Switchnätsalgebra

Dagens föreläsning behandlar:

Läroboken kapitel 3
Arbetsboken kapitel 2,3

Ur innehållet:

- Satslogik och Boolesk algebra
- Grindar
- Funktionstabell
- Binär evaluering
- Normal form/Förenklad form/ Minimal form
- Karnaughdiagram

Negation, ”ICKE” → NOT-grind (Inverterare)

Observera de alternativa skrivsätten inom Boolesk algebra $x' = \overline{x}$

satslogik

sannings-tabell

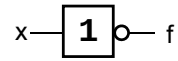
p	¬p
F	S
S	F

Boolesk algebra

funktions-tabell

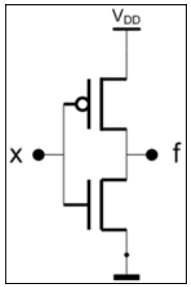
x	f=x’
0	1
1	0

IEC-symbol



Amerikansk symbol

CMOS (Complementary MOS)



logiknivå ***kretsnivå***

Disjunktion, ”ELLER” → OR-grind

satslogik

sannings-tabell

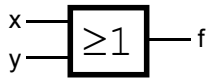
p	q	p∨q
F	F	F
F	S	S
S	F	S
S	S	S

Boolesk algebra

funktions-tabell

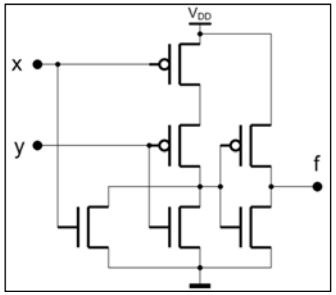
x	y	f=x+y
0	0	0
0	1	1
1	0	1
1	1	1

IEC-symbol



Amerikansk symbol

CMOS (Complementary MOS)



logiknivå ***kretsnivå***

Konjunktion, ”OCH” → AND-grind

satslogik

sannings-tabell

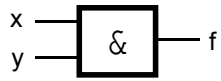
p	q	p∧q
F	F	F
F	S	F
S	F	F
S	S	S

Boolesk algebra

funktions-tabell

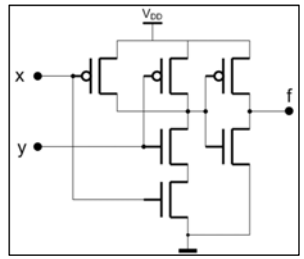
x	y	f=x•y
0	0	0
0	1	0
1	0	0
1	1	1

IEC-symbol

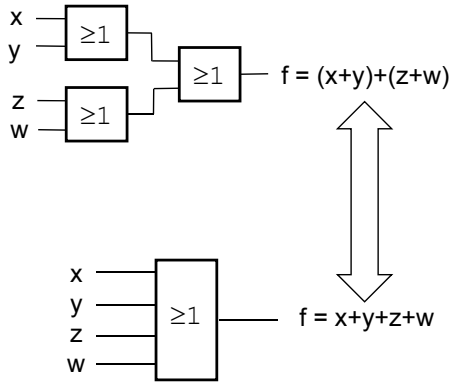


Amerikansk symbol

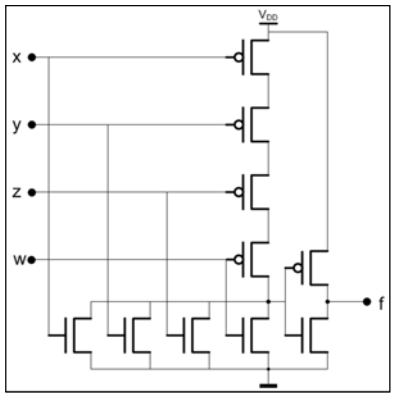
CMOS (Complementary MOS)



