaylı sınırı kapsayan başka bir daire ekliyoruz. Bunu, eklenen terim olan AC 'nin fikir birliği veya gereksiz terim olduğu konsensüs teoremi olarak tanıyabilirsiniz.

Şekil 2.79, aksaklığa dayanıklı devreyi göstermektedir. Eklenen AND kapısı mavi ile vurgulanır. Şimdi, $A = 0$ ve $C = 1$ olduğunda B üzerindeki bir geçiş, çıkışta bir hataya neden olmaz, çünkü mavi AND geçidi, geçiş boyunca 1 çıktısı verir.

Genel olarak, tek bir değişkendeki bir değişiklik bir K-haritasındaki iki asal çarpım arasındaki sınırı geçtiğinde bir aksaklık meydana gelebilir. Bu sınırları kapatmak için K-haritasına fazlalık çıkarımlar ekleyerek aksaklığı ortadan kaldırabiliriz. Bu, elbette ekstra donanım maliyetiyle gelir.

Bununla birlikte, çoklu girişlerdeki eşzamanlı geçişler de hatalara neden olabilir. Bu aksaklıklar, donanım eklenerek düzeltilemez. İlginç sistemlerin büyük çoğunluğunun çoklu girişler üzerinde eşzamanlı (veya neredeyse eşzamanlı) geçişlere sahip olması

nedeniyle, hatalar çoğu devrede hayatın bir gerçeğidir. Bir tür aksaklığı nasıl ortadan kaldıracığımızı göstermiş olsak da, aksaklıkları tartışmanın amacı onları ortadan kaldırmak değil, var olduklarının farkında olmaktır. Bu, özellikle bir simülatör veya osiloskop üzerindeki zamanlama diyagramlarına bakarken önemlidir.

SIRALI MANTIK

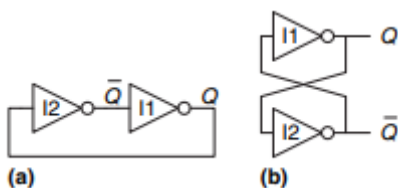
Son bölümde, birleşimsel mantığın nasıl analiz edilip tasarlanacağını gösterdik. Birleşimsel mantığın çıktısı yalnızca mevcut giriş değerlerine bağlıdır. Doğruluk tablosu veya Boole denklemi verildiğinde, gereklilikleri karşılamak için optimize edilmiş bir devre oluşturabiliriz.

Bu bölümde, sıralı mantığı analiz edip tasarlayacağız. Sıralı mantığın çıktıları hem mevcut hem de önceki girdi değerlerine bağlıdır. Dolayısıyla, sıralı mantık hafızaya sahiptir. Sıralı mantık, önceki belirli girdileri açıkça hatırlayabilir veya önceki girdileri, sistemin durumu adı verilen daha küçük bir bilgi miktarına dömtürür. Bir dijital sıralı devrenin durumu, devrenin gelecekteki davranışını açıklamak için gerekli geçmişle ilgili tüm bilgileri içeren, durum değişkenleri adı verilen bir bit kümesidir.

Bölüm, bir bitlik durumu depolayan basit ardışık devreler olan lath ve flip-flop inceleyerek başlar. Genel olarak, sıralı devreleri analiz etmek karmaşıktır. Tasarımı basitleştirmek için, yalnızca kombinasyonel mantık ve devrenin durumunu içeren flip-flop sıralarından oluşan eşzamanlı sıralı devreler inşa etmek için kendimizi disipline ediyoruz. Bölüm, ardışık devreleri tasarlamamanın kolay bir yolu olan sonlu durum makinelerini açıklamaktadır. Son olarak, sıralı devrelerin hızını analiz ediyoruz ve hızı artırmanın bir yolu olarak paralelliği tartışıyoruz.

3.2 Mandallar ve flip-flop lar

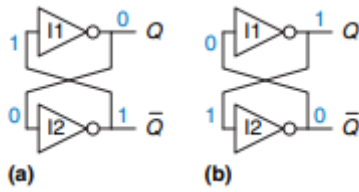
Belleğin temel yapı taşı, iki kararlı duruma sahip bir öğedir. Şekil sol, bir döngüye bağlanmış bir çift eviriciden oluşan basit bir iki durumlu elemanı göstermektedir. Şekil 3.1, simetriyi vurgulamak için aynı devreyi yeniden çizdi. tersleyiciler çapraz bağlanmıştır, yani I1'in girişi I2'nin çıkışıdır ve bunun tersi de geçerlidir. Devrenin girişi yoktur, ancak Q ve Q' olmak üzere iki çıkışı vardır. Bu devreyi analiz etmek, döngüsel olduğu için kombinasyonel bir devreyi analiz etmekten farklıdır: Q, Q' ya bağlıdır ve Q', Q ya bağlıdır.



İki durumu ele alalım, Q , 0 veya Q , 1'dir. Her bir durumun sonuçları üzerinde çalışarak, elimizde:

Durum I: $Q = 0$ Şekil 3.2 (a) 'da gösterildiği gibi, I2 bir YANLIŞ girişi, Q alır, bu nedenle Q' üzerinde bir DOĞRU çıktı üretir: I1 bir DOĞRU girdi alır, Q' , bu nedenle Q üzerinde YANLIŞ çıktı üretir. Bu, orijinal varsayımla tutarlıdır. Bu $Q = 0$, dolayısıyla durumun kararlı olduğu söyleniyor.

Durum II: $Q = 1$ Şekil 3.2 (b) 'de gösterildiği gibi, I2 bir DOĞRU girdi alır ve bir Q' YANLIŞ çıktı: I1 bir YANLIŞ girdi alır ve bir DOĞRU üretir Q üzerinde çıktı. Bu yine kararlı.



Çapraz bağlı invertörlerin iki kararlı durumu olduğundan, $Q = 0$ ve $Q = 1$, devrenin iki kararlı olduğu söylenir. İnce bir nokta, devrenin her iki çıkışının da yaklaşık olarak 0 ile 1 arasında üçüncü bir olası duruma sahip olmasıdır. Buna yarı kararlı durum denir ve Bölüm 3.5.4'te tartışılacaktır.

N kararlı duruma sahip bir eleman, $\log_2 N$ bit bilgi taşır, böylece iki durumlu bir eleman bir bit depolar. Çapraz bağlanmış eviricilerin durumu tek bir ikili durum değişkeni olan Q 'da bulunur. Q 'nın değeri bize devrenin gelecekteki davranışını açıklamak için gerekli olan geçmiş hakkında her şeyi anlatır. Spesifik olarak, eğer $Q = 0$ ise sonsuza kadar 0 olarak kalacak ve $Q = 1$ ise sonsuza kadar 1 kalacaktır. Devrenin başka bir Q' düğümü vardır, ancak Q' herhangi bir ek bilgi içermez çünkü Q biliniyorsa, Q' da bilinir. Öte yandan Q' , durum değişkeni için de kabul edilebilir bir seçimdir.

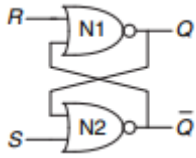
Sıralı bir devreye güç ilk uygulandığında, başlangıç durumu bilinmemektedir ve genellikle öngörülemez. Devre her açıldığında farklılık gösterebilir.

Çapraz bağlanmış invertörler biraz bilgi depolayabilmesine rağmen, pratik değildir çünkü kullanıcının durumu kontrol etmek için herhangi bir girişi yoktur. Bununla birlikte, mandallar ve flip-floplar gibi diğer iki dengeli elemanlar, durum değişkeninin değerini kontrol etmek için girdiler sağlar. Bu bölümün geri kalanı bu devreleri ele almaktadır.

3.2.1 SR Latch

En basit sıralı devrelerden biri, Şekilde 3.3 gösterildiği gibi, iki çapraz bağlı NOR geçidinden oluşan SR mandalıdır. Mandalın S ve R olmak üzere iki girişi ve Q ve Q'

olmak üzere iki çıkışı vardır: SR mandalı çapraz bağlanmış invertörlere benzer, ancak durumu, Q çıkışını ayarlayan ve sıfırlayan S ve R girişleri aracılığıyla kontrol edilebilir.



Tanıdık olmayan bir devreyi anlamamanın iyi bir yolu, doğruluk tablosunu çalışmaktır, böylece başladığımız yer burasıdır. Bir NOR geçidinin herhangi bir giriş TRUE olduğunda bir FALSE çıkışı ürettiğini hatırlayın. Olası dört R ve S kombinasyonunu düşünün.

Durum I: $R = 1, S = 0$ N1, en az bir DOĞRU girdi, R görür, bu nedenle Q'da YANLIŞ çıktı üretir. N2 hem Q hem de S YANLIŞ'ı görür, bu nedenle Q' DOĞRU çıktı üretir:

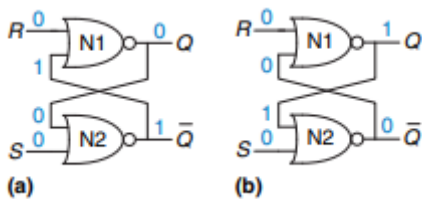
Durum II: $R = 0, S = 1$ N1, 0 ve Q' girişlerini alıyor: Henüz Q' bilmediğimiz için, Q çıkışını belirleyemiyoruz. N2, en az bir DOĞRU girdi, S alıyor, bu nedenle Q' YANLIŞ bir çıktı üretiyor: Şimdi N1'i yeniden ziyaret edebiliriz, her iki girişin de YANLIŞ olduğunu bilerek, Q çıktısı DOĞRU'dur.

