

10.HAFTA

Mantıksal Tasarım

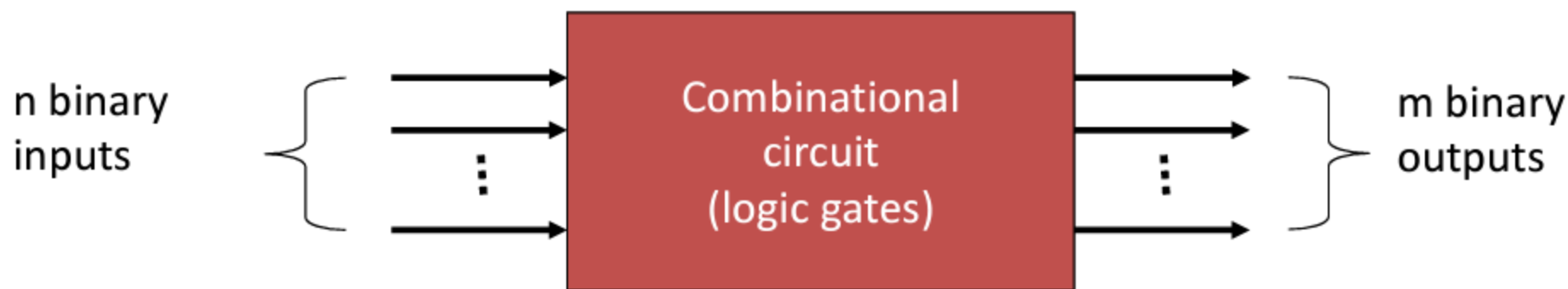
Birleşik Mantık Devreleri

- Birleşik mantık devrelerinin çıktıları sadece girdi değerlerine bağlıdır, geçmişteki girdi değerlerine bağlı değildir.
- Bundan dolayı birleşik mantık devrelerinin tasarımı ardışık mantık devrelerinin tasarımına göre daha kolaydır.

Combinational (birleşik): Devrenin çıkış sinyal(ler)i, yalnızca o anki giriş sinyal(ler)inin değerine bağlıdır. Devrede durum (state) belirten bir bellek (memory) bulunmamaktadır. Giriş sinyali değişir değişmez çıkış sinyaline etki eder.

Sequential (sıralı): Devrenin çıkış sinyal(ler)i, giriş sinyal(ler)inin ve devrede durum (state) belirten bellek (memory) değerine bağlıdır. Giriş sinyalinin değişimi her zaman çıkış sinyaline etki etmez, devrenin durumuna bağlıdır.

NOT: Derste şu ana kadar gördüğümüz bütün fonksiyonlar ve devreler combinational devreydi.

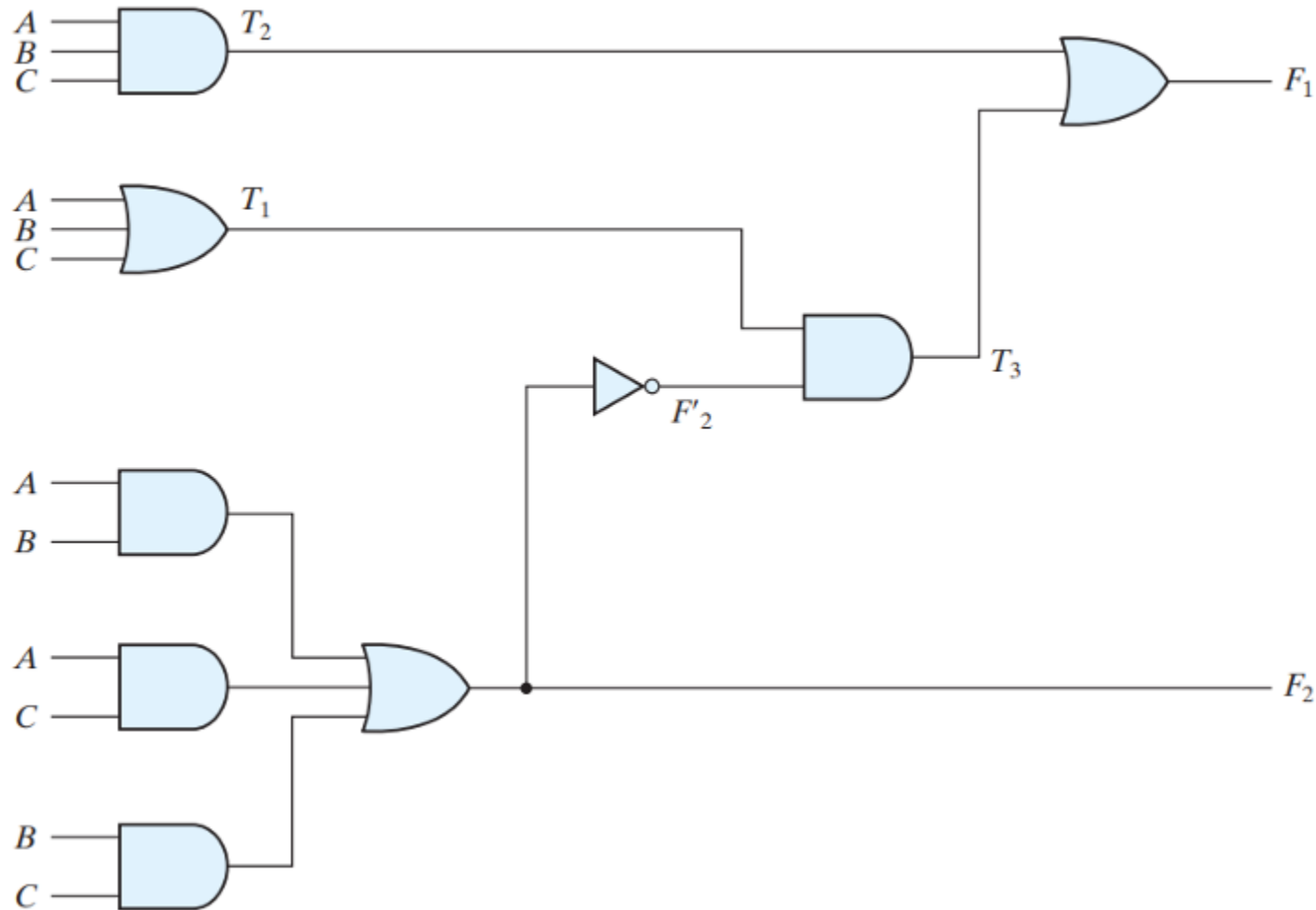


- **n** input bits $\rightarrow 2^n$ possible binary input combinations
- For each possible input combination, there is one possible output value
 - truth table
 - Boolean functions (with n input variables)
- Examples: adders, subtractors, comparators, decoders, encoders, multiplexers.