logiknivå ***kretsnivå***

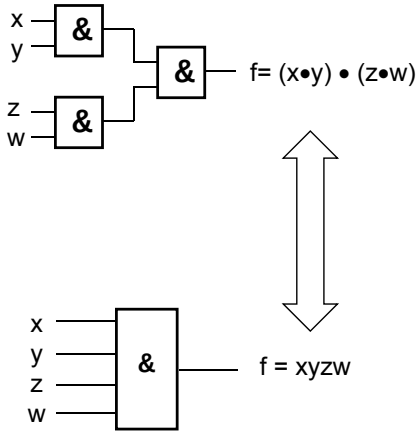


logiknivå

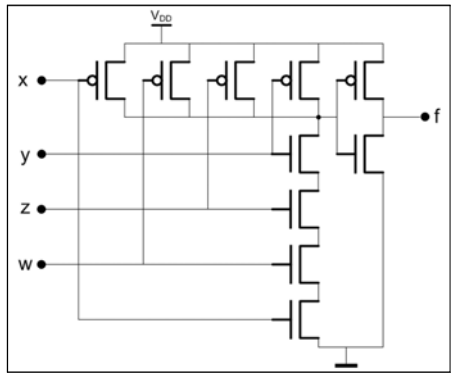


kretsnivå

Antalet ingångar kan utökas



logiknivå



kretsnivå

Antal ingångar (fan-in), begränsas av använd kretsteknologi.

	satslogik		Boolesk algebra
identitet	p∨**F**⇔p p∧**S**⇔p	→ →	x + 0 = x x • 1 = x
dominans	p∨**S**⇔**S** p∧**F**⇔**F**	→ →	x + 1 = 1 x • 0 = 0
tautologi motsägelse	p∨(¬p)⇔**S** p∧ (¬p)⇔**F**	→ →	x + x' = 1 x • x' = 0
idempotens "$a^2 = a$" (alltid)	p∧ p⇔p	→	x • x = x
dubbel negation	¬(¬p) ⇔p	→	(x')' = x
kommutativitet	p∨q⇔q∨p p∧q⇔q∧p	→ →	x + y = y + x x • y = y • x
associativitet	(p∨q)∨r⇔p∨(q∨r) (p∧q) ∧r⇔p∧ (q∧r)	→ →	(x + y) + z = (x + y) + z (x • y) • z = (x • y) • z
distributivitet	p∨(q∧r)⇔(p∨q)∧(p∨r) p∧(q∨r)⇔(p∧q)∨(p∧r)	→ →	x + (y • z) = (x + y) • (x + z) x • (y + z) = (x • y) + (x • z)
deMorgans teorem	¬(p∧q) ⇔¬p∨¬q ¬(p∨q) ⇔¬p∧¬q	→ →	(x • y)' = x' + y' (x + y)' = x' • y'

Binär evaluering

Exempel:

Bevisa deMorgans teorem med hjälp av binär evaluering

Påstående 1: (x • y)' = x' + y'			
x	y	(x • y)'	x' + y'
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0
		VL	HL

Påstående 2: (x + y)' = x' • y'			
x	y	(x + y)'	x' • y'
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0
		VL	HL

deMorgan, generalisering

- Det gäller att:

$$(x_0 \bullet x_1 \bullet ....x_i ... \bullet x_{N-1} \bullet x_N)' = x_0' + x_1' + ...x_i' ... + x_{N-1}' + x_N'$$

- bevisas enklast med induktion:

$$(x_0 \bullet x_1 \bullet ....x_i ... \bullet x_{N-1} \bullet x_N)' = x_0' + x_1' + ...x_i' ... + x_{N-1}' + x_N'$$

$$sätt\ x_a = x_1 \bullet x_2\ och\ skriv$$

$$(x_0 \bullet x_a)' = x_0' + x_a'\ (visat\ tidigare...) \Rightarrow$$

$$(x_0 \bullet (x_1 \bullet x_2))' = x_0' + (x_1 \bullet x_2)' \Rightarrow$$

$$(x_0 \bullet x_1 \bullet x_2)' = x_0' + x_1' + x_2'$$

Ytterligare grindtyper har visat sig användbara

NAND – ”ICKE-OCK” – Negerad AND-grind
Vi ska visa att de grundläggande funktionerna (NOT,AND,OR) samtliga *kan* realiseras med en NAND-typ grind. (NAND-logik)

NOR – ”ICKE-ELLER” – Negerad OR-grind
Vi ska visa att de grundläggande funktionerna (NOT,AND,OR) samtliga *kan* realiseras med en NOR-typ grind. (NOR-logik)

EXCLUSIVE OR - ”EXKLUSIVT ELLER” – XOR-grind
”Härledd funktion” dvs. baserad på användning av NOT/AND/OR. Grindtypen är speciellt användbar vid jämförelseoperationer.