Durum III: $R = 1, S = 1$ N1 ve N2'nin her ikisi de en az bir DOĞRU girdi (R veya S) görür, bu nedenle her biri bir YANLIŞ çıktı üretir. Dolayısıyla Q ve Q' ikisi de YANLIŞ.

Durum IV: $R = 0, S = 0$ N1, 0 ve Q' girdilerini alır: Henüz Q' bilmediğimiz için çıktıyı belirleyemiyoruz. N2, 0 ve Q girdilerini alır. Henüz Q bilmediğimiz için çıktıyı belirleyemiyoruz. Şimdi sıkıştık. Bu, çapraz bağlanmış invertörleri anımsatmaktadır. Ancak Q 0 veya 1 değerlerini aldığını biliyoruz. Dolayısıyla, bu alt durumların her birinde ne olduğunu kontrol ederek sorunu çözebiliriz.

Durum IVa: $Q = 0$ S ve Q YANLIŞ olduğundan, N2, Şekil 3.4 (a) 'da gösterildiği gibi, Q' üzerinde bir DOĞRU çıktı üretir. Şimdi N1 bir DOĞRU girdi, Q' alır, dolayısıyla çıktısı, Q, bizim varsaydığımız gibi YANLIŞ olur.

Durum IVb: $Q = 1$ Q DOĞRU olduğu için, N2, Şekil 3.4 (b) 'de gösterildiği gibi, Q' üzerinde YANLIŞ bir çıktı üretir. Şimdi N1, R ve Q olmak üzere iki FALSE girdisi alıyor, dolayısıyla çıktısı, Q, varsaydığımız gibi DOĞRU.

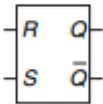


Tüm bunları bir araya getirerek, Q'nun bilinen bir önceki değerinin olduğunu varsayalım, Durum IV'e girmeden önce Qprev adını vereceğiz. Qprev 0 veya 1'dir ve sistemin durumunu temsil eder. R ve S 0 olduğunda, Q hatırlayacaktır bu eski değer, Qprev ve Q' onun tamamlayıcısı olacaktır, Q'prev: Bu devre hafızaya sahiptir.

Şekil 3.5'teki doğruluk tablosu bu dört durumu özetlemektedir. S ve R girişleri Ayarla ve Sıfırla anlamına gelir. Bir bit ayarlamak, onu DOĞRU yapmak demektir. Bir biti sıfırlamak, YANLIŞ yapmak demektir. Çıktılar, Q ve Q', normalde tamamlayıcıdır. R ileri sürüldüğünde, Q 0'a sıfırlanır ve Q' bunun tersini yapar. S ileri sürüldüğünde, Q 1'e ayarlanır ve Q' bunun tersini yapar. Her iki girdi de belirtilmediğinde, Q eski değeri olan Qprev'i hatırlar. Hem S hem de R'yi aynı anda ileri sürmek pek bir anlam ifade etmiyor çünkü bu, mandalın aynı anda ayarlanıp sıfırlanması gerektiği anlamına geliyor ki bu imkansız. Zayıf karışık devre, her iki çıkışı da 0 yaparak yanıt verir.

Case	S	R	Q	\bar{Q}
IV	0	0	Q_{prev}	\bar{Q}_{prev}
I	0	1	0	1
II	1	0	1	0
III	1	1	0	0

SR mandalı, Şekil 3.6'daki sembol ile temsil edilmektedir. Sembolü kullanmak, soyutlama ve modülerliğin bir uygulamasıdır. Farklı mantık kapıları veya transistörler kullanmak gibi bir SR mandalı oluşturmanın çeşitli yolları vardır. Bununla birlikte, Şekil 3.5'teki doğruluk tablosu ve Şekil 3.6'daki sembol ile belirtilen ilişkiye sahip herhangi bir devre elemanına SR mandalı denir.



Çapraz bağlanmış invertörler gibi, SR mandalı, Q'da depolanan bir bitlik durumu olan iki durumlu bir elemandır. Bununla birlikte, durum S ve R girişleri aracılığıyla kontrol edilebilir. R ileri sürüldüğünde, durum 0'a sıfırlanır. S ileri sürüldüğünde, durum 1'e ayarlanır. Hiçbiri ileri sürülmediğinde, durum eski değerini korur. Girişlerin tüm geçmişinin tek durum değişkeni Q tarafından açıklanabileceğine dikkat edin. Geçmişte hangi ayarlama ve sıfırlama modeli olursa olsun, SR mandalının gelecekteki davranışını tahmin etmek için gereken tek şey bunun en yakın zamanda olup olmadığıdır. ayarlayın veya sıfırlayın.

3.2.2 D Latch

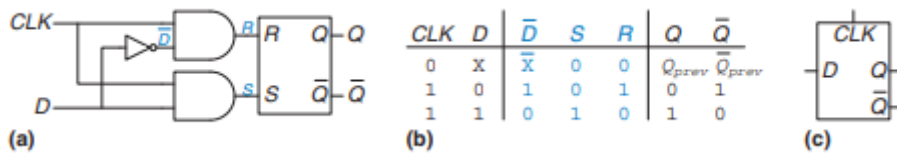
SR mandalı gariptir çünkü hem S hem de R aynı anda öne sürüldüğünde garip davranır. Dahası, S ve R girişleri ne ve ne zaman meselelerini birleştirir. Girdilerden birini ileri sürmek sadece durumun ne olması gerektiğini değil, aynı zamanda ne zaman değişmesi gerektiğini de belirler. Bu sorular ne, ne zaman ayrıldığında

devrelerin tasarlanması kolaylaşır. Şekil 3.7(a) ortadaki mandal bu sorunları çözer. İki girişi vardır. Veri girişi D, bir sonraki durumun ne olması gerektiğini kontrol eder. Saat girişi, CLK, durumun ne zaman değişmesi gerektiğini kontrol eder.

Yine, Şekil 3.7 (b) 'de verilen doğruluk tablosunu yazarak mandalı analiz ediyoruz. Kolaylık sağlamak için önce D, S ve R dahili düğümlerini ele alıyoruz. CLK = 0 ise, D'nin değerine bakılmaksızın hem S hem de R YANLIŞ'tır. CLK = 1 ise, D'nin değerine bağlı olarak bir AND geçidi DOĞRU ve diğeri YANLIŞ üretecektir. S ve R verildiğinde, Q ve Q' Şekil 3.5 kullanılarak belirlenir. CLK = 0 olduğunda, Q'nun eski değeri olan Q_{prev}'i hatırladığını gözlemleyin. CLK = 1 olduğunda, Q = D. Her durumda, mantıksal görüldüğü gibi Q', Q tamamlayıcısıdır. D mandalı, eşzamanlı olarak ileri sürülen R ve S girişlerinin garip durumunu önler.

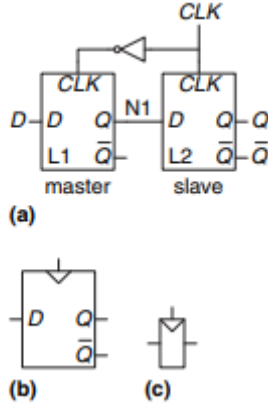
Hepsini bir araya getirdiğimizde, verilerin mandaldan ne zaman aktığını saatin kontrol ettiğini görüyoruz. CLK = 1 olduğunda, mandal şeffaftır. D'deki veriler, sanki mandal sadece bir arabellekmiş gibi Q'ya akar. CLK = 0 olduğunda, mandal donuktur. Yeni verilerin Q'ya akmasını engeller ve Q eski değeri korur. Bu nedenle, D mandalı bazen şeffaf bir mandal veya seviyeye duyarlı bir mandal olarak adlandırılır. D mandalı sembolü Şekil 3.7 (c) 'de verilmiştir.

D mandalı, CLK = 1 iken durumunu sürekli olarak günceller. Bu bölümün ilerleyen kısımlarında, durumu yalnızca belirli bir zamanda güncellenmenin faydalı olduğunu göreceğiz. Bir sonraki bölümde açıklanan D flip-flop tam da bunu yapıyor.



3.2.3 D Flip-Flop

Bir D flip-flop, Şekil 3.8(a) gösterildiği gibi, tamamlayıcı saatler tarafından kontrol edilen iki arka arkaya D mandalından oluşturulabilir. İlk mandala, L1, ana olarak adlandırılır. İkinci mandala, L2, köle olarak adlandırılır. Aralarındaki düğüm N1 olarak adlandırılır. D flip-flopu için bir sembol Şekil 3.8(b) verilmiştir. Q' çıktısına ihtiyaç duyulmadığında, sembol genellikle Şekil 3.8(c) olduğu gibi yoğunlaştırılır.



