Lektion 1

Kombinatoriske Logiske Kredse

Emil Lykke Diget

Computerarkitektur og Operativsystemer
Syddansk Universitet

Introduction

Computerarkitektur og Operativsystemer

Viden

- **Kombinatoriske logiske kredse**
- Memorytyper
- Memoryinterface incl. timing
- Adressedekodning
- Interrupt og exceptions
- Computerdesign
- Register-transfer level
- Datapath
- Control unit
- Instruktionssæt
- Pipeline
- Cache
- Processer og tråde
- Context switch
- Inter-process synkronisering og kommunikation, kritiske sektorer og semaphores

Færdigheder

- Redegøre for principperne og algoritmerne bag operativsystemets centrale funktioner
- **Forstå opbygningen af en moderne CPU**
- Kende de almindeligt forekomne memorytyper
- Forstå centrale begreber omkring et operativsystems afvikling af et program

Kompetencer

- Implementere operativsystemsfunktioner i et RTOS (Real Time Operating System)

- Lektion 1: Kombinatoriske Logiske Kredse
- Lektion 2: En Simpel Computer
- Lektion 3: Hukommelse
- Lektion 4: Mikroarkitektur
- Lektion 5: Micro-assembly og IJVM
- Lektion 6: Optimering af Mikroarkitekturdesign

Lagdelt Computermodel

Level 5	High Level Language (e.g., C, C++)	Compiler
Level 4	Assembly Language	Assembler
Level 3	Operating System	Partial Interpretation
Level 2	Instruction Set Architecture (ISA)	Interpretation
Level 1	Micro Architecture	Hardware
→ Level 0	Digital Logic	

Overblik

En computer er opbygget af **kombinatoriske logiske kredse**, der styres af signaler, der kan have to tilstande: Høj (1) og lav (0).

Derfor skal vi gennemgå talsystemer, og mere specifikt det **binære talsystem**.

Dagsorden

- Talsystemer
  - Hexadecimal
  - Konvertering
  - Negative Binære Tal
- Digitale Kredsløb
- Boolsk Algebra
- ALU
  - Decoder
  - Logisk Enhed
  - Full Adder
  - 8-bit ALU
- Opsummering

Indhold

Titalssystemet kender vi alle sammen.

Hvert ciffer kan være 0-9.

Det består af en række decimaler og måske et punktum.

$$\begin{array}{ccccccccccc} & & 100\text{ere} & 10\text{ere} & 1\text{ere} & & \frac{1}{10}\text{ere} & \frac{1}{100}\text{ere} & \frac{1}{1000}\text{ere} & & \\ d_n & \dots & d_2 & d_1 & d_0 & \cdot & d_{-1} & d_{-2} & d_{-3} & \dots & d_{-\eta} \end{array}$$

$$\text{Tal} = \sum_{i=-\eta}^{n} d_i \cdot k^i \tag{1}$$

hvor radix $k = 10$.

Titalssystemet

Titalssystemet kender vi alle sammen.

Hvert ciffer kan være 0-9.

Det består af en række decimaler og måske et punktum.

$$\begin{array}{ccccccccccc} & & 100\text{ere} & 10\text{ere} & 1\text{ere} & & \frac{1}{10}\text{ere} & \frac{1}{100}\text{ere} & \frac{1}{1000}\text{ere} & & \\ d_n & \dots & d_2 & d_1 & d_0 & . & d_{-1} & d_{-2} & d_{-3} & \dots & d_{-\eta} \end{array}$$

$$\text{Tal} = \sum_{i=-\eta}^{n} d_i \cdot k^i = \sum_{i=-\eta}^{n} d_i \cdot 10^i, \tag{1}$$

hvor radix $k = 10$.

Tallet 1337 i decimaltal ($k = 10$).

d_3		d_2		d_1		d_0		
1		3		3		7		
$1 \cdot 10^3$	$+$	$3 \cdot 10^2$	$+$	$3 \cdot 10^1$	$+$	$7 \cdot 10^0$		
1000	$+$	300	$+$	30	$+$	7	$=$	1337

De Mest Nødvendige Talsystemer

Når man arbejder med computere, er det ofte brugbart at regne med følgende talsystemer:

- $k = 2$, binary
- $k = 8$, octal
- $k = 16$, hexadecimal

$$\text{Tal} = \sum_{i=-\eta}^{n} d_i \cdot k^i$$

Binary

Radix $k = 2$; to cifre

$$0 \quad 1$$

Et ciffer kaldes en **bit**.

$$\text{Tal} = \sum_{i=-\eta}^{n} d_i \cdot 2^i \tag{2}$$

Binary

Radix $k = 2$; to cifre

$$0 \quad 1$$

Tallet 1337_{10} til binær.