NOT EXCLUSIVE OR – ”ICKE EXKLUSIVT ELLER” – Negerad XOR-grind.

Negerad konjunktion, ”ICKE-OCH” → NAND-grind

satslogik ***Boolesk algebra***

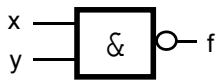
sannings-tabell

p	q	¬(p∧q)
F	F	S
F	S	S
S	F	S
S	S	F

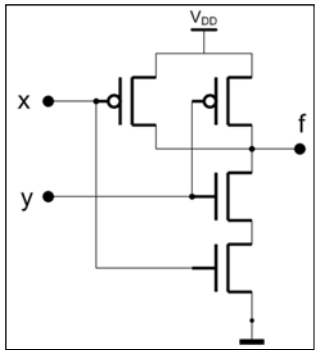
funktions-tabell

x	y	f=(x•y)’
0	0	1
0	1	1
1	0	1
1	1	0

IEC-symbol



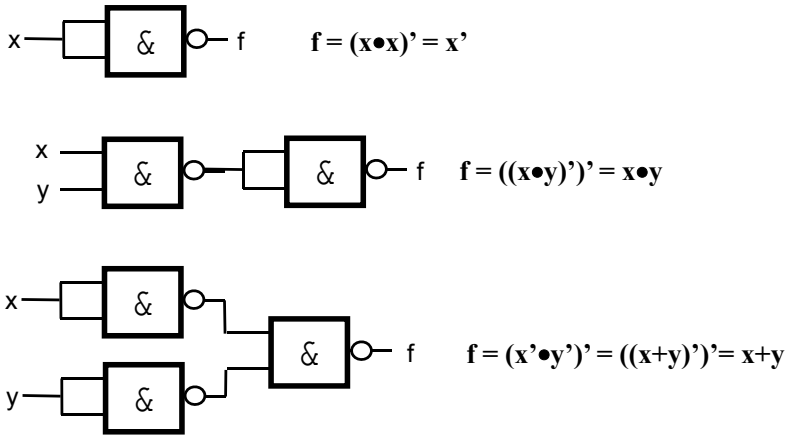
Amerikansk symbol



kretsnivå

logiknivå

NOT/AND/OR-funktioner med NAND-logik



Negerad disjunktion, ”ICKE-ELLER” → NOR-grind

satslogik

sannings-tabell

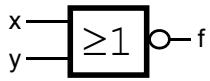
p	q	¬(p∨q)
F	F	S
F	S	F
S	F	F
S	S	F

Boolesk algebra

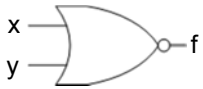
funktions-tabell

x	y	f=(x+y)’
0	0	1
0	1	0
1	0	0
1	1	0

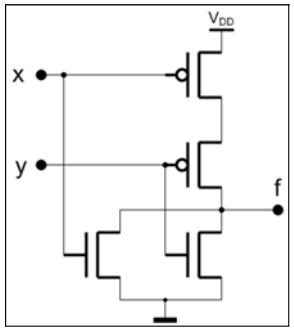
IEC-symbol



Amerikansk symbol

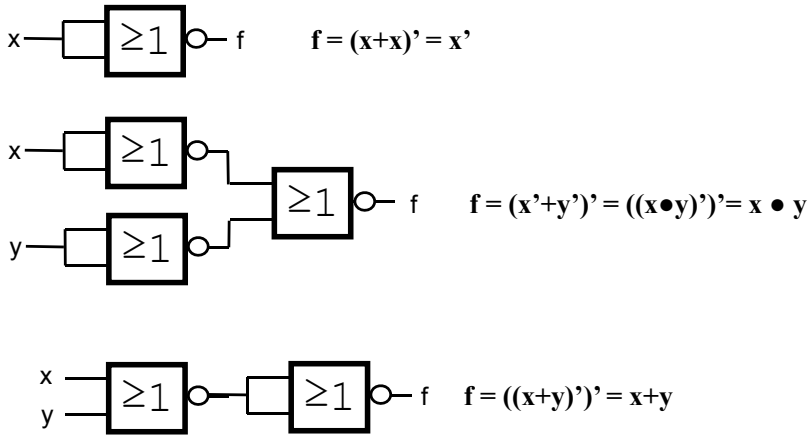


logiknivå



kretsnivå

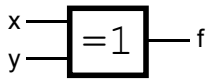
NOT/AND/OR-funktioner med NOR-logik



(NOT) Exkluderande ELLER, (ICKE) XOR-grind

Definition:
x⊕y = x’y+xy’

