Ve (And)

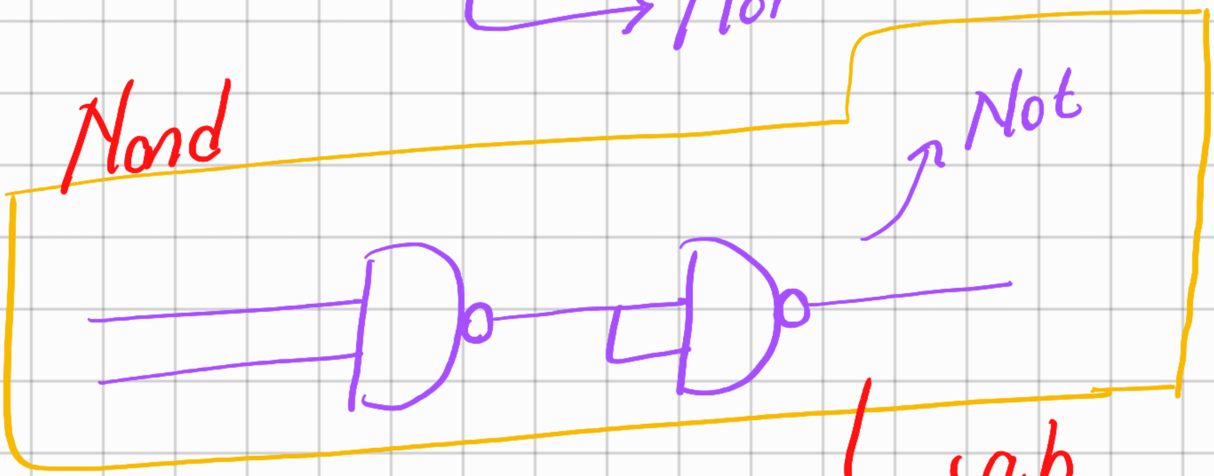
$$f = a.b = \underbrace{\overline{\overline{a.b}}}_{\text{Nandın değili gibi düşünebiliriz}} = (\overline{a.b})'$$

Bunu şöyle de yazarız

↳ $(\overline{a'+b'})$

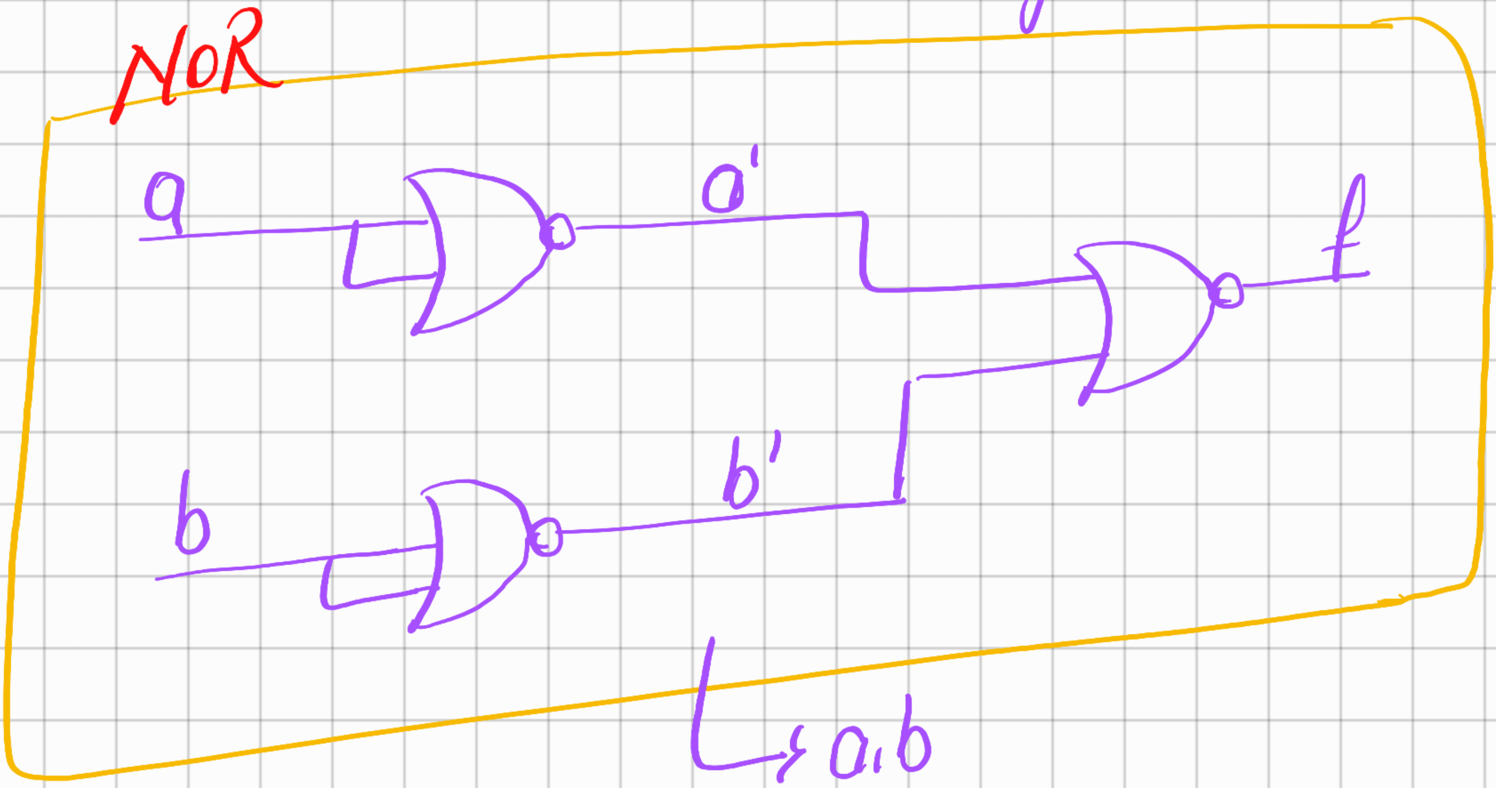
$(\overline{x+y})$

↳ Nor



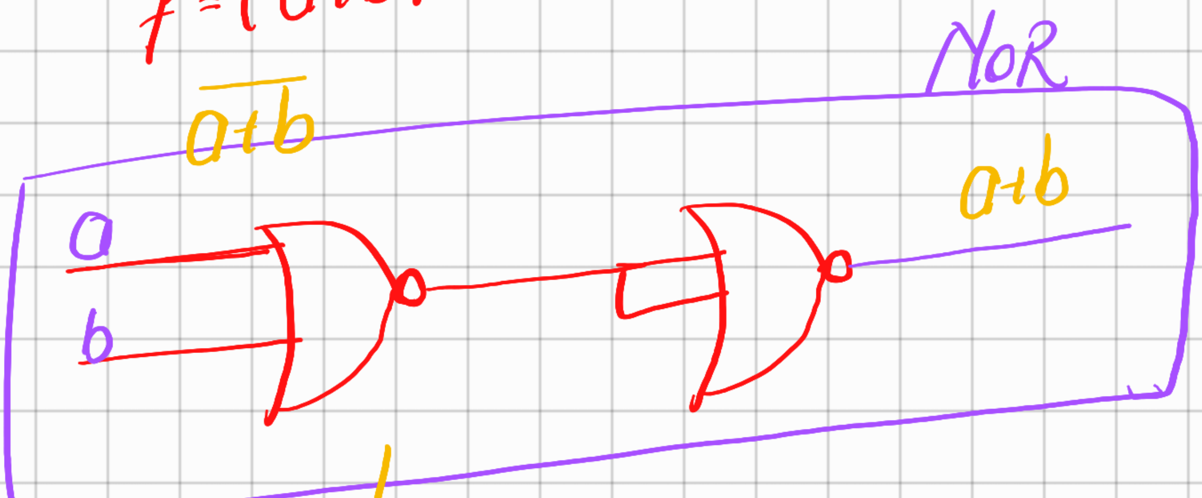
↳ elimizde 2 tane Nand kapısı varsa devreyi Nandlerle yapabiliriz

