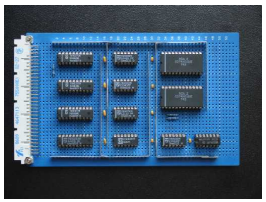
Digital- och datorteknik



Föreläsning #5

Biträdande professor Jan Jonsson

Institutionen för data- och informationsteknik
Chalmers tekniska högskola

Kostnad för grindnät

Vad är ett ”bra” grindnät?

De egenskaper som betraktas som eftersträvansvärda vid realisering av ett grindnät är:

- Att grindnätet tar få fysiska komponenter i anspråk, för att på så sätt hålla både pris, fysiskt utrymme och strömförbrukning på en låg nivå.
- Att grindnätet är snabbt, d v s att löptiden (tiden det tar för en förändring i insignalerna till nätet att visa sig på utsignalen) är kort.

Detaljer i realiseringen som påverkar dessa egenskaper kallar vi för kostnaden för ett grindnät.

Kostnad för grindnät

Vad påverkar kostnaden för ett grindnät?

Det som huvudsakligen påverkar kostnaden hos ett grindnät är:

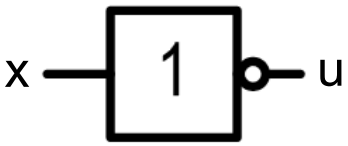
- Antalet transistorer: ju fler transistorer desto större fysiskt utrymme och högre strömförbrukning
- Antalet grindingångar: ju fler ingångar desto fler transistorer
- Antalet grindar: ju fler grindar desto större antal grindingångar

Minimering av kostnaden för ett grindnät bör alltså i första hand ha målet att hålla antalet grindar, grindingångar och transistorer på en låg nivå.

Logikgrindar

De grundläggande logikoperationerna:

INVERTERARE (ICKE, NOT)

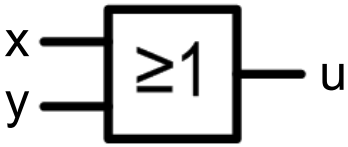


2 transistorer

x	u
0	1
1	0

$u = \overline{x}$

ELLER (OR)



6 transistorer

x	y	u
0	0	0
0	1	1
1	0	1
1	1	1

$u = (x + y)$

OCH (AND)

6 transistorer

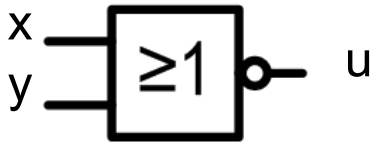
x	y	u
0	0	0
0	1	0
1	0	0
1	1	1

$u = (x * y)$

Logikgrindar

Andra användbara logikoperationer:

NOR

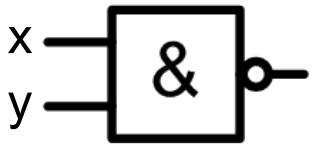


4 transistorer

x	y	u
0	0	1
0	1	0
1	0	0
1	1	0

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

NAND



4 transistorer

x	y	u
0	0	1
0	1	1
1	0	1
1	1	0

$$u = \overline{(x * y)}$$

Kostnad för grindnät

Hur kan vi minimera kostnaden för ett grindnät?

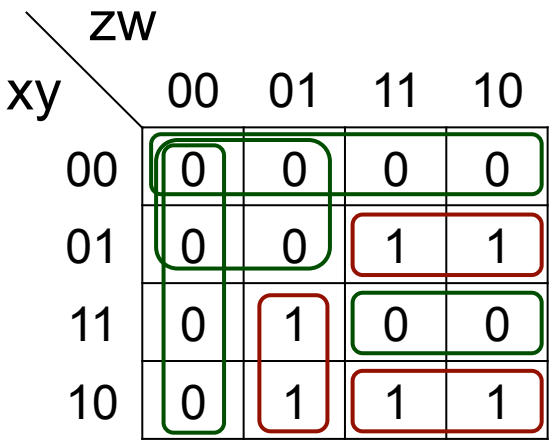
Följande metoder bidrar till att hålla kostnaden nere:

