EEM212 - SAYISAL DEVRE TASARIMI DERS NOTLARI

DERS NOTU 2: BOOLE CEBRİ VE DİJİTAL DEVRE GERÇEKLEŞTİRİMİ

Dr. İsmail Öztürk *

<ismail.ozturk@amasya.edu.tr>

İçindekiler

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

1 Boole Cebri

1.1 Boole Cebrine Giriş

Boole cebrinin kökeni antik Yunan filozofu olan Aristoteles'e kadar dayanır. Her ne kadar Aristoteles'ten önce de insanlar akıl yürütme yapıyor olsa da, Aristoteles akıl yürütme işini belli bir metodolojiye dökmeye çalışan ilk filozof olmuştur. Mesela, belirlediği akıl yürütme yöntemlerinden biri felsefede *tasım* olarak adlandırılmaktadır. Tasım yönteminde elde bulunan iki yargıdan bunlar dışında üçüncü bir yargı elde edilir. Tasım yönteminin en ünlü örneği aşağıdaki gibidir:

1. Tüm insanlar ölümlüdür. (yargı)
2. Tüm Yunanlar insandır. (yargı)
3. Tüm Yunanlar ölümlüdür. (sonuç)

Her ne kadar bu akıl yürütme işlemi ile çok da ufuk açıcı bir sonuç elde etmiş olmasak da, bu akıl yürütme yönteminin asıl gücü bu yöntemi genelleştirdiğimizde ortaya çıkmaktadır. Yukarıdaki tasım örneğindeki spesifik yargıları A, B gibi simgeler kullanarak aşağıdaki gibi genelleştirebiliriz:

1. Tüm B'ler A'dır. (yargı)
2. Tüm C'ler B'dir. (yargı)
3. Tüm C'ler A'dır. (sonuç)

Burada A ve B yargıları yerine (doğru oldukları müddetçe) istediğimiz herhangi bir ifadeyi koyabiliriz. Örneğin, "tüm sporcular sağlıklıdır" ve "tüm futbolcular sporcudur", bu nedenle "tüm futbolcular sağlıklıdır". Buradan görüldüğü üzere, belli problemlere semboller atayıp akıl yürütmeyi yukarıdaki gibi otomatikleştirmemiz mümkündür.

Mantıksal akıl yürütme işleminin otomatikleştirilmesi fikri Aristoteles'ten sonra pek çok bilim insanı ve filozofu etkiledi. Hatta bu araştırmacılardan bazıları bilgisayarların keşfinden çok daha önce mantıksal problemlerin sembollerle ifade edildikten sonra makineler tarafından çözülebileceğini öngördü. Fakat, bu konuda asıl çığır açan çalışma 1854 yılında İngiliz bir matematikçi olan George Boole (d. 1815, ö. 1864) tarafından ortaya kondu. Boole insanın akıl yürütme sürecine dair bir takım ilkeleri gözden geçirerek bu ilkeleri matematiksel olarak ifade etmeyi başarmıştı.

Daha sonraları Boole cebri olarak genelleştirilen bu matematiksel işlemlerde değişkenler sadece 0 ile 1 değerlerini alabilmekte ve 0 mantıksal yanlışı ifade ederken, 1 mantıksal doğruyu ifade etmektedir. Bu cebir türünde kullanılan temel işlemler ise VE (ing. AND), VEYA (ing. OR) ile DEĞİL (ing. NOT) işlemleridir. VE işlemi $\times$ veya . ile temsil edilirken, VEYA işlemi $+$ ile temsil edilir. VE ile VEYA işlemleri iki değişken kullanılarak yapılan işlemlerken, DEĞİL işlemi sadece tek bir değişkene uygulanır. Buna göre x bir değişken olmak üzere x ifadesinin değili x' veya $\bar{x}$ ile gösterilir.

Bu matematik kullanılarak insanın akıl yürütmesine dair belirlenmiş bir takım ilkelerin ifade edilebileceğini belirtmiştik. Mesela, "zıtlık ilkesi" herhangi bir varlığın bir niteliği hem barındırıp hem de barındırmamasının mümkün olmadığını söyler. Boole cebri kullanılarak bu aşağıdaki gibi formülize edilir:

$$x.x' = 0 \tag{1}$$

Yani, x önermesi VE x önermesinin DEĞİLİ (zıttı, tersi) yanlıştır. Burada önermeden kastettiğimiz sadece doğru veya yanlış olabilen yargılardır. Sadece doğru ve yanlış değeri alabildikleri için de Boole değişkenleri cinsinden ifade edilebilirlerdir. Mesela, "tabak yuvarlaktır" yargısı bir önermedir. Eğer elimizde yuvarlak bir tabak varsa "tabak yuvarlaktır" önermesi doğru olacaktır. Bunu Boole denklemi olarak $x = 1$ ile ifade ederiz ($x = 1$ tam olarak tabak yuvarlaktır ifadesi doğrudur anlamına gelir). Bu durumda, tabak yuvarlak değildir önermesi (yani x') yanlış olacaktır ($x' = 0$). Tam tersi, elimizde kare bir tabak olsaydı, bu durumda $x = 0$ (tabak yuvarlaktır ifadesi yanlıştır) olup $x' = 1$ (tabak yuvarlak değildir ifadesi doğrudur) olacaktı.

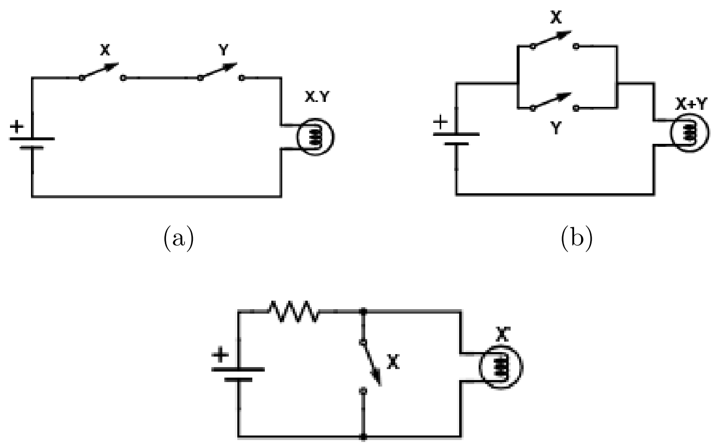
Buna uygun olarak Eşitlik 1'deki x Boole değişkeninin "tabak yuvarlaktır" önermesine karşılık gelen doğruluk değeri olduğunu varsaydığımızda, bu eşitlikteki Boole denklemi "tabak yuvarlaktır VE tabak yuvarlak DEĞİLdir ifadesi YANLIŞtır" şeklinde bir mantıksal çıkarıma karşılık gelecektir. Başka bir deyişle tabağın hem yuvarlak olup hem de yuvarlak olmaması mümkün değildir. Görüldüğü üzere Boole cebri ile mantıksal akıl yürütmeleri Aristoteles örneğindeki gibi sembollerle yapmakla kalmayıp bunları matematiksel eşitlikler olarak da ifade edebiliyoruz.

