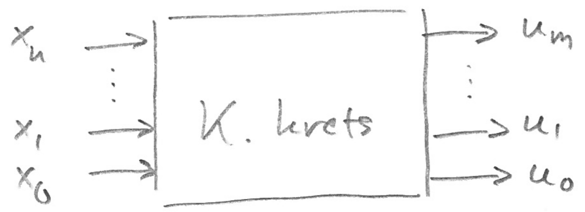
Kombinationskretsar

Def Kombinationskrets



$u = f(x)$ där

$u = (u_m, \ldots, u_1, u_0)$

$x = (x_n, \ldots, x_1, x_0)$

$u_i, x_j \in \{0, 1\}$

Obs: Utsignal u beror endast på x vid samma tidpunkt.

Syntes av k-kretsar

Ex OR-funktionen

Funktionstabell

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

Booleska uttryck

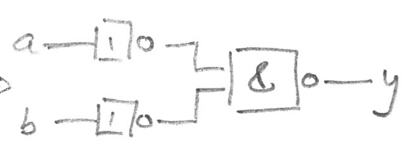
$y = f(a,b) =$

$= a + b =$

$= (a+b)'' =$

$= (a' \cdot b')'$

Realisering med grindar



Notera

- Flera uttryck för samma funktion.
- Uttrycken gå att översätta till grindnät och vise versa.

Syntesfrågeställningar (ev OH)

- Hur ställer vi upp Booleska uttryck för en funktion specificerad av en tabell?

- Hur hittar vi ett uttryck som svarar mot ett billigt grindnät?

  - Billigt: få ingångar, få grindar, liten kiselarea
  - Bivillkor:

    - vissa typer av grindar
    - få plats i en given programmerbar krets (PLD)
    - maximal fördröjning från in- till ut-signal ⇒ begränsat <u>grinddjup</u>

Löpande exempel

Låt $x_2 x_1 x_0$ representera tal där x_2 är msb.

Realisera funktionen som returnerar 1 för de 3 minsta talen.

Funktionstabell

x_2	x_1	x_0	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Grundformer för Booleska funktioner

Def Summa av produkter (SP-form, disjunktiv form)

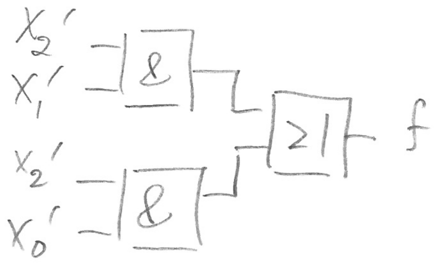
$$f = x_2' x_1' + x_2' x_0'$$

Def Produkt av summor (PS-form, konjunktiv form)

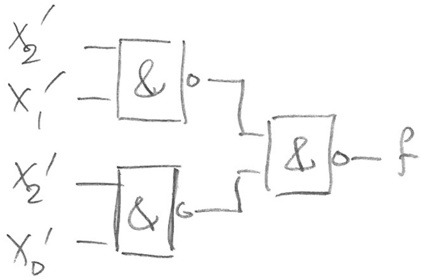
$$f = x_2' (x_1' + x_0')$$

Grindnätstyper för SP-form

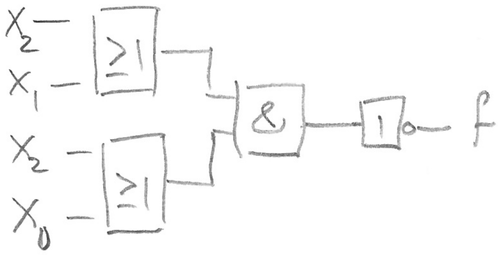
$$f = (x_2'x_1' + x_2'x_0')'' = ((x_2'x_1')'(x_2'x_0')')' = ((x_2+x_1)(x_2+x_0))'$$



AND-OR-nät.