CLK = 0 olduğunda, ana mandal şeffaftır ve ikincil kilit donuktur. Bu nedenle, D'deki değer ne olursa olsun N1'e yayılır. CLK = 1 olduğunda, master donuklaşır ve slave şeffaf hale gelir. N1'deki değer Q'ya yayılır, ancak N1, D'den kesilir. Bu nedenle, saat 0'dan 1'e yükselmeden hemen önce D'de olan değer, saat yükseldikten hemen sonra Q'ya kopyalanır. Diğer tüm zamanlarda, Q eski değerini korur, çünkü D ile Q arasındaki yolu her zaman kapatan donuk bir mandal vardır.

Başka bir deyişle, bir D flip-flop, saatin yükselen kenarında D'yi Q'ya kopyalar ve diğer tüm zamanlarda durumunu hatırlar. Bu tanımı ezberleyene kadar yeniden okuyun; Yeni başlayan dijital tasarımcıların en yaygın sorunlarından biri, bir flip-flop'un ne yaptığını unutmaktır. Saatin yükselen kenarı, kısalık için genellikle sadece saat kenarı olarak adlandırılır. D girişi, yeni durumun ne olacağını belirtir. Saat kenarı, durumun ne zaman güncellenmesi gerektiğini gösterir.

Bir D flip-flop aynı zamanda bir master-slave flip-flop, bir edge tetiklemeli flip-flop veya bir pozitif kenar-tetiklemeli flip-flop olarak da bilinir. Sembollerdeki üçgen, kenarla tetiklenen bir saat girişini belirtir. Q' çıktısı genellikle ihtiyaç duyulmadığında ihmal edilir.

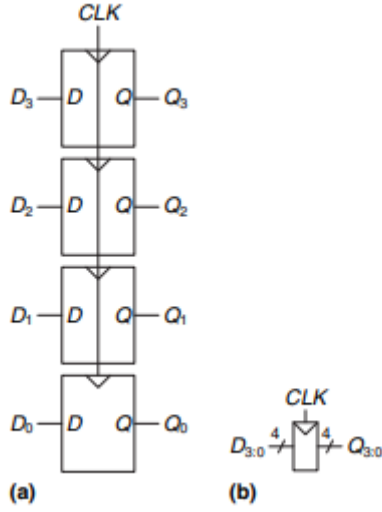
Örnek 3.1 DÖNER FLOP TRANSİSTÖR SAYISI Bu bölümde açıklanan D flip-flop'u oluşturmak için kaç transistöre ihtiyaç vardır?

Çözüm: Bir NAND veya NOR geçidi dört transistör kullanır. NOT geçidi iki transistör kullanır. Bir AND geçidi bir NAND ve bir NOT'dan oluşturulur, bu nedenle altı transistör kullanır. SR mandalı iki NOR geçidi veya sekiz transistör kullanır. D mandalı bir SR mandalı, iki AND geçidi ve bir NOT geçidi veya 22 transistör kullanır. D flip-flop'u iki D mandalı ve bir NOT geçidi veya 46 transistör kullanır. Bölüm 3.2.7, iletim kapılarını kullanan daha verimli bir CMOS uygulamasını açıklar.

3.2.4 Register / Saklayıcı

Bir N-bit saklayıcı, ortak bir CLK girişini paylaşan bir N flip-flop bankasıdır, böylece yazmacın tüm bitleri aynı anda güncellenir. Saklayıcılar, çoğu ardışık devrenin temel

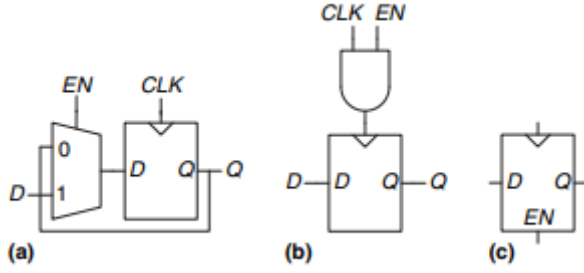
yapı taşıdır. Şekil 3.9, D3: 0 girişleri ve Q3: 0 çıkışları ile dört bitlik bir yazmaç için şematik ve sembolü göstermektedir. D3: 0 ve Q3: 0'ın her ikisi de 4 bit veri yoludur.



3.2.5 Enabled Flip-Flop

Etkinleştirilmiş bir flip-flop, verilerin saat kenarında yüklü olup olmadığını belirlemek için EN veya ENABLE adlı başka bir giriş ekler. EN DOĞRU olduğunda, etkinleştirilen flip-flop sıradan bir D flip-flopu gibi davranır. EN FALSE olduğunda, etkinleştirilen flip-flop saati yok sayar ve durumunu korur. Etkin flip-floplar, her saat kenarı yerine, her zaman bir flip-flop'a yeni bir değer yüklemek istediğimizde kullanışlıdır.

Şekil 3.10, bir D flip-flop ve ekstra bir kapıdan etkinleştirilmiş bir flip-flop oluşturmanın iki yolunu göstermektedir. Şekil 3.10 (a) 'da, bir giriş çoklayıcı, EN DOĞRU ise D'deki değerin geçip geçmeyeceğini veya EN YANLIŞ ise eski durumu Q'dan geri döndürmeyi seçer. Şekil 3.10 (b) 'de saat kapıdır. EN DOĞRU ise, flip-flopun CLK girişi normal olarak değişir. EN YANLIŞ ise, CLK girişi de YANLIŞ'tır ve flip-flop eski değerini korur. CLK = 1 iken EN'nin değişmemesi gerektiğine dikkat edin, flip-flop bir saat arızası görmesin (yanlış bir zamanda geçiş yapın). Genellikle, saat üzerinde mantık yürütmek kötü bir fikirdir. Saat geçitleme, saati geciktirir ve Bölüm 3.5.3'te göreceğimiz gibi zamanlama hatalarına neden olabilir, bu nedenle bunu yalnızca ne yaptığınızı bildiğinizden eminseniz yapın. Etkinleştirilmiş bir flip-flop için sembol Şekil 3.10 (c) 'de verilmiştir.

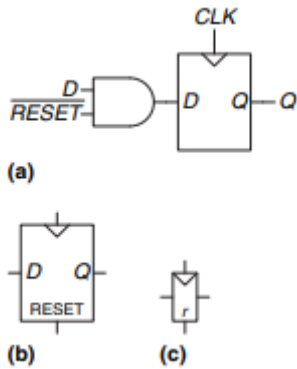


Resetlenebilir Flip-Flop

Sıfırlanabilir bir flip-flop, RESET adı verilen başka bir giriş ekler. SIFIRLA YANLIŞ olduğunda, sıfırlanabilir flip-flop sıradan bir D flip-flop gibi davranır. SIFIRLA DOĞRU olduğunda, sıfırlanabilir flip-flop D'yi yok sayar ve çıkışı 0'a sıfırlar. Sıfırlanabilir flip-floplar, bilinen bir durumu (yani, 0) ilk kez bir sistemdeki tüm flip-flop'lara zorlamak istediğimizde kullanışlıdır.

Bu tür parmak arası terlikler eşzamanlı veya eşzamansız olarak sıfırlanabilir. Eşzamanlı olarak sıfırlanabilen parmak arası terlikler kendilerini yalnızca CLK'nın yükselen kenarında sıfırlar. Zaman uyumsuz olarak sıfırlanabilen parmak arası terlikler, CLK'dan bağımsız olarak RESET TRUE olur olmaz kendilerini sıfırlar.

Şekil 3.11 (a), sıradan bir D flip-flop ve bir AND geçidinden senkronize olarak sıfırlanabilir bir flip-flopun nasıl oluşturulacağını gösterir. SIFIRLAMA YANLIŞ olduğunda, AND geçidi, flip-flop'un girişine 0'ı zorlar. RESET TRUE olduğunda, AND geçidi D'yi flip-flop'a geçirir. Bu örnekte RESET, aktif bir düşük sinyaldir, yani sıfırlama sinyali 1 değil 0 olduğunda işlevini yerine getirir. Bir invertör ekleyerek devre bunun yerine aktif bir yüksek sıfırlama sinyalini kabul edebilirdi. Şekil 3.11 (b) ve 3.11 (c), aktif yüksek sıfırlamalı sıfırlanabilir flip-flop için sembolleri gösterir.



Tahmin edebileceğiniz gibi, zaman zaman ayarlanabilir parmak arası terlikler de kullanılmaktadır. SET öne sürüldüğünde flip-flop'a bir 1 yüklerler ve onlar da senkronize ve asenkron aromalarda gelirler. Sıfırlanabilir ve ayarlanabilir flip-floplar ayrıca bir etkinleştirme girişine sahip olabilir ve N-bit kayıtları halinde gruplanabilir.