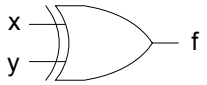
IEC-symbol



funktionstabell

x	y	f=x⊕y
0	0	0
0	1	1
1	0	1
1	1	0

Amerikansk symbol

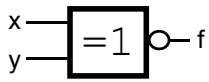


(x⊕y)’ = x’y’+xy

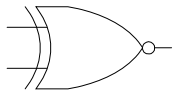
funktionstabell

x	y	f=(x⊕y)’
0	0	1
0	1	0
1	0	0
1	1	1

IEC-symbol



Amerikansk symbol



Evalueringsordning för operatorer

Evalueringsordning (prioriteter) i avsaknad av parenteser för de grundläggande operatorerna är:

1. NOT
2. AND
3. OR

Exempel: Detaljera evalueringsordningen genom att sätta ut parenteser i följande uttryck:

`f(x,y,z,w)=x’+ x·y⊕w`

Lösning:

`f(x,y,z,w)= x’+x·y⊕w =(x’)+ x·y·w’+y’·w =`

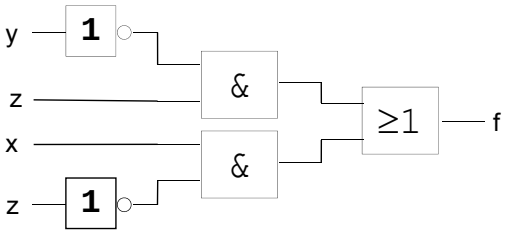
`(x’)+(x·y·(w’))+((y’)·w)`

Booolesk disjunktiv form
(Sum Of Products = SOP-form)

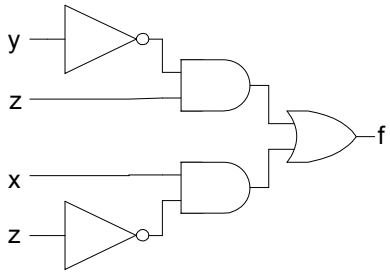
$$(a_1 \bullet a_2 \bullet ... \bullet a_n) + (b_1 \bullet b_2 \bullet ... \bullet b_m) + ... + (c_1 \bullet c_2 \bullet ... \bullet c_p)$$

Exempel: `f(x,y,z)= y'z + xz'` realiseras av grindnätet:

IEC-symboler



Amerikanska symboler



Mintermer

Med ”minterm” menar vi varje unik konjunktion av boolska variabler, dessa kan förekomma i grund- och inverterad form.

Exempel: Mintermer vid tre variabler:

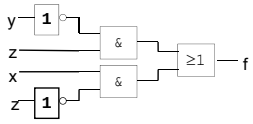
rad	x	y	z	minterm
0	0	0	0	m_0 = x’y’z’
1	0	0	1	m_1 = x’y’z
2	0	1	0	m_2 = x’yz’
3	0	1	1	m_3 = x’yz
4	1	0	0	m_4 = xy’z’
5	1	0	1	m_5 = xy’z
6	1	1	0	m_6 = xyz’
7	1	1	1	m_7 = xyz

Vi kan bekvämt specificera en boolesk funktion genom att ange dess mintermer.

Exempel:
$f(x,y,z)$= x’y’z + x’yz + xy’z =
$m_1 + m_3 + m_5$ =
$\sum(m_1, m_3, m_5)$

Ett vanligt kompakt skrivsätt:
$f(x,y,z) = \sum m\,(1, 3, 5)$

Exempel: Ange mintermerna i funktionen
`f(x,y,z)= y'z + xz'`



Lösning: Ställ upp ***funktionstabell*** för f

x	y	z	y’z	xz’	f=y’z+xz’	
0	0	0				
0	0	1	**1**		**1**	m_1
0	1	0				
0	1	1				
1	0	0		**1**	**1**	m_4
1	0	1	**1**		**1**	m_5
1	1	0		**1**	**1**	m_6
1	1	1				