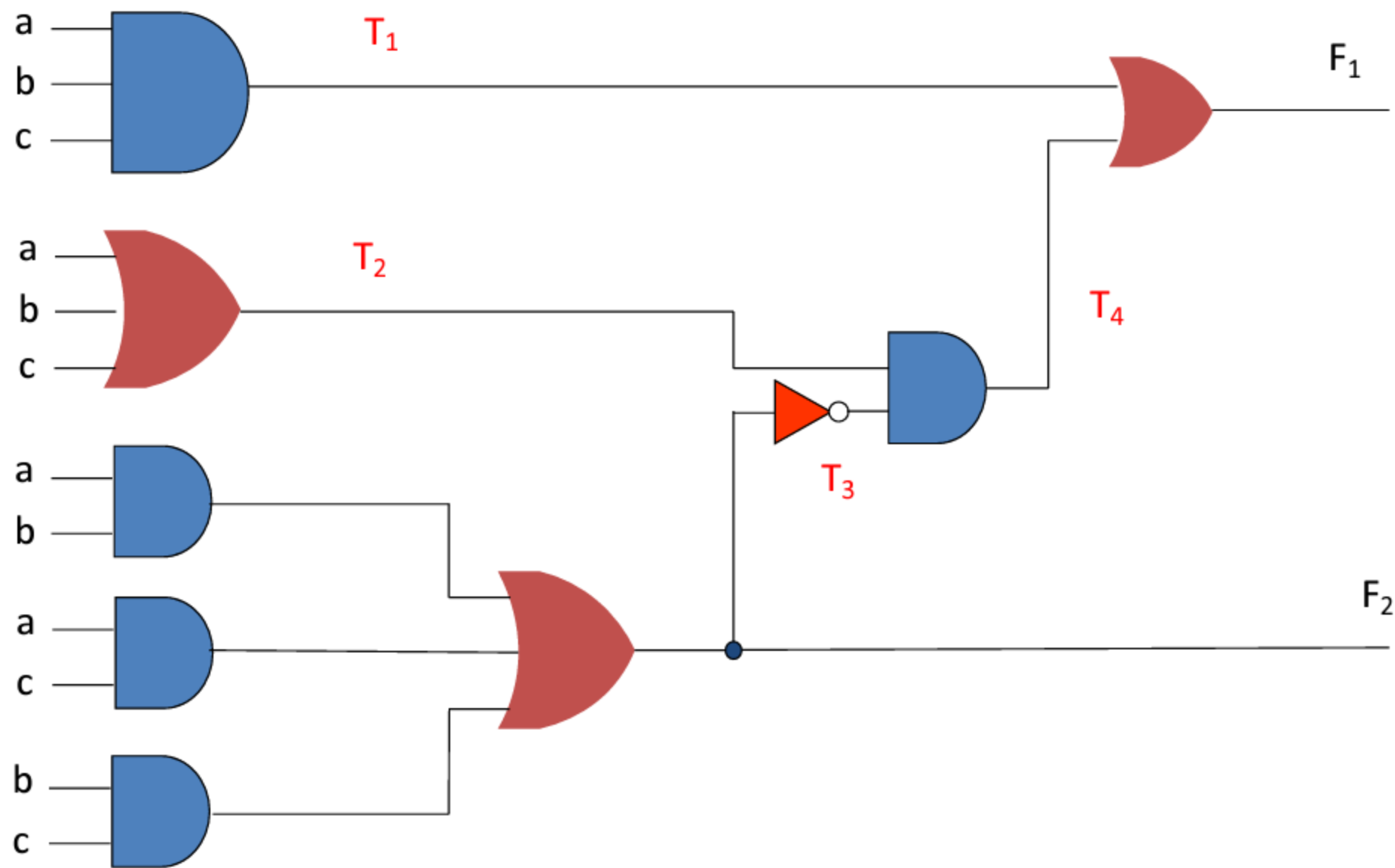
Mantık Devrelerinin Tasarım Aşamaları

1. Devresi hazırlanacak problem net bir şekilde herhangi bir bulanıklık kalmayacak tarzda net bir şekilde ifade edilir.
2. Tasarlanacak devrenin kaç adet girdisi ve çıktısı olduğu belirlenmelidir.
3. Devrenin çıktıları ile girdileri arasındaki ilişkiyi belirten doğruluk tablosu oluşturulur.
4. Oluşturulan doğruluk tablosu kullanılarak çıktılar ve girdiler arasındaki bağlantılar Boolean fonksiyonları ile ifade edilir. Bunun için Karnaugh yerleşkeler kullanılır ve elde edilen ifadelerin minimum sayıda terim içermesine dikkat edilir.
5. Bir önceki adımda elde edilen Boolean fonksiyonlarının gerçekleştirimleri mantık kapıları ile yapılır. Gerçekleştirim esnasında en uygun ve en ucuz kapıların kullanılmasına dikkat edilmelidir.

Birleşik Devre Analizi



F1 & F2				
Giriş			Çıkış	
A	B	C	F1	F2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



- $T_1 = abc$

- $T_2 = a + b + c$

- $F_2 =$

- $T_3 =$

- $T_4 = T_3 T_2$

- $F_1 = T_1 + T_4$

$=$

$=$

$=$

$=$

- $T_1 = abc$
- $T_2 = a + b + c$
- $F_2 = ab + ac + bc$
- $T_3 = F_2' = (ab + ac + bc)'$
- $T_4 = T_3 T_2 = (ab + ac + bc)' (a + b + c)$
- $F_1 = T_1 + T_4$
 - $= abc + (ab + ac + bc)' (a + b + c)$
 - $= abc + ((a' + b')(a' + c')(b' + c')) (a + b + c)$
 - $= abc + ((a' + a'c' + a'b' + b'c')(b' + c')) (a + b + c)$
 - $= abc + (a'b' + a'c' + a'b'c' + b'c') (a + b + c)$
 - $= abc + (a'b' + a'c' + b'c') (a + b + c)$

- $F_2 = ab + ac + bc$

- $F_1 =$

- $F_1 =$

- $F_1 =$

- $F_1 =$

- $F_2 = ab + ac + bc$
- $F_1 = abc + (a'b' + a'c' + b'c')(a + b + c)$
- $F_1 = abc + a'b'c + a'bc' + ab'c'$
- $F_1 = a(bc + b'c') + a'(b'c + bc')$
- $F_1 = a(b \oplus c)' + a'(b \oplus c)$
- $F_1 = (a \oplus b \oplus c) : \text{Odd Function}$

$$F_1 = a \oplus b \oplus c$$

$$F_2 = ab + ac + bc$$

carry

sum

a	b	c	T_1	T_2	T_3	T_4	F_2	F_1
0	0	0	0	0	1	0	0	0
0	0	1	0	1	1	1	0	1
0	1	0	0	1	1	1	0	1
0	1	1	0	1	0	0	1	0
1	0	0	0	1	1	1	0	1
1	0	1	0	1	0	0	1	0
1	1	0	0	1	0	0	1	0
1	1	1	1	1	0	0	1	1

This is what we call **full-adder (FA)**

Toplayıcılar

3.2.1 Toplayıcılar (Adders)

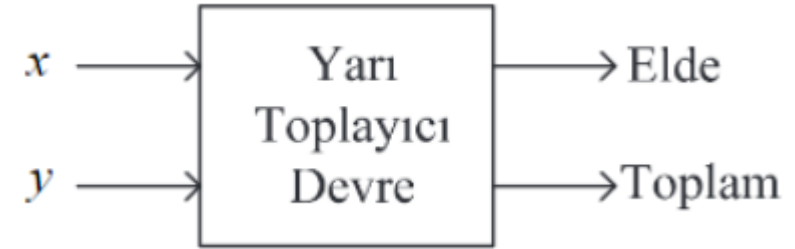
Toplayıcılar ikili sistemde verilen n -bitlik iki sayıyı toplamak amacı ile tasarlanırlar. n en az '1' olabilir.

Yarı Toplayıcı (Half Adder)

Bir bitlik iki adet sayıyı toplamak amacı ile yapılır. Yarı toplayıcı devresinin iki adet girdisi vardır. Bir bitlik iki sayının toplamının en büyük değeri iki bitlik en küçük sayı olacağından yarı toplayıcı devresinin iki adet çıktısı olacaktır. Sayılardan birini ' x ' diğerini ' y ' ile gösterirsek yarı toplayıcı devresinin yapacağı işlem