1. Karnaughminimering
2. Användning av grindar med N > 2 ingångar
3. Konvertering av AND/OR (SP-form) till NAND/NAND
4. Konvertering av OR/AND (PS-form) till NOR/NOR
5. Identifiering av XOR/XNOR-funktion
6. Identifiering av ”don’t care”-termer

Kostnad för grindnät

1. Karnaughminimering:

Ett Karnaughdiagram är ett mycket kraftfullt verktyg som gör det möjligt att härleda en minimal lösning för nät på både AND/OR-form (SP-form) och OR/AND-form (PS-form).



SP-form:

$$f = x\bar{z}w + \bar{x}yz + x\bar{y}z$$

PS-form:

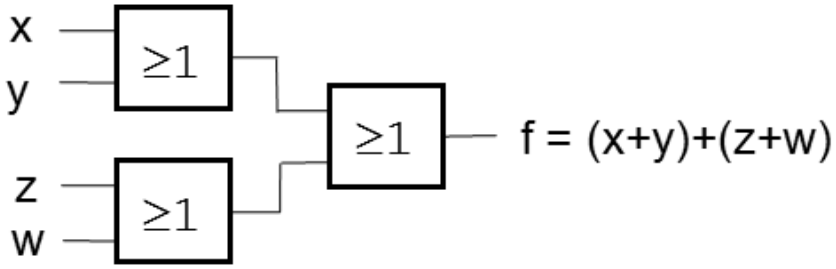
$$f = (x + y)(x + z)(z + w)(\bar{x} + \bar{y} + \bar{z})$$

Kostnad för grindnät

2. Användning av grindar med N > 2 ingångar:

Om enbart grindar med maximalt två ingångar används krävs det flera grindar av samma sort när en logisk funktion kräver tre eller fler invariabler.

En sådan lösning kan användas vid enstaka tillfällen, men bör generellt sett undvikas av följande skäl:

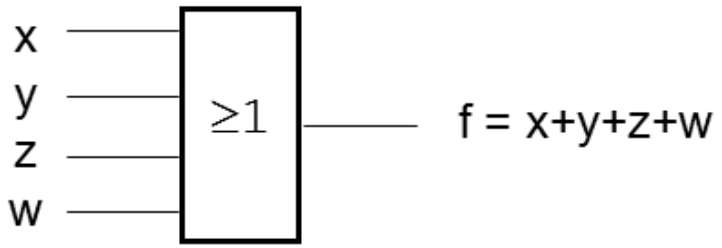


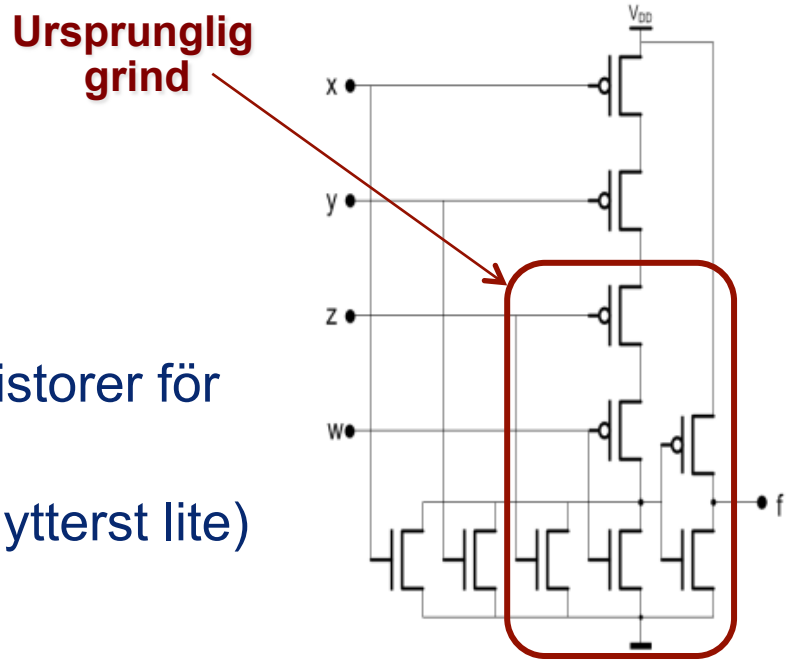
- Det krävs en extra logikgrind för varje ytterligare invariabel.
- Totala löptiden växer logaritmiskt med varje ytterligare invariabel.
  - Exempel 1: fyra invariabler ger två grindars fördröjning
  - Exempel 2: åtta invariabler ger tre grindars fördröjning