Boole'un ölümünden tam 73 yıl sonra Claude Shannon adlı Amerikalı bir elektrik mühendisi yüksek lisans tezinde [1] Boole'un matematiksel eşitliklerinin o zamanlar telefon ağlarını anahtarlamak için kullanılan elektromekanik röleler yardımıyla elektrik devreleri olarak kurulabileceğini gösterdi. Bu sayede, elektromekanik rölelerin kullanımını Boole cebri ile sadeleştirerek telefon ağlarında daha az rölenin kullanılmasını sağlamış oldu. Hatta, tezinin en sonunda Boole cebrini kullanarak elektromekanik röleler ile 4-bitlik bir sayıcı bile tasarladı. Bu dijital devre tasarımının ilk örneklerinden birini teşkil etmektedir. Üretilen devreler mantıksal işlemlerden ibaret Boole cebrine dayandığı için sonraları bu tür devreler mantık devreleri [2] olarak adlandırıldı.

Temel Boole işlemlerinin elektrik devreleri olarak karşılıkları Şekil 1'de görülmektedir. Burada, anahtarlar elektromekanik röleleri temsil etmekte ve anahtar kapalıyken (iletimde) 1 durumu; açıkken 0 durumunun sağlandığı varsayılmaktadır. 1940'lı yılların sonuna doğru gelindiğinde vakum tüplerinin yaygınlaşmasıyla Şekil 1'deki anahtarlar vakum tüpleriyle yer değiştirdi. 1960'lı yıllarda transistörlerin yaygınlaşmaya başlamasıyla da mantık devrelerinde anahtarlama yapmak için transistörler

[1] http://www.cs.virginia.edu/~evans/cs6501-s13/shannon38.pdf

[2] ing. logic circuits



(a) (b) (c)

Şekil 1: Temel Boole işlemlerinin elektriksel devre karşılıkları: (a) VE işlemi; (b) VEYA işlemi; (c) DEĞİL işlemi

kullanılmaya başlandı. Transistörlerin git gide küçülmesiyle de günümüzde artık milyonlarca transistörü bünyesinde barındıran mikroçipler üzerinde oldukça karmaşık mantık devrelerinin kurulması mümkün hale geldi.

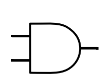
1.2 Boole Cebrinin Temel İşlemleri

Boole cebrinin temel işlemleri yukarıda bahsetmiş olduğumuz üzere VE, VEYA ile DEĞİL işlemleridir. Dijital devre tasarımında bu işlemi gerçekleştiren elemanlar **mantık kapıları** olarak adlandırılmaktadır. x ile y Boole değişkenleri kullanılarak yapılan VE işlemi $\times$ veya . sembolleri kullanılarak sırasıyla $x \times y$ ve $x.y$ şeklinde; ya da arada herhangi bir sembol kullanılmaksızın kısaca xy şeklinde ifade edilebilir.

Tanım 1.1: Doğruluk Tablosu

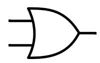
Boole işlemlerinin veya bu işlemleri kullanarak oluşturulan Boole fonksiyonlarının çıkış değerlerinin tüm olası giriş değerlerine karşı sıralandığı tablolara **doğruluk tablosu** adı verilir.

VE işlemine ait doğruluk tablosu ile dijital devre tasarımda kullanılan VE mantık kapısı sembolü aşağıdaki gibidir:



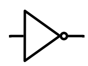
x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

VEYA işlemi ise + sembolü ile temsil edilir. VEYA işlemine ait doğruluk tablosu ile dijital devre tasarımda kullanılan VEYA mantık kapısı sembolü aşağıdaki gibidir:



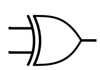
x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

DEĞİL işlemi ise tek bir değişkene uygulanır. x bir değişken olmak üzere bu değişkenin DEĞİL ifadesi x' veya $\bar{x}$ ile gösterilir. DEĞİL işlemine ait doğruluk tablosu ile dijital devre tasarımda kullanılan DEĞİL mantık kapısı sembolü aşağıdaki gibidir:



x	x'
0	1
1	0

Bu temel işlemler kullanılarak farklı Boole işlemleri de tanımlanabilmektedir. Dijital devre tasarımında en çok kullanılan işlemlerden biri olan ÖZEL VEYA (ing. XOR [3]) işlemi yukarıdaki temel işlemler kullanılarak elde edilir. x ve y değişkenleri arasında uygulanan ÖZEL VEYA işlemi $\oplus$ sembolü ile $x \oplus y$ şeklinde gösterilse de aslında bu işlem $x \oplus y = xy' + x'y$ şeklinde VE, VEYA, DEĞİL işlemlerinin bir kombinasyonudur. ÖZEL VEYA işlemine ait mantık kapısı sembolü ve bu işleme ait doğruluk tablosu aşağıdaki gibidir:



x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

VE ile VEYA işlemlerinin sembolleri $\times$ ile $+$ olsa da bu mantıksal işlemler aritmetik çarpma ve aritmetik toplama işlemleri ile karıştırılmamalıdır. Sayılar üzerinde yapılan aritmetik işlemlerle mantıksal Boole işlemleri farklıdır. Mesela, VEYA işleminin $+$ sembolü aritmetik toplam işleminin sembolü ile aynı olsa da doğruluk tablolarına baktığımızda aritmetik toplama işlemine en yakın Boole işleminin VEYA işlemi değil, $\oplus$ ile gösterilen ÖZEL VEYA işlemi olduğu görülür.
Her ne kadar VE işlemi bazen çarpma; VEYA işlemi bazen toplama olarak isimlendirilse de, böyle söylerken aritmetik toplama veya aritmetik çarpmadan bahsetmediğimizi hatırlamanız gerekmektedir.

[3] Exclusive OR

1.3 Boole Cebrinin Temel Özellikleri

Boole cebri kullanılırken işlemlerin öncelik sırası şu şekildedir:

1. Parantez
2. DEĞİL işlemi
3. VE işlemi
4. VEYA işlemi

Boole cebrinde kullanılan temel kurallar ise aşağıdaki gibidir:

1. Değişme Kuralı:
   $x + y = y + x$
   $x.y = y.x$
2. Birleşme Kuralı:
   $x + y + z = (x + y) + z = x + (y + z)$
   $x.y.z = (x.y).z = x.(y.z)$
3. Dağılma Kuralı:
   $x.(y + z) = x.y + x.z$
   $x + (y.z) = (x + y).(x + z)$
4. Etkisiz Eleman Kuralı:
   $x.1 = x$
   $x + 0 = x$
5. Yutan Eleman Kuralı:
   $x.0 = 0$
   $x + 1 = 1$
6. Özdeşlik Kuralı:
   $x.x = x$
   $x + x = x$
7. Tümleyen Kuralı:
   $x + x' = 1$
   $x.x' = 0$
8. Çift Tersleme Kuralı:
   $(x')' = x$
9. Yutma Kuralı:
   $x.(x + y) = x$
   $x + (x.y) = x$
10. De Morgan Kuralı:
   $(x.y)' = x' + y'$
   $(x + y)' = x'.y'$

Bu kurallar dışında Boole cebrinin sahip olduğu bir diğer özellik **dualite prensibi** olarak adlandırılmaktadır. Bu özellik aşağıdaki gibi tanımlanır:

Tanım 1.2: Dualite Prensibi

Dualite prensibi bir Boole ifadesini o ifadenin duali adı verilen eşdeğer bir formda ifade etmemizi sağlar. Bu prensibe göre, eğer orjinal ifade geçerli bir Boole ifadesiyse ifadenin duali de geçerli bir Boole ifadesi olmak zorundadır. Bir Boole ifadesinin dualini bulmak için yapmamız gereken işlemler şunlardır:

- $\times$ işlemini $+$ işlemiyle; $+$ işlemini $\times$ işlemiyle değiştir,
- 1'leri 0; 0'ları ise 1 yap.