Dvs:
$f(x,y,z) = \sum m\,(1, 4, 5, 6)$ =
x’y’z + xy’z’ + xy’z + xyz’

Normal och Förenklad disjunktiv form

I föregående exempel såg vi hur:

`f(x,y,z)= y'z + xz'` = `x'y'z + xy'z' + xy'z + xyz'`

Förenklad — ***Normal form (=kanonisk) summa av mintermer***

Exempel: Visa algebraiskt att uttrycken för f är ekvivalenta.

```
f(x,y,z)    = x'y'z + xy'z' + xy'z + xyz' =
            = (x'+x)y'z + x(y'+y)z' =
            = y'z + xz'
```

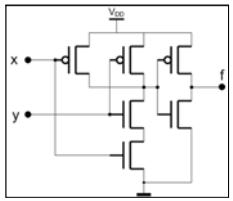
En boolesk funktion kan i allmänhet skrivas på många olika sätt. Det finns dock bara en ***disjunktiv normal form (kanonisk SoP, ”canonical SOP”)***.
Övriga former sägs vara *förenklade*. En form som inte kan förenklas ytterligare kallas ***minimal***.

Realisering

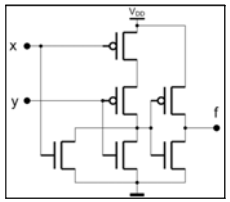
Booleska disjunktiva uttryck ger direkt realisering med NOT/AND/OR-logik, exempelvis:

`f(x,y,z)= y'z + xz'` (2 st. 2-ing AND och 1 st. 2-ing OR)

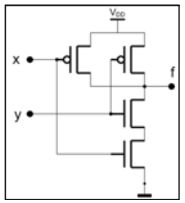
Det kan finnas skäl för att implementera med annan logik



2-ing AND



2-ing OR



2-ing NAND

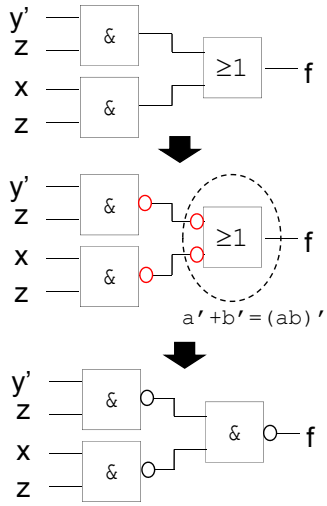
Kan vi realisera f med NAND i stället för AND/OR så kan vi kanske spara transistorer...

AND/OR → NAND

Kan”visualiseras” enligt följande...

Omskrivning av disjunktiv form, skriv f = (f')', och tillämpa deMorgans lag...

```
f(x,y,z)   = y'z + xz' =
     (f')'= [(y'z + xz')']'
          = [(y'z)'(xz')']'
```



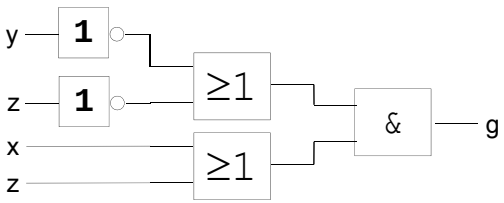
Booolesk konjunktiv form

(Product Of Sums= POS-form)

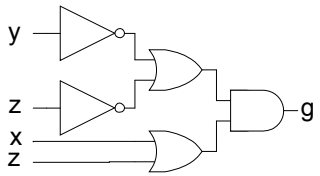
$$(a_1 + a_2 + ... + a_n) \bullet (b_1 + b_2 + ... + b_m) \bullet .... \bullet (c_1 + c_2 + ... + c_p)$$

Exempel: `g(x,y,z)= (x+z)(y'+z')` realiseras av grindnätet:

IEC-symboler



Amerikanska symboler



Maxtemer

Med ”maxterm” menar vi varje unik disjunktion av boolska variabler, ***sådan att dess logiska värde är 0***. Dessa kan förekomma i grund- och inverterad form.