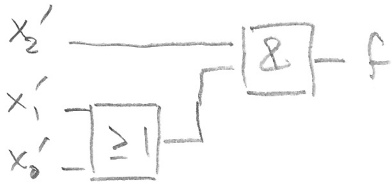
NAND-NAND-nät



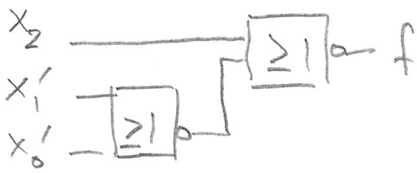
OR-AND-NOT-nät

Grindnätstyper för PS-form

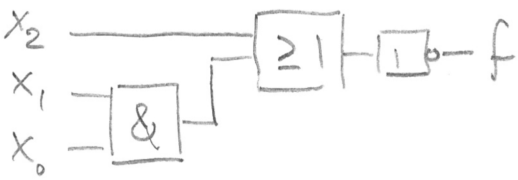
$$f = (x_2'(x_1'+x_0'))'' = (x_2 + (x_1'+x_0')')' = (x_2 + x_1x_0)'$$



OR-AND-nät



NOR-NOR-nät



AND-OR-NOT-nät

Booleska uttryck från funktionstabell

Min- och max-termer

i	x_2	x_1	x_0	$m_0 = x_2'x_1'x_0'$	$m_1 = x_2'x_1'x_0$	$M_0 = x_2+x_1+x_0$	$M_1 = x_2+x_1+x_0'$
0	0	0	0	1	0	0	1
1	0	0	1	0	1	1	0
2	0	1	0	0	0	1	1
3	0	1	1	0	0	1	1
4	1	0	0	0	0	1	1
5	1	0	1	0	0	1	1
6	1	1	0	0	0	1	1
7	1	1	1	0	0	1	1

Def: Till varje insignalkomb. i definieras en

- minterm m_i som blir 1 endast för insignalkomb. i.
- maxterm M_i som blir 0 —"—

$M_0 = m_0' = (x_2'x_1'x_0')' = x_2 + x_1 + x_0$

Generellt gäller att $M_i = m_i'$.

Normalformer

Def: SP-normalform är en SP-form med mintermer.

Ex $f = m_0 + m_1 + m_2 = x_2'x_1'x_0' + x_2'x_1'x_0 + x_2'x_1x_0'$

Förenklat skrivsätt $f = \Sigma(0,1,2)$

$f=1$ för dessa värden

Def: PS-normalform är en PS-form med maxtermer.

Ex: $f = M_3 M_4 M_5 M_6 M_7$

Förenklat skrivsätt $f = \Pi(3,4,5,6,7)$

$f=0$ för dessa värden

Kommentarer

- Normalformerna är entydiga och enkla att teckna.
- Blir stora uttryck och grindnät.

Förenkling av Booleska funktioner

$$f = \underbrace{x_2'x_1'x_0'}_{=0} + \underbrace{x_2'x_1'x_0}_{=1} + \underbrace{x_2'x_1x_0'}_{=2} + \underbrace{x_2'x_1'x_0'}_{=0}$$

är lika i alla utom en bit

$$= \underbrace{x_2'x_1'\overbrace{(x_0'+x_0)}^{=1}}_{\text{term: } 0+1} + \underbrace{x_2'x_0'\overbrace{(x_1+x_1')}^{=1}}_{\text{term } 0+2} =$$

$$= x_2'x_1' + x_2'x_0'$$

Grafisk metod: Karnaughdiagram

(omorganiserad funktionstabell)

$x_2'x_1'$

x_2 \ x_1x_0	00	01	11	10
0	0: 1	1: 1	3: 0	2: 1
1	4: 0	5: 0	7: 0	6: 0

$x_2'x_0'$

Närliggande rutor motsvarar termer med <u>en</u> avvikande bit.

Metod

- Ringa in alla 1:or (inga 0:or)
- Få ringar, stora ringar ger stora förenklingar
- Giltiga sammanslagningar har formen $2^i \times 2^j$ rutor för $i, j \in \{0, 1, 2\}$ (4 var)
- Visa OH

Ex: K-diag. i 4 var.