Et ciffer kaldes en **bit**.

$$\text{Tal} = \sum_{i=-\eta}^{n} d_i \cdot 2^i \tag{2}$$

Binary

Et ciffer kaldes en **bit**.

Radix $k = 2$; to cifre

$$0 \quad 1$$

$$\text{Tal} = \sum_{i=-\eta}^{n} d_i \cdot 2^i \tag{2}$$

Tallet 1337_{10} til binær.

d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	
1	0	1	0	0	1	1	1	0	0	1	
$1 \cdot 2^{10}$	$+0 \cdot 2^9$	$+1 \cdot 2^8$	$+0 \cdot 2^7$	$+0 \cdot 2^6$	$+1 \cdot 2^5$	$+1 \cdot 2^4$	$+1 \cdot 2^3$	$+0 \cdot 2^2$	$+0 \cdot 2^1$	$+1 \cdot 2^0$	
$1024 +$	0	$+ 256 +$	0	$+ \; 0$	$+ \; 32$	$+ \; 16$	$+ \; 8$	$+ \; 0$	$+ \; 0$	$+ \; 1$	$=1337$

10100111001_2 eller 0b10100111001.

Oktal

Radix $k = 8$; otte cifre

0 1 2 3 4 5 6 7

$$\text{Tal} = \sum_{i=-\eta}^{n} d_i \cdot 8^i \tag{3}$$

Oktal

Radix $k = 8$; otte cifre

$$\text{Tal} = \sum_{i=-\eta}^{n} d_i \cdot 8^i \tag{3}$$

0 1 2 3 4 5 6 7

Tallet 1337 til oktal.

d_3		d_2		d_1		d_0		
2		4		7		1		
$2 \cdot 8^3$	$+$	$4 \cdot 8^2$	$+$	$7 \cdot 8^1$	$+$	$1 \cdot 8^0$		
1024	$+$	256	$+$	56	$+$	1	$=$	1337

2471_8 eller 02471.

Hexadecimal

Radix $k = 16$; seksten cifre

0	1	2	3	4	5	6	7
8	9	A	B	C	D	E	F

$$\text{Tal} = \sum_{i=-\eta}^{n} d_i \cdot 16^i \tag{4}$$

Hexadecimal

Radix $k = 16$; seksten cifre

$$\text{Tal} = \sum_{i=-\eta}^{n} d_i \cdot 16^i \tag{4}$$

0	1	2	3	4	5	6	7
8	9	A	B	C	D	E	F

Tallet 1337 til hexadecimal.

d_2		d_1		d_0		
5		3		9		
$5 \cdot 16^2$	$+$	$3 \cdot 16^1$	$+$	$9 \cdot 16^0$		
1280	$+$	48	$+$	9	$=$	1337

539_{16} eller 0x539.

Konvertering

Konvertering mellem oktal, hexadecimal og binær er simpel.

Fra binær til oktal:

$$\underbrace{010}\ \underbrace{100}\ \underbrace{111}\ \underbrace{001}$$

Saml tre bits sammen fra højre og konverter til oktal ciffer.

Konvertering

Konvertering mellem oktal, hexadecimal og binær er simpel.

Fra binær til oktal:

$$\underbrace{010}_{2}\ \underbrace{100}_{4}\ \underbrace{111}_{7}\ \underbrace{001}_{1}$$

Saml tre bits sammen fra højre og konverter til oktal ciffer.

Konvertering

Konvertering mellem oktal, hexadecimal og binær er simpel.

Fra binær til oktal:

$$\underbrace{010}_{2}\ \underbrace{100}_{4}\ \underbrace{111}_{7}\ \underbrace{001}_{1}$$

Saml tre bits sammen fra højre og konverter til oktal ciffer.