↳ Bu iki devre de aynıdır

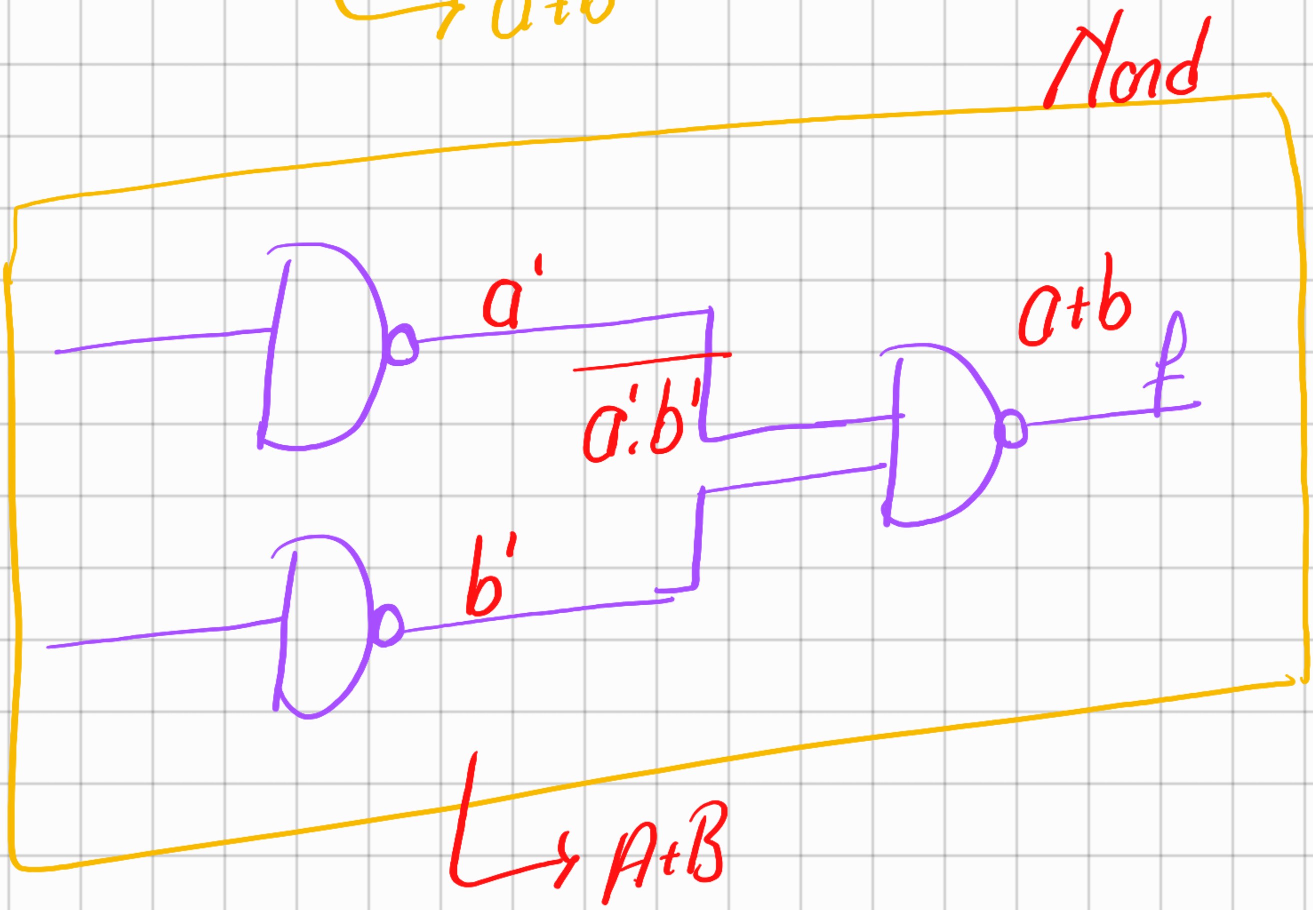


Veya (OR)

$f = (a+b) \rightarrow \overline{\overline{(a+b)}}$



↳ $a+b$



↳ Bu iki devrede birbirinin aynısıdır

↳ $= (\overline{a+b})'$

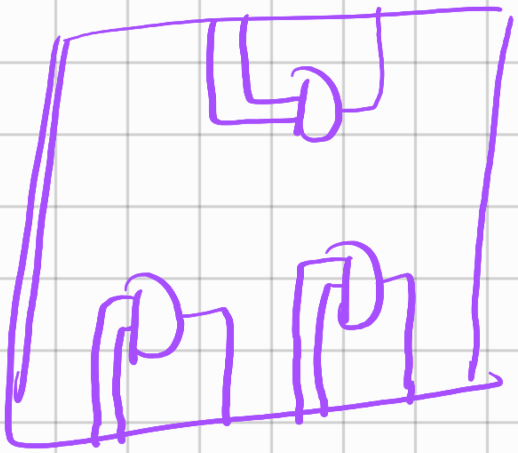
$= (a'.b')$

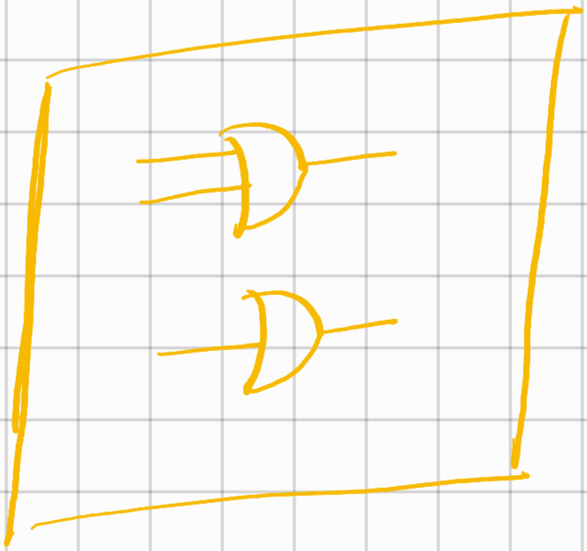
$= (\overline{x.y})$

↳ Nand

Türetilmiş kapılarla temel lojik kapıların yapılışı

↳ olabildiğince aynı kapıları kullanmak bizim için avantajlı halde





Nand

Nor

Tümleme Kapısı (Not)