Örneğin, $1 + 0 = 1$ ifadesi geçerli bir Boole ifadesidir. Bu ifadenin duali olan $0.1 = 0$ ifadesi de görmüş olduğunuz üzere geçerli bir Boole ifadesidir. Dikkat ederseniz yukarıda vermiş olduğumuz her bir kuraldaki (çift tersleme hariç) iki ifade birbirinin dualidir. Yani ifadelerden birini ispatlarsanız dualite prensibi gereği diğer ifade de doğru olacaktır.

Yukarıda vermiş olduğumuz kuralları ispatlamak için en kolay yöntem doğruluk tablosu oluşturmaktır. Doğruluk tablosu oluşturmanın yanı sıra diğer kuralları ve dualite prensibini kullanarak da ispat yapmamız mümkündür.

Örnek 1.1:

Yutma kuralını ispatlayınız.

Doğruluk tablosu kullanarak ispat:

x	y	$(x+y)$	$x(x+y)$
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Tablodan görüldüğü üzere x'in sütunu ile $x(x+y)$'nin sütunu aynıdır. Dolayısıyla, $x(x+y) = x$ ilişkisi doğrudur. Dualite prensibine göre $x + (xy) = x$ ifadesi de doğru olmalıdır. ■

Boole kurallarını kullanarak ispat:

$$\begin{aligned} x(x+y) &= xx + xy && \text{(Dağılma Kuralı)} \\ &= x + xy && \text{(Özdeşlik Kuralı)} \\ &= x(1+y) && \text{(Dağılma Kuralı)} \\ &= x && \text{(Yutan Eleman Kuralı)} \quad \blacksquare \end{aligned}$$

Tabi bu ispatın geçerli olabilmesi için yukarıda kullanılan kuralların da daha önceden ispatlanmış olması lazım.

Yukarıda iki değişken için vermiş olduğumuz kurallar daha fazla sayıda değişken için genelleştirilebilir. Mesela, üç değişken için De Morgan kuralı

$$(x.y.z)' = x' + y' + z'$$

olur. Bunu ispatlamak için yapmamız gereken iki değişken için ispatladığımız kuralı üç değişken için aşağıdaki gibi (tekrar tekrar) kullanmaktır:

$$\begin{aligned}(x.y.z)' &= (x.(y.z))' \quad \text{(Birleşme Kuralı)}\\ &= x' + (y.z)' \quad \text{(De Morgan Kuralı)}\\ &= x' + y' + z' \quad \text{(De Morgan Kuralı)}\end{aligned}$$

Dualite prensibi gereği $(x + y + z)' = x'.y'.z'$ ifadesi de doğru olmalıdır.

1.4 Boole Fonksiyonları

$B = \{0, 1\}$ olmak üzere $F : B^k \to B$ şeklindeki fonksiyonlara Boole fonksiyonları denir. $k = 0$ için F sabit bir fonksiyondur. Matematik kullanmadan aynı tanımı şu şekilde yapabiliriz: Bir F Boole fonksiyonu k adet Boole değişkeni alıp çıkışta bu k değişkenin farklı kombinasyonlarına bağlı tek bir değer döndüren bir fonksiyondur. Eğer fonksiyon değişken almıyorsa ($k = 0$), bu durumda fonksiyon $F = 0$ veya $F = 1$ şeklinde sabit bir fonksiyondur.

Örnek olarak $F(x, y) = xy + yx'$ fonksiyonu iki değişkenli ($k = 2$) bir fonksiyondur. Bu fonksiyon kısaca $F = xy+yx'$ şeklinde de ifade edilebilir. Bu fonksiyonun doğruluk tablosu aşağıdaki gibidir (doğruluk tablosu x ve y değerleri için F'nin alacağı değer hesaplanarak doldurulur):

x	y	F
0	0	0
0	1	1
1	0	0
1	1	1

Yukarıdaki doğruluk tablosundan $F = y$ olduğu görülmektedir (çünkü y ve F sütunları aynı). Bu da $F(x, y) = xy + yx'$ fonksiyonunun $F(x, y) = y$ fonksiyonu ile aynı sonucu verdiği anlamına gelir. Bunu Boole kurallarını kullanarak aşağıdaki gibi de görebiliriz:

$$F = xy + yx' = y(x + x')$$
$$= y$$

Yukarıdaki örnekte görüldüğü gibi Boole fonksiyonları kendisinden daha az değişkene sahip, daha basit fonksiyonlarla aynı sonucu verebilir.

Tanım 1.3:

Bir Boole fonksiyonunun daha az değişken kullanarak ifade edilmesi işlemine **sadeleştirme** adı verilir. Mevcut değişken sayısından daha az değişkenle ifade edilmesinin mümkün olmadığı Boole fonksiyonları için **en sade halinde**dir denilir.
Örneğin $F = x$ fonksiyonu en sade halindedir.

$F(x, y, x) = xyz + x'$ üç değişkenli ($k = 3$) bir Boole fonksiyonu örneğidir. Bu fonksiyonun doğruluk tablosu aşağıdaki gibidir (xyz ve x' sütunlarını yazmak zorunlu değildir; fakat bu ara işlemleri yapmak doğruluk tablosunun doldurulmasını epeyce kolaylaştırır):

Doğruluk tablosunda değişken değerlerinin yan yana oluşturduğu ikili değerlerin 0,1,. . .,7 şeklinde onluk değerlere karşılık geldiğine dikkat edin. Doğruluk tablolarında değişken değerleri bu şekilde sıralı olmalıdır. Bu sırayı elde etmek için alttaki satırın değerini üstteki satıra +1 ekleyerek (ikili tabanda toplamayla) bulabilirsiniz.

x	y	z	xyz	x'	F
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	0	1

Görüldüğü üzere iki değişkenli bir fonksiyonun doğruluk tablosu 4 satır içerirken, üç değişkenli bir fonksiyonun doğruluk tablosu 8 satır içermektedir.

Genel olarak k değişkenli bir fonksiyonun doğruluk tablosu 2^k satır içermelidir.

Bu (bir önceki ders notunda görmüş olduğumuz üzere) k-bitlik bir verinin tüm olası kombinasyonlarının sayısıdır. Bu k-bitlik tüm olası kombinasyonlar, doğruluk tablosunda değişkenlerin hemen altında sıralanır.

Eğer k değişkenli bir doğruluk tablosu 2^k satır içeriyorsa, k değişkenli bir F Boole fonksiyonunun alabileceği değerler de 2^k satırlık bir veri olacaktır. Peki bu 2^k satırlık ikili tabandaki veri (yani 2^k-bitlik veri) kaç farklı kombinasyon oluşturabilir? Daha önce görmüş olduğumuz üzere 2^{2^k} farklı kombinasyon oluşturabilir. Bu ise aşağıdaki sonucu elde etmemizi sağlar:

k adet değişken kullanarak 2^{2^k} **farklı** Boole fonksiyonu oluşturulabilir.

