Velkommen!

Abstraksjonsnivå, Datamaskinarkitektur og ALU

Gruppe 27

(Hanne-)Kristin Schrøder Leiros

hkleios@uio.no

Plan for dagen

- Litt repetisjon av pensum
  - Foilene er ikke de beste men i år har dere også fått tekstlig sammendrag og bakgrunn
- Jobbe med ukesoppgaver i plenum

Oblig 3 kommer snart

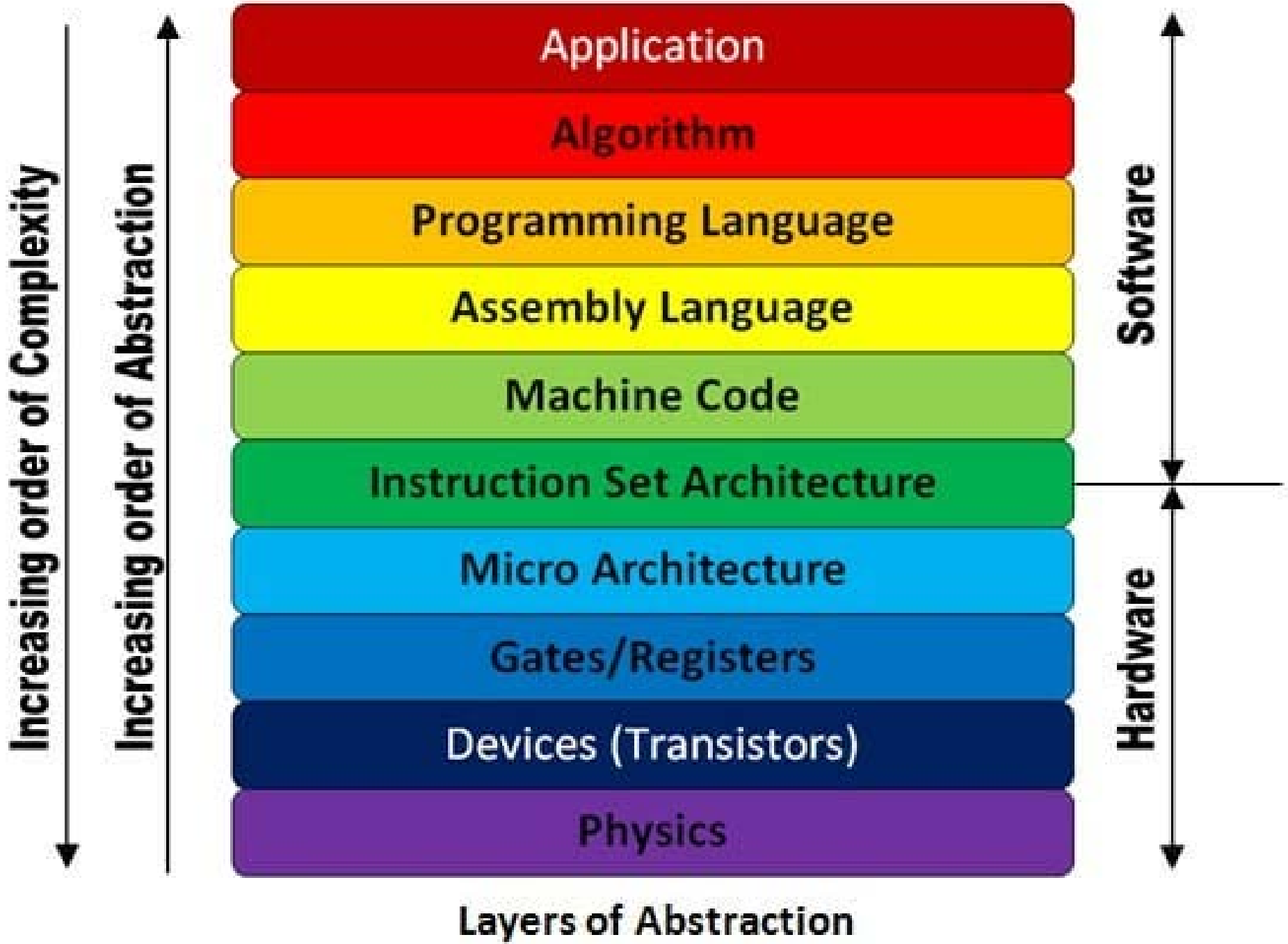
- Frist 25. oktober kl. 23:59
- Kretsanalyse
- Funksjonsbeskrivelse
- Forbedringstiltak
- Og mer ?