Fra binær til hexadecimal:

$$\underbrace{0101}\ \underbrace{0011}\ \underbrace{1001}$$

Saml fire bits sammen fra højre og konverter til hexadecimal ciffer.

Konvertering

Konvertering mellem oktal, hexadecimal og binær er simpel.

Fra binær til oktal:

$$\underbrace{010}_{2}\ \underbrace{100}_{4}\ \underbrace{111}_{7}\ \underbrace{001}_{1}$$

Saml tre bits sammen fra højre og konverter til oktal ciffer.

Fra binær til hexadecimal:

$$\underbrace{0101}_{5}\ \underbrace{0011}_{3}\ \underbrace{1001}_{9}$$

Saml fire bits sammen fra højre og konverter til hexadecimal ciffer.

Konvertering

Konvertering mellem oktal, hexadecimal og binær er simpel.

Fra binær til oktal:

$$\underbrace{010}_{2}\underbrace{100}_{4}\underbrace{111}_{7}\underbrace{001}_{1}$$

Saml tre bits sammen fra højre og konverter til oktal ciffer.

Fra binær til hexadecimal:

$$\underbrace{0101}_{5}\underbrace{0011}_{3}\underbrace{1001}_{9}$$

Saml fire bits sammen fra højre og konverter til hexadecimal ciffer.

Konvertering den anden vej er lige så simpel.

Algoritme til Konvertering fra Decimal til Binær

```
function dec2bin(int number)
{
    if number == 0
    {
        return '0';
    }
    else
    {
        return dec2bin(floor(number / 2)) \
            + str(mod(number, 2));
    }
}
```

Negative Binære Tal

Indtil videre kan vi repræsentere positive tal i binær. Men hvad med de negative?
Fire forskellige systemer. Vi kigger på et 8-bit tal.

$$N = 00110010_2 = 50_{10}$$

- Signed magnitude
  Bittet længst til venstre er fortegnsbit (0 er + og 1 er -).
  Resten repræsenterer den absolutte værdi af tallet.

$$-N = 10110010$$

Negative Binære Tal

Indtil videre kan vi repræsentere positive tal i binær. Men hvad med de negative?
Fire forskellige systemer. Vi kigger på et 8-bit tal.

$$N = 00110010_2 = 50_{10}$$

- Et-komplement
  Bittet længst til venstre er fortegnsbit (0 er + og 1 er -).
  Erstat 0 med 1 og 1 med 0 i alle 8.

$$-N = 11001101$$

Negative Binære Tal

Indtil videre kan vi repræsentere positive tal i binær. Men hvad med de negative?
Fire forskellige systemer. Vi kigger på et 8-bit tal.

$$N = 00110010_2 = 50_{10}$$

- To-komplement
  Bittet længst til venstre er fortegnsbit (0 er + og 1 er -).
  Negering kræver to skridt.
  1. Erstat alle 0 med 1 og 1 med 0.
  2. Læg 1 til resultatet.

$$-N = 11001110$$

Negative Binære Tal

Indtil videre kan vi repræsentere positive tal i binær. Men hvad med de negative?
Fire forskellige systemer. Vi kigger på et 8-bit tal.

$$N = 00110010_2 = 50_{10}$$

- Excess $2^{m-1} = 128$
  Repræsenterer et tal som summen af tallet selv og $2^{m-1} = 2^7 = 128$.
  Svarer til to-komplement med inverteret MSB.

$$-N = 01001110$$

Negative Tal
Sammenligning

Decimal	Unsigned	Sign-magnitude	Ones' complement	Two's complement	Excess-8 (biased)
15	1111	–	–	–	–
14	1110	–	–	–	–
13	1101	–	–	–	–
12	1100	–	–	–	–
11	1011	–	–	–	–
10	1010	–	–	–	–
9	1001	–	–	–	–
8	1000	–	–	–	–
7	0111	0111	0111	0111	1111
6	0110	0110	0110	0110	1110
5	0101	0101	0101	0101	1101
4	0100	0100	0100	0100	1100
3	0011	0011	0011	0011	1011
2	0010	0010	0010	0010	1010
1	0001	0001	0001	0001	1001
0	0000	0000	0000	0000	1000
-0		1000	1111		
-1	–	1001	1110	1111	0111
-2	–	1010	1101	1110	0110
-3	–	1011	1100	1101	0101
-4	–	1100	1011	1100	0100
-5	–	1101	1010	1011	0011
-6	–	1110	1001	1010	0010
-7	–	1111	1000	1001	0001
-8	–	–	–	1000	0000

Indhold

Digitale Kredsløb

I et digitalt kredsløb kan et signal kun antage to værdier: 0 og 1.
Binært tal!