Boole fonksiyonlarına uygulanan DEĞİL işlemi F' (ya da $\bar{F}$) ile gösterilir. Mesela, yukarıdaki $F = xyz + x'$ fonksiyonu için $F' = (xyz + x')'$ şeklindedir. DEĞİL'i (ya da tersi) alınmış fonksiyonları sadeleştirmek için De Morgan kuralı uygulanmalıdır. Mesela, F' fonksiyonu De Morgan kuralı kullanılarak şu şekilde sadeleştirilebilir:

$$\begin{aligned} F' = (xyz + x')' &= (xyz)'.x \\ &= x.(x' + y' + z') \\ &= x.x' + x.(y' + z') \\ &= xy' + xz' \end{aligned}$$

1.5 Fonksiyonların Kanonik Formları

Bir önceki kısımda görmüş olduğumuz üzere, farklı Boole fonksiyonları çıkışta aynı sonucu verebilmektedir. Bu ise, farklıymış gibi görünen birden çok fonksiyonun aslında aynı fonksiyon olduğu anlamına gelir. Örneğin, daha önce görmüş olduğumuz üzere $F = xy + yx'$ ile $F = x$ fonksiyonları aynı sonucu üretmektedir. Üç değişkenli $F = xyz + x'$ fonksiyonu için Boole kurallarını kullanarak pek çok eşdeğer fonksiyon bulabiliriz. Mesela,

$$\begin{aligned} F &= xyz + x' \\ &= xyz + x'(y + y') \\ &= xyz + x'y + x'y' \end{aligned}$$

olduğu için $F = xyz + x'$ ile $F = xyz + x'y + x'y'$ fonksiyonları aynıdır.

Olaya daha genel bakarsak, yine önceki kısımdan görmüş olduğumuz üzere, k değişken ile sadece 2^{2^k} farklı fonksiyon oluşturabileceğimizi görürüz. Mesela, tek bir değişken ile oluşturabileceğimiz fonksiyonlar en sade hallleriyle $F_0(x) = x$, $F_1(x) = x'$, $F_2(x) = 0$ ve $F_3(x) = 1$ olmak üzere $2^{2^1} = 4$ tanedir. $G(x) = x + x'$, $H(x) = 1 + xx'$ benzeri diğer tüm tek değişkenli fonksiyonlar bu 4 fonksiyondan biriyle aynı olmalıdır (ki Boole kuralları ile bunu görmek kolaydır).

İki değişken kullanarak oluşturabileceğimiz tüm farklı fonksiyonlar aşağıdaki tabloda sıralanmıştır:

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Görmüş olduğunuz üzere 2 değişken $2^2 = 4$ satırlık bir doğruluk tablosu oluşturmuş ve 4 satırlık ikili veri ile F_0 ile F_{15} arasında $2^4 = 16$ farklı fonksiyon oluşturulmuştur. Bu 16 fonksiyon en sade halleriyle aşağıdaki gibidir:

$$\begin{array}{llll} F_0 = 0 & F_4 = x'y & F_8 = x'y' & F_{12} = x' \\ F_1 = xy & F_5 = y & F_9 = xy + x'y' & F_{13} = x' + y \\ F_2 = xy' & F_6 = x \oplus y & F_{10} = y' & F_{14} = x' + y' \\ F_3 = x & F_7 = x + y & F_{11} = x + y' & F_{15} = 1 \end{array}$$

Yukarıdaki $F(x, y) = xy + yx'$ örneğimizin bu listede bulunmamasından bu fonksiyonun aslında F_0 ile F_{15} arasındaki fonksiyonlardan birine eşdeğer olması gerektiğini anlarız. Yine $G(x, y) = xy + (x' + y)'$ ve $H(x, y) = (x' + y)(x'y)'$ benzeri diğer fonksiyonların da sadeleştirilerek yukarıdaki fonksiyonlardan birine dönüştürülebileceği aşikardır.

Vermiş olduğumuz örnekler basit sadeleştirmelerle birbirlerine dönüştürülebildiği için bu fonksiyonların aynı olduğunu görebilmek kolaydır. Fakat, daha çok değişken işin içine girerse veya sadeleştirme için arada daha fazla kural kullanılması gerekirse, verilen iki fonksiyonun aynı mı olup olmadığını belirlemek kolay olmayacaktır. Hatta $F_8 = x'y' = (x+y)'$ ve $F_{14} = x'+y' = (xy)'$ örneklerinde olduğu gibi en sade haldeki fonksiyonlar bile birden fazla şekilde ifade edilebileceğinden, bazen en sade haldeki aynı fonksiyonları bile farklı fonksiyonlar olarak değerlendirmek mümkündür.

İşte bu nedenle, matematikçiler ve mühendisler Boole fonksiyonlarını ifade etmek için kanonik bir form geliştirme ihtiyacı duymuşlardır. Burada **kanonik form**dan kastımız şudur: Fonksiyonları öyle bir formda ifade edelim ki, bu formdayken aynı fonksiyonu sadece tek bir ifadeyle göstermek mümkün olsun. Yani bu formdayken aynı fonksiyonu iki veya daha fazla şekilde ifade etmek mümkün olmasın. Fonksiyonları böyle bir formda ifade etmek onları bir nevi etiketlememizi sağlayacaktır. Bu sayede, iki fonksiyonu alıp bunları kanonik formda ifade ettiğimizde, eğer fonksiyonların ifadeleri aynı elde ediliyorsa (yani fonksiyonların etiketleri aynıysa), bu fonksiyonların aslında aynı fonksiyona karşılık geldiğini belirlemiş oluruz. Bu fonksiyonlar sadeleştirme veya Boole kuralları uygulanarak birbirlerine dönüştürülebilirler. Benzer şekilde, kanonik formları farklı olan (etiketleri farklı olan) fonksiyonlar farklı fonksiyonlar olmalıdır. Doğal olarak, bu tür fonksiyonları birbirlerine dönüştürmek mümkün değildir.

Fonksiyonları kanonik formda göstermek için, **miniterimlerin toplamı** ve **maksiterimlerin çarpımı** şeklinde iki gösterim formu kullanılır.

1.5.1 Miniterimlerin Toplamı

Tanım 1.4: Miniterim

Miniterimler [a], mevcut tüm değişkenlerin kendisinin veya tersinin (sadece bir kez) seçilmesiyle elde edilen ifadelerin VE'sinin alındığı (çarpıldığı) terimlerdir. Örneğin, iki değişken için miniterimler: xy, xy', $x'y$ ve $x'y'$ olacaktır. Üç değişken için ise miniterimler: xyz, xyz', $xy'z$, $xy'z'$, $x'yz$, $x'yz'$, $x'y'z$ ve $x'y'z'$ şeklindedir.

Genel olarak miniterimleri oluştururken çarpımda değişkenin kendisi veya tersini almak gibi 2 seçeneğimiz olduğundan, k değişken için 2^k farklı miniterim olmalıdır.