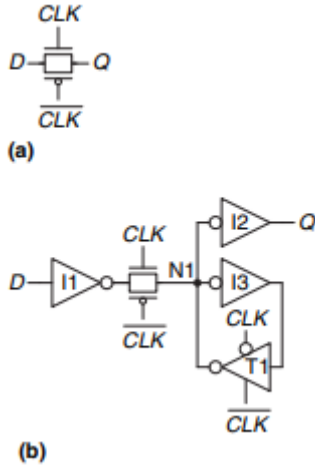
3.2.7 Transistör Seviyesinde Latch ve Flip-Flop Tasarımları

Örnek 3.1, mandalların ve flip-flopların mantık kapılarından yapıldığında çok sayıda transistör gerektirdiğini gösterdi. Ancak bir flip-flop'un temel rolü, tıpkı bir anahtar gibi şeffaf veya opak olmaktır. Bölüm 1.7.7'den bir iletim geçidinin bir CMOS anahtarı oluşturmanın etkili bir yolu olduğunu hatırlayın, bu nedenle transistör sayısını azaltmak için iletim kapılarından faydalanabileceğimizi bekleyebiliriz.

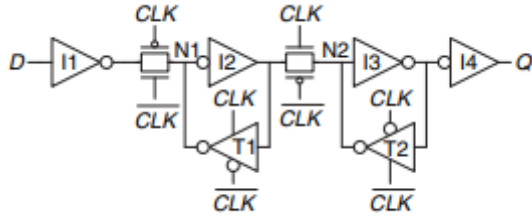
Şekil 3.12(a)'da gösterildiği gibi, tek bir iletim kapısından kompakt bir D mandalı oluşturulabilir. $CLK = 1$ ve $CLK = 0$ olduğunda, iletim kapısı AÇIK'tır, bu nedenle D, Q'ya akar ve mandal şeffaftır. $CLK = 0$ ve $CLK = 1$ olduğunda, iletim kapısı KAPALI, dolayısıyla Q, D'den yalıtılır ve mandal opaktır. Bu mandal iki ana sınırlamadan muzdariptir:

- Kayan çıktı düğümü: Mandal opak olduğunda, Q hiçbir kapı tarafından değerin tutulmaz. Böylece Q, yüzen veya dinamik bir düğüm olarak adlandırılır. Bir süre sonra gürültü ve şarj sızıntısı Q değerini bozabilir.
- Arabellek yok: Tampon eksikliği, birkaç ticari çipte arızalara neden oldu. D'yi negatif bir voltaja çeken bir gürültü artışı, nMOS transistörünü açarak, $CLK = 0$ olduğunda bile mandalı şeffaf hale getirebilir. Benzer şekilde, VDD'nin üzerindeki D'deki bir artış, $CLK = 0$ olduğunda bile pMOS transistörünü açabilir. Ve iletim kapısı simetrik, bu nedenle Q üzerindeki gürültü D girişini etkilerken geriye doğru sürülebilir. Genel kural, ne iletim kapısının girişinin ne de sıralı devrenin durum düğümünün dış dünyaya maruz kalmaması gerektirir. gürültünün muhtemel olduğu yer.

Şekil 3.12(b), modern ticari çiplerde kullanılan daha sağlam bir 12 transistörlü D mandalını göstermektedir. Hala saatli bir iletim kapısı etrafında inşa edilmiştir, ancak giriş ve çıkışı tamponlamak için I1 ve I2 invertörlerini ekler. Mandalın durumu N1 düğümünde tutulur. Evirici I3 ve üç durumlu arabellek, T1, N1'i statik bir düğüme dönüştürmek için geri bildirim sağlar. $CLK = 0$ iken N1'de az miktarda gürültü oluşursa, T1, N1'i geçerli bir mantık değerine geri götürür.



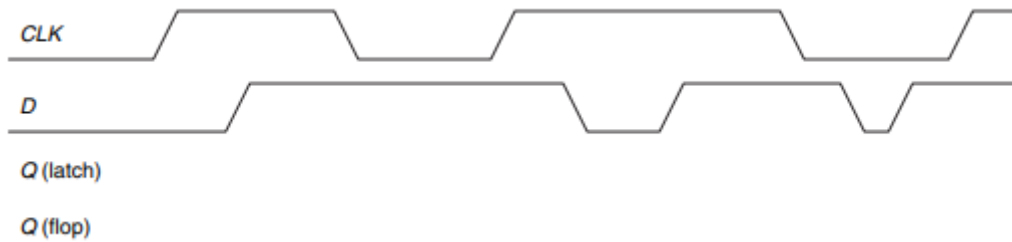
Şekil 3.13, CLK ve CLK tarafından kontrol edilen iki statik mandaldan yapılmış bir D parmak arası terlik göstermektedir. Bazı yedekli dahili invertörler kaldırılmıştır, bu nedenle flip-flop sadece 20 transistör gerektirir.



3.2.8 Tümden inceleme

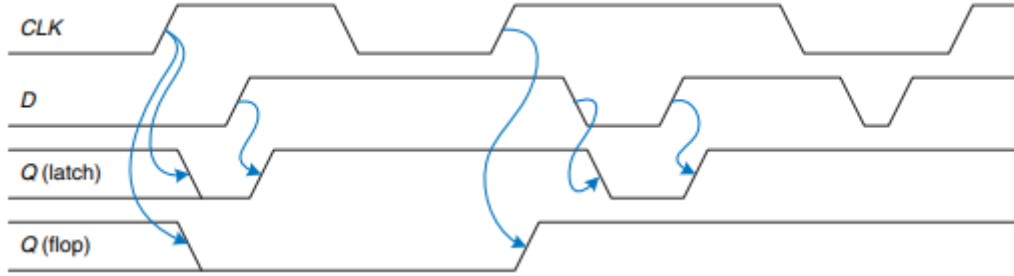
Mandallar ve flip-floplar, sıralı devrelerin temel yapı taşlarıdır. Bir D mandalının seviyeye duyarlı olduğunu, oysa bir D flip-flopunun kenarla tetiklendiğini unutmayın. D mandalı, CLK = 1 olduğunda saydamdır ve D girişinin Q çıkışına akmasına izin verir. D flip-flop, CLK'nın yükselen kenarında D'yi Q'ya kopyalar. Diğer tüm zamanlarda, mandallar ve flip-floplar eski durumlarını korurlar. Saklayıcı ise ortak bir CLK sinyalini paylaşan birkaç D flip-flop'tan oluşan bir bankadır.

Örnek 3.2 : Ben Bitdiddle, Şekil 3.14 gösterilen D ve CLK girişlerini bir D mandalı ve bir D flip-flopuna uygular. Her cihazın Q çıktısını belirlemesine yardımcı olun.



Çözüm: Şekil 3.15, Q'nun giriş değişikliklerine yanıt vermesi için küçük bir gecikme olduğu varsayılarak çıkış dalga biçimlerini gösterir. Oklar, bir çıktı değişikliğinin

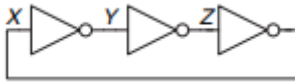
nedenini gösterir. Q'nun başlangıç değeri bilinmemektedir ve yatay çizgi çiftiyle gösterildiği gibi 0 veya 1 olabilir. İlk önce mandalı düşünün. CLK'nın ilk yükselen kenarında, $D = 0$, dolayısıyla Q kesinlikle 0 olur. CLK = 1 iken D her değiştiğinde, Q da onu takip eder. CLK = 0 iken D değiştiğinde, dikkate alınmaz. Şimdi flip-flop'u düşünün. CLK'nın her yükselen kenarında D, Q'ya kopyalanır. Diğer tüm zamanlarda, Q durumunu korur.



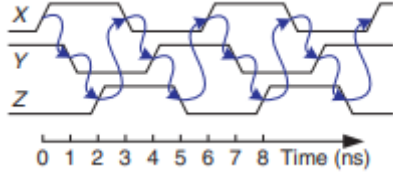
3.3 Senkron Lojik Tasarım

Genel olarak, sıralı devreler, kombinasyonel olmayan, yani çıkışı sadece akım girişlerine bakılarak belirlenemeyen tüm devreleri içerir. Bazı sıralı devreler sadece tuhaftır. Bu bölüm, bu ilginç devrelerden bazılarını inceleyerek başlar. Daha sonra senkron ardışık devreler kavramını ve dinamik disiplini tanıtır. Kendimizi senkron ardışık devrelere disipline ederek, sıralı sistemleri analiz etmek ve tasarlamak için kolay, sistematik yollar geliştirebiliriz.

Örnek 3.3: Alyssa P. Hacker, Şekil 3.16'da gösterildiği gibi, kendilerini bir döngüye bağlamış üç yanlış yazılmış evirici ile karşılaşır. Üçüncü inverterin çıkışı birinci invertere geri beslenir. Her inverterin 1 ns yayılma gecikmesi vardır. Devrenin ne yaptığını belirleyin



Çözüm: X düğümünün başlangıçta 0 olduğunu varsayalım. Sonra $Y = 1$, $Z = 0$ ve dolayısıyla $X = 1$, bu bizim orijinal varsayımımızla tutarsız. Devrenin kararlı durumu yoktur ve kararsız veya kararsız olduğu söylenir. Şekil 3.17 devrenin davranışını göstermektedir. X 0 zamanında yükselirse, Y 1 ns'de düşecek, Z 2 ns'de yükselecek ve X 3 ns'de tekrar düşecek. Sırayla, Y 4 ns'de yükselecek, Z 5 ns'de düşecek ve X tekrar 6 ns'de yükselecek ve ardından model tekrar edecek. Her düğüm, 6 ns'lik bir periyotla (tekrar süresi) 0 ile 1 arasında salınır. Bu devreye halka osilatör denir.