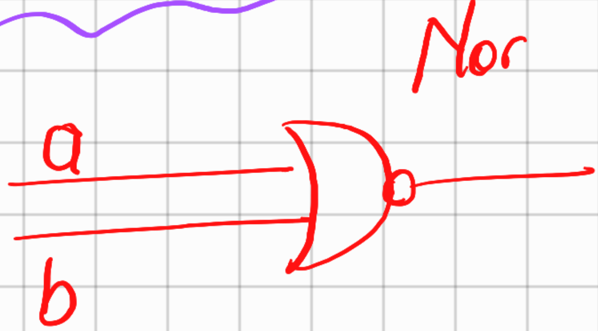
$f = (a.b)'$

$f = (a.a')$

$\hookrightarrow a'$

$a.a$

$\hookrightarrow a$

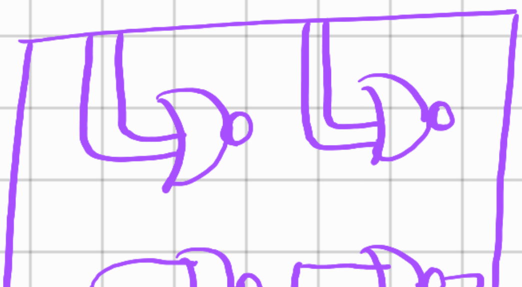


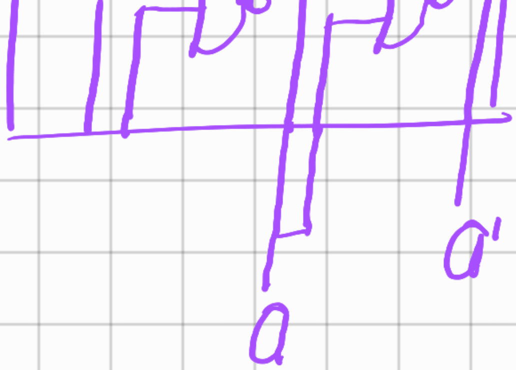
$f = (a+b)' = (\overline{a+b})$

$f = (\overline{a+a})$

$\hookrightarrow a'$

$a+a \rightarrow a$





↳ Mesela böyle bir devremiz var
ve Norlarla işlem yapıyoruz
ama bize bir tane de Not lazım
o zaman kapılardan bir tanesini
kısa devre yapıp not kapısına
çevire biliriz

İndirgenmiş ifadelerin aynı tür
kapılarla gerçeklenmesi

Çarpımların
toplamı

$f \rightarrow abc + b'c + cd$

↳ indirgenmiş ifadedeki çarpım şeklindeki ifadelerimiz

↳ Çarpımlar toplamının en genel ifadesi şöyledir

$x \rightarrow a, b, c$

x, y, z

↳ $x_1\, x_2\, x_3$

$$\boxed{f = \sum \pi x_i}$$

↳ pi dediğimiz şey bunların çarpımlarını gösterir

↳ Bu da toplamasıdır

Toplamlar Çarpımı

$$f = (a+b)(b'+c+a)(c+d')$$

↳ Bunlarda toplam şeklindeki ifadelerimiz

$$\boxed{f \to \pi \sum x_i}$$

Çarpımlar toplamı

TVE (Nand) tasarımı

↳ $\overline{f} = \overline{\sum \pi x_i}$

$\overline{(ab+cd)}$

$(\overline{ab})(\overline{cd})$

Burdaki toplama çarpmaya döndü

$\to \underbrace{\overline{f} = \pi \overline{\pi x_i}}$ $\to \overline{\overline{f}} = \overline{\pi \overline{\pi x_i}}$

Burda da toplama çarpmaya döndü

$\overline{(\overline{ab})(\overline{cd})}$

Nand (x y

$$f = \overline{\prod \overline{\prod x_i}}$$

↳ işlem sonucunda Nandler ile dönüştüre-bileceğim yapılara dönüşmüş olucak

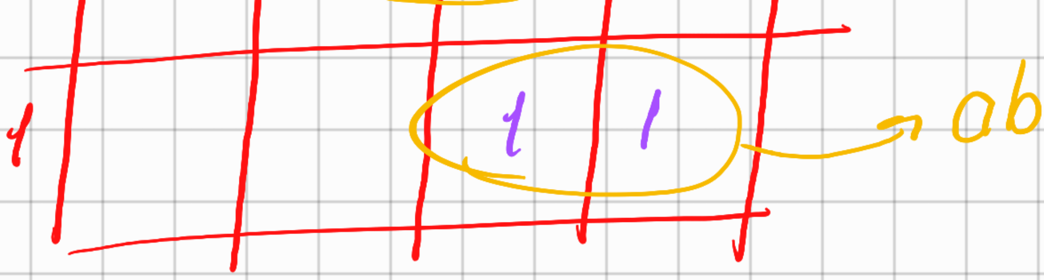
---

ÖR → $\bar{a}\bar{b}c + \bar{a}bc + ab\bar{c} + abc$ Nandler-le gerçek-leyiniz

a	b	c	d	f
0	0	0	0	
1	1	1	1	

a \ bc	00	01	11	10
0		1	1	

→ $a'c$



↳ $f = ab + a'c$

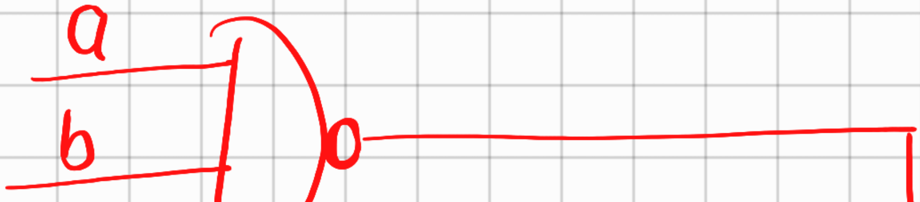
↳ $f = \overline{\overline{ab + a'c}}$

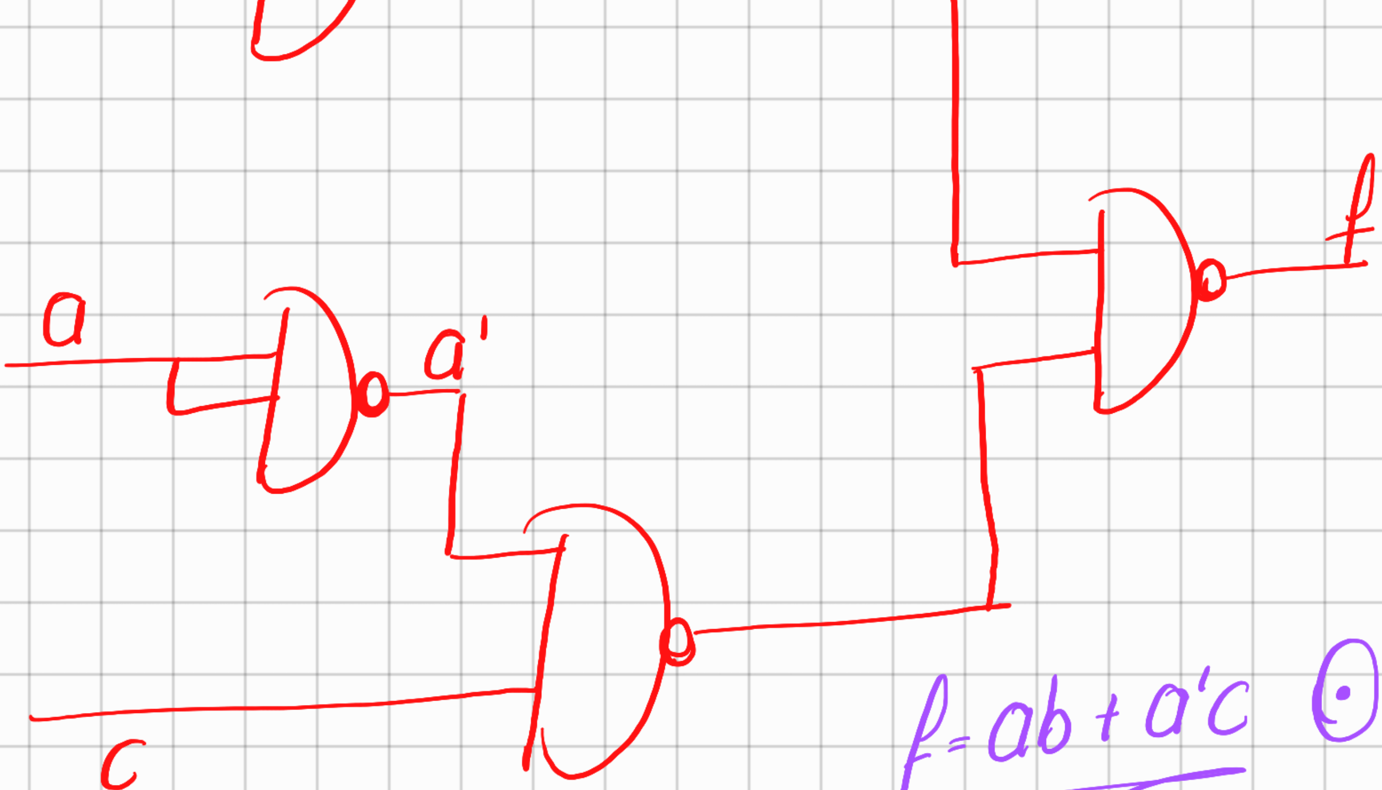
↳ $\overline{(ab)'(a'c)'}$

x y

$(x \cdot y)'$

↳ Çarpımlar toplamı şeklindeki ifadeyi 2 kere değilini alınca zaten Nandler ile yapılır hale geliyor