$$\begin{array}{r} x \\ + y \\ \hline \text{Elde Toplam} \end{array}$$

x	y	Elde	Toplam
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Elde

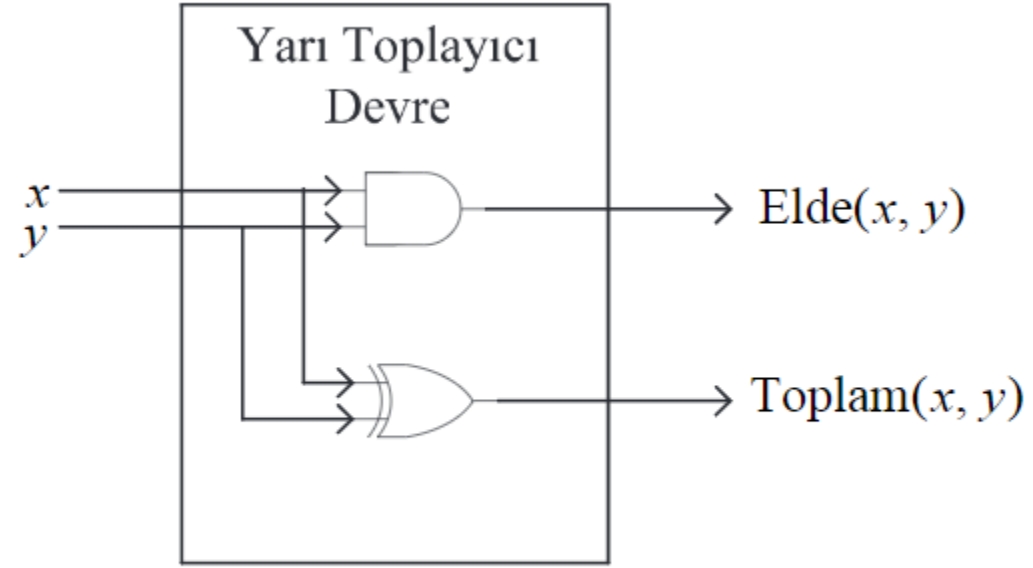
		y		y'	y
				0	1
x	0	0	0	0	
	1	0	1		xy

Toplam

		y		y'	y
				0	1
x	0	0	1		
	1	1	0		

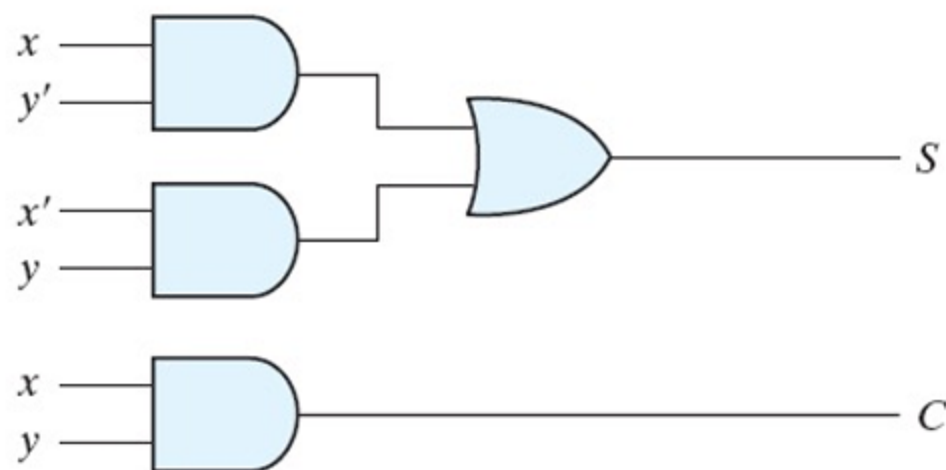
$x'y + xy'$

Şekil 3.3 Yarı Toplayıcı Fonksiyonunun Karnaugh Yerleşkesi



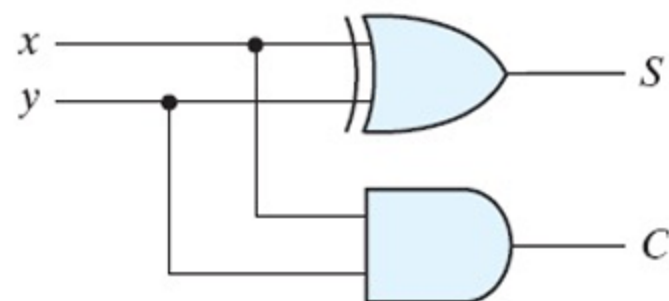
Şekil 3.4 Yarı Toplayıcı Mantık Devresi

- the flexibility for implementation
- $S = x \oplus y$
- $S = (x+y)(x'+y')$
- $S' = xy + x'y'$
- $S = (C + x'y')'$
- $C = xy = (x' + y')'$



(a) $S = xy' + x'y$
 $C = xy$

- $S = x'y + xy'$
- $C = xy$



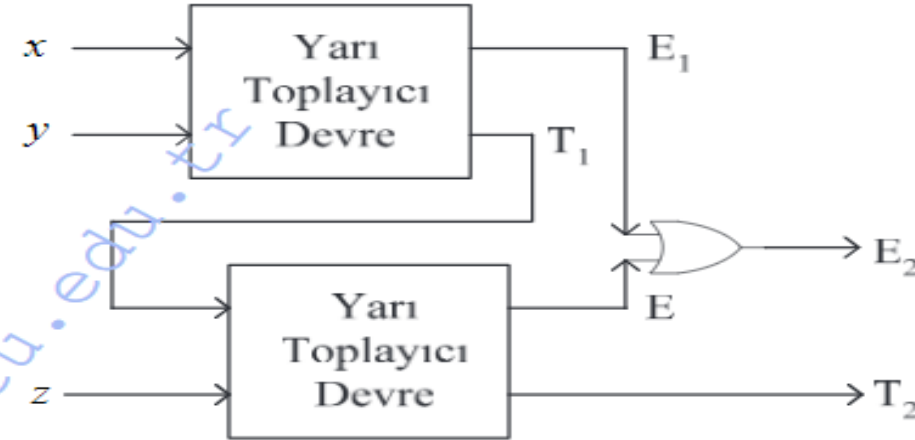
(b) $S = x \oplus y$
 $C = xy$

Tam Toplayıcı (Full Adder)

Tam toplayıcı bir bitlik 3 adet sayının toplama işlemini yapan devredir. Bir bitlik sayılar 'x', 'y' ve 'z' olsun. Bu üç sayının toplamı iki aşama halinde şöyle yapılır, ilk aşamada 'x' ve 'y' toplanır daha sonra bu toplamın sonucuna 'z' eklenir. Bu işlem şöyledir,

$$\begin{array}{r} x \\ + \quad y \\ \hline E_1 T_1 \end{array} \quad \begin{array}{r} E_1 T_1 \\ + \quad z \\ \hline E_2 T_2 \end{array}$$

buradaki 'E' ve 'T' harfleri elde ve toplam sözcüklerinin kısaltmalarıdır. İki adet yarı toplayıcı devreyi kullanarak bir adet tam toplayıcı devreyi oluşturabiliriz. Aşağıdaki şekilde bu işlem anlatılmıştır.



Şekil 3.5 Tam Toplayıcı Devresinin İki Adet Yarı Toplayıcı ve Bir Adet OR Kapısı ile Gerçekleştirimi

Tam toplayıcı devresinin gerçekleştirimi için 3 adet yarı toplayıcı devresine ihtiyaç var gibi görünse de üçüncü yarı toplayıcı devresinin yapacağı işlemi tek bir OR kapısı ile de yapabiliriz, çünkü

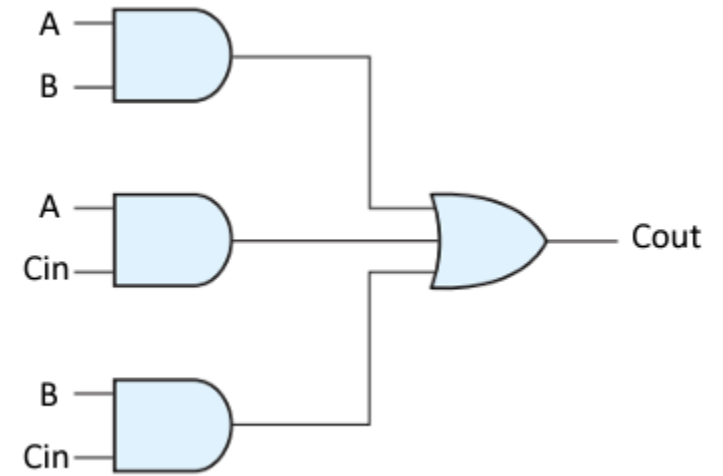
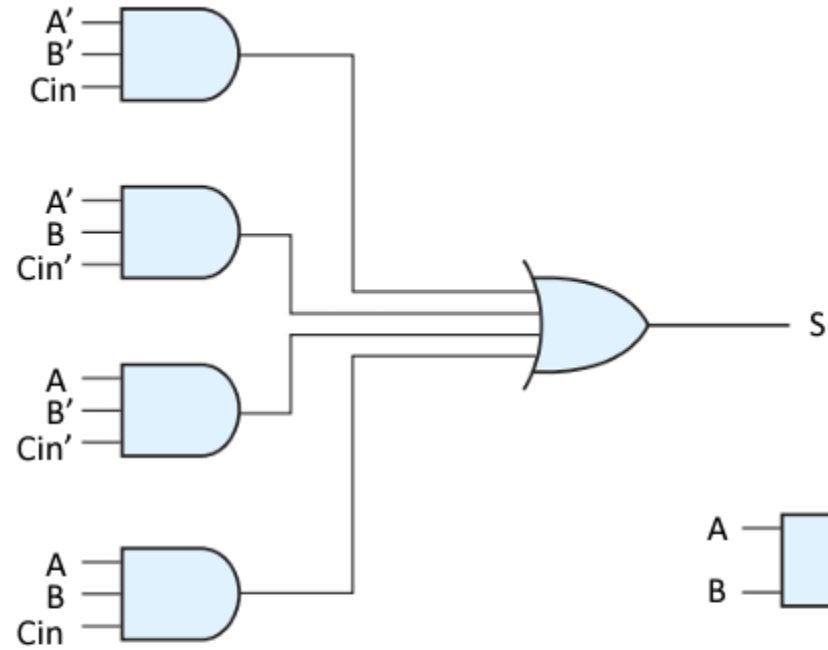
$$\begin{array}{r} E_1 T_1 \\ + \quad z \\ \hline E_2 T_2 \end{array}$$

işleminde $E_1 T_1$ ifadesinin alacağı değerler 01 ya da 10 olmaktadır. Çünkü iki adet bir bitlik sayının toplamı ikiden büyük olamaz. $E_1 T_1 = 01$ değerlerine 'z' nin alabileceği 0 ve 1 değerleri eklenince, sonuçlar 01 ya da 10 olmaktadır. Benzer şekilde $E_1 T_1 = 10$ değerlerine 'z' nin alabileceği 0 ve 1 değerleri eklenince, sonuçlar 10 ya da 11 olmaktadır. Bu sonuçlar göz önünde bulundurularak tam toplayıcı devresi Şekil 3.4'teki gibi çizilebilir.