Exempel: Maxtermer vid tre variabler:

rad	x	y	z	maxterm
0	0	0	0	M_0 = x+y+z
1	0	0	1	M_1 = x+y+z'
2	0	1	0	M_2 = x+y'+z
3	0	1	1	M_3 = x+y'+z'
4	1	0	0	M_4 = x'+y+z
5	1	0	1	M_5 = x'+y+z'
6	1	1	0	M_6 = x'+y'+z
7	1	1	1	M_7 = x'+y'+z'

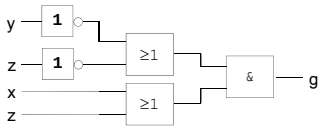
En boolesk funktion kan specificeras även i form av maxtermer.

Exempel:

$$f = \prod M(1,3,5) = M_1 \bullet M_3 \bullet M_5 =$$
$$(x + y + \bar{z}) \bullet (x + \bar{y} + \bar{z}) \bullet (\bar{x} + y + \bar{z})$$

Exempel: Ange maxtermerna i funktionen

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



Lösning: Ställ upp ***funktionstabell*** för g

x	y	z	x+z	y'+z'	g=(x+y)(y'+z')	
0	0	0	0		0	M_0
0	0	1				
0	1	0	0		0	M_2
0	1	1		0	0	M_3
1	0	0				
1	0	1				
1	1	0				
1	1	1		0	0	M_7

Dvs:

$g(x,y,z) = \prod M(0, 2, 3, 7) =$

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

Normal och Förenklad konjunktiv form

I föregående exempel såg vi hur:

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

Förenklad — ***Normal form (=kanonisk) produkt av maxtermer***

Exempel: Visa algebraiskt att uttrycken för g är ekvivalenta.

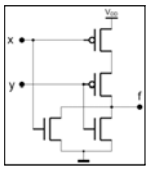
$$
\begin{aligned}
g &= (x+y+z)(x+y'+z)(x+y'+z')(x'+y'+z') \\
&= (xx+xy'+xz+xy+yy'+yz+xz+zy'+zz) \\
&\quad (xx'+xy'+xz'+x'y+y'y'+y'z+x'z'+y'z'+z'z') \\
&= (x+xy'+xz+xy+yz+zy'+z) \\
&\quad (xy'+xz'+x'y'+y'z'+x'z+z') \\
&= (x+z+x(y'+y)+z(y+y')+xz) \\
&\quad ((x+x')y'+z'(x+x')+y'z'+z') \\
&= (x+z+x+z+xz)(y'+z'+y'z'+z') \quad \text{(redundans/dominans)} \\
&= (x+z)(y'+z')
\end{aligned}
$$

Realisering, OR/AND → NOR

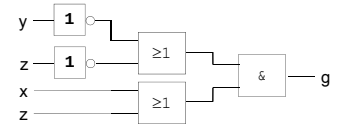
Booleska konjunktiva uttryck ger direkt realisering med NOT/AND/OR-logik.
Även här kan vi dock spara transistorer genom att realisera med en alternativ grindtyp. Då vi utgår från konjunktiv form är NOR-realisering lämpligt.

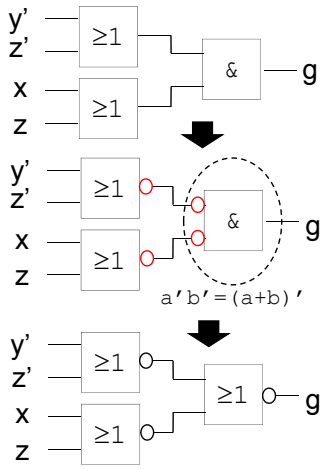
Omskrivning av konjunktiv form, skriv g = (g')', och tillämpa deMorgans lag...

$$
\begin{aligned}
(g')' &= [((x+z)(y'+z'))'] \\
&= [(x+z)'+(y'+z')']
\end{aligned}
$$



2-ing NOR





Minimering av booleska uttryck

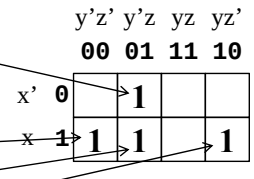
- Vi har sett hur funktionellt ekvivalenta booleska uttryck kan uttryckas på normal/förenklad form med varierande ”kostnad” för realiseringen.
- Vi har använt algebraiska metoder för att förenkla uttryck.
- Det kan i bland vara svårt att se om en förenkling verkligen resulterat i en *minimal form* eller inte.
- För komplexa uttryck är algebraiska metoder väldigt opraktiska och det har därför utvecklats åtskilliga metoder för ***minimering*** av Booleska uttryck.
- I denna kurs använder vi ***Maurice Karnaugh's*** metod med ”Karnaughdiagram”. Vi ger här metoden med praktiska exempel dock utan bevis.