Gates

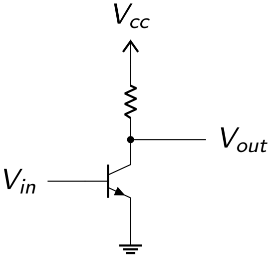
En gate kan beregne forskellige funktioner med binære input.

På elektronik-niveau bliver der brugt en transistor.

Gates

En gate kan beregne forskellige funktioner med binære input.

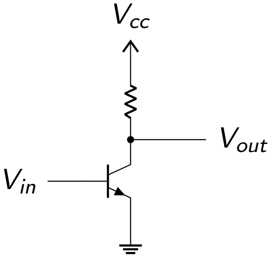
På elektronik-niveau bliver der brugt en transistor.



Gates

En gate kan beregne forskellige funktioner med binære input.

På elektronik-niveau bliver der brugt en transistor.

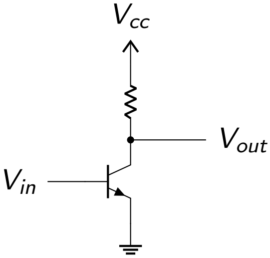


Figur: Inverter, NOT

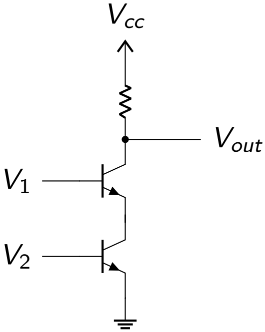
Gates

En gate kan beregne forskellige funktioner med binære input.

På elektronik-niveau bliver der brugt en transistor.



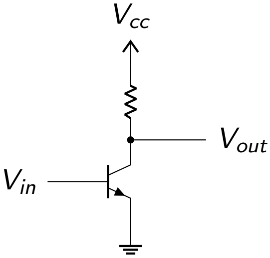
Figur: Inverter, NOT



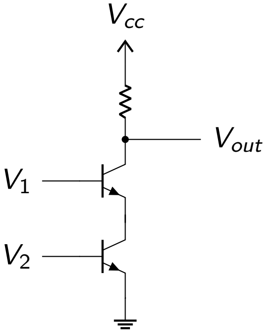
Gates

En gate kan beregne forskellige funktioner med binære input.

På elektronik-niveau bliver der brugt en transistor.



Figur: Inverter, NOT

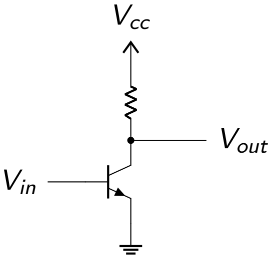


Figur: NAND gate

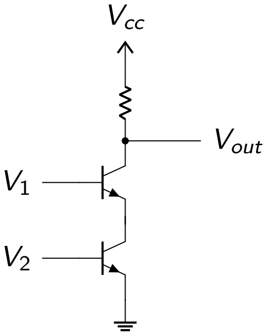
Gates

En gate kan beregne forskellige funktioner med binære input.

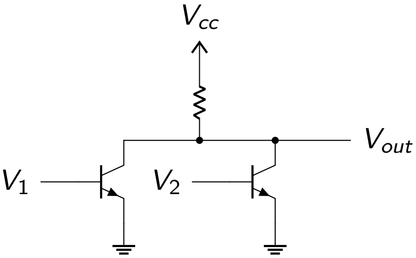
På elektronik-niveau bliver der brugt en transistor.



