Computeren inderst inde

DM573

Rolf Fagerberg

Mål

Målet for disse slides er at beskrive bits, Boolsk algebra og gates, som er de basale byggesten, når man laver CPU'er.

Bits

Information = valg mellem forskellig muligheder.

Simpleste situation: valg mellem to muligheder. Kald dem 0 og 1. Et sådant valg kaldes en bit.

Mere information = valg mellem flere muligheder = brug flere bits:

1011001100111010

F.eks. 16 bits: valg mellem $2^{16} = 65.536$ muligheder.

Beregning: Ny information fra gammel. Dvs. nye bits fra gamle bits.

Computere

Fysisk repræsentation af bits i elektroniske computere:

- 0 = ingen strøm i ledning
- 1 = strøm i ledning

Computere er essentielt et (stort) system af ledninger og simpel elektronik, som kan repræsentere bits og som kan lave nye bits fra gamle.

Dagens emne: hvordan strukturerer man opbygningen af et sådant system?

Svar:

- Boolsk algebra: Regning med værdierne 0 og 1.
- Gates: Elektronik som implementerer simple beregninger fra Boolsk algebra. De basale byggesten i opbygning af computere.

Boolsk algebra

Opkaldt efter den engelske matematiker George Boole.

George Boole 1815-1864

Algebra

Almindelig regning med tal (almindelig "algebra"):

- Værdier er reelle tal: 2, 0, 3.1415, -13.5
- Simple beregninger: plus ($+$), minus ($-$), gange ($\cdot$), dividere ($/$), negering ($-$), ... Disse er funktioner fra tal til tal.
- Mere avancerede beregninger: sæt simple beregninger sammen, f.eks. $2 \cdot ((-3) + 4)$

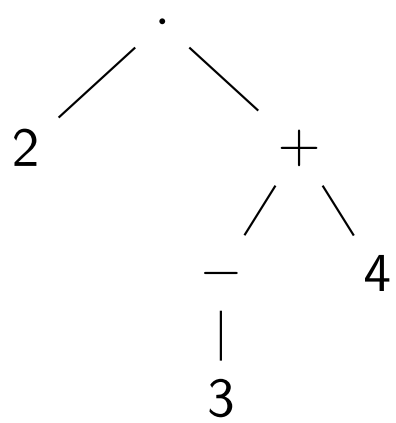
Boolsk algebra:

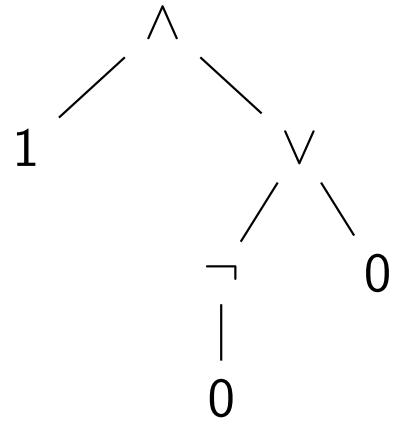
- Værdier er 0 og 1.
- Simple beregninger: NOT ($\neg$), OR ($\vee$), AND ($\wedge$),... Disse er funktioner fra $\{0, 1\}$ til $\{0, 1\}$.
- Mere avancerede beregninger: sæt simple beregninger sammen, f.eks. $1 \wedge ((\neg 0) \vee 0)$.

Alt, hvad vi mangler, er at definere de grundlæggende funktioner NOT ($\neg$), OR ($\vee$), AND ($\wedge$),... i den Boolske algebra.

Udtrykstræer

Bemærk, at både normale algebraiske udtryk og boolske udtryk kan beskrives som udtrykstræer, hvor beregningen går nedefra og op:





Input: tal
Output: tal

$2 \cdot ((-3) + 4)$

Input: $\{0, 1\}$
Output: $\{0, 1\}$

$1 \wedge ((\neg 0) \vee 0)$

Til daglig bruger vi parenteser til at lave en lineær beskrivelse af denne hierarkiske struktur.

Funktioner

Funktioner kan gives via regneforskrifter:

$$y = f(x_1, x_2) = 3x_1 + 5x_2 + 7$$

Funktioner kan også beskrives ved en tabel:

x_1	x_2	y
0	1	12
1	1	15
2	1	18
0	2	17
⋮	⋮	⋮

For almindelige tal er der uendeligt mange input, så en tabel vil altid være en ufuldstændig beskrivelse.

Men: i Boolsk algebra har hver input-variabel kun to mulige værdier. Derfor er der et endeligt antal forskellige input, så tabeller kan beskrive funktioner fuldstændigt.

Funktioner på bits

Beregning = ny information fra gammel. Dvs. nye bits fra gamle.

Dvs. en funktion fra bits til bits.