Giriş			Çıkış	
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = A'B'Cin + A'BCin' + AB'Cin' + ABCin \rightarrow Cin'(A'B + AB') + Cin(A'B' + AB) \\ \rightarrow Cin'(A \oplus B) + Cin(A \oplus B)' \rightarrow A \oplus B \oplus Cin$$

$$Cout = AB + ACin + BCin$$



■ **Specification:**

- A combinational circuit that forms the arithmetic sum of three bits and generates a sum and a carry

■ **Inputs:**

- Three inputs: x, y, z
- Two outputs: S, C

■ **Truth table:**

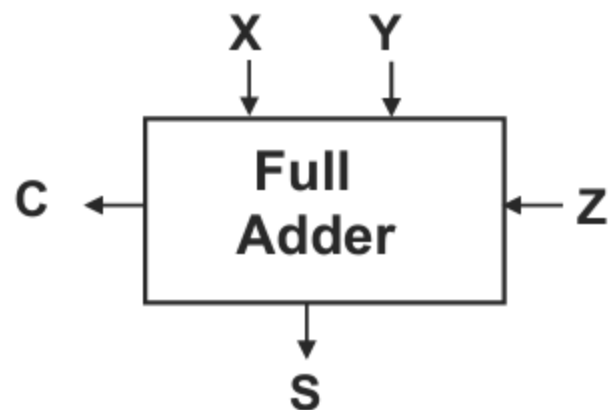
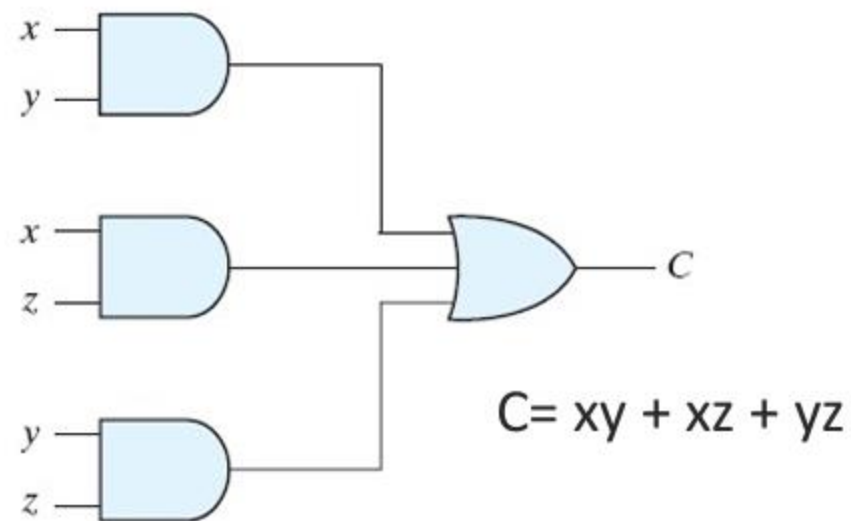
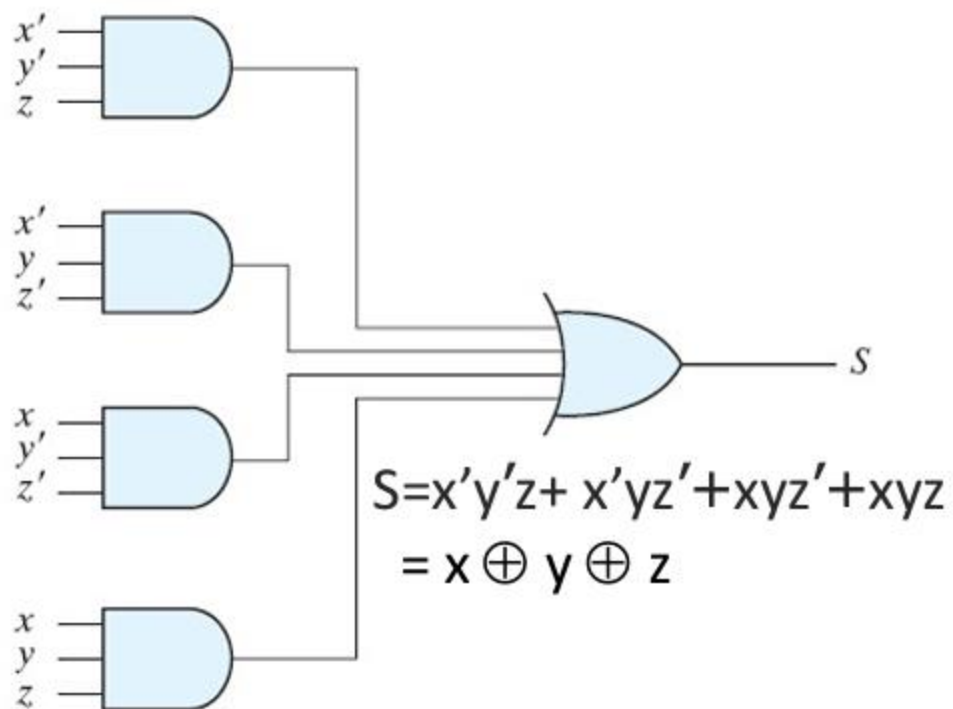


Table 4.4
Full Adder

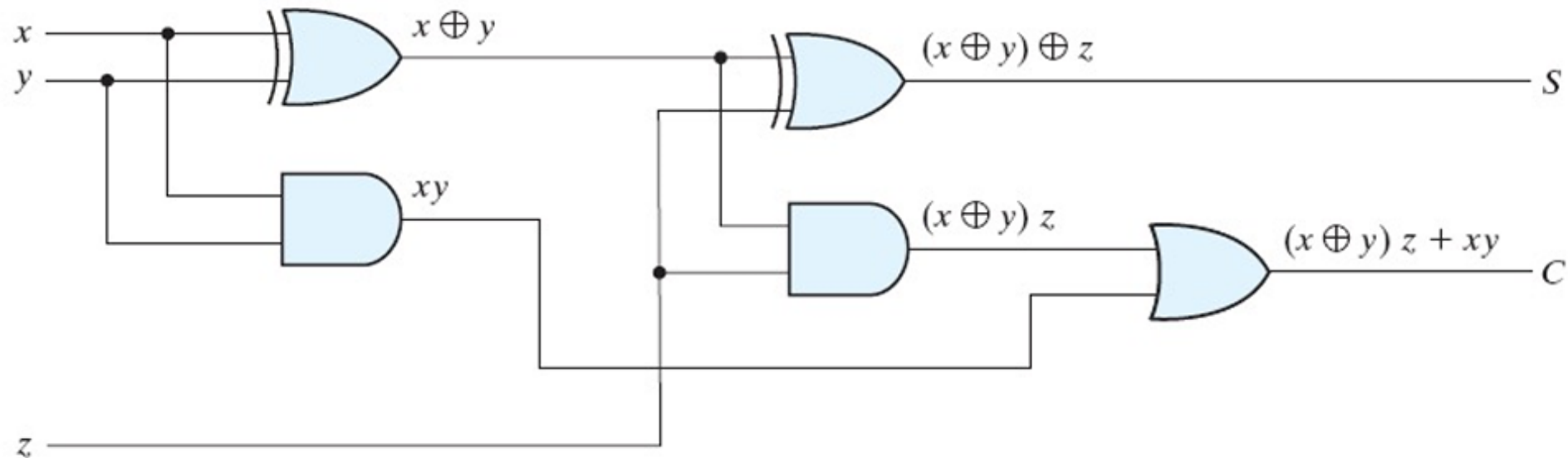
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$x \backslash yz$		y			
		00	01	11	10
x	0	m_0	m_1 1	m_3	m_2 1
	1	m_4 1	m_5	m_7 1	m_6

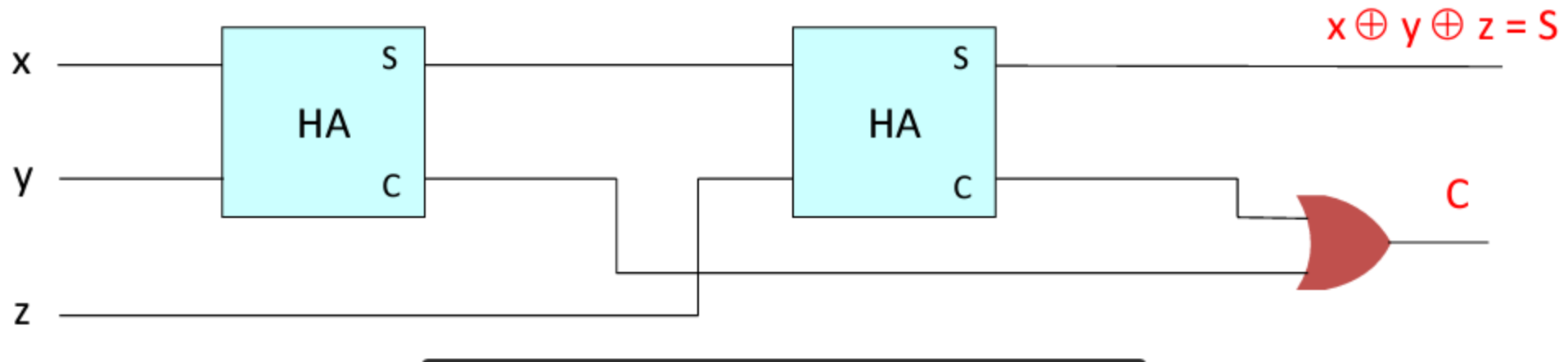
$x \backslash yz$		y			
		00	01	11	10
x	0	m_0	m_1	m_3 1	m_2
	1	m_4	m_5 1	m_7 1	m_6 1