[a]ing. minterms (minimum terms)

Miniterimlerin neden kanonik gösterimde kullanılabileceğini anlamak için belli bir miniterimi 1 yapan değişken değerleri için diğer tüm miniterimlerin 0 olması gerektiğini görmeniz gerekmektedir. Örnek olarak, iki değişken için oluşturulmuş aşağıdaki doğruluk tablosunu inceleyin:

x	y	Miniterim	Miniterim Sembolü	m_0	m_1	m_2	m_3
0	0	$x'y'$	m_0	1	0	0	0
0	1	$x'y$	m_1	0	1	0	0
1	0	xy'	m_2	0	0	1	0
1	1	xy	m_3	0	0	0	1

Yukarıdaki tabloda miniterimlerin sembollerini etiketlerken giriş değerlerinin bit kombinasyonu ondalık tabanlı sayıya çevrilmiştir. Mesela, $x = 0$ ve $y = 0$ için kombinasyon $(00)_2 = 0$ olduğundan bu giriş kombinasyonuna karşılık gelen miniterim m_0 olarak etiketlenmiştir. Benzer şekilde, $x = 1$ ve $y = 0$ için kombinasyon $(10)_2 = 2$ olduğundan bu giriş kombinasyonuna karşılık gelen miniterim m_2 olarak etiketlenmiştir.

Yukarıdaki tablodan görüldüğü üzere, giriş $x = 0$, $y = 0$ iken $m_0 = x'y'$ değeri 1 olmakta ve diğer tüm miniterimler (m_1, m_2, m_3) 0 olmaktadır. Bu durum diğer tüm giriş kombinasyonları için aynıdır. Yani aynı anda sadece bir tane miniterim 1 olmaktadır. İşte m_0 olarak $x'y'$ miniterimini seçmiş olmamızın nedeni de budur. Genel olarak, bir m_i sembolüne karşılık gelen miniterim seçilirken, verilen giriş kombinasyonuna göre 1 değeri alan miniterim seçilir. Mesela, $m_2 = xy'$ seçmemizin nedeni m_2 için girişlerin $2 = (10)_2$ olması ve $x = 1$, $y = 0$ için sadece xy' teriminin 1 olmasıdır.

Yukarıdaki tablonun benzerini kaç değişken kullanarak oluşturursanız oluşturun, giriş değerlerinin bir kombinasyonuna karşılık miniterimlerden sadece birinin 1 olduğunu göreceksiniz. Bu nedenle, elde ettiğimiz sonuçlar k adet değişken kullanıldığında da benzer şekildedir.

Örnek 1.2:

Üç değişken için m_5 ve m_8 miniterimlerinin karşılıklarını bulunuz.

$5 = (101)_2$ olduğu için m_5 için girişler $x = 1$, $y = 0$ ve $z = 1$'dir. m_5 bu girişlere karşılık 1 değeri döndüren miniterim olmalıdır. Dolayısıyla, $m_5 = xy'z$ olmalıdır. (Bu seçimi yaparken VE işleminin 1 döndürebilmesi için işleme katılan tüm değerlerin 1 olması gerektiğine dikkat etmeniz gerekmektedir.)

Üç değişken için m_0 ile m_7 arasında toplam $2^3 = 8$ miniterim tanımlanır. Üç değişken için m_8 yoktur!
(*Tabi siz kendiniz dört değişken için m_8'in ne olması gerektiğini bulabilirsiniz.*)

Buradan, miniterimleri 1 yapan değerleri kullanarak bir Boole fonksiyonunu eşsiz bir şekilde ifade edebileceğimizi tahmin etmiş olmanız lazım. Örnek olarak aşağıdaki doğruluk tablosuna sahip iki değişkenli fonksiyonu ele alalım:

x	y	F
0	0	1
0	1	0
1	0	1
1	1	0

Tablodan görüldüğü üzere bu fonksiyon $x = 0$, $y = 0$ ve $x = 1$, $y = 0$ giriş kombinasyonları için 1 olmakta; diğer tüm girişler için 0 olmaktadır. Hatırlayacağınız üzere, m_0 miniterimi sadece $x = 0$, $y = 0$ girişi için 1 olmakta ve diğer tüm girişler için 0 olmaktadır. Benzer şekilde, m_2 miniterimi sadece $x = 1$, $y = 0$ girişi için 1 olmakta ve diğer tüm girişler için 0 olmaktadır. Aradaki ilişkiyi görebildiniz mi? Aslında F fonksiyonu m_0 "veya" m_2 şartlarını sağlayan giriş değerleri için 1 olmakta; diğer girişlerde 0 olmaktadır. Dolayısıyla, F fonksiyonu aşağıdaki formda ifade edilebilir:

$$F = m_0 + m_2 = x'y' + xy'$$

Herhangi bir giriş kombinasyonu için tek bir miniterim 1 olabileceğinden bu tür bir gösterim eşsizdir. Yani aynı fonksiyonu başka miniterimler kullanarak ifade etmeniz mümkün değildir. İşte bu nedenle, bu tür gösterime kanonik formda gösterim adı verilir. Kanonik gösterim çoğunlukla en sade gösterim değildir. Mesela, yukarıdaki fonksiyon

$$F = x'y' + xy' = y'(x' + x) = y'$$

şeklinde sadeleştirilir.

Genel olarak, bir fonksiyonu miniterimler kullanarak kanonik formda ifade etmek istediğimizde o fonksiyonun doğruluk tablosundan o fonksiyonu 1 yapan giriş değerlerini seçerek bu giriş değerlerine karşılık gelen miniterimlerin VEYA'sını alırız (başka bir deyişle miniterimleri toplarız). Dolayısıyla, bu şekilde elde ettiğimiz kanonik form **miniterimlerin toplamı** olarak adlandırılır. Eğer, bu toplam $F = m_0 + m_2 + m_7$ gibiyse bunu kısaca

$$F = \sum(0, 2, 7)$$

ile toplam sembolüyle ifade ederiz. (Tabi bu toplam VEYA işlemi anlamına gelmektedir. Aritmetik toplama ile herhangi bir alakası yoktur.)

Ek Bilgi

Miniterimlerin kendisi çarpımla (VE işlemi ile) elde edildiğinden miniterimlerin toplamı aynı zamanda **kanonik çarpımların toplamı** (ing. *canonical sum of products*, veya *canonical SOP*) olarak da adlandırılır.

Örnek 1.3:

Aşağıda doğruluk tablosu verilmiş olan fonksiyonu miniterimlerin toplamı şeklinde ifade ediniz.

x	y	z	F
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

Tablodan görüldüğü üzere, fonksiyon sadece $(001)_2 = 1$, $(011)_2 = 3$, $(110)_2 = 6$ ve $(111)_2 = 7$ giriş kombinasyonları için 1 olmaktadır. Buna göre,

$$F = \sum(1, 3, 6, 7) = m_1 + m_3 + m_6 + m_7$$

olarak ifade edilir. Tabi bu da $F = x'y'z + x'yz + xyz' + xyz$ ifadesine karşılık gelir.

Verilen bir fonksiyonu miniterimlerin toplamı şeklinde ifade edebilmek için doğruluk tablosu oluşturabileceğimiz gibi Boole kurallarını kullanarak da fonksiyon ifadesini kanonik forma dönüştürebiliriz.

Örnek 1.4:

$F(x,y) = xy + yx'$, $G(x,y) = y$ ve $H = (x'+y)(x'y)'$ fonksiyonlarını miniterimlerin toplamı şeklinde ifade ediniz.