For én bit ind (x) og én bit ud (y) er alle mulighederne for en funktion:

x	y
0	0
1	0

Konstant 0

x	y
0	1
1	1

Konstant 1

x	y
0	0
1	1

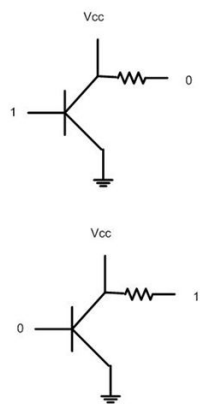
Identiteten

x	y
0	1
1	0

Negering (NOT)

[Der er 2 forskellige input (rækker), og derfor $2^2 = 4$ forskellige muligheder for valg af output (sidste søjle)]

Fact: De tre første er trivielle at lave med elektriske kredsløb (forbind output til jord, forbind output til strøm, forbind output til input). NOT kan laves med transistorer og andre simple elektroniske komponenter, som vist til højre.



NOT-gate

Tabel over NOTs funktion:

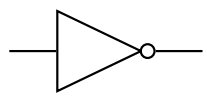
x	NOT(x)
0	1
1	0

Matematisk notation for NOT:

$$\neg 0 = 1; \quad \neg 1 = 0;$$

En kredsløbsdel, som implementerer NOT kaldes en NOT-*gate*.

Symbol for en NOT-gate i et kredsløb:



Beregning

Beregning = en funktion fra bits til bits.

For to bits ind (x_1 og x_2) og én bit ud (y) er mulighederne:

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	0

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	0

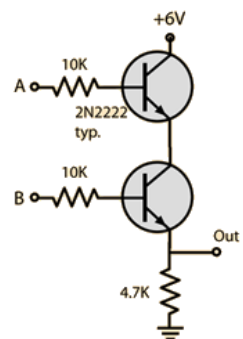
x_1	x_2	y
0	0	0
0	1	1
1	0	0
1	1	0

16 stk i alt .....

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	1

[Der er $2^2 = 4$ forskellige input (rækker), og derfor $2^4 = 16$ forskellige muligheder for valg af output (sidste søjle)]

Vi skal lære navnene (AND, OR, NAND, ...) for en del af disse. **Fact:** Også de kan laves med transistorer og andre simple elektroniske komponente. Her er f.eks. en AND:



AND-gate

Tabel over ANDs funktion:

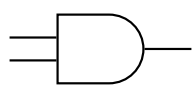
x_1	x_2	$\mathrm{AND}(x_1, x_2)$
0	0	0
0	1	0
1	0	0
1	1	1

Matematisk notation for AND:

$$0 \wedge 0 = 0;\ 0 \wedge 1 = 0;\ 1 \wedge 0 = 0;\ 1 \wedge 1 = 1;$$

En kredsløbsdel, som implementerer AND kaldes en AND-*gate*.

Symbol for en AND-gate i et kredsløb:



OR-gate

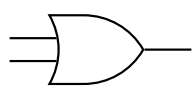
Tabel over ORs funktion:

x_1	x_2	$\mathrm{OR}(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	1

Matematisk notation for OR:

$$0 \vee 0 = 0;\ 0 \vee 1 = 1;\ 1 \vee 0 = 1;\ 1 \vee 1 = 1;$$

Symbol for en OR-gate i et kredsløb:



XOR-gate

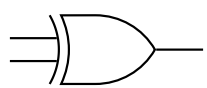
Tabel over XORs funktion:

x_1	x_2	$\text{XOR}(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

Matematisk notation for XOR:

$$0 \oplus 0 = 0;\ 0 \oplus 1 = 1;\ 1 \oplus 0 = 1;\ 1 \oplus 1 = 0;$$

Symbol for en XOR-gate i et kredsløb:



NAND-gate

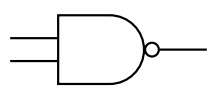
Tabel over NANDs funktion:

x_1	x_2	NAND(x_1, x_2)
0	0	1
0	1	1
1	0	1
1	1	0

Der er ikke en etableret matematisk notation for NAND. Vi bruger:

$$0 \text{ nand } 0 = 1;\ 0 \text{ nand } 1 = 1;\ 1 \text{ nand } 0 = 1;\ 1 \text{ nand } 1 = 0;$$

Symbol for en NAND-gate i et kredsløb:



NOR-gate

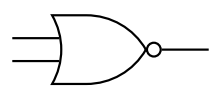
Tabel over NORs funktion:

x_1	x_2	NOR(x_1, x_2)
0	0	1
0	1	0
1	0	0
1	1	0

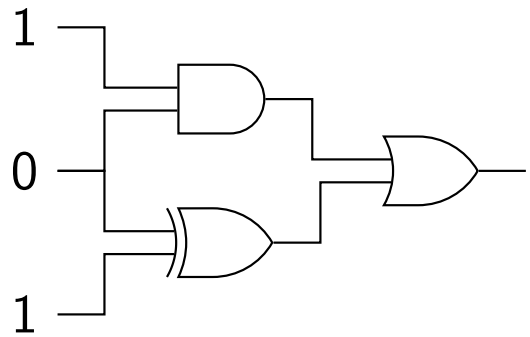
Der er ikke en etableret matematisk notation for NOR. Vi bruger:

$$0 \text{ nor } 0 = 1;\ 0 \text{ nor } 1 = 0;\ 1 \text{ nor } 0 = 0;\ 1 \text{ nor } 1 = 0;$$

Symbol for en NOR-gate i et kredsløb:



Eksempel på kredsløb

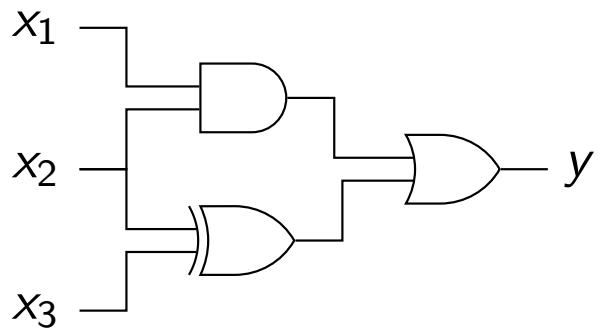


Hvad er gates? Gates er: AND, XOR og OR.

Hvad er output? Output er: 1.

Eksempel på kredsløb

Samme kredsløb set som generel funktion:



Udfyld tabel:

x_1	x_2	x_3	y
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

Svar:
Kredsløbet beregner det matematiske udtryk $(x_1 \wedge x_2) \vee (x_2 \oplus x_3)$.

x_1	x_2	x_3	y
0	0	0	0

Alle tabeller

Vi viser nu, at man alene med AND, OR og NOT kan lave kredsløb som beregner enhver ønsket tabel (dvs. enhver ønsket beregning):

1. Find de rækker (input) i tabellen, hvor output y er lig 1.
2. For hver række (input) brug AND og NOT til at lave et kredsløb, som giver output 1 for dette og kun dette input (se eksemplet på næste side for metoden).
3. Sæt disse kredsløb sammen med en masse OR.

Det resulterende kredsløb giver output 1 præcis for de rækker (input), hvor tabellen har 1 som output.

Et eksempel gives på næste side, med fire inputs x_1, x_2, x_3 og x_4. Det udtrykkes med matematisk notation, men kan nemt konverteres til et kredsløb med fire input ledninger x_1, x_2, x_3 og x_4 (brug AND-gates for $\wedge$, NOT-gates for $\neg$ og OR-gates for $\vee$).

Eksempel

I følgende tabel er der to rækker, hvor output y er lig 1:

x_1	x_2	x_3	x_4	y	
0	0	0	0	0	
1	0	0	0	0	
⋮	⋮	⋮	⋮	0	
0	1	1	0	1	←
⋮	⋮	⋮	⋮	0	
1	0	1	0	1	←
⋮	⋮	⋮	⋮	0	

Følgende to udtryk har værdi 1 præcis når input (x_1, x_2, x_3, x_4) er $(0, 1, 1, 0)$, henholdsvis $(1, 0, 1, 0)$:

$$((\neg x_1) \wedge x_2) \wedge (x_3 \wedge (\neg x_4))$$
$$(x_1 \wedge (\neg x_2)) \wedge (x_3 \wedge (\neg x_4))$$

Derfor har følgende udtryk værdi 1 præcis for de input, hvor tabellen har 1 i output:

$$(((\neg x_1) \wedge x_2) \wedge (x_3 \wedge (\neg x_4))) \vee ((x_1 \wedge (\neg x_2)) \wedge (x_3 \wedge (\neg x_4)))$$

Boolske funktioner og kredsløb

Opsummering:

- Boolske funktioner kan beskrives ved tabeller.
- Kredsløb bygget op af gates (og med input- og outputledninger) kan ofte[1] ses som implementerende Boolske funktioner.
- *Enhver* Boolsk funktion kan implementeres via et kredsløb.

[1] Se de næste sider for flip-flops, som ikke så nemt kan siges at implementere en Boolsk funktion.

Boolske funktioner og kredsløb

Fact: De basale operationer (plus, gange, division, invertér bits, . . .) i en computer kan beskrives via Boolske funktioner og derfor implementeres via kredsløb bygget på af gates.