$f = ab + a'c$ ⊙

Ç

Çarpımlar toplamı (T veya) NOR tasarımı

$(a+b)'$

ifadeyi bu hale getirmeye çalışcaz

$f = \sum \pi x_i \rightarrow \pi x_i = \overline{\sum \overline{x_i}}$

$(ab) \rightarrow (\overline{\overline{ab}}) \rightarrow (\overline{a'+b'})$

$$\sum\overline{\sum\overline{x_i}} = f$$

$$\hookrightarrow \overline{f} = \overline{\sum\overline{\sum\overline{x_i}}}$$

$$\overline{\overline{f}} = f$$

istedigimiz f fonksiyonunu elde etmiş oluruz

ÖR $\rightarrow f(abc) \rightarrow ab + a'c$

$$\hookrightarrow (a'+b') + \overline{(\overline{a}'+\overline{c})}$$

$$\hookrightarrow (\overline{a'+b'}) + (\overline{a+\overline{c}}) = f$$

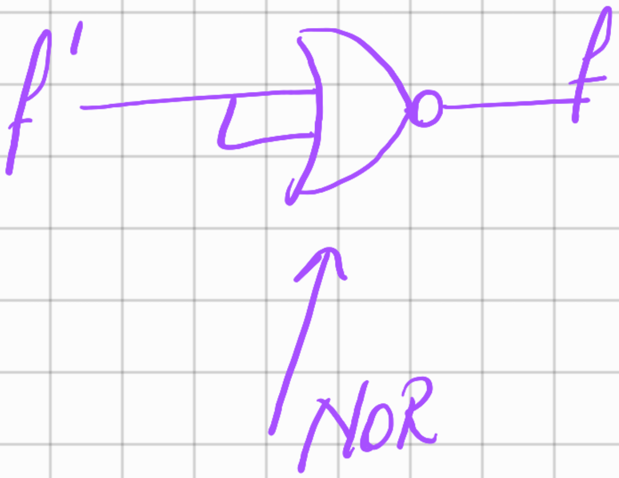
$$(\overline{x+y}) + (\overline{x+y})$$

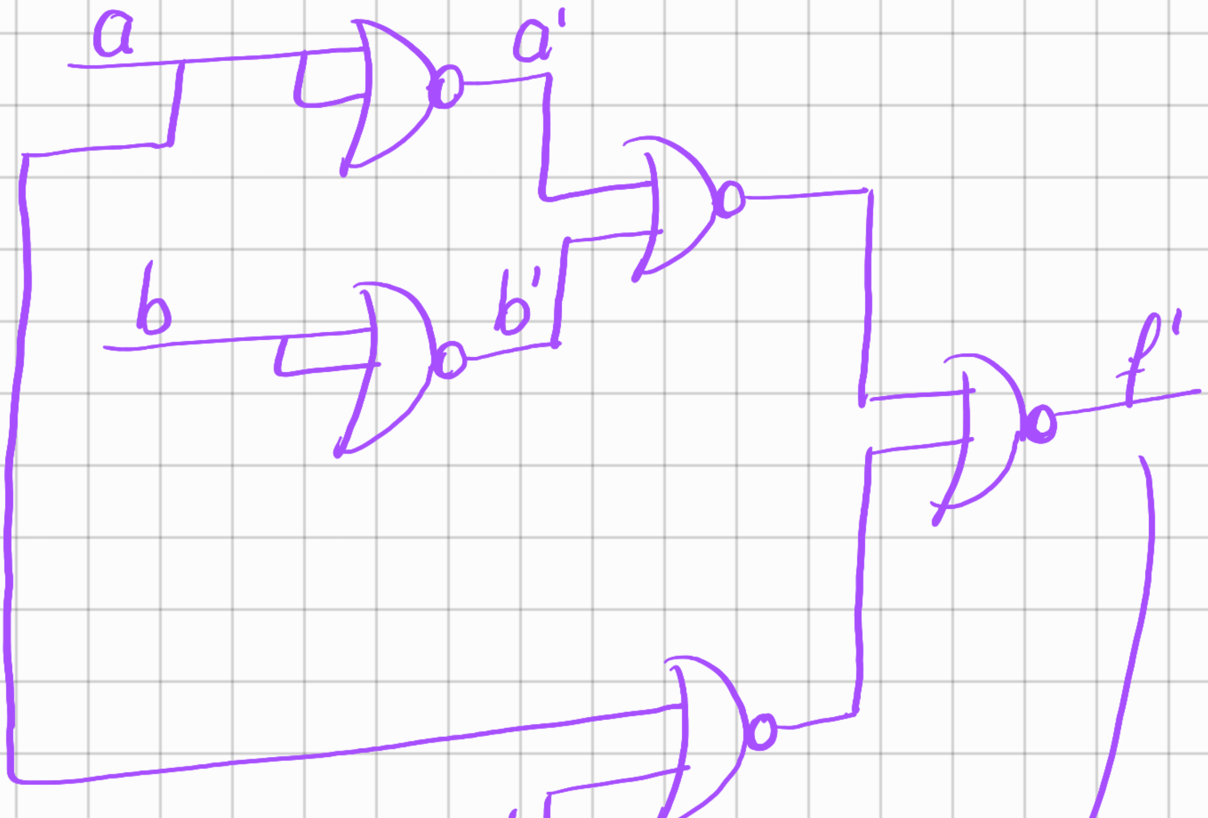
$f.(x+y)$

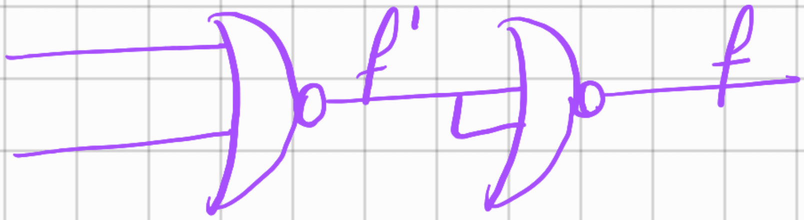
$f' = \overline{(a'+b')} + \overline{(a+c')}$

$\overline{(x+y)}$

Bunun Nor formunda olmasını sağlamak için







Toplamlar Çarpımı (Nand) tasarımı

t Ve
(tümleyen Ve)