Figur: Inverter, NOT

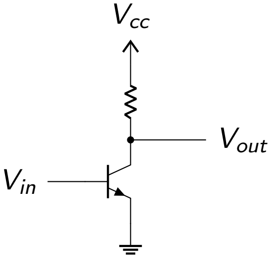


Figur: NAND gate

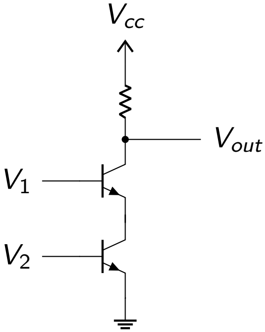


En gate kan beregne forskellige funktioner med binære input.

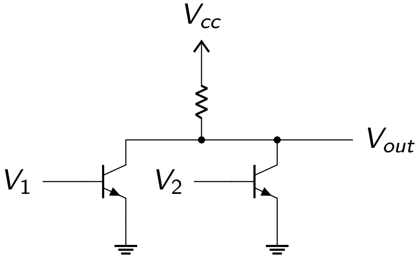
På elektronik-niveau bliver der brugt en transistor.



Figur: Inverter, NOT



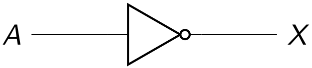
Figur: NAND gate



Figur: NOR gate

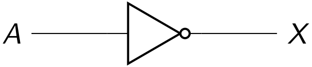
Basisgates

NOT



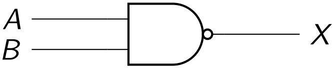
A	X
0	1
1	0

NOT



A	X
0	1
1	0

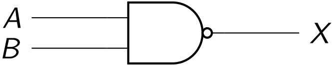
NAND



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

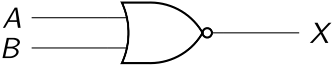
Basisgates

NAND



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

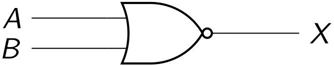
NOR



A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

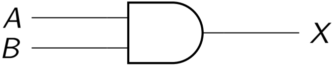
Basisgates

NOR



A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

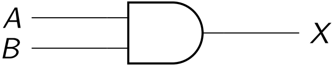
AND



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

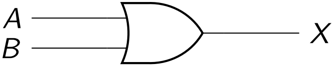
Basisgates

AND



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

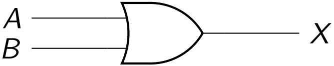
OR



A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

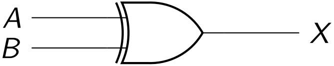
Basisgates

OR



A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

XOR



A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Indhold

Boolsk Algebra

Algebra til at beskrive logiske kredsløb.
En boolsk funktion har en eller flere input- og output-variabler. Alle værdier er binære.

$$X = f(A, B) \tag{5}$$

$X = 0$ når $A = 0, B = 0$, $\quad X = 1$ når $A = 0, B = 1$,
$X = 1$ når $A = 1, B = 0$ og $X = 1$ når $A = 1, B = 1$.
Hvilken gate?

Boolsk Algebra

Algebra til at beskrive logiske kredsløb.
En boolsk funktion har en eller flere input- og output-variabler. Alle værdier er binære.

$$X = f(A, B) \tag{5}$$

$X = 0$ når $A = 0, B = 0,$ $\quad X = 1$ når $A = 0, B = 1,$
$X = 1$ når $A = 1, B = 0$ $\quad$ og $\quad X = 1$ når $A = 1, B = 1.$
Hvilken gate? OR.

Boolsk Algebra

Algebra til at beskrive logiske kredsløb.

En boolsk funktion har en eller flere input- og output-variabler. Alle værdier er binære.

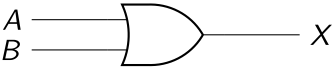
$$X = f(A, B) \tag{5}$$

$X = 0$ når $A = 0, B = 0,$ $X = 1$ når $A = 0, B = 1,$
$X = 1$ når $A = 1, B = 0$ og $X = 1$ når $A = 1, B = 1.$

Hvilken gate? OR.

En boolsk funktion med n input kan beskrives af en **sandhedstabel** med 2^n rækker.

OR



A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Matematiske symboler bruges i boolske ligninger.
Addition $+$ er OR.
Multiplikation $\cdot$ er AND.
En streg over $\overline{A}$ er den inverse, NOT.
På den måde kan man gå fra en sandhedstabel til en ligning.