Læringsmål

- Ha oversikt over hvordan kode eksekverer på en simplifisert Von-Neumann arkitektur
- Kunne drøfte hvordan en PC fungerer fra de forskjellige abstraksjonsnivåene
- Kunne grunnleggende forenklet port design
  - (mer denne uke om porter)

Abstraksjonsnivå

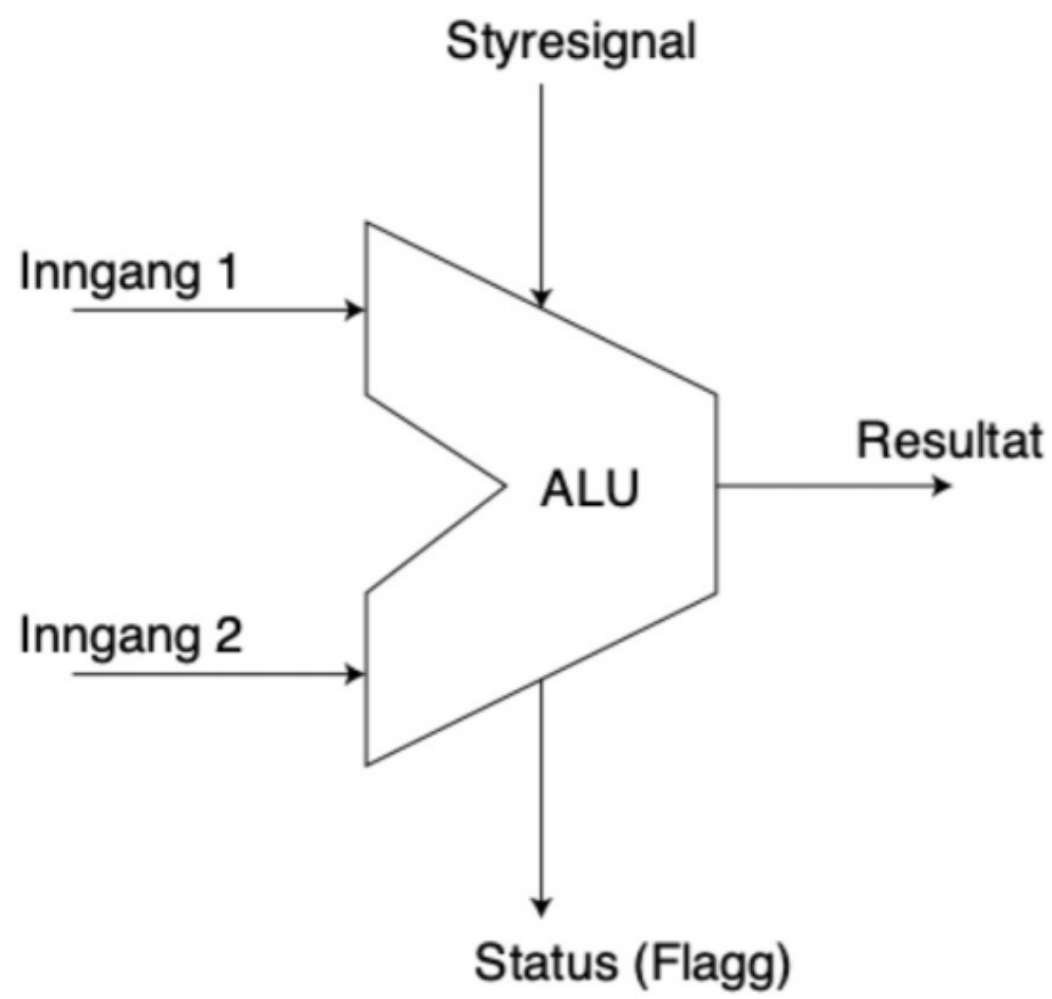
- Vi TRENGER ikke alltid forstå hvordan noe fungerer, bare at det fungerer
- Men dere lærer om dette så dere KAN forstå det når dere trenger og det er greit å forstå hardwaren man programmerer på
- Vi deler ofte i forskjellige lag av abstraksjon eller abstraksjonsnivåer
- Gjemmer kompleksitet
- Eksempel i notatet om bil og gaming


Increasing order of Complexity
Increasing order of Abstraction
Application
Algorithm
Programming Language
Assembly Language
Machine Code
Instruction Set Architecture
Micro Architecture
Gates/Registers
Devices (Transistors)
Physics
Software
Hardware
Layers of Abstraction