Kostnad för grindnät

2. Användning av grindar med N > 2 ingångar:

Associativa lagarna (T5) säger att evalueringsordningen är godtycklig för OR- och AND-grindar. Det är därför enkelt att konstruera sådana grindar med tre eller fler ingångar:



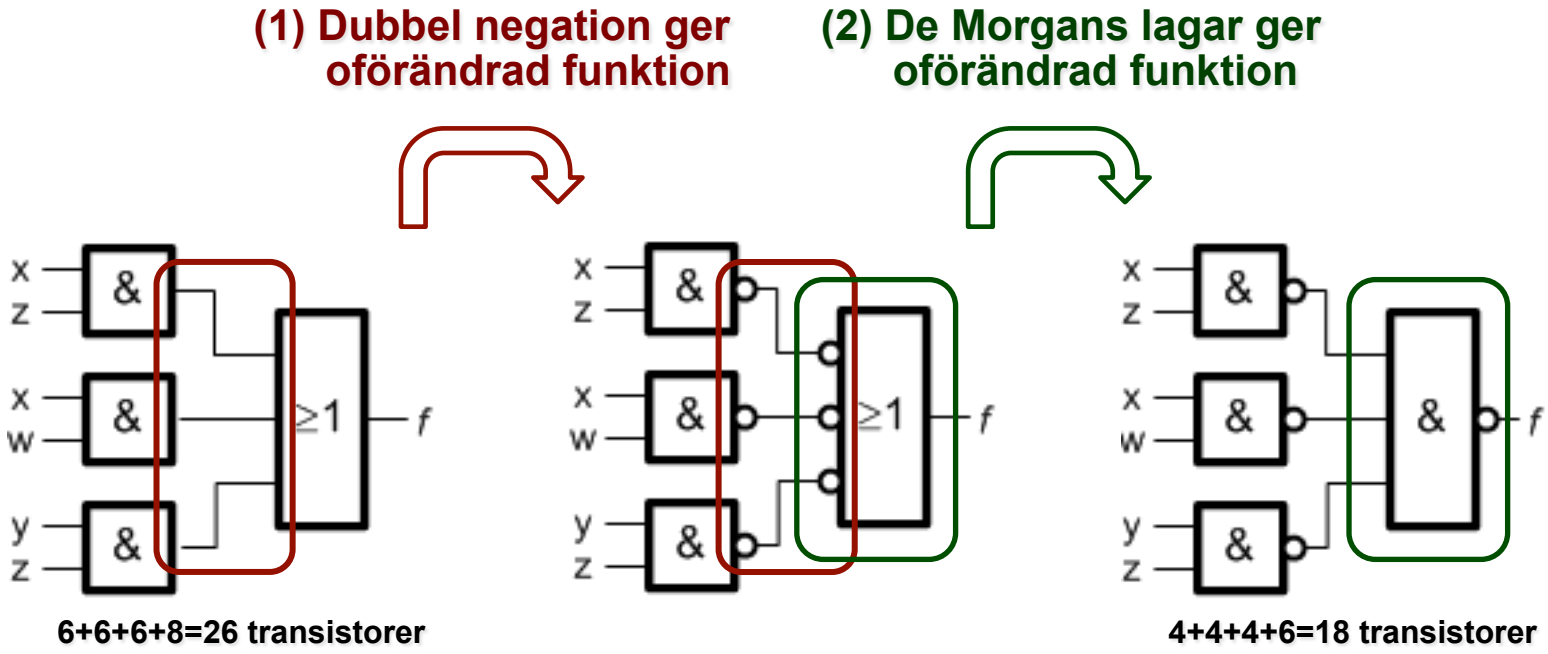


- Det krävs enbart ett extra par transistorer för varje ytterligare invariabel.