$F = x'y + xy = m_1 + m_3$ olduğu için zaten miniterimlerin toplamı formunda. ($x'y$'nin m_1 olduğunu bulmak için $x'y$'yi 1 yapan giriş kombinasyonunun $(01)_2 = 1$ olması gerektiğine dikkat edin.) G'yi bu formatta göstermek için sadeleştirmenin tersini yapmamız lazım:

$$\begin{aligned} G = y &= y(x + x') \\ &= xy + x'y \\ &= m_1 + m_3 \end{aligned}$$

Görüldüğü üzere F ve G fonksiyonları aynı kanonik forma sahip ki bu fonksiyonların aynı olduğunu daha önce görmüştük. H için de benzer bir işlem yaparsak

$$\begin{aligned} H &= (x' + y)(x + y') \\ &= x'x + x'y' + xy + yy' \\ &= x'y' + xy \\ &= m_0 + m_3 \end{aligned}$$

buluruz. Kanonik formu bulurken hata yapıp yapmadığımızı, doğruluk tablosu oluşturarak kontrol edelim:

x	y	H
0	0	1
0	1	0
1	0	0
1	1	1

Buradan da yine $H = \sum(0,3) = m_0 + m_3$ sonucuna ulaşırız.

1.5.2 Maksiterimlerin Çarpımı

Tanım 1.5: Maksiterim

Maksiterimler [a], mevcut tüm değişkenlerin kendisinin veya tersinin (sadece bir kez) seçilmesiyle elde edilen ifadelerin VEYA'sının alındığı (toplandığı) terimlerdir. Örneğin, iki değişken için maksiterimler: $x + y$, $x + y'$, $x' + y$ ve $x' + y'$ olacaktır. Üç değişken için ise maksiterimler: $x + y + z$, $x + y + z'$,

$x+y'+z$, $x+y'+z'$, $x'+y+z$, $x'+y+z'$, $x'+y'+z$ ve $x'+y'+z'$ şeklindedir.

Genel olarak maksiterimleri oluştururken toplamda değişkenin kendisi veya tersini almak gibi 2 seçeneğimiz olduğundan, k değişken için 2^k farklı maksiterim olmalıdır.

[a] ing. maxterms (maximum terms)

Maksiterimlerde, miniterimlerin aksine herhangi bir giriş kombinasyonu için sadece bir maksiterim 0 olurken diğer tüm maksiterimler 1 değerini alır. Örnek olarak aşağıdaki iki değişkenli maksiterim tablosunu inceleyiniz:

x	y	Maksiterim	Maksiterim Sembolü	M_0	M_1	M_2	M_3
0	0	$x+y$	M_0	0	1	1	1
0	1	$x+y'$	M_1	1	0	1	1
1	0	$x'+y$	M_2	1	1	0	1
1	1	$x'+y'$	M_3	1	1	1	0

Maksiterimleri ifade etmek için büyük M sembolünü kullanmamız dışında maksiterimleri yine miniterimleri etiketlediğimiz gibi etiketleriz: İkili tabandaki giriş kombinasyonunun ondalık karşılığı maksiterimin indisi olur. Mesela, $x=0$, $y=1$ girişi için $(01)_2 = 1$ olduğundan bu girişe karşılık gelen maksiterim sembolü M_1'dir. M_1'e karşılık gelen maksiterim değeri olarak $x+y'$ seçmemizin nedeni ise $x=0$, $y=1$ girişi için sadece bu maksiterimin 0 olmasıdır. Diğer tüm maksiterimleri benzer bir şekilde seçtiğimizde, yukarıdaki tablodan herhangi bir giriş kombinasyonu için sadece tek bir maksiterimin 0 olurken geri kalan tüm maksiterimlerin 1 olduğunu görürüz. Bu durum, tıpkı miniterimlerde olduğu gibi k değişken için genelleştirilebilir.

Örnek 1.5:

Dört değişken için M_{12} ve M_{15} maksiterimlerinin karşılıklarını bulunuz.

$12 = (1100)_2$ olduğu için M_{12} için girişler $x=1$, $y=1$, $z=0$ ve $w=0$'dır. M_{12} bu girişlere karşılık 0 değeri döndüren maksiterim olmalıdır. Dolayısıyla, $M_{12} = x'+y'+z+w$ olmalıdır. (Bu seçimi yaparken VEYA işleminin 0 döndürebilmesi için işleme katılan tüm değerlerin 0 olması gerektiğine dikkat etmeniz gerekmektedir.)

Benzer şekilde $15 = (1111)_2$ olduğundan, girişler $x=1$, $y=1$, $z=1$ ve $w=1$ olup $M_{15} = x'+y'+z'+w'$ 'dir.

Herhangi bir giriş kombinasyonu için sadece bir tane maksiterim 0 olabiliyorsa, bu özellik fonksiyonları 0 yapan giriş kombinasyonlarına karşılık gelen maksiterimleri kullanarak o fonksiyonu kanonik bir şekilde ifade edebilmemizi sağlar. Yine iki değişkenli örneğimizi ele alalım:

x	y	F
0	0	1
0	1	0
1	0	1
1	1	0

Yukarıdaki fonksiyon girişler $x = 0$, $y = 1$ veya $x = 1$, $y = 1$ olduğunda 0 olmakta; diğer tüm durumlarda 1 olmaktadır. başka bir deyişle, F fonksiyonu M_1 "ve" M_3 şartlarını sağlayan giriş değerleri için 0 olmakta; diğer girişlerde 1 olmaktadır. Dolayısıyla, F fonksiyonu aşağıdaki formda ifade edilebilir:

$$F = M_1 M_3 = (x + y')(x' + y')$$

Genel olarak, bir fonksiyonu maksiterimler kullanarak kanonik formda ifade etmek istediğimizde o fonksiyonun doğruluk tablosundan o fonksiyonu 0 yapan giriş değerlerini seçerek bu giriş değerlerine karşılık gelen maksiterimlerin VE'sini alırız (başka bir deyişle maksiterimleri çarparız). Dolayısıyla, bu şekilde elde ettiğimiz kanonik form **maksiterimlerin çarpımı** olarak adlandırılır. Eğer, bu çarpım $F = M_1 M_4 M_5$ gibiyse bunu kısaca

$$F = \prod(1, 4, 5)$$

ile çarpım sembolüyle ifade ederiz. (Tabi bu çarpım VE işlemi anlamına gelmektedir. Aritmetik çarpma ile herhangi bir alakası yoktur.)

Ek Bilgi

Maksiterimlerin kendisi toplamla (VEYA işlemi ile) elde edildiğinden maksiterimlerin çarpımı aynı zamanda **kanonik toplamların çarpımı** (ing. *canonical product of* sums, veya *canonical POS*) olarak da adlandırılır.

Örnek 1.6:

Aşağıda doğruluk tablosu verilmiş olan fonksiyonu maksiterimlerin çarpımı şeklinde ifade ediniz.

x	y	z	F
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

Tablodan görüldüğü üzere, fonksiyon sadece $(000)_2 = 0$, $(010)_2 = 2$, $(100)_2 = 4$ ve $(101)_2 = 5$ giriş kombinasyonları için 0 olmaktadır. Buna göre,

$$F = \prod(0, 2, 4, 5) = M_0 M_2 M_4 M_5$$

olarak ifade edilir. Tabi bu da $F = (x+y+z)(x+y'+z)(x'+y+z)(x'+y+z')$ ifadesine karşılık gelir. Bu örnekte,

$$F = \sum(1, 3, 6, 7) = m_1 + m_3 + m_6 + m_7$$

olduğuna dikkat ediniz. Yani maksiterimlerin çarpımı ifadesindeki indislerin dışında kalan indisleri kullanarak miniterimlerin toplamı ifadesi elde edebiliriz (ya da tam tersi).