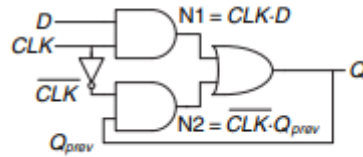


Halka osilatör periyodu, her bir inverterin yayılma gecikmesine bağlıdır. Bu gecikme, inverterin nasıl üretildiğine, güç kaynağı voltajına ve hatta sıcaklığa bağlıdır. Bu nedenle, halka osilatör periyodunu doğru bir şekilde tahmin etmek zordur. Kısacası, halka osilatör, sıfır girişli ve periyodik olarak değişen bir çıkışlı sıralı bir devredir.

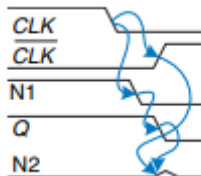
Örnek 3.4: Yarış durumu Ben Bitdiddle, daha az kapı kullandığı için Şekil 3.7'dekinden daha iyi olduğunu iddia ettiği yeni bir D mandalı tasarladı. İki girdi (D ve CLK) ve mandalın eski durumu Q_{prev} verilen Q çıktısını bulmak için doğruluk tablosunu yazdı. Bu doğruluk tablosuna dayanarak Boole denklemlerini türetmiştir. Q_{prev} çıktısını Q çıktısını geri besleyerek elde eder. Tasarımı Şekil 3.18'de gösterilmektedir. Mandalı, her kapının gecikmesinden bağımsız olarak doğru çalışıyor mu?

CLK	D	Q_{prev}	Q
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

$$Q = CLK \cdot D + \overline{CLK} \cdot Q_{prev}$$



Çözüm: Şekil 3.19, devrenin, belirli kapılar diğerlerinden daha yavaş olduğunda devre dışı kalmasına neden olan bir yarış koşuluna sahip olduğunu göstermektedir. $CLK = D = 1$ varsayalım. Mandal şeffaftır ve $Q = 1$ yapmak için D'den geçer. Şimdi, CLK düşer. Mandal, $Q = 1$ 'i koruyarak eski değerini hatırlamalıdır. Bununla birlikte, inverterin CLK'dan CLK'ya olan gecikmesinin AND ve OR geçitlerinin gecikmelerine kıyasla oldukça uzun olduğunu varsayalım. O zaman N1 ve Q düğümlerinin her ikisi de CLK yükselmeden önce düşebilir. Böyle bir durumda N2 asla yükselmez ve Q 0'da takılı kalır.



Bu, çıkışların doğrudan girişlere geri beslendiği bir asenkron devre tasarımı örneğidir. Asenkron devreler, devrenin davranışının, mantık kapılarından geçen iki yoldan hangisinin en hızlı olduğuna bağlı olduğu yarış koşullarına sahip oldukları için meşhurdur. Bir devre çalışabilirken, biraz farklı gecikmelere sahip kapılardan yapılmış

görünüşte aynı olan bir devre çalışmayabilir. Veya devre, yalnızca gecikmelerin tam olarak doğru olduğu belirli sıcaklıklarda veya voltajlarda çalışabilir. Bu arızaların izini sürmek son derece zordur

3.3.2 Senkron Ardışık Devreler

Önceki iki örnek, çıkışların doğrudan girişlere geri beslendiği, döngüsel yollar adı verilen döngüler içerir. Kombinasyon devrelerinden ziyade sıralı devrelerdir. Kombinasyonel mantığın döngüsel yolları ve yarışları yoktur. Girişler birleşimsel mantığa uygulanırsa, çıkışlar her zaman bir yayılma gecikmesi içinde doğru değere oturacaktır. Bununla birlikte, döngüsel yollara sahip sıralı devreler, istenmeyen yarışlara veya kararsız davranışlara sahip olabilir. Bu tür devreleri problemler için analiz etmek zaman alıcıdır ve birçok zeki insan hata yapmıştır.

Bu sorunlardan kaçınmak için tasarımcılar, yolun herhangi bir yerine kayıtlar ekleyerek döngüsel yolları kırarlar. Bu, devreyi bir kombinasyonel mantık ve kayıtlar koleksiyonuna dönüştürür. Kayıtlar, yalnızca saat kenarında değişen sistemin durumunu içerir, bu nedenle durumun saatle senkronize olduğunu söylüyoruz. Saat, tüm kayıtlara girişler bir sonraki saat kenarından önce yerleşmek üzere yeterince yavaşsa, tüm yarışlar elenir. Geri besleme yolundaki her zaman yazmaçları kullanma disiplini benimsemek, bizi bir senkron sıralı devrenin resmi tanımına götürür.

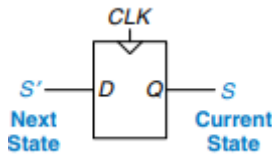
Bir devrenin giriş ve çıkış terminalleri ve işlevsel ve zamanlama özellikleri ile tanımlandığını hatırlayın. Sıralı bir devrenin sonlu ayırık durumları $\{S_0, S_1, \dots, S_{k-1}\}$ vardır. Bir senkron ardışık devre, yükselen kenarları durum geçişlerinin meydana geldiği bir dizi zamanı gösteren bir saat girişine sahiptir. Sistemin o andaki durumunu, bir sonraki saat kenarında gireceği durumdan ayırt etmek için genellikle mevcut durum ve sonraki durum terimlerini kullanırız. İşlevsel belirtim, bir sonraki durumu ve mevcut durum ve giriş değerlerinin olası her bir kombinasyonu için her bir çıkış değerini ayrıntılarıyla belirtir. Zamanlama belirtimi, saatin yükselen kenarından çıkış değişene kadar geçen süre için bir üst sınır, tpcq ve bir alt sınırdan, tccq ve ayrıca girişlerin ne zaman olduğunu gösteren kurulum ve tutma süreleri, tsetup ve thold'dan oluşur. saatin yükselen kenarına göre sabit olmalıdır.

Senkron ardışık devre kompozisyonunun kuralları bize, bir devrenin, birbirine bağlı devre elemanlarından oluşuyorsa, senkron ardışık bir devre olduğunu öğretir.

- Her devre elemanı ya bir kayıt ya da birleşimsel devredir.
- En az bir devre elemanı bir kayıttır
- Tüm kayıtlar aynı saat sinyalini alır
- Her döngüsel yol en az bir kayıt içerir.

Senkron olmayan sıralı devrelere asenkron denir.

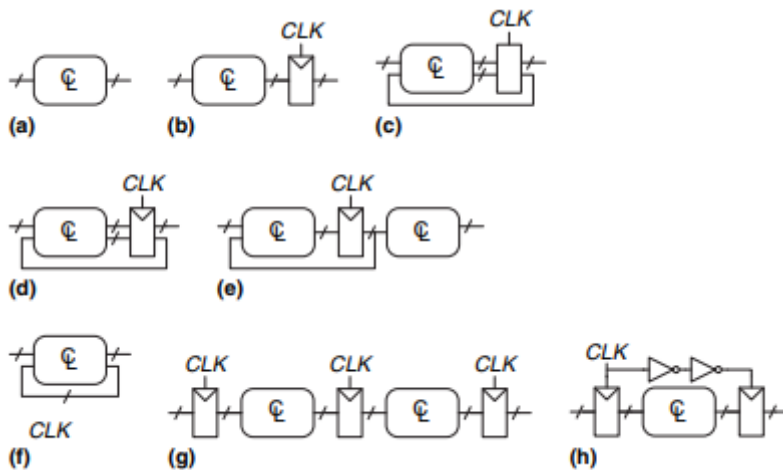
Bir flip-flop, en basit senkron ardışık devredir. Bir girişi, D, bir saati, CLK, bir çıkışı, Q ve iki durumu, {0, 1} vardır. Bir flip-flop için işlevsel belirtim, Şekil 3.20'de gösterildiği gibi, sonraki durumun D olması ve Q çıktısının mevcut durum olmasıdır.



Sıklıkla mevcut durum değişkenini S ve sonraki durum değişkenini S' olarak adlandırırız. Bu durumda, S'den sonraki asal bir sonraki durumu belirtir, tersine çevirmeyi değil. Sıralı devrelerin zamanlaması Bölüm 3.5'te analiz edilecektir.

Senkron ardışık devrelerin diğer iki yaygın türü, sonlu durum makineleri ve boru hatları olarak adlandırılır. Bunlar bu bölümde daha sonra ele alınacaktır.

Örnek 3.5 SENKRON SIRALI DEVRELER Şekil 3.21'deki devrelerden hangileri senkron ardışıl devrelerdir?