$(a.b)'$
$(x.y)'$

$f = \Pi \underbrace{\Sigma x_i}$

$\Sigma x_i = \overline{\Pi \overline{x_i}}$

$\overline{(a+b)} = (a'.b')$

$f = \prod \overline{\prod \overline{x_i}}$

$\rightarrow \overline{(\overline{x_1}\,\overline{x_2})}$

$\rightarrow \overline{(a.b)} \rightarrow$ Nand

$\rightarrow \bar{f} = \overline{\prod \overline{\prod \overline{x_i}}}$

$\rightarrow f'$  f

ÖR $\rightarrow f(abc) \rightarrow [a+b][\bar{a}+c]$

$\rightarrow$ Bize burda bunu Nandleyiniz diyor

$\rightarrow (x.y)'$

Aslında biz bu hale getirmeye çalışıyoruz

$$\overline{\left[\overline{a+b}\right]} = (\overline{a'.b'})$$

$$\hookrightarrow (\overline{a'.b'})(\overline{a'+c'})$$

$$\hookrightarrow f$$

$$\hookrightarrow f = \underbrace{(\overline{a'.b'})}_{X} \cdot \underbrace{(\overline{a+c'})}_{Y}$$

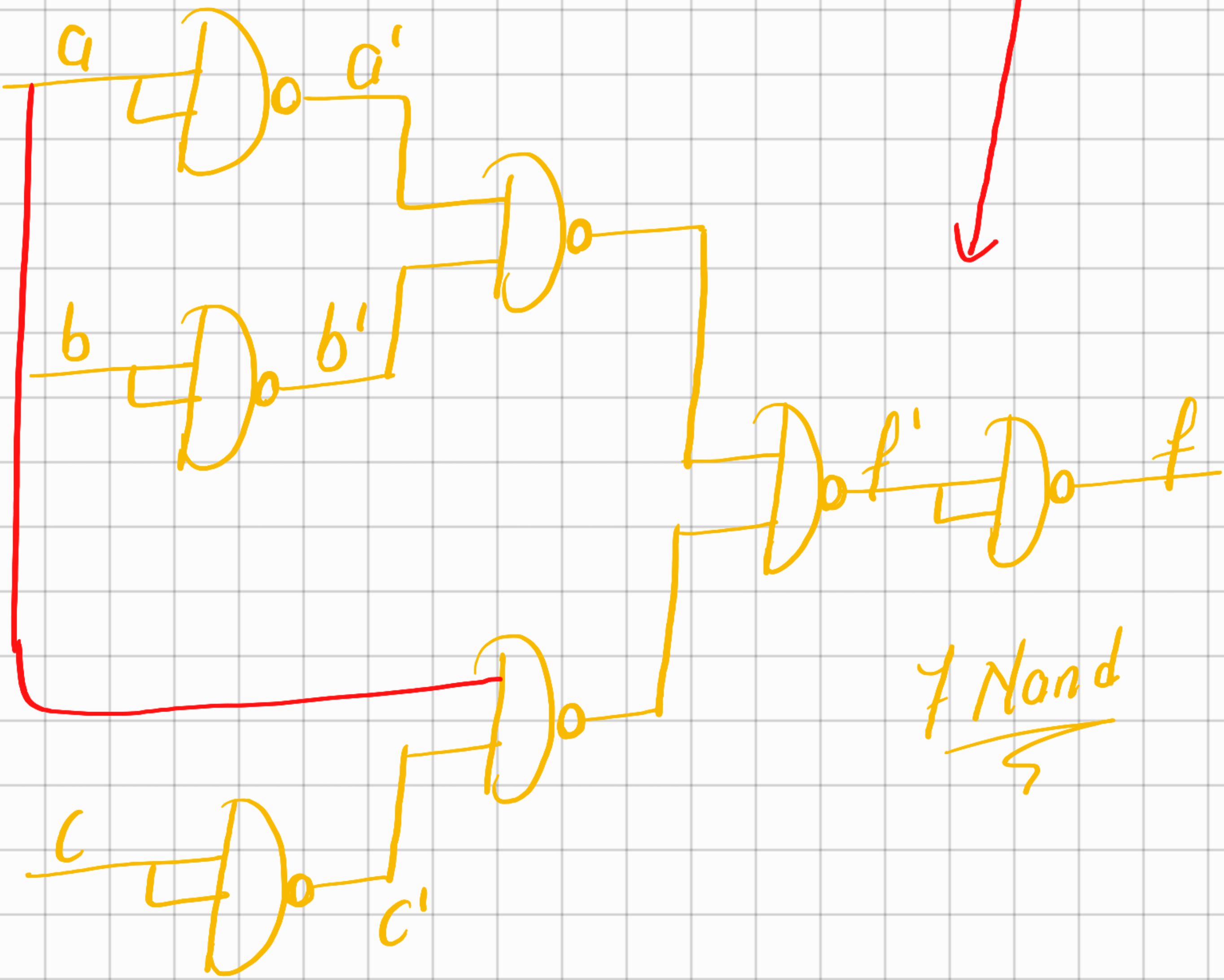
Bakıyoruz burası normal bir and işlemi bunu dönüştürmek için değilini almamız gerekir

yani

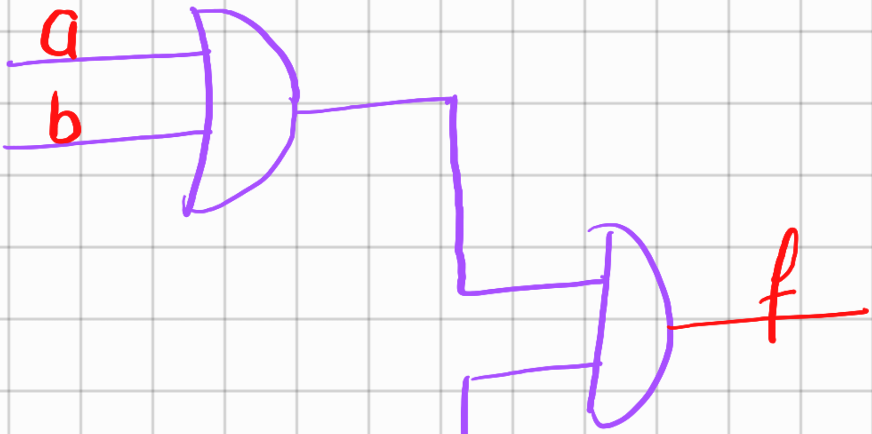
$(x.y)'$

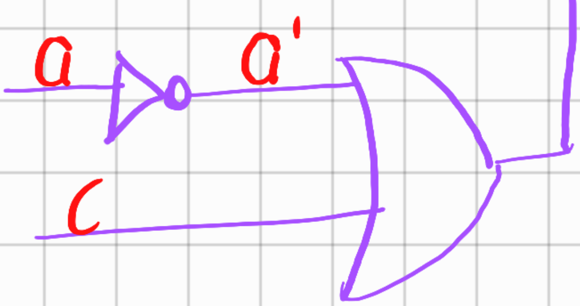
$\overline{(x.y)}$

$f' = \overline{\overline{(a'b')}\;\overline{(a+c')}}$



7 Nand



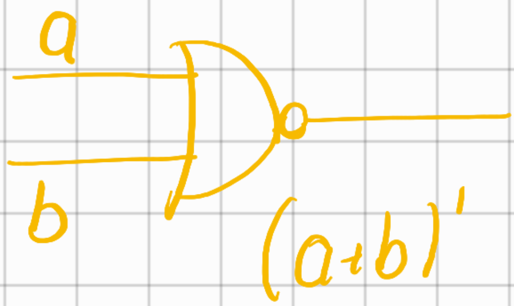


eğer bu işlemleri yapmasaydık 3 değişik kapı kullanıcaktık farklı kapılar kullanmak kötü bir şey onun yerine bu işlemleri 7 Nand ile hallettik

---

Toplamlar Çarpımı (NOR) tasarımı
(Π veya)

$f = \Pi \Sigma x_i$



$f' = \overline{\Pi \Sigma x_i}$

$$f' = \sum \sum x_i$$

$$\overline{(f')} = f = \overline{\sum \overline{\sum x_i}}$$

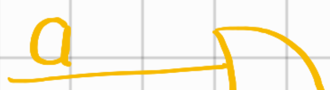
↳ Bu toplamın değilini almak demek Nor kullanmak demek