Applikasjoner	3D-Rendering program, Word, Zoom, etc. Operativsystem er det nederste «laget» av applikasjons-laget.	Software
Algoritmer	Hvordan et applikasjon skal kjøres elns.	
Programmeringsspråk	Java, Python, etc.	
Assembly kode	Simpleste koden som ikke består av kun binære tall. Varierer basert på hvilken maskin som brukes.	
Maskin kode	Binære tall som maskinen kan «forstå». Antall binære tall er avhengig av antall bits som er avhengig av maskin-arkitektur.	Fra SW Til HW
Instruction Set Architecture	En abstrakt modell av en datamaskin som beskriver hva maskinen består av. data typer, registre, etc.	
Micro Architecture	Kretsløpet i en datamaskin	Hardware
Gates/Registers	Enheter som utfører boolske funksjoner. AND, NOT, OR, XOR,	
Devices (Transistors)	Spenning, ikke spenning	
Fysikk	HvA eR sPeNiNg	

ALU – Aritmetic Logic Unit

- Brukes til aritmetiske og logiske beregninger
- Input: to for data, en instruksjon
- Output: resultat og status
- Viktigste delen av CPU-en



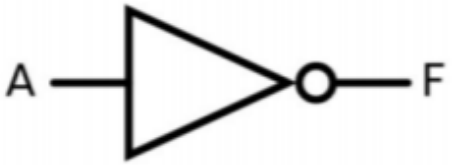
Logiske porter

- Base for de logiske funksjonene i ALU-en
- Bruker boolsk logikk
- Tre viktigste er NOT (INV), AND og OR
- Har egne regler, symboler og sannhetsverditabeller

INV – port

$F = A'$

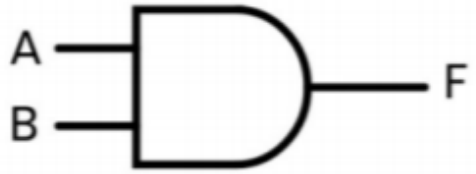
A	F
0	1
1	0



AND – port

$F = AB$

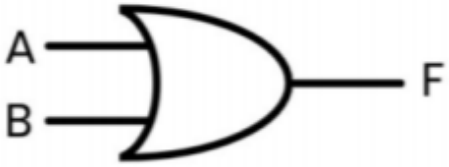
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1



OR – port

$F = A + B$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1



Sannhetsverditabeller

- Beskriver de mulige resultatene av alle kombinasjoner i en port
- Én sannhetsverditabell kan ha mange funksjonsbeskrivelser
  - (sammensetninger av porter)
- Én funksjon har bare én sannhetsverditabell

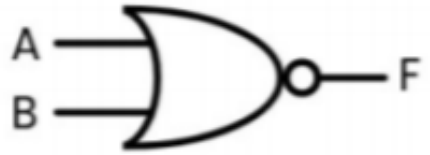
Inverterte porter

- Med de tre logiske portene nå kan vi også lage:
  - NAND (NotAND)
  - NOR (NotOR)
- Disse gir motsatt output fra AND og OR porter

NOR – port

$F = (A + B)'$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0



NAND – port

$F = (AB)'$

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



XOR-porten

- Viktig for å kunne utføre binær addisjon pga sannhetsverditabellen

XOR - port

$F = A \oplus B$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



XNOR - port

$F = (A \oplus B)'$

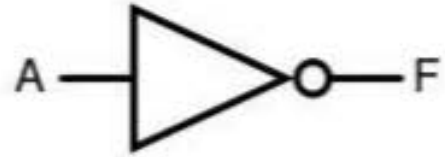
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1



INV - port

$F = A'$

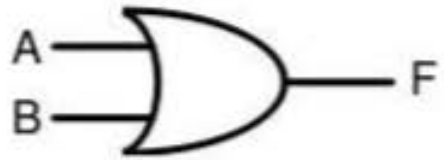
A	F
0	1
1	0



OR - port

$F = A+B$

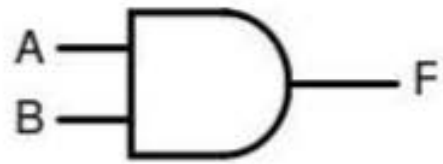
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1



AND - port

$F = AB$

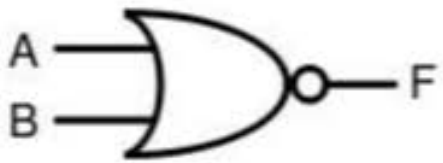
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1



NOR - port

$F = (A + B)'$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0



NAND - port

$F = (AB)'$

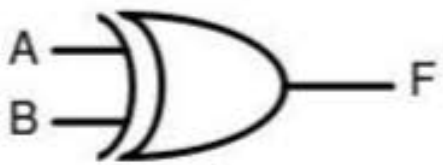
A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