Matematiske symboler bruges i boolske ligninger.
Addition $+$ er OR.
Multiplikation $\cdot$ er AND.
En streg over $\overline{A}$ er den inverse, NOT.
På den måde kan man gå fra en sandhedstabel til en ligning.

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Matematiske symboler bruges i boolske ligninger.
Addition $+$ er OR.
Multiplikation $\cdot$ er AND.
En streg over $\overline{A}$ er den inverse, NOT.
På den måde kan man gå fra en sandhedstabel til en ligning.

$$X = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Tabel Over Boolske Identiteter

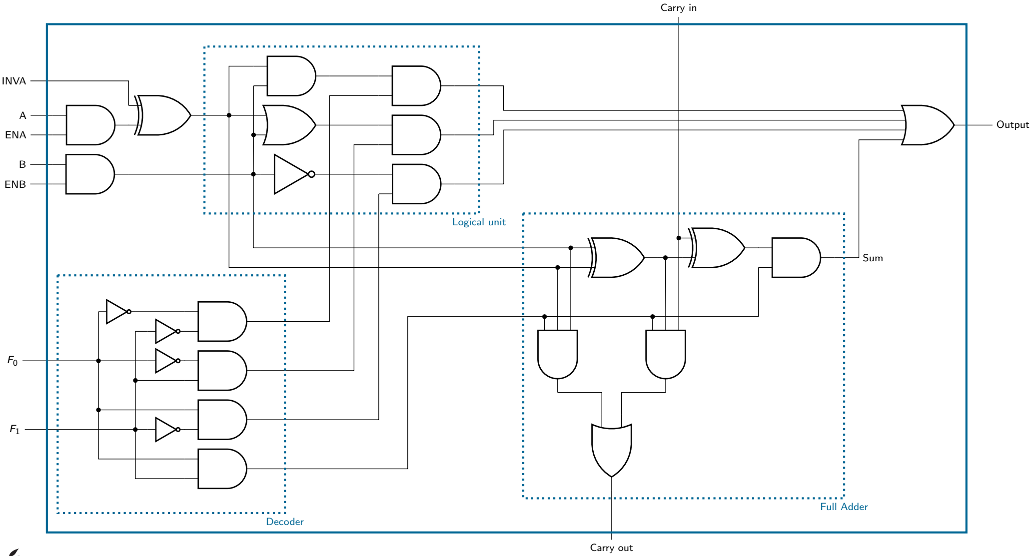
Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\ \overline{B}$

Indhold

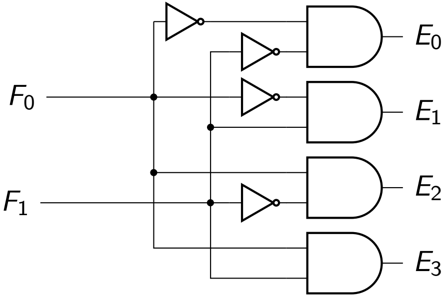
En computer skal kunne regne.
Til det benyttes et kredsløb kaldet en **Arithmetic Logic Unit** (ALU).
Fire funktioner

F_0	F_1	out
0	0	A AND B
0	1	A OR B
1	0	$\overline{B}$
1	1	$A + B$

1-bit ALU-kredsløb



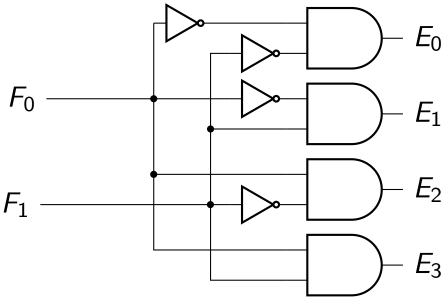
Decoder



Decoder

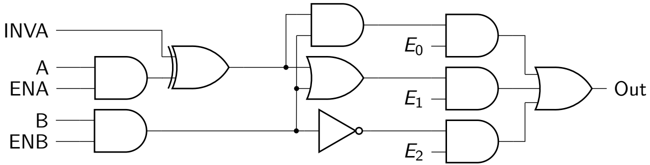
Funktion: Vælg et output vha. input-adresse.
Et output højt og de andre output lave.