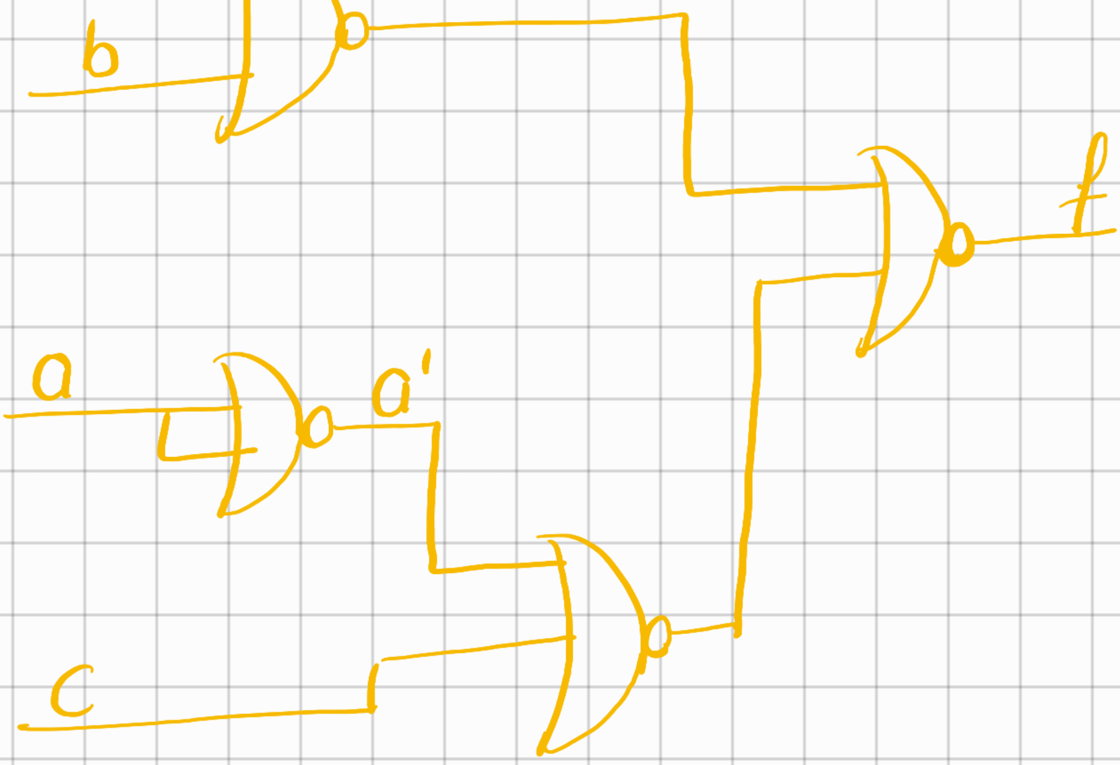
ÖR $\rightarrow f(a,b,c) \rightarrow [a+b][a'+c]$

↳ $f' = \overline{[a+b]} + \overline{[a'+c]}$

$$f' = x+y$$

$$\overline{(f')} = \overline{\underbrace{\overline{[a+b]}}_{x} + \underbrace{\overline{[a'+c]}}_{y}}$$

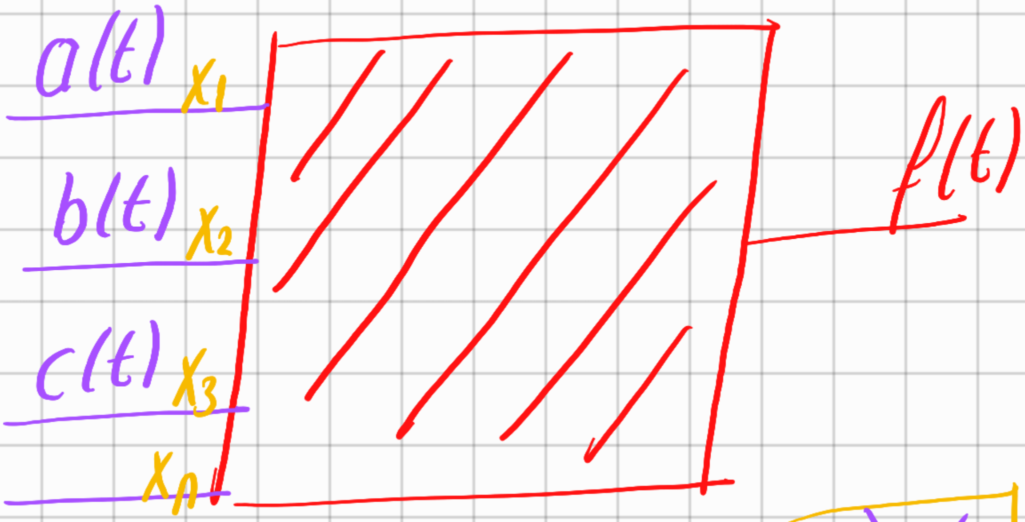




Kombinezonsal Devreler

↳ Çıkışı sadece o anki giriş değerlerine bağlı olan ve bilinen kapıların biraraya getirilmesiyle oluşan lojik devreler

↳ Burda temel kapıları ve türetilmiş kapıları kullanıcaz



~~hafıza~~

çıkış bunların t anındaki girişlerine bağlı

hafıza bilgisine yapısına gerek yok

çıkış anında t anındaki girişlerden t anındaki çıkış üretiliyor

Aritmetik toplama/çıkarma devreleri

yarı toplayıcı/çıkarıcı

...layıcı/çıkarıcı

tam toplayıcı

↳ Multiplekser (seçiciler)

↳ Demultiplekser (Dağıtıcılar)

↳ Decoders (Kod çözücü)

↳ encoders (Kodlayıcı)

↳ ALU (Arithmetic logic Unit)

↳ 7 segment display

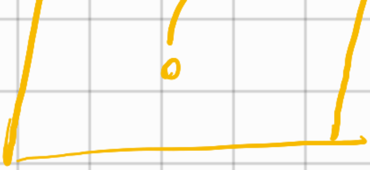
Kombinezonsal Devre tasarımı

① → problemin söz ile tanımı yapılır

Meselа

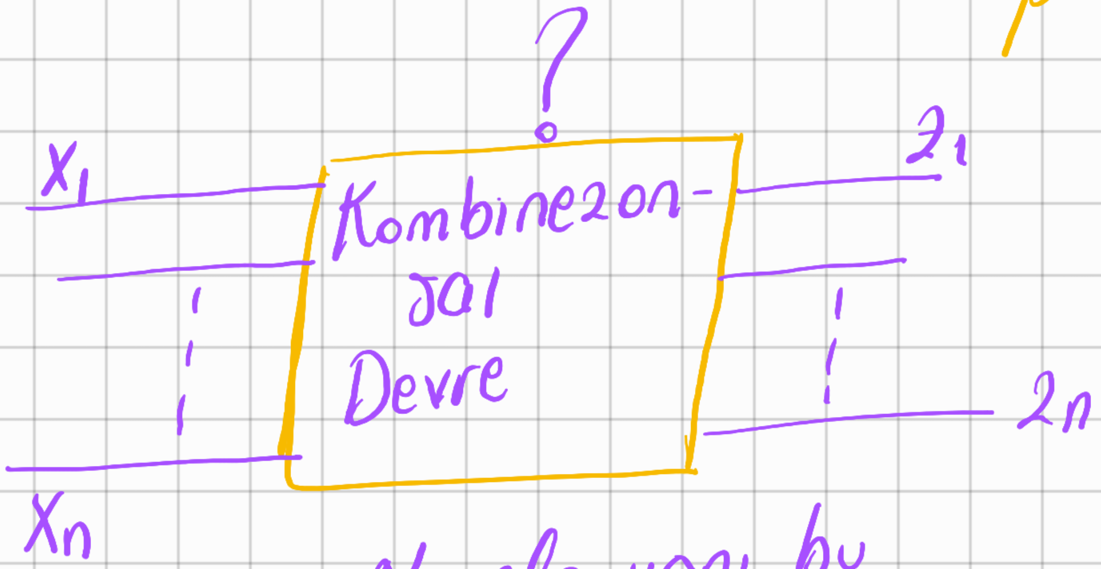
?