Karnaughdiagram - metod

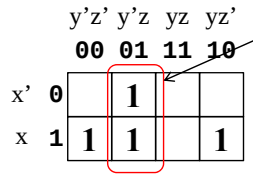
Betrakta funktionstabellen för
f(x,y,z)= ∑ m (1 , 4, 5 , 6) =
x'y'z + xy'z' + xy'z + xyz'

rad	x y z	f
0	0 0 0	
1	0 0 1	1
2	0 1 0	
3	0 1 1	
4	1 0 0	1
5	1 0 1	1
6	1 1 0	1
7	1 1 1	

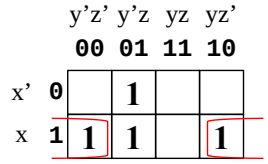
..ger Karnaughdiagram för tre variabler.



Karnaughdiagram - metod

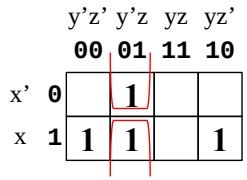


Betrakta två intilliggande mintermer.
x'y'z+xy'z=y'z(x'+x)=y'z
Vi ser detta direkt i diagrammet genom att konstatera att för de inringade mintermerna är y och z konstanta medan x varierar (x'+x) varför x kan tas bort.



Kodningen innebär att diagrammet kan vikas (som en cylinder), med inringningen täcker vi nu de båda återstående mintermerna.
xy'z'+xyz'=xz'(y'+y)=xz'

Den minimerade funktionen.
f = y'z+xz'

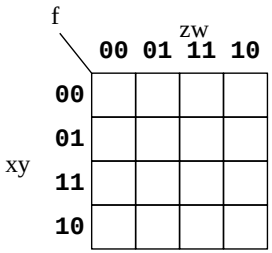


Vi hade följaktligen lika gärna kunnat täckt de båda tidigare mintermerna med denna inringning.
Vilket hade gett oss samma resultat.

Hur vet vi att denna form av funktionen är minimal?
-Studera Karnaugh's bevis.

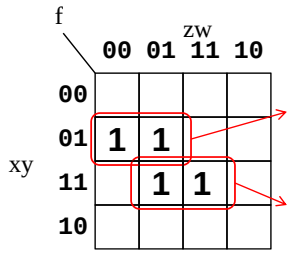
- Samtliga mintermer måste "täckas", dvs. samtliga ettor måste ringas in.
- Endast intilliggande ettor kan ringas in samtidigt (ej diagonalt...)
- Antalet ettor i en inringning måste vara 2^n (n=0,1,2,3...), dvs 1,2,4,8,16...
- Minimal form om samtliga mintermer täckts med minimalt antal inringningar.

Karnaughdiagram för 4 variabler
f(x,y,z,w)



Exempel:
Minimera
f(x,y,z,w)= ∑ m (4, 5 ,13,15)

Lösning:
För in mintermerna i Karnaughdiagrammet
Ringa in mintermer



Identifiera f

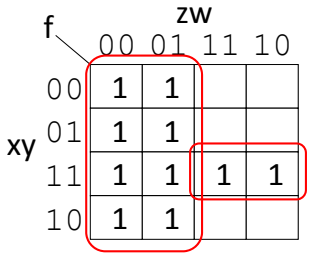
x'yz'w'+x'yz'w = x'yz'(w+w')

xyz'w+xyzw = xy(z'+z)w

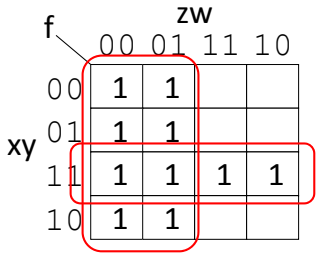
$f(x,y,z,w) = x'yz'+xyw$

Minimal form

Låt varje inringning omfatta *så många mintermer* som möjligt.
Samma minterm kan ringas in flera gånger.
Exempel:



EJ MINIMAL



MINIMAL

Sammanfattning - Karnaughdiagram