Çözüm: Devre (a) birleşimseldir, sıralı değildir, çünkü kaydı yoktur. (b) geri beslemesi olmayan basit bir ardışık devredir. (c) ne birleşimsel devre ne de senkron ardışık devredir, çünkü ne yazmaç ne de birleşimsel devre olan bir mandalı vardır. (d) ve (e) senkron ardışık mantıktır; bunlar, Bölüm 3.4'te tartışılan iki sonlu durumlu makine biçimidir. (f) ne birleşimsel ne de senkron ardışık değildir, çünkü birleşimsel mantığın çıkışından aynı mantığın girişine kadar döngüsel bir yola sahiptir, ancak yolda kayıt yoktur. (g) Kısım 3.6'da inceleyeceğimiz bir ardışık düzen şeklinde senkron ardışık mantıktır. (h) kesinlikle senkron ardışık devre değildir, çünkü ikinci kayıt birinciden farklı bir saat sinyali alır, iki invertör gecikmesi tarafından geciktirilir.

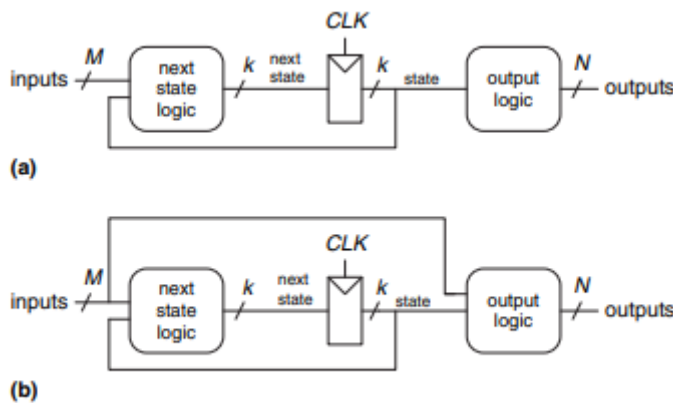
3. 3. 3 Senkron ve Asenkron Devreler

Teoride asenkron tasarım, senkronize tasarımdan daha geneldir, çünkü sistemin zamanlaması saatli kayıtlarla sınırlı değildir. Analog devreler herhangi bir voltajı kullanabildikleri için analog devrelerin dijital devrelerden daha genel olması gibi, asenkron devreler de her türlü geri beslemeyi kullanabildikleri için senkron devrelerden daha geneldir. Bununla birlikte, dijital devrelerin analog devrelerden daha kolay olması gibi, senkron devrelerin tasarımı ve kullanımının asenkron devrelerden daha kolay olduğu kanıtlanmıştır. Asenkron devreler üzerinde onlarca yıllık araştırmalara rağmen, neredeyse tüm dijital sistemler esasen senkronizedir.

Elbette, farklı saatlere sahip sistemler arasında iletişim kurarken veya rastgele zamanlarda girişler alırken, sürekli voltajların gerçek dünyası ile iletişim kurarken analog devrelerin gerekli olması gibi, zaman uyumsuz devreler bazen gereklidir. Ayrıca, asenkron devrelerdeki araştırmalar, bazıları senkron devreleri de geliştirebilecek ilginç bilgiler üretmeye devam ediyor.

3.4 Sonlu Durum Makinaları

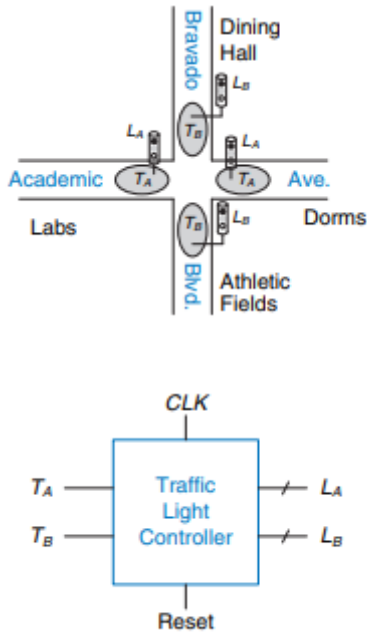
Şekilde 3.22 gösterilen senkron sıralı devreler sonlu durum makineleri (FSM'ler) denir. Sonlu durum makinelerinde k yazmaçlı bir devre sonlu sayıdaki (2^k) durumlardan birinde olabilir. Bir FSM'nin M girişi, N çıkışı ve k bit durumu bir saat ve isteğe bağlı olarak bir reset (sıfırlama) sinyali alır. FSM, iki kombinasyonel mantık bloğu, bir sonraki durum mantığı ve çıkış mantığı ve durumu depolayan bir yazmaçtan oluşur. Her bir saat kenarında, FSM mevcut işlevsel özellikleri ile karakterize edilen iki sonlu durum makinesi sınıfı vardır. Moore makinelerinde, çıktılar yalnızca makinenin mevcut durumuna bağlıdır. Mealy makinelerinde çıktılar hem mevcut duruma hem de girişlerine bağlıdır. Sonlu durum makineleri, özellikleri verilen eşzamanlı sıralı devreleri tasarlamak için sistematik bir yol sağlar. Bu yöntem, bir örnekle başlayarak bu bölümün geri kalanında açıklanacaktır.



3.4.1 Sonlu Durum Makinaları Örneği

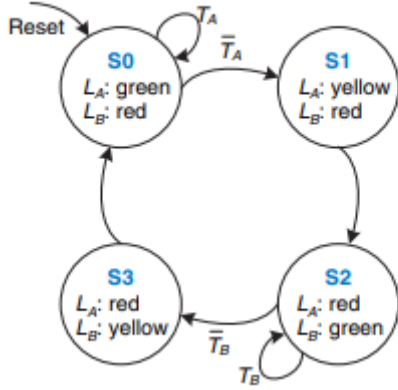
FSM'lerin tasarımını göstermek için, kampüsteki yoğun bir kavşakta trafik ışığı için bir kontrolör tasarlayalım. Mühendislik öğrencileri, yurtları ile Academic Ave'deki laboratuvarlar arasında dolaşırlar. En sevdikleri ders kitaplarında FSM'ler hakkında okumakla meşguller ve nereye gittiklerine bakmıyorlar. Futbolcular atletik alanlar ve Bravado Bulvarı'ndaki yemek salonu arasında koşuşturuyor. Topu ileri geri fırlatıyorlar ve nereye gittiklerine de bakmıyorlar. Bu iki yolun kesişme noktasında çok sayıda ciddi yaralanma meydana geldiğinden Dekan, ölümler olmadan önce bir trafik ışığı takmasını istiyor.

Ben, sorunu bir FSM ile çözmeye karar verir. Academic Ave. ve Bravado Blvd.'ye sırasıyla TA ve TB olmak üzere iki trafik sensörü kuruyor. Her sensör, öğrenciler mevcutsa DOĞRU, cadde boşsa YANLIŞ gösterir. Ayrıca trafiği kontrol etmek için LA ve LB olmak üzere iki trafik lambası kuruyor. Her ışık, yeşil, sarı veya kırmızı dijital girişleri alır. Dolayısıyla, FSM'sinin TA ve TB olmak üzere iki girişi ve LA ve LB olmak üzere iki çıkışı vardır. Işıklar ve sensörler Şekil 2.23 gösterilmektedir. Ben, 5 saniyelik periyotlu bir saati vardır. Her saat işaretinde (yükselen kenar), ışıklar trafik sensörlerine göre değişebilir. Ayrıca, Fiziksel Tesis teknisyenlerinin denetleyiciyi açtıklarında bilinen bir başlangıç durumuna getirebilmeleri için bir sıfırlama düğmesi vardır. Şekilde 3.24, durum makinesinin kara kutu görünümünü göstermektedir.



Ben'in bir sonraki adımı, sistemin tüm olası durumlarını ve bu durumlar arasındaki geçişleri göstermek için Şekil 3.25 gösterilen durum geçiş diyagramını çizmektir. Sistem sıfırlandığında, ışıklar Academic Ave'de yeşil ve Bravado Blvd'de kırmızıdır. Her 5 saniyede bir kontrolör trafik düzenini inceler ve daha sonra ne yapılacağına karar verir. Academic Ave.'de trafik olduğu sürece ışıklar değişmez. Academic Ave.'de artık trafik olmadığında, Academic Ave.'deki ışık kırmızıya ve Bravado Bulvarı'nın ışığı yeşile dönmeye önce 5 saniye boyunca sarıya döner. Benzer şekilde, Bravado Blvd.

Bulvarda trafik olduğu sürece ışık yeşil kalır, ardından sarıya ve sonunda kırmızıya döner.