F.eks. kan addition

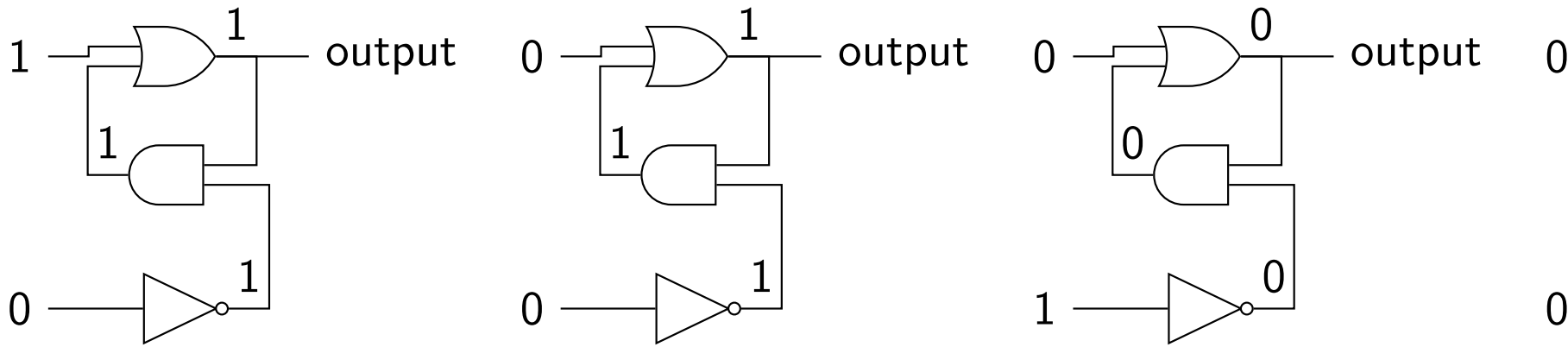
$$\begin{array}{r} {\scriptstyle 1\,1\,1} \\ 101010_2 \\ +001111_2 \\ \hline = 111001_2 \end{array}$$

på hver ciffer-position beskrives ved to funktioner

- $\textsc{Resultat}(x_1,x_2,x_3)$, som ud fra de to input-cifre x_1 og x_2 samt menten x_3 på denne plads giver output-cifferet på denne plads.
- $\textsc{Mente}(x_1,x_2,x_3)$, som ud fra de to input-cifre x_1 og x_2 samt menten x_3 på denne plads giver menten på næste plads.

[Menten på første plads er altid 0.]

Flip flop



Aktiver (sæt til 1) øverste input $\Rightarrow$ output bliver 1.

Bemærk at output er stabilt.

Aktiver (sæt til 1) nederste input $\Rightarrow$ output bliver 0.

Bemærk at output er stabilt.

Alt i alt: vi kan lave kredsløb som kan *gemme* en ønsket bit (sålænge der er strøm). Dvs. at vi kan lave hukommelsesceller.

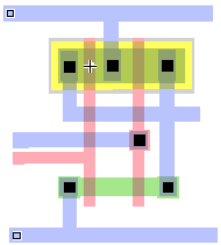
CPU'er

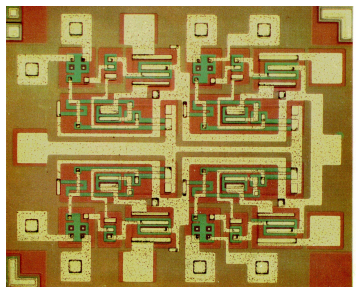
Opsummering: Med gates kan vi

- Lave enhver funktion fra bits til bits.
- Lave hukommelsesceller, der kan gemme bits

Gates bruges derfor som de basale byggeklodser til at bygge CPU'er, og Boolsk algebra bruges til at strukturere byggeprocessen.

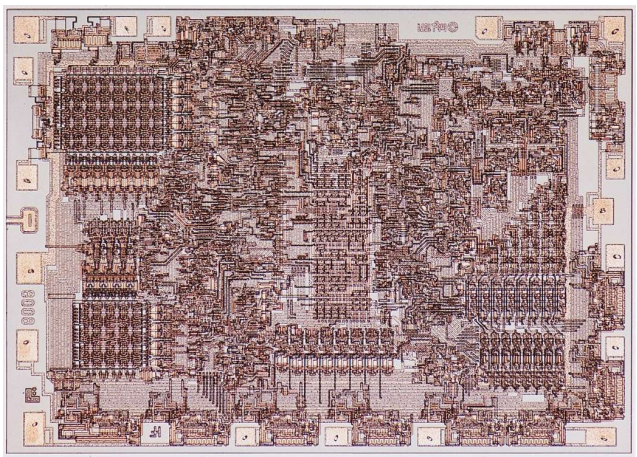
"Ledningerne" for en enkelt transistor på en printplade.





4-vejs NAND-gate, 7 transistorer. Anden halvdel af 1960erne.

Intel 8008 CPU, 3098 transistorer, 1972. Printplade vist.



Intel Core i7-6950X, 3.4 mio transistorer, 2016. For- og bagside vist.