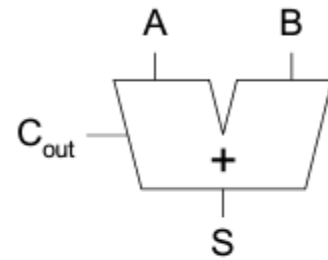
- Sum
 - $S = x \oplus y \oplus z = (x \oplus y) \oplus z$
- Carry
 - $C = xyz' + xyz + xy'z + x'yz$
 $= xy + (xy' + x'y)z$
 $= xy + (x \oplus y)z$
- This allows us to implement a full-adder using 2 half adders and 1 OR gate.



- Sum
 - $S = x \oplus y \oplus z = (x \oplus y) \oplus z$
- Carry
 - $C = xyz' + xy'z + x'y'z + x'yz$
 $= xy + (xy' + x'y)z$
 $= xy + (x \oplus y)z$
- This allows us to implement a full-adder using 2 half adders and 1 OR gate.



Half Adder

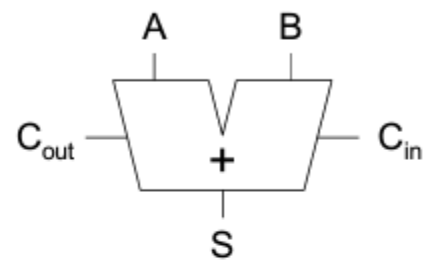


A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = A \oplus B$$

$$C_{out} = AB$$

Full Adder



C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

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

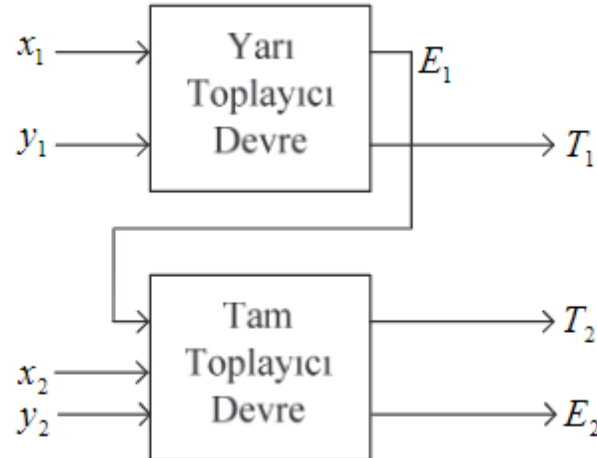
$$C_{out} = AB + AC_{in} + BC_{in}$$

Örnek 3–1: İki bitlik iki adet sayı verilmiş olsun. Bu sayıların toplamının sonucunu veren mantık devresi yarı ve tam toplayıcılar kullanarak oluşturunuz.

Çözüm 3–1: İki bitlik sayılar x_2x_1 ve y_2y_1 olsun. Bu iki sayının toplamı iki aşamalı bir işlemdir. İlk aşamada x_1 ve y_1 sayıları toplanır. İkinci aşamada ise ilk aşamadan gelen elde ile x_2 ve y_2 sayıları toplanır. Bu işlemler aşağıda açıklanmıştır,

$$\begin{array}{r} x_1 \\ + \quad y_1 \\ \hline E_1T_1 \\ x_2 \\ + \quad y_2 \\ \hline E_2T_2T_1 \end{array}$$

burada $E_2T_2T_1$ iki bitlik sonuç sayısı olmaktadır. İlk toplama işlemini yarı toplayıcı ile ikinci toplama işlemini ise tam toplayıcı ile gerçekleştirmek mümkündür. Bu iki işlemi yapan devre aşağıda verilmiştir.

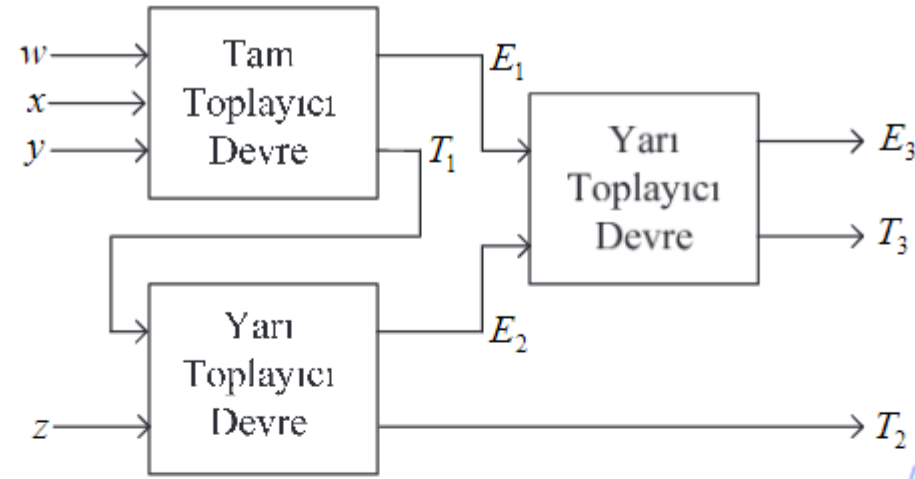


Örnek 3–2: Bir bitlik 4 adet sayının toplamını veren mantık devresini tasarlayınız.

Çözüm 3–2: Bir bitlik sayılar ' w ', ' x ', ' y ' ve ' z ' olsun. Bu dört sayının toplamı iki aşamada yapılabilir. İlk aşamada ' w ', ' x ' ve ' y ' sayıları toplanır. İkinci aşamada ise ' w ', ' x ' ve ' y ' toplam sonucu ' z ' ile toplanır. Matematiksel olarak ifade edecek olursak

$$\begin{array}{r} w \\ x \\ + \quad y \\ \hline E_1T_1 \\ + \quad z \\ \hline E_3T_3T_2 \end{array}$$

burada $E_3T_3T_2$ üç bitlik sonucu göstermektedir. Yukarıdaki üç bitlik toplamaları tam toplayıcı ile iki bitlik toplam işlemlerini ise yarı toplayıcı ile yapmak mümkündür. O halde yukarıdaki matematiksel ifadeyi göz önüne alarak bir bitlik adet sayının toplamını veren devre aşağıdaki şekilde tasarlanabilir.



Şekil 3.7 4-bitlik Toplayıcı Mantık Devresi

Yukarıdaki şekilde elde edilen mantık devresi kapalı kutu gösterimi ile aşağıdaki gibi ifade edilebilir.



$$w + x + y + z = T_c T_b T_a$$

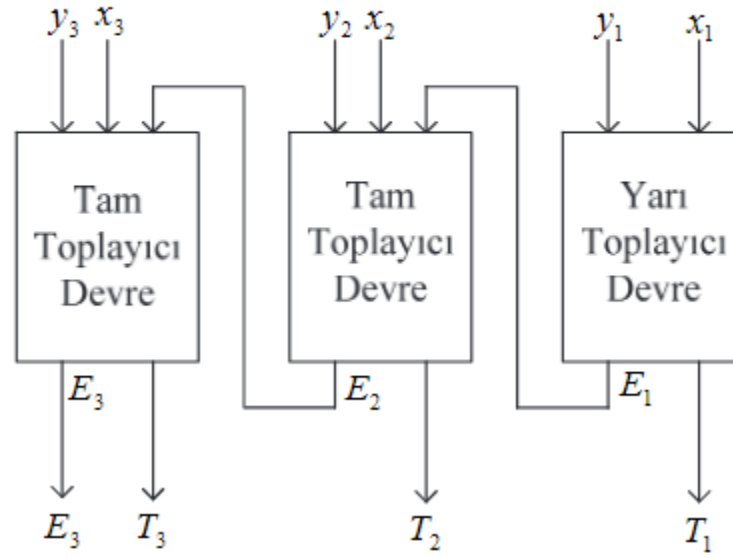
Şekil 3.8 4-bitlik Toplayıcı Devresinin Kapalı Kutu Gösterimi

Örnek 3–4: Üç bitlik 2 adet sayının toplamını veren mantık devresini tasarlayınız.