- Totala löptiden växer inte alls (eller ytterst lite) med varje ytterligare invariabel.

Kostnad för grindnät

3. Konvertering av AND/OR (SP-form) till NAND/NAND:

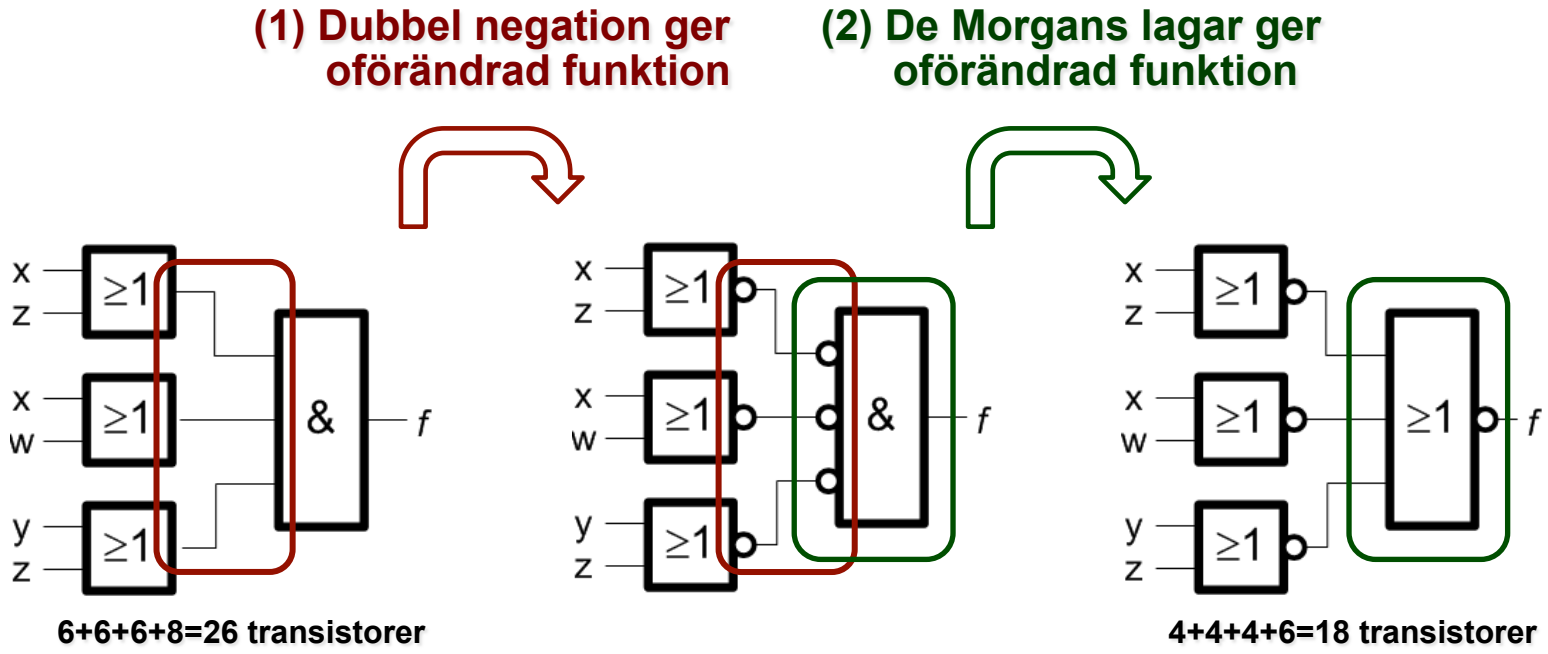
Med hjälp av regeln för dubbel negation (T9) och De Morgans lagar (T6) kan man enkelt konvertera ett AND/OR-nät till ett NAND/NAND-nät med ekvivalent funktion:



Kostnad för grindnät

4. Konvertering av OR/AND (PS-form) till NOR/NOR:

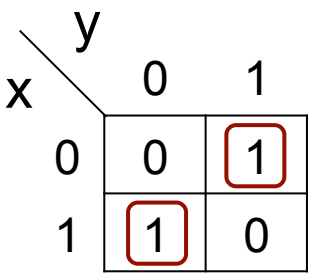
Med hjälp av regeln för dubbel negation (T9) och De Morgans lagar (T6) kan man enkelt konvertera ett OR/AND-nät till ett NOR/NOR-nät med ekvivalent funktion:



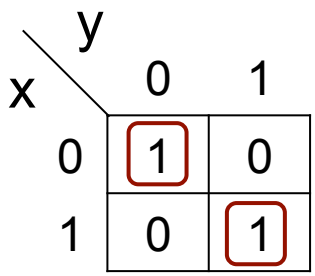
Kostnad för grindnät

5. Identifiering av XOR- och NXOR-funktion:

I Karnaughdiagram kan man ofta se följande diagonala mönster:



$$u = x \cdot \bar{y} + \bar{x} \cdot y$$



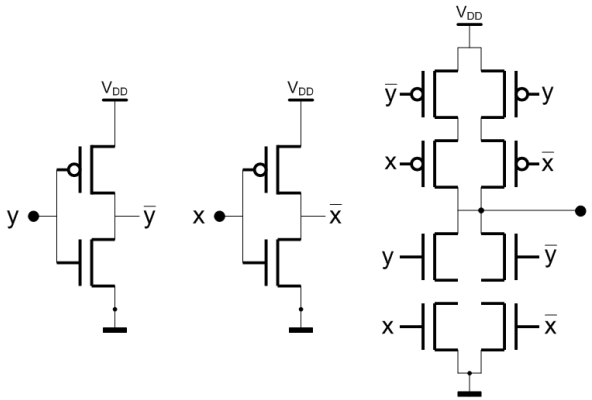
$$u = \bar{x} \cdot \bar{y} + x \cdot y$$

Att realisera endera av funktionerna i ett grindnät med enbart de grundläggande logikgrindarna (INV, OR, AND) skulle kräva fem grindar (2 INV, 1 OR, 2 AND) med totalt 2+2+6+6+6=22 transistorer

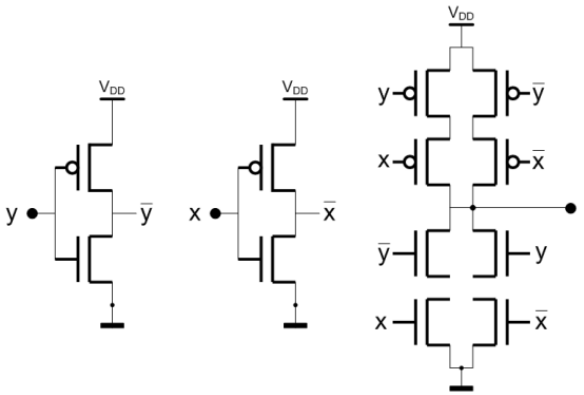
Kostnad för grindnät

5. Identifiering av XOR- och NXOR-funktion:

Det visar sig dock att endera funktion kan realiseras mer effektivt enligt följande lösningar med 12 transistorer vardera:



$$u = x \cdot \bar{y} + \bar{x} \cdot y$$

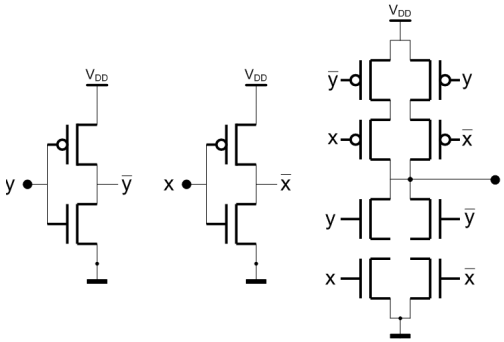


$$u = \bar{x} \cdot \bar{y} + x \cdot y$$

Dessa funktioner kallas för exklusivt ELLER (XOR) respektive icke exklusivt ELLER (NXOR), och är mycket användbara i olika sammanhang för att reducera kostnaden för ett grindnät.

Kostnad för grindnät

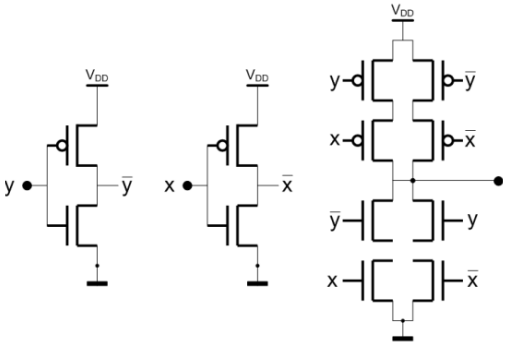
XOR (”exclusive OR”)



x	y	u
0	0	0
0	1	1
1	0	1
1	1	0

$$u = x \oplus y = x \cdot \overline{y} + \overline{x} \cdot y$$

NXOR (”negated XOR”)



x	y	u
0	0	1
0	1	0
1	0	0
1	1	1

$$u = \overline{x \oplus y} = \overline{x} \cdot \overline{y} + x \cdot y$$

Dessa logikgrindar kan ej ha fler ingångar än två. Genom att kaskadkoppla flera grindar kan man dock möjliggöra att fler än två signaler kopplas till nätet.

Kostnad för grindnät

6. Identifiering av ”don’t care”-termer:

I en funktionstabell eller ett Karnaughdiagram kan man ibland stöta på så kallade ”don’t care’-termer. Sådana anges som ’–’ i tabellen/diagrammet, och betyder att funktionens värde inte är av intresse (t ex på grund av att motsvarande kombination av invariabler aldrig kan förekomma).

xy \ zw	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	–	1	–	–
10	–	1	1	1

SP-form:

$$f = x\bar{z}w + \bar{x}yz + x\bar{y}z$$

Tips!
En ”don’t care”-term kan tolkas på valfritt sätt (1 eller 0), vilket gör det möjligt att hitta grindnät med lägre kostnad!

Kostnad för grindnät

6. Identifiering av ”don’t care”-termer:

I en funktionstabell eller ett Karnaughdiagram kan man ibland stöta på så kallade ”don’t care’-termer. Sådana anges som ’–’ i tabellen/diagrammet, och betyder att funktionens värde inte är av intresse (t ex på grund av att motsvarande kombination av invariabler aldrig kan förekomma).

xy \ zw	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	–	1	–	–
10	–	1	1	1

SP-form:

$$f = x + \bar{x}yz$$

Tips!
En ”don’t care”-term kan tolkas på valfritt sätt (1 eller 0), vilket gör det möjligt att hitta grindnät med lägre kostnad!

