

# Computeren inderst inde

DM534

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.

Dette emne er et uddrag af kurset *DM548 Computerarkitektur og systemprogrammering* (3. semester).

# Bits

Information = valg mellem forskellige muligheder.

Simpleste situation: valg mellem to muligheder. Kald dem 0 og 1. Denne valgmulighed kaldes en **bit**.

# Bits

**Information** = valg mellem forskellig muligheder.

Simpleste situation: valg mellem to muligheder. Kald dem 0 og 1. Denne valgmulighed kaldes en **bit**.

Større samling information: brug flere bits:

1011001100111010

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

# Bits

**Information** = valg mellem forskellige muligheder.

Simpleste situation: valg mellem to muligheder. Kald dem 0 og 1. Denne valgmulighed kaldes en **bit**.

Større samling information: 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.

# 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?

# 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. Den basale byggesten i opbygning af computere.



# Algebra

# 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 ( $-$ ), ...
- ▶ Mere avancerede beregninger: sæt simple beregninger sammen, f.eks.  $2 \cdot (3 + (-4))$

# 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 ( $-$ ), ...
- ▶ 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 ( $\wedge$ ), ...
- ▶ Mere avancerede beregninger: sæt simple beregninger sammen, f.eks.  $0 \wedge (1 \vee (\neg 0))$

# Funktioner

Funktioner kan gives via regneforskrifter:

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

# 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
$\vdots$	$\vdots$	$\vdots$

# 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
$\vdots$	$\vdots$	$\vdots$

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

# 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
$\vdots$	$\vdots$	$\vdots$

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.



# 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)

# 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)]

# 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

$x$	$y$
0	1
1	1

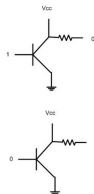
$x$	$y$
0	0
1	1

$x$	$y$
0	1
1	0

Konstant 0    Konstant 1    Identiteten    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

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

Symbol for en NOT-gate i et kredsløb:



Tabel over NOTs funktion:

$x$	$\text{NOT}(x)$
0	1
1	0

Matematisk notation (Boolsk udtryk) for NOT:

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

# Beregning

Beregning = en funktion fra bits til bits.

# 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

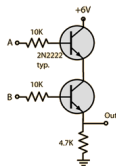
$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	0
1	1	0

16  
stk  
.....

$x_1$	$x_2$	$y$
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

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

Symbol for en AND-gate i et kredsløb:



Tabel over ANDs funktion:

$x_1$	$x_2$	AND( $x_1, x_2$ )
0	0	0
0	1	0
1	0	0
1	1	1

Matematisk notation (Boolsk udtryk) for AND:

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

# OR-gate

Symbol for en OR-gate i et kredsløb:



Tabel over ORs funktion:

$x_1$	$x_2$	OR( $x_1, x_2$ )
0	0	0
0	1	1
1	0	1
1	1	1

Matematisk notation (Boolsk udtryk) for OR:

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



## XOR-gate

Symbol for en XOR-gate i et kredsløb:



Tabel over XORs funktion:

$x_1$	$x_2$	XOR( $x_1, x_2$ )
0	0	0
0	1	1
1	0	1
1	1	0

Matematisk notation (Boolsk udtryk) for XOR:

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

# NAND-gate

Symbol for en NAND-gate i et kredsløb:



Table over NANDs funktion:

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

Matematisk notation (Boolsk udtryk) for NAND:

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

# NOR-gate

Symbol for en NOR-gate i et kredsløb:



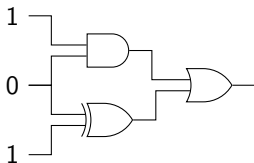
Tabel over NORs funktion:

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

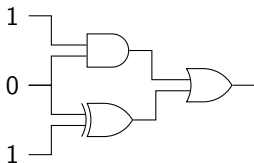
Matematisk notation (Boolsk udtryk) for NOR:

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

## Eksempel på kredsløb

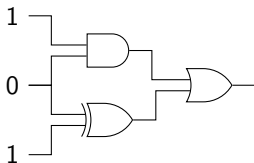


## Eksempel på kredsløb



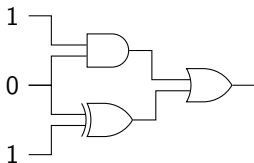
Hvad er gates?