Çözüm 3–4: Sayılar $x_3x_2x_1$ ve $y_3y_2y_1$ olsun. Bu iki sayı toplama işlemi aşağıda verilmiştir,

$$\begin{array}{r} x_3x_2x_1 \\ + y_3y_2y_1 \\ \hline E_3T_3T_2T_1 \end{array}$$

buradaki bit-bit toplama işlemi incelenirse bu işlemi yapabilecek olan mantık devresini bir yarı toplayıcı ve iki tam toplayıcı kullanarak tasarlamak mümkündür. İlk olarak x_1 ve y_1 sayılarının toplamı yarı toplayıcı ile yapılabilir. Bu toplamdan gelen elde ile x_2 ve y_2 sayılarının toplamı tam toplayıcı ile gerçekleştirilebilir. Bu toplamdan gelen elde ile x_3 ve y_3 sayılarının toplamı da bir tam toplayıcı ile yapılabilir. Bu hususları göz önünde bulundurarak üç bitlik iki sayının toplamını veren devreyi aşağıdaki şekildeki gibi tasarlayabiliriz.

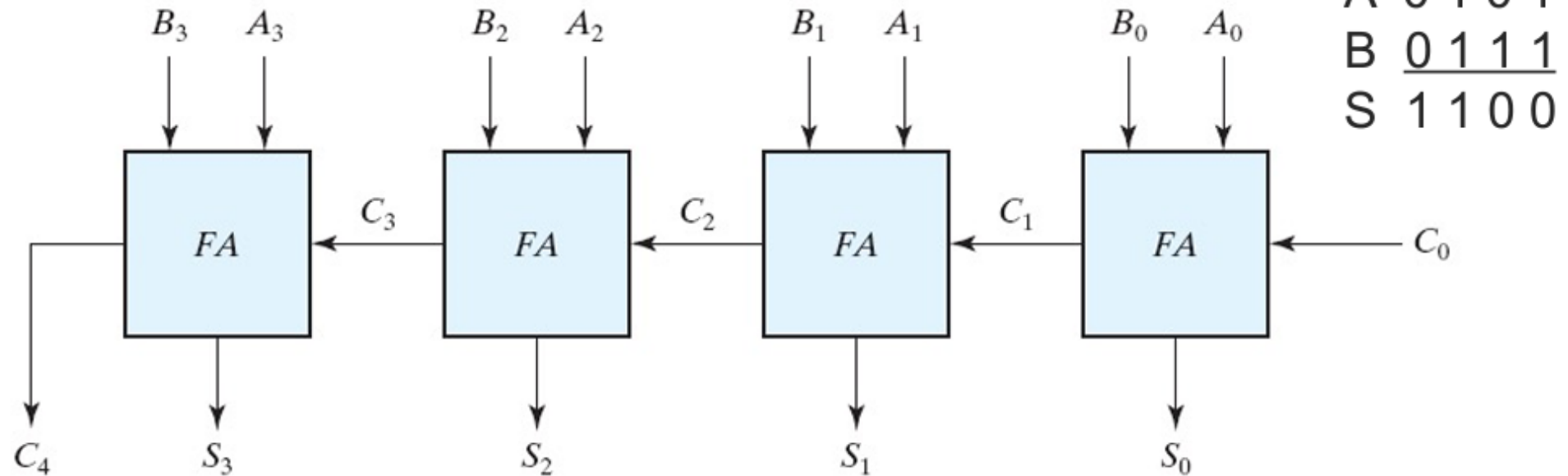


Şekil 3.10 5-bitlik Toplayıcı Mantık Devresi

Buraya kadar çözdüğümüz örneklerden görüleceği üzere, ilk olarak basit mantık devrelerinin tasarımını yapıyoruz. Daha sonra ise basit mantık devrelerinin değişik birleşimlerini kullanarak daha karmaşık mantık devrelerini oluşturuyoruz. Sayısal tasarım bir tür sanat olayı ya da mimari çalışma gibi düşünülebilir. Belli temel taşları kullanarak büyük sistemleri belli mimari esaslara ve mantığa göre oluşturuyoruz.

Example: 4-bit binary adder

■ 4-bit Ripple Carry Adder



■ Classical example of standard components

- Would require truth table with 2^9 entries!

Çıkarıcılar

3.2.2 Çıkarıcılar (Subtractors)

Çıkarıcılar iki sayıyı birbirinden çıkarma amacı ile tasarlanırlar. İki adet temel çıkarıcı mantık devresi mevcuttur, bunlar ‘yarı çıkarıcı’ ve ‘tam çıkarıcı’ olarak isimlendirilmişlerdir.

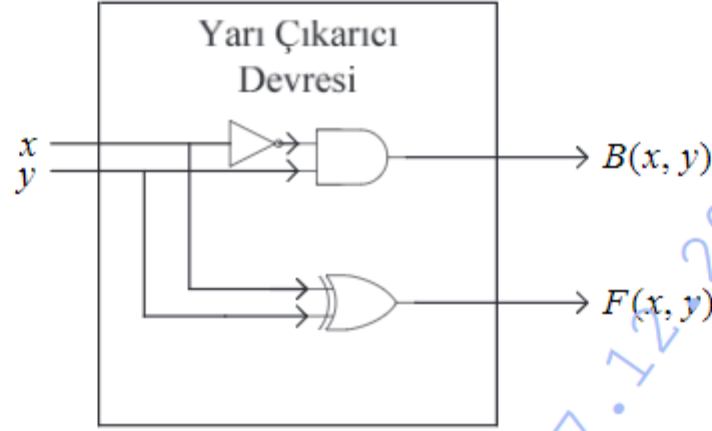
Yarı Çıkarıcı (Half Subtractor)

Yarı çıkarıcı devresi $x - y$ işlemini yapar. Bu devreyi tasarlamak için ilk olarak $x - y$ işleminin doğruluk tablosunu oluşturalım, sonra bu doğruluk tablosunu kullanarak yarı çıkarıcı devresinin tasarımın Karnaugh yerleşkeler kullanarak yapalım. $x - y$ işleminin doğruluk tablosu aşağıda verilmiştir.

Tablo 3.3 $x - y$ İşleminin Doğruluk Tablosu

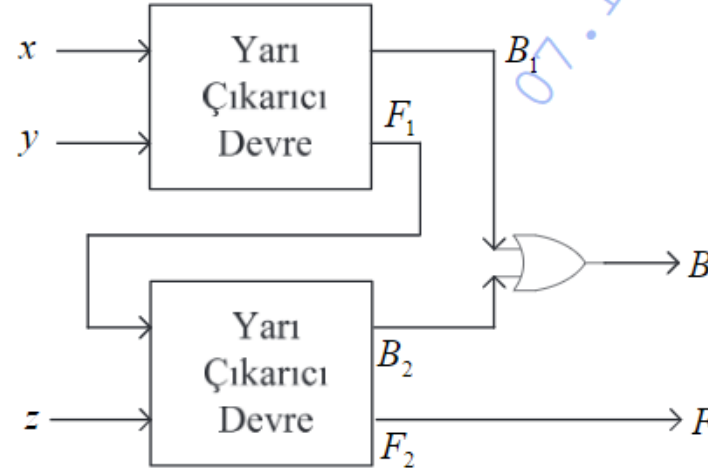
x	y	Borç	Fark
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Yukarıdaki tabloda verilen ‘Borç’ fonksiyonu $x = 0, y = 1$ durumunda 1 diğer birleşimler için ise 0 değerini almaktadır. ‘Fark’ fonksiyonu ise çıkarma işleminin sonucunu göstermektedir. Yukarıdaki tablodaki ‘Borç’ ve ‘Fark’ fonksiyonları $B(x, y)$ ve $F(x, y)$ olarak gösterilirse tablodan da yararlanarak bu fonksiyonları $B(x, y) = x' y$ $F(x, y) = x' y + y' x$ şeklinde yazabiliriz ve gerçekleştirimini aşağıdaki devrede olduğu gibi yapabiliriz.



Şekil 3.11 Yarı Çıkarıcı Mantık Devresi

‘Borç’ ve ‘Fark’ fonksiyonları Boolean değişkenleri x, y, z türünden ifade edilebilir. Elde edilen fonksiyon ifadelerin gerçekleştirimleri mantık kapıları kullanılarak yapılabilir. Biz ise tam çıkarıcı devrenin gerçekleştirimini daha önce tasarlamış olduğumuz yarı çıkarıcı devreyi kullanarak yapacağız. Bunun için takip edeceğimiz yöntem şöyledir. $x - y - z$ çıkarma işlemi iki aşamada yapılabilir. İlk aşamada $x - y$ çıkarma işlemi yapılır, bu işlem sonucunda oluşan fark ve borç alma sonuçlarını F_1 ve B_1 ile gösterelim, yani çıkarma işleminin sonucu $F_1 = x - y$ olmaktadır. İkinci çıkarma işlemi ise $F_1 - z$ ile gerçekleştirilmektedir. Bu iki çıkarma işlemi esnasından bir önceki sayıdan elde alma (Borç alma) ya ilk çıkarma işleminde ya da ikinci çıkarma işleminde gerçekleşebilir. Bu iki durumu bir OR kapı vasıtasıyla birleştirmek mümkündür. Yukarıdaki ifadeleri öz önüne alarak tam çıkarıcı devresini yarı çıkarıcı devreler ve OR kapısı kullanarak aşağıdaki gibi gerçekleştirebiliriz.