XOR - port

$F = A \oplus B$

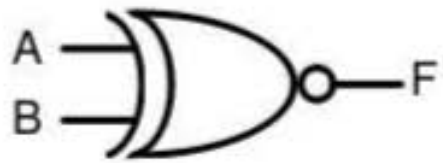
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



XNOR - port

$F = (A \oplus B)'$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

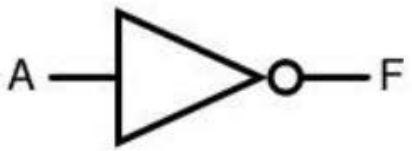


Alle logiske port-typer

INV - port

$F = A'$

A	F
0	1
1	0



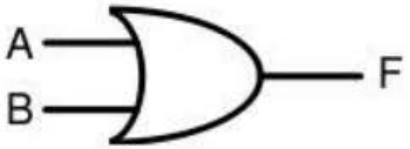
Invert - snur om inputet

OR - et eller begge tall er 1

OR - port

$F = A+B$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

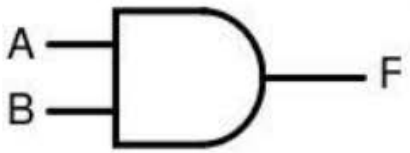


AND - begge tallene er 1

AND - port

$F = AB$

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

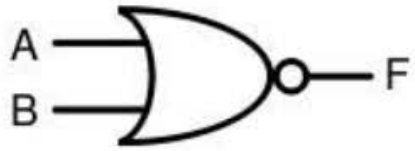


Not OR - ingen av tallene er 1

NOR - port

$F = (A + B)'$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

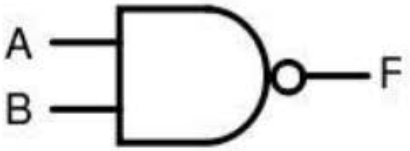


Not AND - minst ett tall er ikke 1

NAND - port

$F = (AB)'$

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

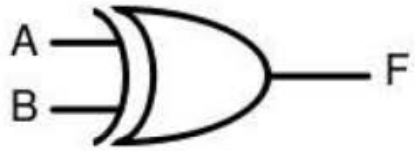


eXclusive OR - ett av tallene er 1, men ikke begge

XOR - port

$F = A \oplus B$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

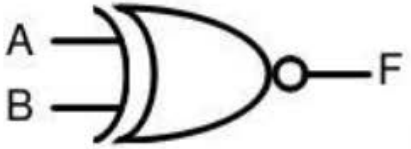


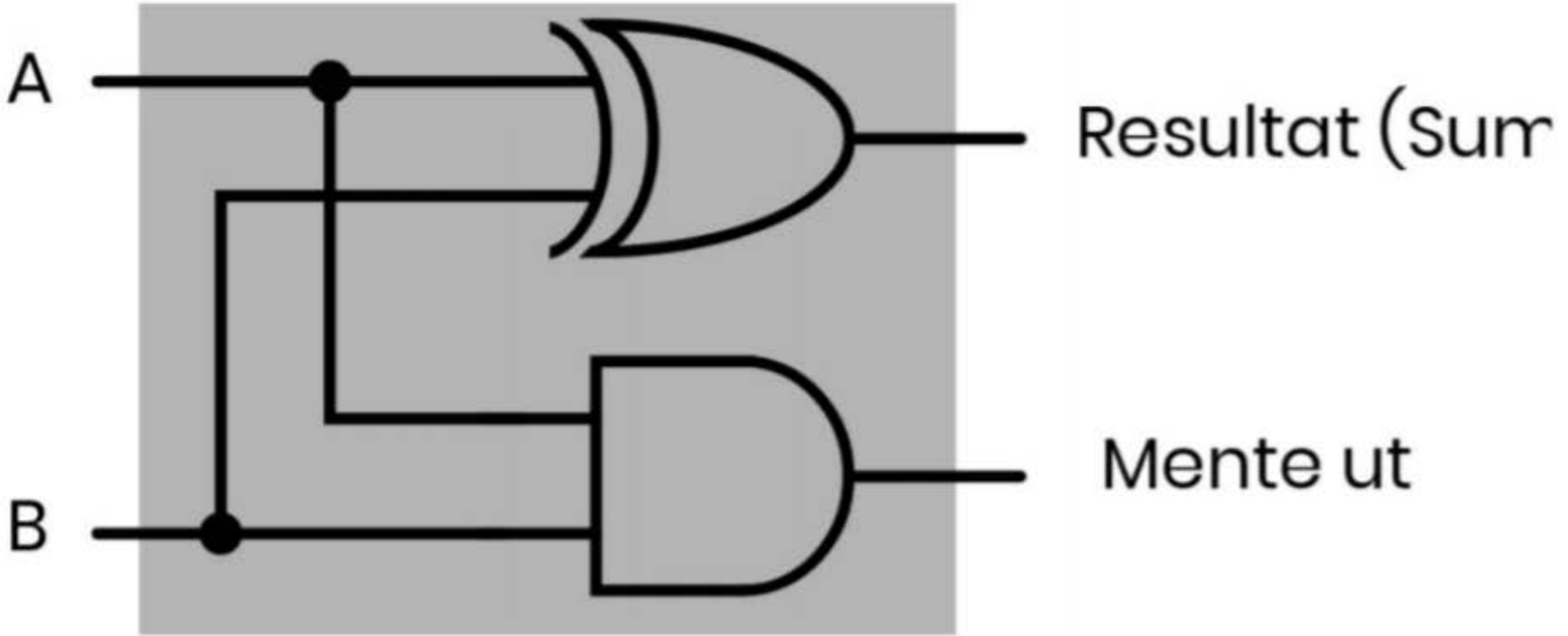
eXclusive Not OR - begge tall har samme verdi