Biz bu kutunun ne iş yapmasını istiyoruz



şını istiyo

kaç input alı-cak kaç out-put alıcak



Mesela yani bu kombinezonsal devre ne yapıcak önce bunun tanımı yapılmalı

(2) → giriş ve çıkış değişkenlerinin sayısı / isimlendirmesi

(3) → Probleme ilişkin doğruluk tablosu

(4) → her bir çıkışa ait lojik fonk minimum maliyet olması için indirgenir (görüşe dayalı

K-Map
Quene-Mc Clusky

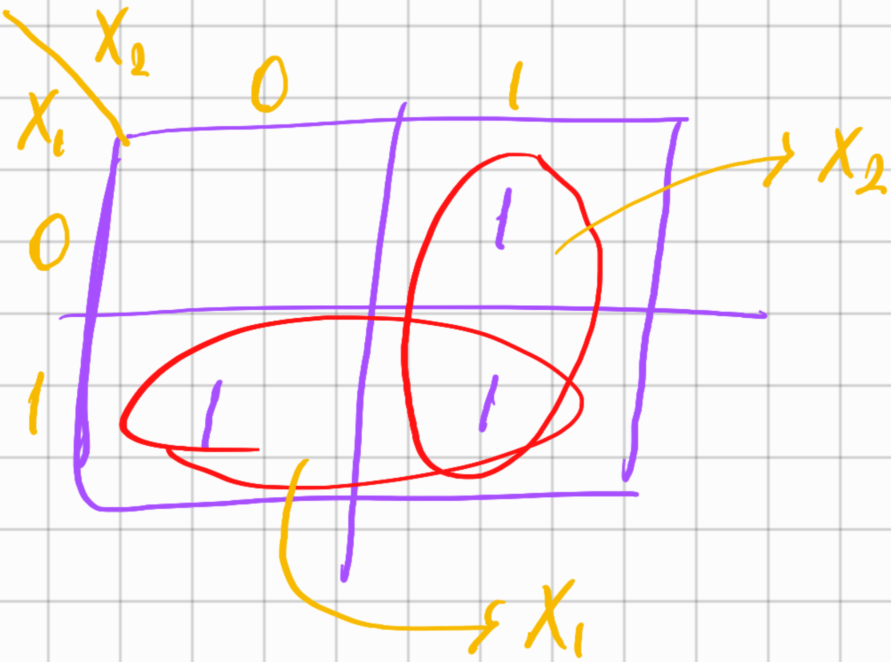
(5) → (Kullanılcak lojik devre elemanları belirleniyor) tek tip kapı kullanımı istemiyorsa dönüşümler gerçekleştirilir

(6) → Devreye ilişkin devre şeması çizilir

ÖR →

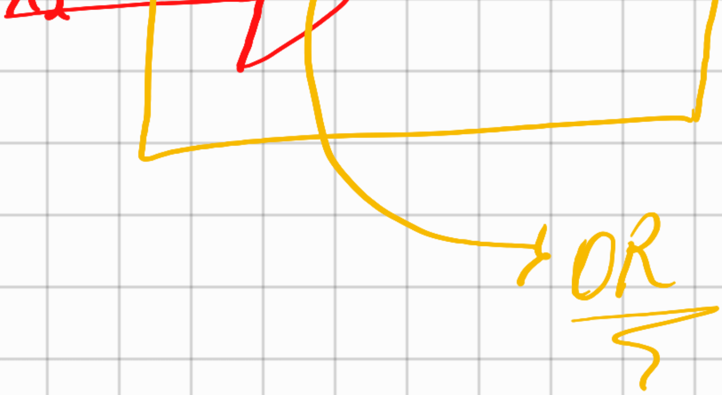
↳ iki girişli bir çıkışlı bir kombinezonsal devre olsun ve bu kombinezonsal devre de girişlerin ikisi de 0 ise çıkışta lojik 0 aksi durumda devrenin çıkışında lojik 1 olması isteniyor

x_1	x_2	z
0	0	0
0	1	1
1	0	1
1	1	1



$f \to x_1 + x_2$

OR

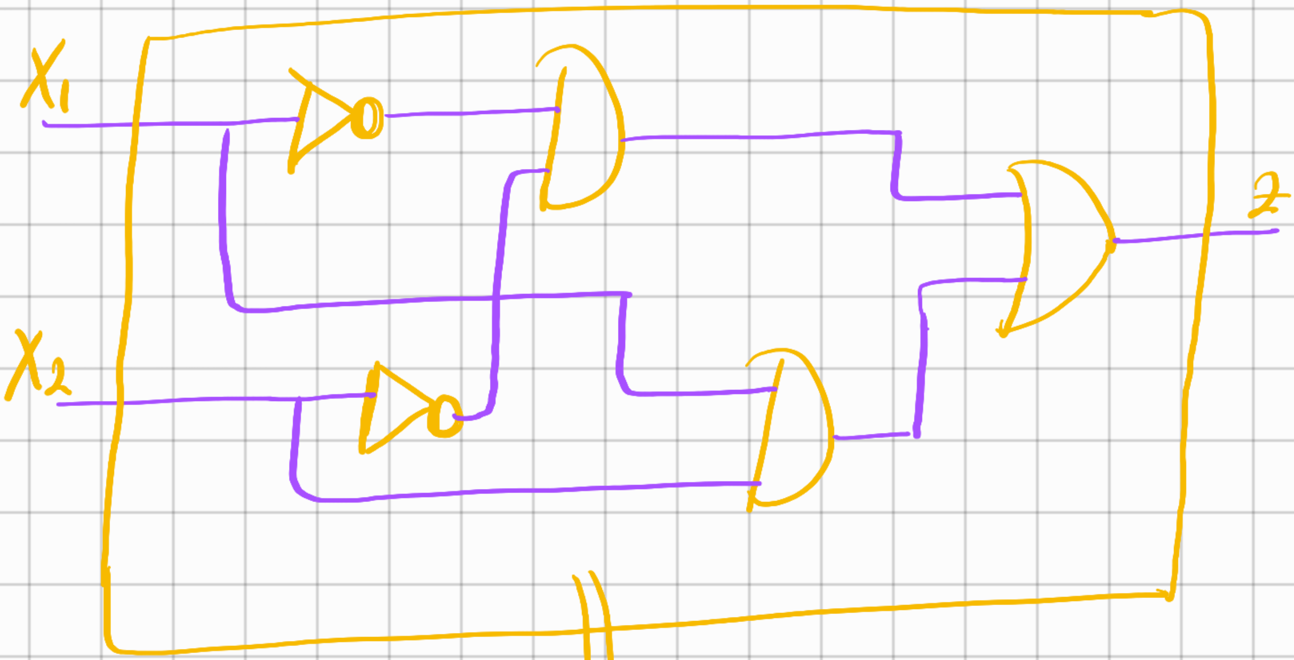


x_1	x_2	z
0	0	1
0	1	0
1	0	0
1	1	1

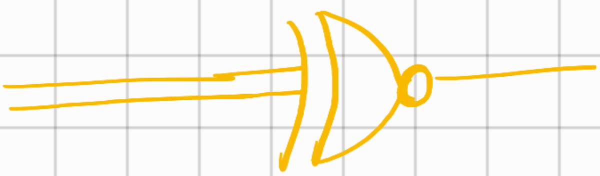
girişler birbirine eşitse 1 onun haricindeki yerlerde 0 versin