Bir durum geçiş diyagramında, daireler ile temsil eder ve yaylar durumlar arasındaki geçişleri temsil eder. Geçişler saatin yükselen kenarında gerçekleşir; saati diyagram üzerinde gösterme zahmetine girmeyiz, çünkü her zaman eşzamanlı sıralı bir devrede mevcuttur. Dahası, saat basitçe geçişlerin ne zaman gerçekleşmesi gerektiğini kontrol ederken, diyagram hangi geçişlerin gerçekleştiğini gösterir. Dış dünya S0 durumuna işaret eden yay, sistemin, önceki durumdan bağımsız olarak, sıfırlama üzerine bu duruma girmesi gerektiğini belirtir. Bir durumda birden fazla yay varsa, yaylar, hangi giriş tarafından tetiklendiğini göstermek için etiketlenir. Örneğin, S0 durumunda iken, TA TRUE ise sistem bu durumda kalacak ve TA FALSE ise S1'e geçecektir. Bir durumdan çıkan tek bir yaya sahipse, bu geçiş her zaman girişlerden bağımsız olarak gerçekleşir. Örneğin, S1 durumunda, sistem her zaman S2'ye geçecektir. Çıkışların belirli bir durumdayken sahip olduğu değer, durumda belirtilir. Örneğin, S2 durumunda, LA kırmızıdır ve LB yeşildir.

Ben, durum geçiş diyagramını bir durum geçiş Tablo 3.1 olarak yeniden yazar; bu, her durum ve giriş için bir sonraki durum olan S'nin ne olması gerektiğini gösterir. Tabloda, sonraki durum belirli bir girdiye bağlı olmadığında, önemsiz semboller (X) kullanıldığını dikkat edin. Ayrıca Reset tablodan çıkarıldığını unutmayın. Bunun yerine, girişlerden bağımsız olarak, sıfırlama sırasında her zaman S0 durumuna giden sıfırlanabilir flip flop kullanıyoruz.

Current State S	Inputs T_A T_B		Next State S'
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0

Durum geiş diyagramı, {S0, S1, S2, S3} etiketli durumları ve {kırmızı, sarı, yeşil} etiketli ıktıları kullandığından soyuttur. Gerek bir devre oluşturmak için, durumlara ve ıktılara ikili kodlamalar atanmalıdır. Ben, Tablo 3.2 ve 3.3 verilen basit kodlamaları seer. Her durum ve her ıkış iki bit ile kodlanır: S1:0, LA1:0 ve LB1:0.

State	Encoding $S_{1:0}$
S0	00
S1	01
S2	10
S3	11

Output	Encoding $L_{1:0}$
green	00
yellow	01
red	10

Ben, Tablo 3.4 gösterildiğı gibi bu ikili kodlamaları kullanmak için durum geiş tablosunu günceller. Revize edilmiş durum geiş tablosu, bir sonraki durum mantığını belirten bir doğruluk tablosudur. Sonraki durumu, S' , mevcut durum, S ve girişlerin bir fonksiyonu olarak tanımlar.

Current State		Inputs		Next State	
S_1	S_0	T_A	T_B	S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

Bu tablodan, bir sonraki durum için Boole denklemlerini arpımları toplamı biçiminde okumak kolaydır.

$$S'_1 = S'_1S'_0 + S_1S'_0TB' + S_1S_0TB \quad S'_0 = S'_1S'_0TA' + S_1S'_0TB'$$

Denklemler, Karnaugh haritaları kullanılarak basitleştirilebilir, ancak genellikle bunu elle yapmak daha kolaydır. Örneğın, S'_1 denklemindeki TB ve $(TB)'$ terimleri açıka gereksizdir. Böylece S'_1 , bir XOR işlemine indirgenir. Aşğıdaki, basitleştirilmiş bir sonraki durum denklemlerini verir.

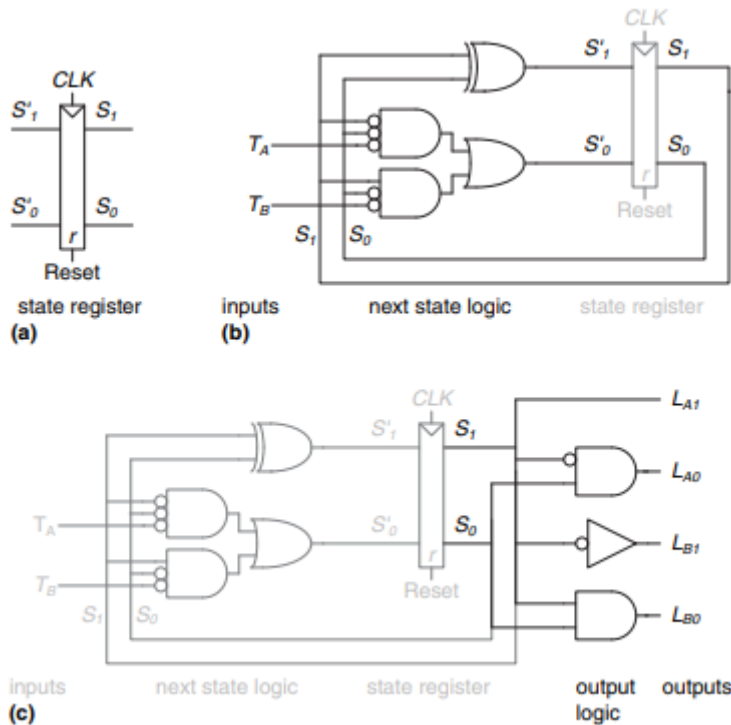
$$\text{Denklem 3.2 } S'_1 = S_1 \oplus S_0 \quad S'_0 = S'_1S'_0TA' + S_1S'_0TB'$$

Benzer şekilde, Ben her durum için çıktının o durumda ne olması gerektiğini gösteren bir çıktı tablosu (Tablo 3.5) yazar. Yine, çıktılar için Boole denklemlerini okumak ve basitleştirmek kolaydır. Örneğin, LA1'in yalnızca S1'in TRUE olduğu satırlarda TRUE olduğunu gözlemleyin.

Current State		Outputs			
S_1	S_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

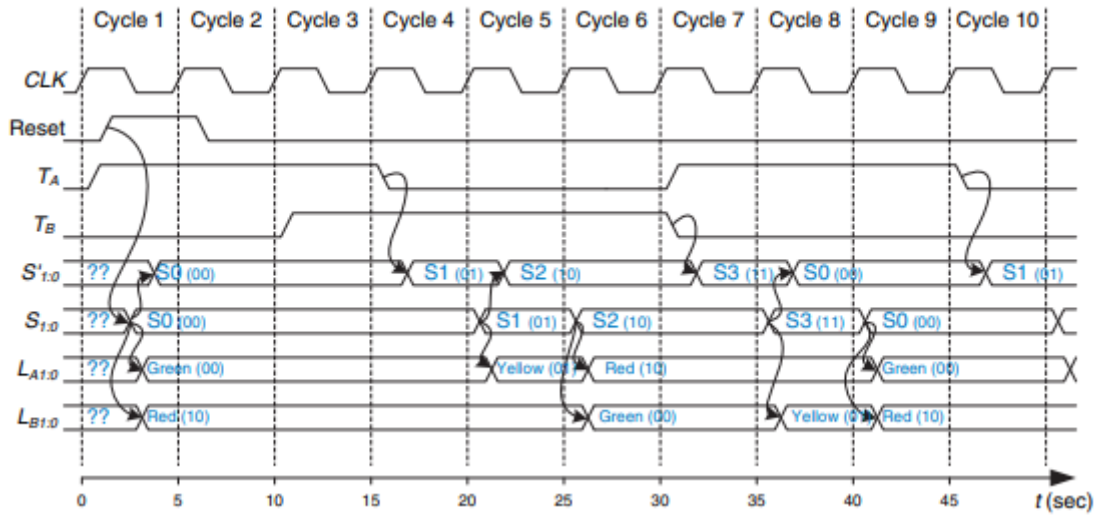
$$\text{Denklem 3.3 } LA1 = S1 \quad LA0 = S1'S0 \quad LB1 = S1' \quad LB0 = S1S0$$

Son olarak Ben, Moore FSM'sini Şekil 3.22(a) şeklinde çiziyor. İlk olarak, Şekil 3.26(a)'da gösterildiği gibi 2 bitlik durum kaydını çizer. Her saat kenarında, durum kaydı, S1:0 durumu olmak için sonraki durumu S1:0 ' kopyalar. Durum kaydı, başlangıçta FSM'yi başlatmak için senkron veya asenkron bir sıfırlama alır. Ardından, Şekil 3.26(b)'de gösterildiği gibi, mevcut durum ve girdilerden sonraki durumu hesaplayan Denklem 3.2'ye dayalı olarak bir sonraki durum mantığını çizer. Son olarak, Şekil 3.26(c)'de gösterildiği gibi, mevcut durumdan çıktıları hesaplayan Denklem 3.3'e dayalı olarak çıktı mantığını çizer.



Şekil 3.27, bir dizi durumdan geçen trafik ışığı kontrolörünü gösteren bir zamanlama diyagramını göstermektedir. Diyagramda CLK, Reset, TA ve TB girişleri, sonraki S' durumu, S durumu ve LA ve LB çıkışları gösterilmektedir. Oklar nedenselliği gösterir;

örneğin, durumu değiştirmek çıktıların değişmesine ve girdilerin değiştirilmesi bir sonraki durumun değişmesine neden olur. Kesikli çizgiler, durum değiştiğinde CLK'nın yükselen kenarlarını gösterir.