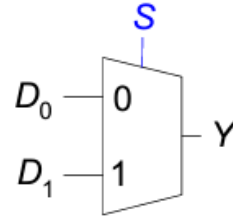
Şekil 3.12 Tam Çıkarıcı Mantık Devresini

Yukarıdaki devrenin çıktıları B ve F olmaktadır ve $B(x, y)$ ve $F(x, y)$ fonksiyonlarını işaret etmektedir. Çıkarma sonucundaki borç alma durumunu ve çıkarma sonucunu göstermektedirler.

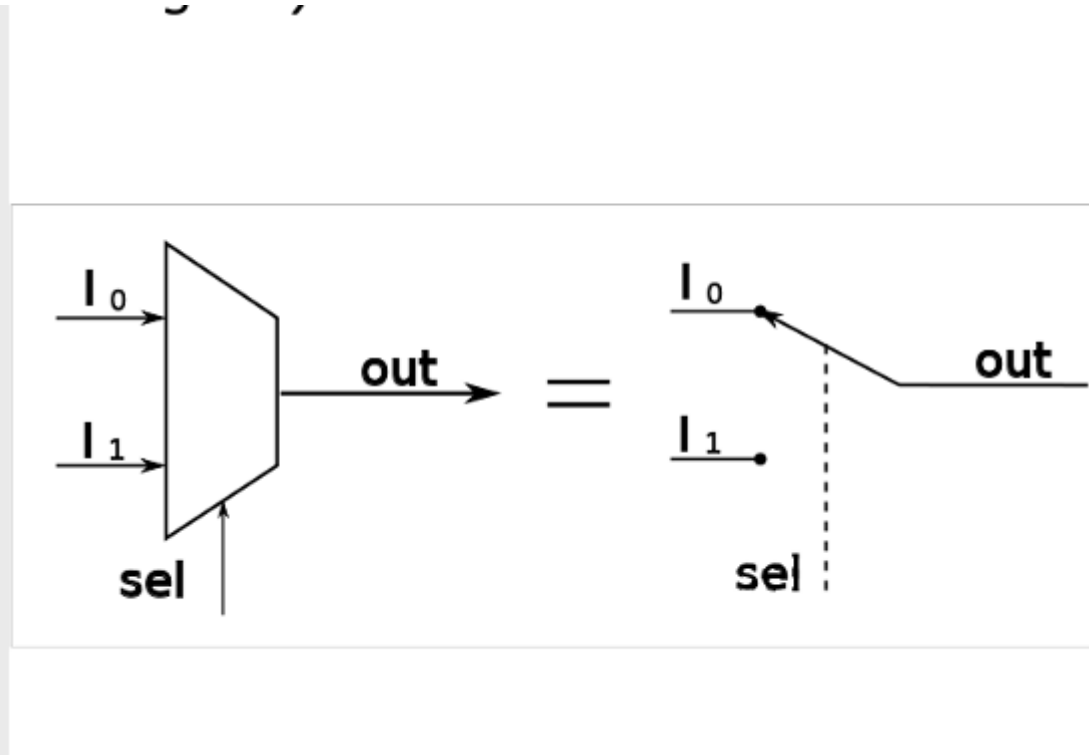
Multiplexer

- Selects between one of N inputs and directs it to a single output.
- $\log_2 N$ -bit select input – control input
- **Example:**

2:1 Mux



S	D_1	D_0	Y	S	Y
0	0	0	0	0	D_0
0	0	1	1	1	D_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		



sel	Giriş		Çıkış
	I_1	I_0	out
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

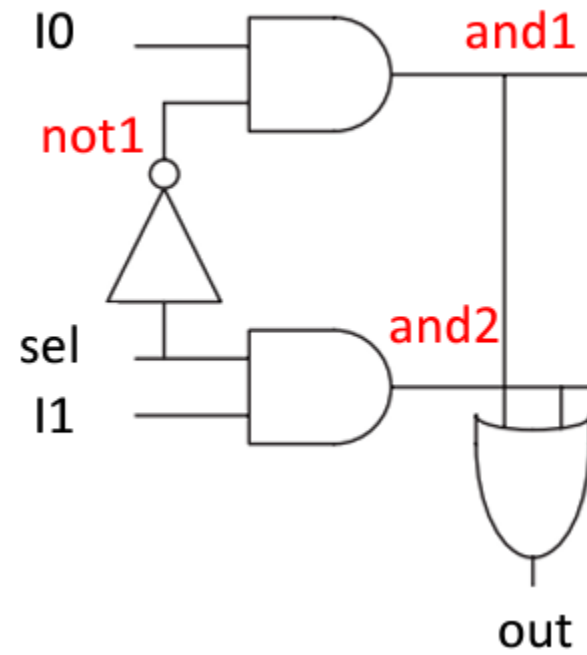
The Karnaugh map for the output out is shown. The columns are labeled with $I_1 I_0$ values: 00, 01, 11, 10. The rows are labeled with sel values: 0 and 1. The cells containing 1 are circled: a red circle around the 1s in the $sel=0$ row (for $I_1 I_0 = 01$ and 11), and a blue circle around the 1s in the $sel=1$ row (for $I_1 I_0 = 10$ and 11).

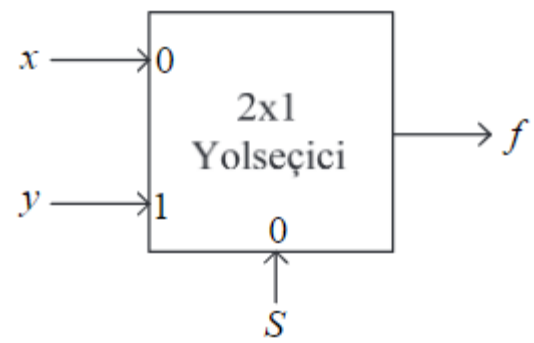
	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$$out = sel' I_0 + I_1 sel$$

Multiplexer 2x1

$$\text{out} = \text{sel}'I0 + I1\text{sel}$$





Şekil 3.30 2x1'lik Yolseçici

S	f				
0	x				
1	y				

→

x	y	S	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

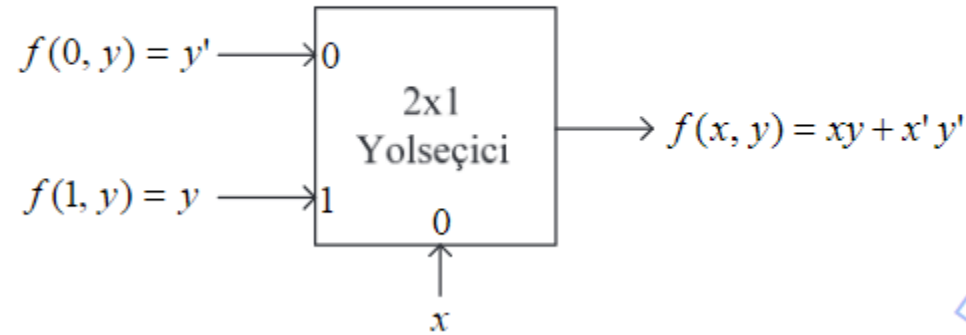
Örnek 3-14: $f(x, y) = xy + x'y'$ fonksiyonunun gerçekleştirimini bir adet 2×1 'lik yolseçici kullanarak yapınız.

Çözüm 3-14: Seçme teline bağlanacak değişken x olsun. Bu değişkene değerler vererek fonksiyonu

$$x = 0 \rightarrow f(0, y) = y'$$

$$x = 1 \rightarrow f(1, y) = y$$

şeklinde parçalara ayıralım. Son olarak fonksiyon parçalarını yolseçicinin girdilerine seçme telinin aldığı değerlere göre aşağıdaki gibi bağlayalım.