0	1	
1		1

$\hookrightarrow \boxed{f \rightarrow x_1'x_2' + x_1x_2}$



Aynıdı



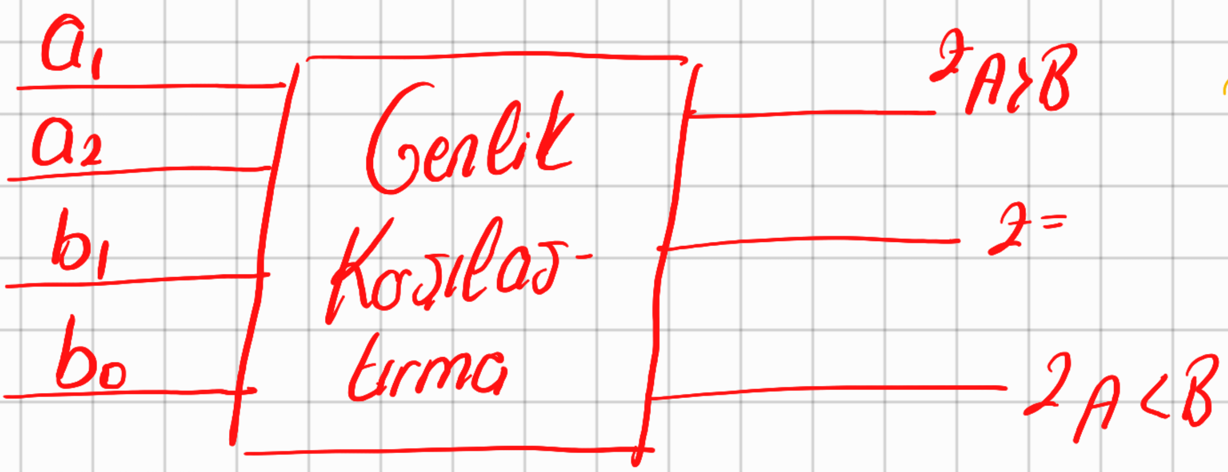
XNOR

[illegible] 2 sayı olsun

OR → 2 bitlik 2 sayı

MSB LSB
$a_1 a_0$ $b_1 b_0$

↳ 2 bitlik genlik karşılaştırma (ampitude comparater)



3 çıkıştan yalnızca biri olabilir

	girişler				çıkışlar		
a	a_1	a_0	b_1	b_0	$a>b$	$a=b$	$a<b$
	0	0	0	0	0	1	0
	0	0	0	1	0	0	1
	0	0	1	0	0	0	1
	0	0	1	1	0	0	1
	0	1	0	0	1	0	0

0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

16 input

→ 3.çıkış

→ 2.çıkış

→ 1.çıkış

Bunların ayrı ayrı karno mapları alarak fonksiyonları oluşturcaz

z a>b

$b_1b_0 \backslash a_1a_0$	00	01	11	10
00		1	1	1

→ $a_0 b_1' b_0'$

→ $a_1 \bar{b}_1$

01			1	1
11				
10			1	

$\rightarrow a_1 a_0 b_0'$

$\hookrightarrow f_1 = z_{a>b} = a_1\bar{b}_1 + a_0 b_1' b_0' + a_1 a_0 b_0'$

$b_1b_0 \backslash a_1a_0$	00	01	11	10
00	1			
01		1		
11			1	
10				1

$\hookrightarrow f_2 = z_{a=b}$

$\hookrightarrow a_1'a_0'b_1'b_0' + a_1'a_0 b_1' b_0 + a_1 a_0 b_1 b_0$

$a_1 a_0' b_1 b_0'$

$2 \quad a < b = f_3$

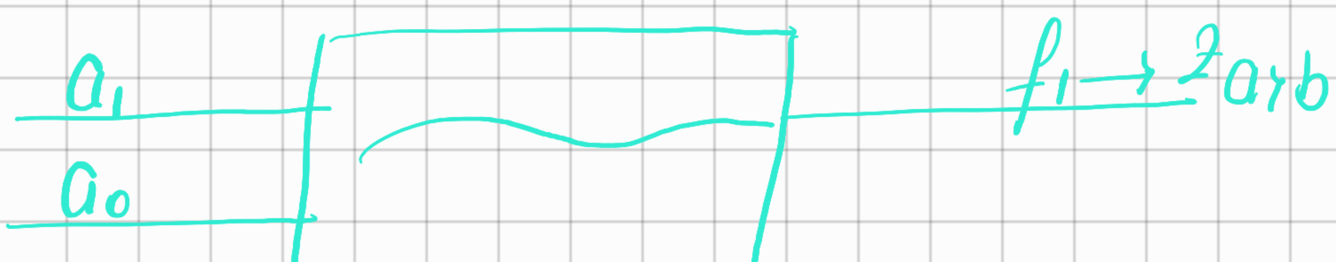
$b_1b_0 \backslash a_1a_0$	00	01	11	10
00				
01	1			
11	1	1		1
10	1	1		

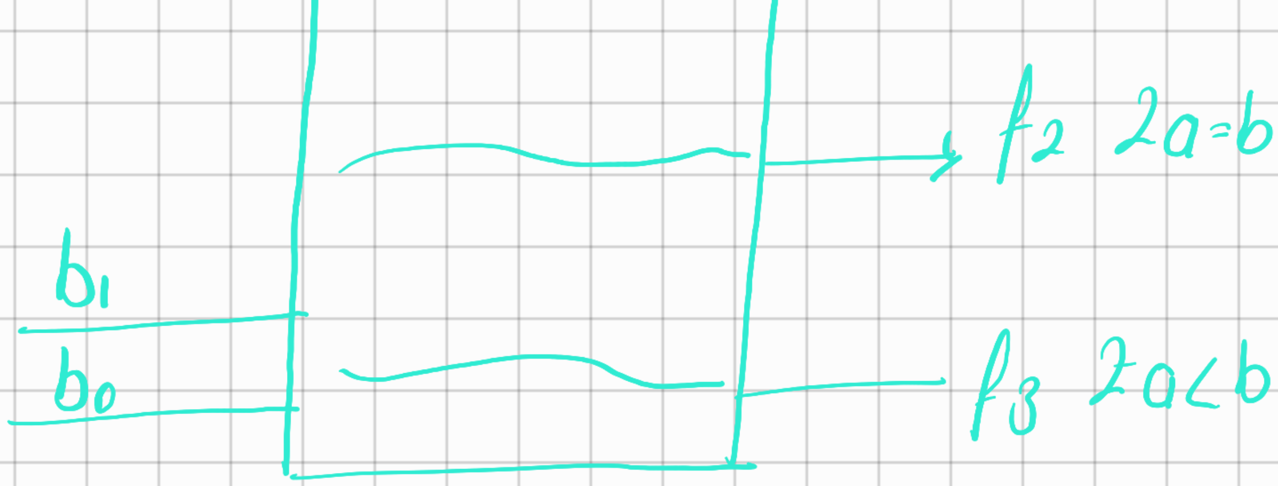
$a_1' a_0' b_0$

$b_1 b_0 a_0'$

$a_1' b_1$

$\hookrightarrow 2 \quad a < b \rightarrow a_1' b_1 + a_1' a_0' b_0 + a_0' b_1 b_0$





b1
b0
f2 2a=b
f3 2a<b