Örnek 1.7:

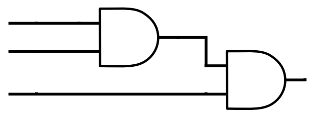
$F(x, y, z) = xy + xz'$ fonksiyonunu maksiterimlerin çarpımı şeklinde ifade edin.

Fonksiyonu miniterimlerin toplamı olarak göstermek daha kolaydır:

$$\begin{aligned} F &= xy(z + z') + xz'(y + y') \\ &= xyz + xyz' + xyz' + xy'z' \\ &= m_7 + m_6 + m_3 + m_1 \\ &= \sum(1, 3, 6, 7) \end{aligned}$$

Miniterimlerin toplamında kullanılmayan indisler maksiterimlerin çarpımında kullanılması gereken indisler olduğundan, tıpkı bir önceki örnekte olduğu gibi:

$$F = \prod(0, 2, 4, 5)$$



Şekil 2: Üç girişli VE kapısının iki tane iki girişli VE kapısı ile oluşturulması.

Maksiterimler ve miniterimler arasında

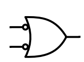
$$m_i = M_i'$$

ilişkisi vardır. Örneğin (üç değişken için),

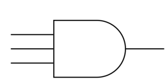
$$m_2' = (x'yz')' = (x + y' + z) = M_2$$

2 Devre Gerçekleştirimi

Daha önce Bölüm 1.2'de temel Boole işlemlerine karşılık gelen mantık kapılarını görmüştük. Tablo 1'de ise dijital devre tasarımında kullanılan temel mantık kapılarının bir listesi verilmektedir. Bu tabloda kullanılan kapılar dışında bazen girişlerinde ○ sembolü olan aşağıdaki gibi mantık kapılarıyla da karşılaşabilirsiniz:

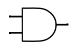
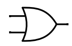
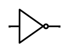
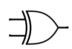
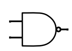
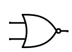
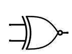
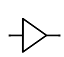


Yukarıdaki kapının çıkışı $F = x' + y'$ şeklindedir. Yani kapı girişlerindeki yuvarlak sembol girişin terslendirildiği anlamına gelir. Bu kapının De Morgan kuralına göre VEDEĞİL kapısına eşdeğer olduğuna dikkat ediniz. Bunun dışında, Tablo 1'deki iki girişli kapıların ikiden fazla girişli şekilde olabileceğini de söylemek gerekir. Mesela, üç girişli bir VE kapısı aşağıdaki gibidir.

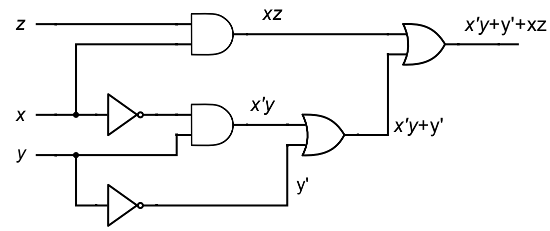


Laboratuvarda en çok sorulan sorulardan biri: "Hocam üç girişli kapı yok ne yapalım?"

Eğer elinizde üç girişli bir VE kapısı yoksa, Şekil 2'deki gibi iki tane VE kapısı kullanarak üç girişli bir VE kapısı oluşturmanız mümkündür:

Sembol	Kapı Adı	Çıkış Eşitliği
	VE (AND)	$F = xy$
	VEYA (OR)	$F = x + y$
	DEĞİL (NOT)	$F = x'$
	ÖZEL VEYA (XOR)	$F = x \oplus y$
	VEDEĞİL (NAND)	$F = (xy)'$
	VEYA DEĞİL (NOR)	$F = (x + y)'$
	ÖZEL VEYA DEĞİL (XNOR)	$F = (x \oplus y)'$
	TAMPON (BUFFER)	$F = x$

Tablo 1: Lojik Devre Tasarımında Kullanılan Temel Mantık Kapıları



Şekil 3: $F = y' + xz + x'y$ fonksiyonunun devre gerçekleştirimi.

Boole fonksiyonları verilen bu mantık kapıları yardımıyla dijital devre olarak kurulabilir. Devre kurulurken Boole fonksiyonunun nasıl ifade edildiği önem arz etmektedir. Şimdi sırasıyla farklı Boole fonksiyonu formlarına karşı dijital devrelerin nasıl kurulacağını inceleyeceğiz.

2.1 Çarpımların Toplamı Formu

Daha önce miniterimlerin toplamında görmüş olduğumuz üzere VE işlemi ile oluşturulmuş farklı terimlerin VEYA işlemi ile birleştirildiği forma **çarpımların toplamı** (ing. *sum of products, SOP*) adı verilir. Çarpımların toplamı formu kanonik olmak zorunda değildir. Bu nedenle, bu form miniterimlerin çarpımıyla aynı şey değildir.

Çarpımların toplamına örnek olarak aşağıdaki fonksiyonu inceleyelim:

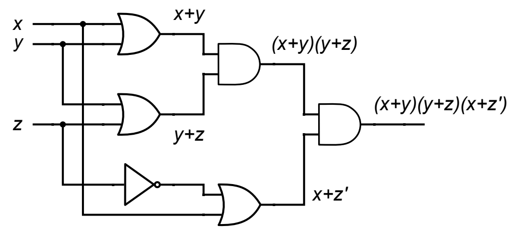
$$F(x, y, z) = y' + xz + x'y$$

Bu formda dikkat etmeniz gereken VE ile oluşturulmuş terimlerin VEYA'sının alındığıdır. Buna göre devreyi kurmaya öncelikle VE işlemlerinden başlanır. VE işlemine değişkenin tersi giriyorsa, VE kapısına girmeden o değişken girişi DEĞİL kapısı ile terslendirilmelidir. Yukarıdaki örnekte y' tek başına kullanıldığı için bu terimde bir VE kapısı kullanmamıza gerek yok. Fakat, xz ve $x'y$ terimleri için iki tane VE kapısı kullanılmalı. Daha sonra, üç terim VEYA işlemine sokularak devre tamamlanır. Bunu üç girişli bir VEYA kapısı ile temsil edebileceğimiz gibi iki tane VEYA kapısını art arda kullanarak da yapabilir. Buna göre, yukarıdaki fonsiyonun devre karşılığı Şekil 3'deki gibi olacaktır.

Genel olarak bu formdaki fonksiyonlar bu şekilde kurulmaktadır.

2.2 Toplamların Çarpımı Formu

Daha önce maksiterimlerin çarpımında görmüş olduğumuz üzere VEYA işlemi ile oluşturulmuş farklı terimlerin VE işlemi ile birleştirildiği forma **toplamların çar-**



Şekil 4: $F = (x + y)(x + z')(y + z)$ fonksiyonunun devre gerçekleştirimi.

pımı (ing. *product of sums, POS*) adı verilir. Toplamların çarpımı formu da kanonik olmak zorunda değildir. Yani, bu form maksiterimlerin çarpımıyla aynı şey değildir.