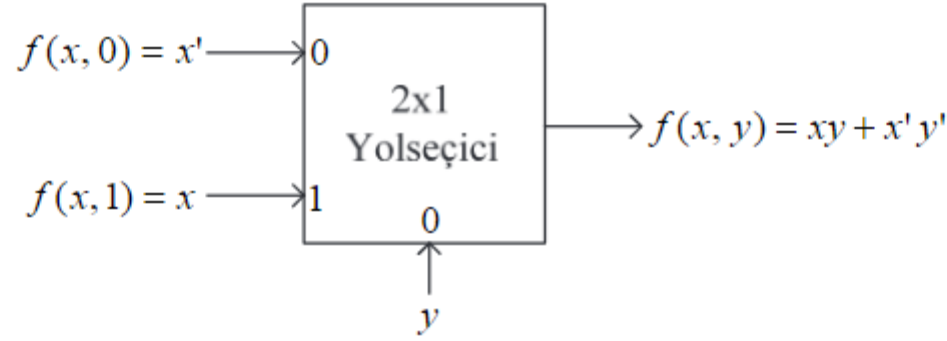


Seçme teline bağlanacak değişken x değil de y olarak seçilirse fonksiyonun parçalanması

$$y = 0 \rightarrow f(x, 0) = x'$$

$$y = 1 \rightarrow f(x, 1) = x$$

şeklinde olur. Fonksiyonun gerçekleştirimi de aşağıdaki gibi daha değişik bir şekilde yapılabilir.

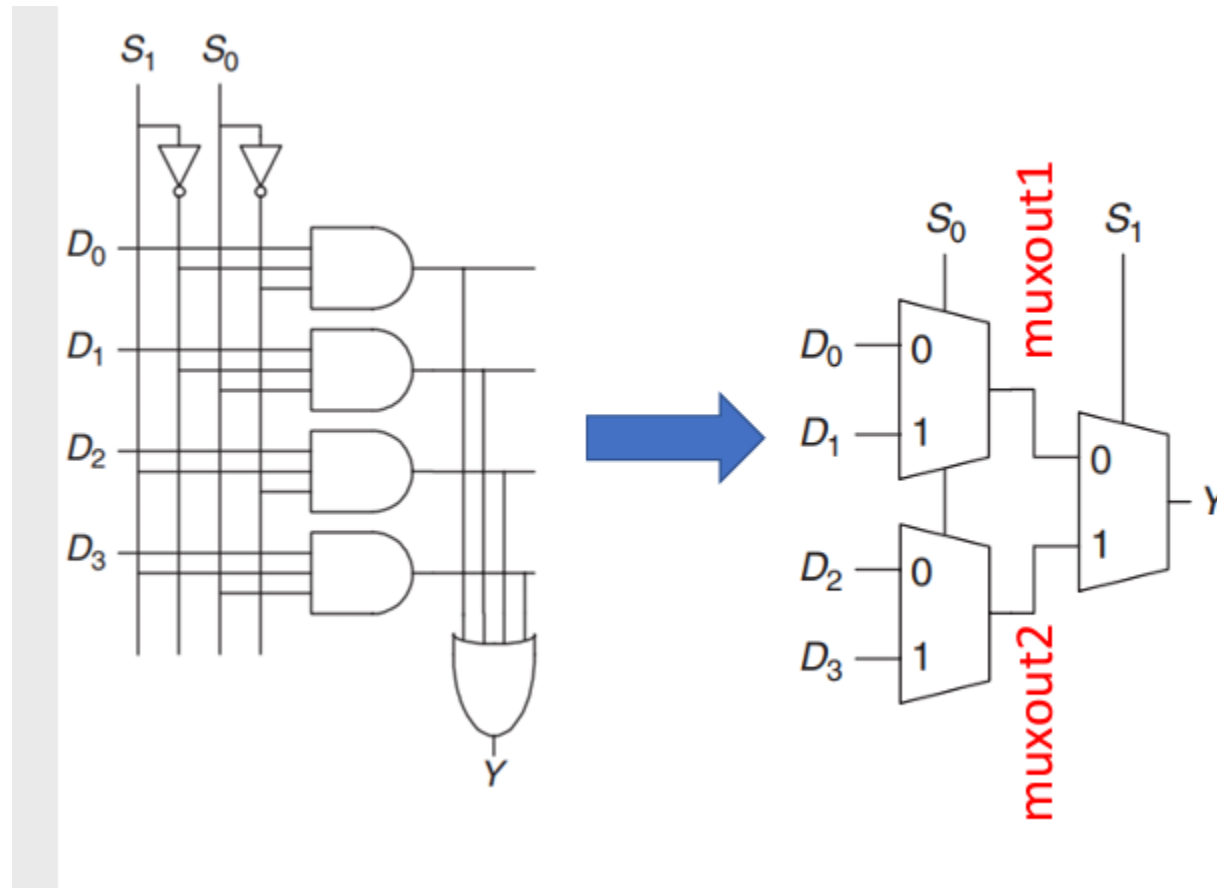


Şekil 3.33 $f(x, y) = xy + x'y'$ Fonksiyonunun 2x1'lik Yolseçici ile Gerçekleştirimi

Diğer bir soru sınıfında da yolseçicili gerçekleştirim devresinin hangi Boolean fonksiyonunu için kurulduğu istenir. Bu tür sorular için takip edilecek yöntem şöyledir.

1. Seçme tellerine bağlanan değişkenlerin aldığı değerler için yolseçicinin girdilerindeki Boolean ifadeleri belirlenir.
2. Seçici tellerin bağlanan değişkenlerle yolseçicilerin girdileri seçici tellerine bağlanan değişkenlerin aldığı değerlere göre çarpım ifadeleri ile birleştirilerek Boolean fonksiyonun ifadesi elde edilir.

4x1 Multiplexer



4-to-1-Line Multiplexer

- 4 input lines: I_0, I_1, I_2, I_3
- 1 output line: Y
- 2 selection lines: S_1, S_0 .

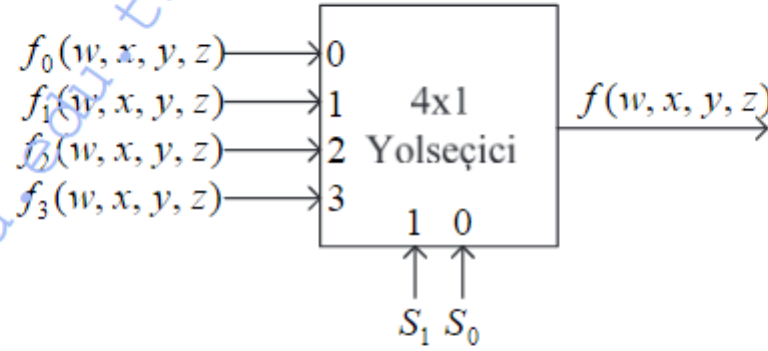
S_1	S_0	Y
0	0	
0	1	
1	0	
1	1	

$$Y = ?$$

Interpretation:

- In case $S_1 = 0$ and $S_0 = 0$, Y selects I_0
- In case $S_1 = 0$ and $S_0 = 1$, Y selects I_1
- In case $S_1 = 1$ and $S_0 = 0$, Y selects I_2
- In case $S_1 = 1$ and $S_0 = 1$, Y selects I_3

Şu ana kadarki örneklerimizde 2x1'lik yolseçici kullandık. Şimdi 4x1'lik yol seçici ünitesini görelim. 4x1'lik yol seçici ünitesinin 2 adet seçme teli ve 4 adet girdisi ve 1 adet çıkışı vardır. 4x1'lik yol seçici ünitesinin kapalı kutu gösterimi aşağıdaki şekilde gösterilmiştir.



Şekil 3.39 4x1'lik Yol Seçici Ünitesinin Kapalı Kutu Gösterimi

Yukarıdaki 4x1'lik yolseçici ünitesininin seçme tellerine bağlanan değişkenler S_1 ve S_0 'ın alacağı değerlere göre ünitenin çıkışındaki değer girdilerden birine eşit olur. Yani seçici telleri bir çeşit trafik kontrolü yaparlar. 4x1'lik yolseçici ünitesinin doğruluk tablosu aşağıda verilmiştir.

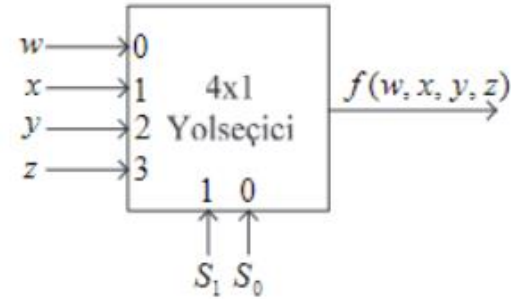
Tablo 3.7 4x1'lik Yolseçici Doğruluk Tablosu

S_1	S_0	f
0	0	f_0
0	1	f_1
1	0	f_2
1	1	f_3

↔

f_0	f_1	f_2	f_3	S_1	S_0	f
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	1
0	1	0	1	0	1	1
0	1	1	0	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	0	1	1	1	0	1
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	1
1	1	1	1	1	1	1

Yolseçici ünitesinin girdilerinde Boolean değişkenleri varsa, ünitenin kapalı kutu gösterimi aşağıdaki gibi olur.



Şekil 3.40 4x1'lik Yolseçici

Yukarıdaki şekilde kapalı kutu gösterimi verilen 4x1 'lik yolseçici ünitesinin mantık kapıları ile tasarımı aşağıda verilmiştir.