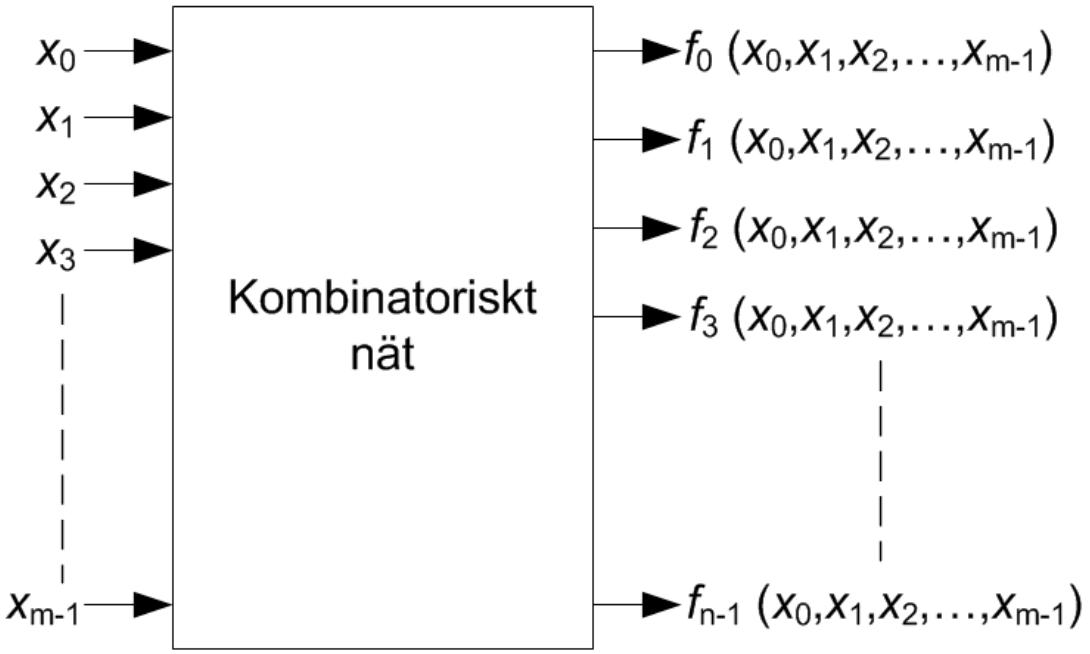
Kombinatoriska nät

Vad kännetecknar ett kombinatoriskt nät?

Ett kombinatoriskt nät är uppbyggt av logikgrindar, och varje utsignal är entydigt definierad av insignalernas värden.

I ett kombinatoriskt nät ändras utsignaler i samma ögonblick som dess insignaler ändras.

Ett kombinatoriskt nät saknar minne, d v s en utsignal beror inte på gamla värden på insignaler.



Kombinatoriska nät

Vad kännetecknar ett kombinatoriskt nät?

Några exempel på kombinatoriska nät är:

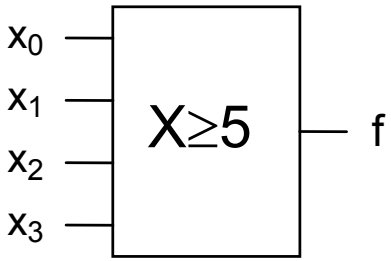
- Jämförare
- Kodomvandlare
- Omkodare
- Väljare (”multiplexer”)
- Fördelare (”de-multiplexer”)
- ALU (”arithmetic-logic unit”)

Flera av de ovanstående exemplen är viktiga byggblock i en dator, och vi återkommer snart till dessa i kursen.

Kombinatoriska nät

Demonstrationsexempel – jämförare:

Figuren gäller för ett kombinatoriskt nät som har utsignalen f = 1 om den NBCD-kodade siffran $X = (x_3x_2x_1x_0)_{NBCD}$ är större än eller lika med fem ($X \geq 5$).



För alla andra NBCD-kodade siffror skall gälla att f = 0.
Tal som inte tillhör NBCD-koden skall betraktas som "don't care".

Konstruera ett minimalt kombinatoriskt nät med NAND-grindar och INVERTERARE.