F_0	F_1	E_3	E_2	E_1	E_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



Logisk Enhed (Logical Unit)

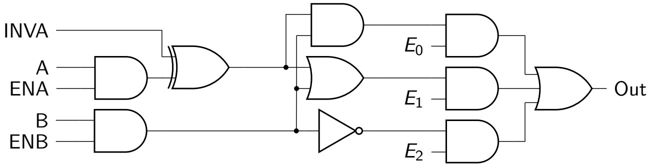
Antag $\text{INVA} = 0$, $\text{ENA} = 1$, $\text{ENB} = 1$.



Logisk Enhed (Logical Unit)

Antag INVA $= 0$, ENA $= 1$, ENB $= 1$.
Forskellig funktion afhængig af E.

A	B	E_2	E_1	E_0	Out
0	0	0	0	1	0
0	1	0	0	1	0
1	0	0	0	1	0
1	1	0	0	1	1
0	0	0	1	0	0
0	1	0	1	0	1
1	0	0	1	0	1
1	1	0	1	0	1
0	0	1	0	0	1
0	1	1	0	0	0
1	0	1	0	0	1
1	1	1	0	0	0

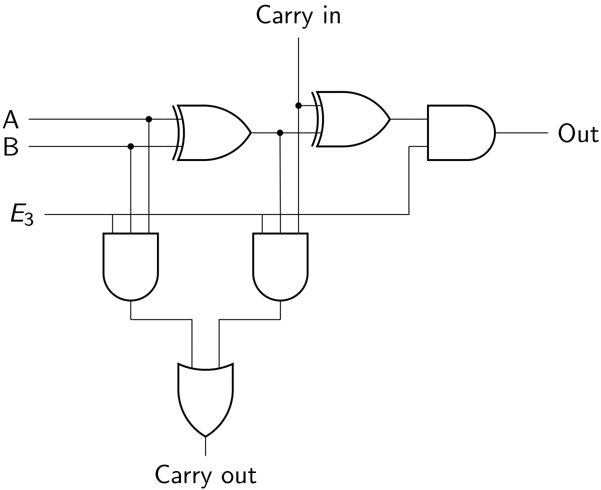


E_0: $A \cdot B$.
E_1: $A + B$.
E_2: $\overline{B}$.

Full Adder

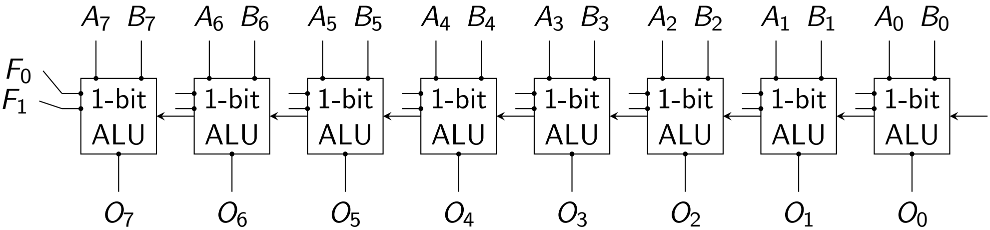
Antag $E_3 = 1$.

A	B	Carry in	Carry out	Out
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1



8-bit ALU

Inputs: INVA, A, ENA, B, ENB, Carry in, F0, F1.
Outputs: Output, Carry out.



Andre Brugbare Kredsløb

- Multiplexer
- Demultiplexer
- Decoder
- Komperator
- Shifters
- Latches
  - SR latch (set-reset)
  - D lat
  - Flip-flops
- ...

Indhold

Opsummering

- Talsystemer, herunder binary, octal, hexadecimal og konvertering imellem disse.
- Digitale kredsløb og nødvendigheden af især det binære talsystem til forståelse heraf.
- Boolsk algebra og sandhedstabeller.
- En simpel 1-bit ALU og dens elementer. ALUen er "hjernen" af CPUen.

Opgave

- Opskriv og reducer udtrykket, der kommer af tabellen til højre.
- Reducer følgende udtryk:

$$X = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

- ALU
  - Installér en digital-logik simulator, f.eks. **Digital** `https://github.com/hneemann/Digital`.
  - Lav et 1-bit ALU-kredsløb som præsenteret i lektionen.
  - Sæt otte af dem sammen for at lave en 8-bit ALU.

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1