Çarpımların toplamına örnek olarak aşağıdaki fonksiyonu inceleyelim:

$$F(x, y, z) = (x + y)(x + z')(y + z)$$

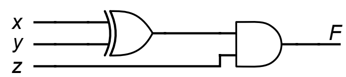
Bu formda dikkat etmeniz gereken VEYA ile oluşturulmuş terimlerin VE'sinin alındığıdır. Buna göre, devreyi kurmaya öncelikle VEYA işlemlerinden başlanır. VEYA işlemine değişkenin tersi giriyorsa, VEYA kapısına girmeden o değişken girişi DEĞİL kapısı ile terslendirilmelidir. Yukarıdaki örnekte üç tane VEYA ile oluşturulmuş terim VE işlemine sokulmaktadır. Bu nedenle, öncelikle üç tane VEYA kapısı kullanılarak $x + y$, $x + z'$ ve $y + z$ terimleri oluşturulmalıdır. Tabi bu yapılırken ikinci terimdeki z girişi kapıya girmeden önce terslendirilmelidir. Üç tane VEYA kapısı kullanıldıktan hemen sonra bu kapıların çıkışlarının VE işlemine tabi tutulmasıyla devre tamamlanmış olur. Bunu üç girişli bir VE kapısı kullanarak yapabileceğiniz gibi iki tane VE kapısını art arda kullanarak da yapabilirsiniz. Buna göre, yukarıdaki fonsiyonun devre karşılığı Şekil 4'deki gibi olacaktır.

Genel olarak bu formdaki fonksiyonlar bu şekilde kurulmaktadır.

2.3 Serbest Form

Boole fonksiyonları ne çarpımların toplamı ne de toplamların çarpımı olmayan serbest bir formda da bulunabilir. Böyle serbest formdaki bir fonksiyonu çarpımların toplamı veya toplamların çarpımı formuna dönüştürerek kurmamız mümkündür. Fakat, bazı durumlarda serbest formdaki bir devreyi doğrudan kurmak yukarıdaki iki formu kullanarak devreyi kurmaktan daha pratik olabilir. Örnek olarak aşağıdaki fonksiyonu inceleyelim:

$$F(x, y, z) = z(xy' + x'y)$$



Şekil 5: $F = z(xy' + x'y)$ fonksiyonunun devre gerçekleştirimi.

Görüldüğü üzere bu form ne toplamların çarpımı ne de çarpımların toplamıdır. Tabi ki dağılma kuralını uygulayarak bu fonksiyonu kolayca $F = xy'z + x'yz$ şeklinde çarpımların toplamı formuna sokabiliriz. Fakat, bu devreyi kurmak orijinal devreyi kurmaktan daha zor olacaktır. Çünkü, orijinal formda devre z ve $xy'+x'y$ ifadelerinin VE'sinin alınmasıdır. Bölüm 1.2'den hatırlarsanız $xy' + x'y$ ifadesi $x \oplus y$ işlemine karşılık gelir. Dolayısıyla, bu devreyi hiç DEĞİL kapısı kullanmadan sadece bir VE ile bir XOR kapısı kullanarak Şekil 5'deki gibi kurmak daha pratik olacaktır.

2.4 Pozitif ve Negatif Lojik

Dijital devreleri mantık kapıları ile kurarken mantık devrelerinin girişlerine hangi voltaj değerlerini verdiğimizde mantıksal 1 ve mantıksal 0 değerlerini elde edeceğimiz o mantık kapısının üretim türüne ve hangi tür lojik kullanıldığına bağlıdır. Mantık kapılarının datasheet'lerine baktığınızda [4]doğruluk tablolarının 0 ve 1 yerine H ve L ifadeleri içerdiğini görürsünüz. H ve L sırasıyla "Higher signal value" (tr. Yüksek sinyal değeri) ve "Lower signal value" (tr. Düşük sinyal değeri) ifadelerinin kısaltılmışıdır. Bunun nedeni, dijital sistemlerde mantıksal 1 ve 0 ifadelerini temsil eden iki seviyeli bir sinyalin kullanılıyor olmasıdır. Bu sinyalin yüksek değeri H ile ifade edilirken düşük değeri L ile ifade edilir. Mesela kullanılan sinyalin seviyeleri (farazi olarak) +15V ve +10V olsun. Bu durumda, H= +15V ve L= +10V olacaktır.

Tanım 2.1:

Eğer, H voltajı mantıksal 1 değeri ve L voltajı mantıksal 0 değeri olarak kullanılıyorsa, bu tür kullanıma **pozitif lojik** adı verilir. Tam tersi, H voltajı mantıksal 0 değeri ve L voltajı mantıksal 1 değeri olarak kullanılıyorsa, bu tür kullanıma **negatif lojik** adı verilir.

Bu dersin laboratuvar uygulamalarının tümünde pozitif lojik kullanılacaktır. Yani H değeri mantıksal 1; L değeri mantıksal 0 olarak kullanılacaktır. Peki ama H ve L voltajlarının değeri ne olmalı? Bu ise kullanılan mantık kapısının üretim teknolojilerine bağlıdır. Bu üretim teknolojileri TTL (transistor-transistor logic), CMOS (complementary metal-oxide semiconductor) ve ECL (emitter-coupled logic) olmak üzere

[4]Mesela 7408 entegresinin datasheet'ine bakın: https://www.electroschematics.com/wp-content/uploads/2013/07/7408-datasheet.pdf

gruplara ayrılır. Düşük güç tüketimi nedeniyle günümüzde CMOS tabanlı mantık kapıları yaygın olarak kullanılmaktadır. Bu teknolojiler için H ve L değerleri aşağıdaki tablodaki gibidir:

Teknoloji	L Voltajı	H Voltajı	Besleme
CMOS	0V ile $1/3V_{DD}$	$2/3V_{DD}$ ile V_{DD}	V_{DD}
TTL	0V ile 0.8V	2V ile V_{CC}	$V_{CC} = +5V \mp \%10$
ECL	V_{EE} ile −1.4V	−1.2V ile 0V	$V_{EE} \cong -5.2$V

Görüldüğü üzere teoride H ve L değerleri sabit değerler olsa da pratikte belli aralıktaki değerlerdir. Mesela TTL devreler için L voltajı (yani mantıksal 0 değeri) 0 ile +0.8V arasındaki voltaj değerleri iken; H voltajı (yani mantıksal 1 değeri) +2V ile +5V arasındaki voltaj değerleridir. Tabi siz laboratuvarda TTL entegreler kullanırken H ve L olarak sırasıyla +5V ve 0V kullanacaksınız.

Laboratuvarda kullandığınız entegreler 74LS ile başlıyorsa TTL'dir. Eğer entegre 74HC ile başlıyorsa entegre CMOS'tur. CMOS entegrenin besleme voltajı +2V ile +6V arasında olabilir fakat bunlar TTL ile beraber kullanıma uygun değildirler. 74HCT ile başlayan entegreler ise hem CMOS olup hem de TTL ile beraber kullanıma uygundurlar.

Pratik olarak kurduğunuz devrelerde giriş çıkış ve besleme voltajları işte burada anlatıldığı gibi ayarlanmaktadır.