Saatin 5 saniyelik bir periyodu vardır, bu nedenle trafik ışıkları en fazla 5 saniyede bir değişir. Sonlu durum makinesi ilk açıldığında, soru işaretleriyle gösterildiği gibi durumu bilinmemektedir. Bu nedenle, bilinen bir duruma getirmek için sistem sıfırlanmalıdır. Bu zamanlama şemasında, eşzamansız olarak sıfırlanabilen iki duraklıların kullanıldığını belirten hemen S0'a sıfırlanır. S0 durumunda, ışık LA yeşil ve ışık LB kırmızıdır.

Bu örnekte trafik hemen Academic Ave'e gelir. Bu nedenle, Bravado Blvd'ye trafik gelse bile denetleyici S0 durumunda kalır ve LA'yı yeşil tutar. ve beklemeye başlar. 15 saniye sonra Academic Ave. üzerindeki trafik tamamen geçer ve TA düşer. Takip eden saat kenarında, kontrolör LA'yı sarıya çevirerek S1 durumuna geçer. 5 saniye sonra kontrolör, LA'nın kırmızıya ve LB'nin yeşile döndüğü S2 durumuna geçer. Denetleyici, Bravado Blvd'deki tüm trafik oluşana kadar S2 durumunda bekler. geçti. Daha sonra LB'yi sarıya çevirerek S3 durumuna geçer. 5 saniye sonra kontrolör S0 durumuna girer ve LB kırmızı ve LA yeşile döner. İşlem tekrarlanır.

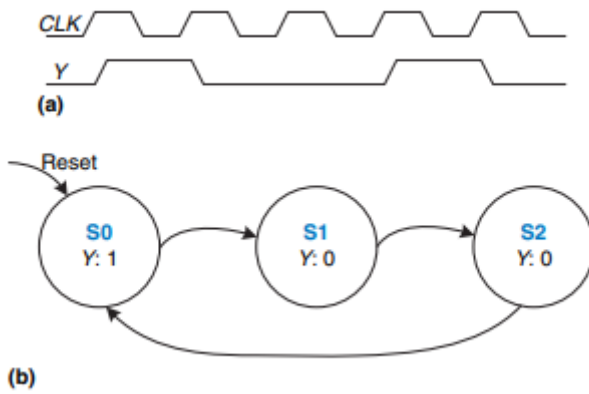
3.4.2 Durum Kodlama

Önceki örnekte, durum ve çıktı kodlamaları keyfi olarak seçilmiştir. Farklı bir seçim, farklı bir devre ile sonuçlanabilirdi. Doğal bir soru, en az mantık kapısına veya en kısa yayılma gecikmesine sahip devreyi üreten kodlamanın nasıl belirleneceğidir. Ne yazık ki, durum sayısı fazla olduğunda mümkün olmayan tüm olasılıkları denemek dışında en iyi kodlamayı bulmanın basit bir yolu yoktur. Bununla birlikte, ilgili durumların veya çıktıların bitleri paylaşması için inceleme yoluyla iyi bir kodlama seçmek çoğu zaman mümkündür. Bilgisayar destekli tasarım (CAD) araçları, olası kodlamaları arama ve makul bir kodlama seçme konusunda da iyidir.

Durum kodlamasında önemli bir karar, ikili kodlama ile tek-sıcak kodlama arasındaki seçimdir. Trafik ışığı denetleyicisi örneğinde kullanıldığı gibi ikili kodlamada, her durum bir ikili sayı olarak temsil edilir. K ikili sayıları $\log_2 K$ bitleriyle temsil edilebildiğinden, K durumlu bir sistem yalnızca $\log_2 K$ bit durumlarına ihtiyaç duyar.

Tek sıcak kodlamada, her durum için ayrı bir durum biti kullanılır. Tek-sıcak olarak adlandırılır, çünkü herhangi bir zamanda yalnızca bir bit "sıcak" veya DOĞRUDUR. Örneğin, üç durumlu tek-sıcak kodlanmış bir FSM, 001, 010 ve 100 durum kodlamalarına sahip olacaktır. Her bir durum biti bir flip-flop'ta depolanır, bu nedenle onehot kodlama, ikili kodlamadan daha fazla flip-flop gerektirir. Ancak, one-hot kodlama ile sonraki durum ve çıkış mantığı genellikle daha basittir, dolayısıyla daha az geçit gerekir. En iyi kodlama seçimi, belirli FSM'ye bağlıdır.

Örnek 3.6 : N'ye bölme sayacının bir çıkışı vardır ve girişi yoktur. Y çıkışı, her N'den bir saat döngüsü için YÜKSEK'tir. Başka bir deyişle, çıkış saatin frekansını N'ye böler. Bir 3'e bölme sayacı için dalga biçimi ve durum geçiş diyagramı Şekil 3.28'de gösterilmektedir. İkili ve tek sıcak durum kodlamalarını kullanarak böyle bir sayaç için devre tasarımları çizin.



Çözüm: Tablo 3.6 ve 3.7, kodlamadan önceki soyut durum geçişini ve çıktı tablolarını gösterir.

Current State	Next State
S0	S1
S1	S2
S2	S0

Current State	Output
S0	1
S1	0
S2	0

Tablo 3.8, üç durum için ikili ve tek sıcak kodlamaları karşılaştırır.

State	One-Hot Encoding			Binary Encoding	
	S_2	S_1	S_0	S_1	S_0
S0	0	0	1	0	0
S1	0	1	0	0	1
S2	1	0	0	1	0

İkili kodlama iki bit durum kullanır. Bu kodlama kullanılarak durum geçiş tablosu Tablo 3.9'da gösterilmiştir. Giriş olmadığını unutmayın; sonraki durum yalnızca mevcut duruma bağlıdır. Çıktı tablosu okuyucuya alıştırma olarak bırakılmıştır. Sonraki durum ve çıktı denklemleri:

Current State		Next State	
S_1	S_0	S'_1	S'_0
0	0	0	1
0	1	1	0
1	0	0	0

$$S'_1 = S_1 S_0 \quad S'_0 = S_1 S_0 \quad (3.4)$$

$$Y = S_1 S_0 \quad (3.5)$$

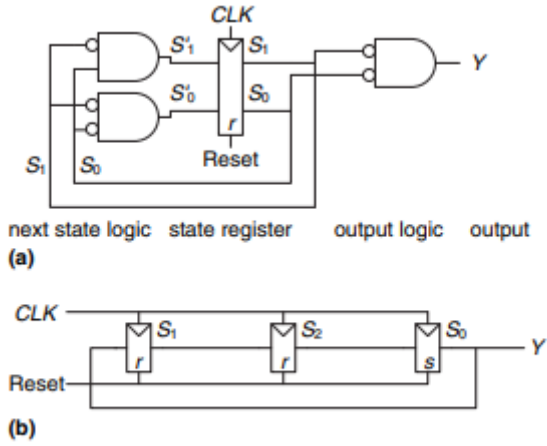
Tek sıcak kodlama, üç bit durum kullanır. Bu kodlama için durum geçiş tablosu Tablo 3.10'da gösterilmiş ve çıktı tablosu yine bir alıştırma olarak okuyucuya bırakılmıştır. Sonraki durum ve çıktı denklemleri aşağıdaki gibidir:

Current State			Next State		
S_2	S_1	S_0	S'_2	S'_1	S'_0
0	0	1	0	1	0
0	1	0	1	0	0
1	0	0	0	0	1

$$S'_2 = S_1 \quad S'_1 = S_0 \quad S'_0 = S_2 \quad (3.6)$$

$$Y = S_0 \quad (3.7)$$

Şekil 3.29, bu tasarımların her biri için şemaları göstermektedir. İkili kodlanmış tasarım donanımının, Y ve S'_0 için aynı geçidi paylaşacak şekilde optimize edilebileceğini unutmayın. Ayrıca, tek-sıcak kodlamanın, makineyi sıfırlama sırasında S'_0 'a başlatmak için hem ayarlanabilir(ler) hem de sıfırlanabilir (r) flip-floplar gerektirdiğini gözlemleyin. En iyi uygulama seçimi, kapıların ve parmak arası terliklerin göreceli maliyetine bağlıdır, ancak bu özel örnek için genellikle tek-sıcak tasarım tercih edilir.



İlgili bir kodlama, K durumunun tam olarak biri YANLIŞ olan K bitleri ile temsil edildiği tek-soğuk kodlamadır.

3.4.3 Moore and Mealy Machines

Şimdiye kadar, çıktının yalnızca sistemin durumuna bağlı olduğu Moore makinelerinin örneklerini gösterdik. Bu nedenle, Moore makineleri için durum geçiş diyagramlarında çıktılar daireler içinde etiketlenir. Mealy makinelerinin Moore makinelerine çok benzediğini hatırlayın, ancak çıktılar mevcut duruma olduğu kadar girdilere de bağlı olabilir. Bu nedenle, Mealy makineleri için durum geçiş diyagramlarında, çıktılar daireler yerine yaylar üzerinde etiketlenir. Çıkışları hesaplayan kombinasyonel mantık bloğu, Şekil 3.22(b)'de gösterildiği gibi mevcut durumu ve girişleri kullanır.