## Eksempel på kredsløb



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

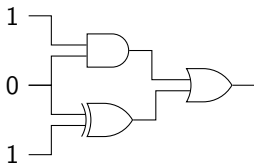
## Eksempel på kredsløb



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

Hvad er output?

## Eksempel på kredsløb



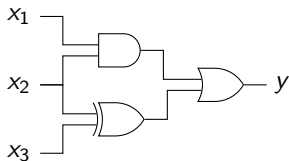
Hvad er gates? Gates er: AND, XOR og OR.

Hvad er output? Output er: 1.



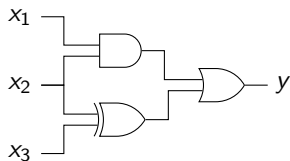
## Eksempel på kredsløb

Samme kredsløb set som generel funktion:



## Eksempel på kredsløb

Samme kredsløb set som generel funktion:

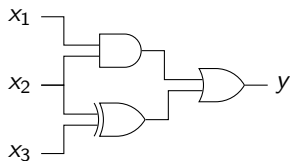


Udfyld tabel:

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

## Eksempel på kredsløb

Samme kredsløb set som generel funktion:



Udfyld tabel:

$x_1$	$x_2$	$x_3$	$y$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

# Alle tabeller

Bemærk 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  $y$  (output) 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.

# Alle tabeller

Bemærk 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  $y$  (output) 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 med output ( $y$ ) lig 1:

$x_1$	$x_2$	$x_3$	$x_4$	$y$	
0	0	0	0	0	
1	0	0	0	0	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	0	
0	1	1	0	1	←
$\vdots$	$\vdots$	$\vdots$	$\vdots$	0	
1	0	1	0	1	←
$\vdots$	$\vdots$	$\vdots$	$\vdots$	0	

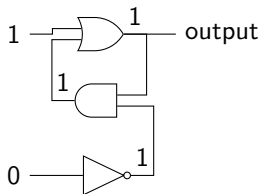
Følgende to udtryk har værdi 1 præcis når input  $(x_1, x_2, x_3, x_4)$  er  $(0, 1, 1, 0)$ , henholdsvis  $(1, 0, 1, 0)$ :

$$\begin{aligned} & ((\neg x_1) \wedge x_2) \wedge (x_3 \wedge (\neg x_4)) \\ & (x_1 \wedge (\neg x_2)) \wedge (x_3 \wedge (\neg x_4)) \end{aligned}$$

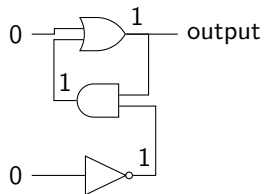
Derfor har følgende udtryk værdi 1 præcis for de input hvor tabellen har 1 i output:

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

# Flip flop



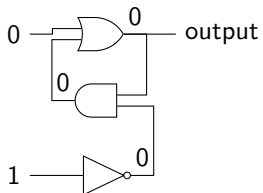
## Flip flop



Bemærk at output er stabilt.



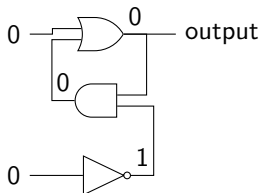
# Flip flop



Bemærk at output er stabilt.

Vi kan ændre output.

# Flip flop

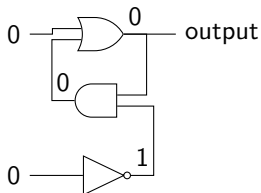


Bemærk at output er stabilt.

Vi kan ændre output.

Bemærk at output er stabilt.

# Flip flop



Bemærk at output er stabilt.

Vi kan ændre output.

Bemærk at output er stabilt.

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

# CPU'er

Pointe: 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.