$x_1'x_0'$

x_3x_2 \ x_1x_0	00	01	11	10
00	1	0	0	0
01	1	1	0	0
11	1	1	1	1
10	1	1	1	1

x_2x_1'

x_3

$$g = x_1'x_0' + x_2x_1' + x_3$$

- Fel: ej största möjliga inringning

	00
00	1

$x_3'x_2'x_1'x_0'$ är mer komplex än $x_1'x_0'$

- Fel: fel storlek

x_3x_2 \ x_1x_0	00	01	11	10
00				
01	1	1		
11	1	1		
10	1	1		

$(x_2+x_3)\,x_1' = x_2x_1' + x_3x_1'$

Motsvarar ej <u>en</u> produkt!

<u>Def</u>: En produktterm hörande till en maximal inringning kallas <u>primimplikator</u>.

<u>Ex</u>: $x_1'x_0'$ är en primimplikator

$x_3'x_2'x_1'x_0'$ är <u>ej</u> en primimplikator

Minimal SP-form

Def: En SP-form bestående av en minimal mängd primimplikatorer är en <u>minimal SP-form</u>.

En minimal SP-form realiserar ett minimalt AND-OR-nät.

Metod

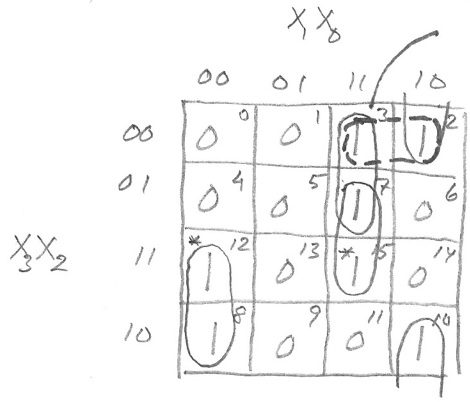
1) Hitta 1:or som bara täcks av en primimplikator.

2) Täck övriga 1:or med primimplikatorer

3) Summera valda primimplikatorer

Ex Bestäm en minimal SP-form till

$$g = \sum(2, 3, 7, 8, 10, 12, 15)$$



Ex på alternativ inringning

$$g = X_3X_1'X_0' + \ldots$$

Notera: Minimal SP-form är <u>ej</u> unik!

Minimal PS-form

Minimal PS-form erhålls med samma metod fast applicerad på funktionens invers.

Ex

f

$x_2 \backslash x_1x_0$	00	01	11	10
0	1	1	0	1
1	0	0	0	0

f'

$x_2 \backslash x_1x_0$	00	01	11	10
0	0	0	1	0
1	1	1	1	1

$$f' = x_2 + x_1x_0$$

AND-OR-NOT-form

$$f = (f')' = (x_2 + x_1x_0)' =$$ de Morgans

$$= x_2' \cdot (x_1x_0)' =$$ de Morgans

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

Oftast ringas 0:or i K-diagrammet för f istället.

Ofullständigt specificerade funktioner

Ibland används inte alla insignalkombinationer och då kvittar det vilken utsignal det blir för dessa värden.

$\Rightarrow$ Ger designfrihet att minimera kretsen.

__Ex__ Bestäm en minimal SP-form för

$$g(x_3,x_2,x_1,x_0) = \Sigma(1,3,8,10,12) + d(5,6,7,13,14,15)$$

don't care, g kan vara 0 eller 1 för dessa värden.

x_3x_2 \ x_1x_0	00	01	11	10
00	(0) 0	(1) 1	(3) 1	(2) 0
01	(4) 0	(5) –	(7) –	(6) –
11	(12) 1	(13) –	(15) –	(14) –
10	(8) 1	(9) 0	(11) 0	(10) 1

don't care

"–" kan ringas in om ringarna kan göras större, men behöver ej ringas in

$$g = x_3x_0' + x_3'x_0 \quad (= x_3 \oplus x_0)$$

Observera att uttrycket är enklare än om '–' satts till 0 och en minimal SP-form beräknats.

__Övning:__ Beräkna detta uttryck och jämför med uttrycket ovan.