XNOR - port

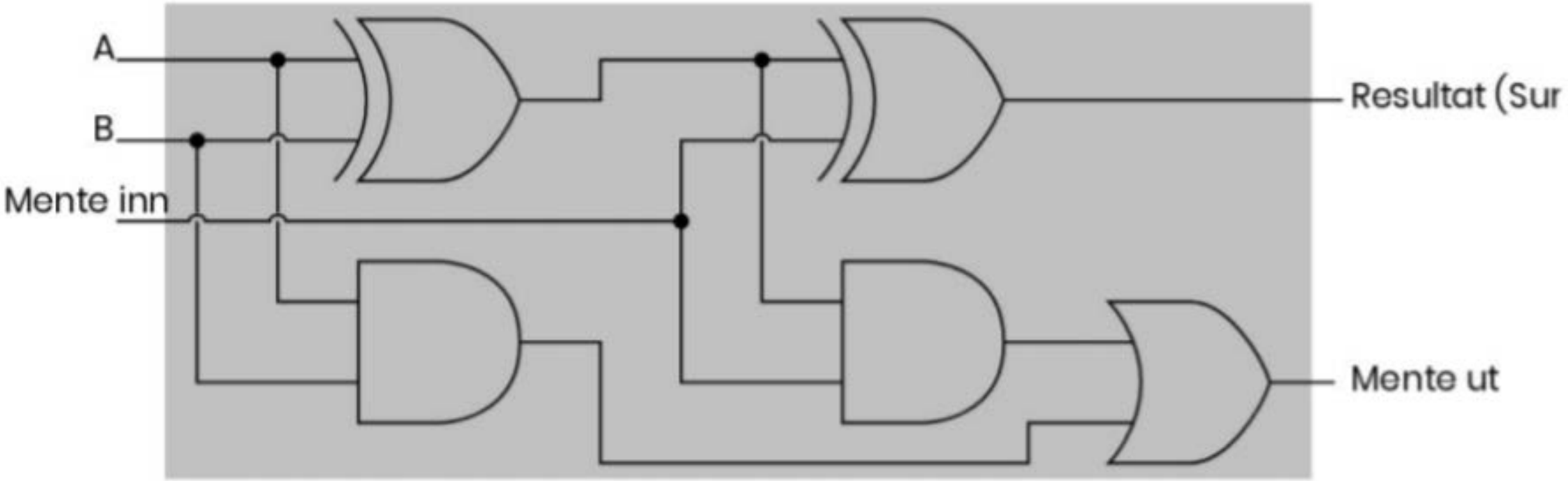
$F = (A \oplus B)'$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1





HALV-ADDER (HA)



FULL-ADDER (FA)

Addere

- Brukes for å kunne utføre binær addisjon
- Halv-adder kan legge sammen to tall , men gir ikke om vi har mente inn også
- Derfor setter vi sammen flere Logiske Porter og får ne Full-Adder, som også tar mente inn
- Hvis vi kombinerer fire Full-Addere kan lage en Seriell Adder, som kan legge samme 4-bit tall.

Ukesoppgaver

OPPGAVE 11

	0
+	1
=	1

OPPGAVE 12
Binær addisjon av 1+1
0 i sum og 1 i mente
1 i mente og 0 i sum
10_2

OPPGAVE 13
Binær addisjon av 0+0
0 i sum og 0 i mente

OPPGAVE 14
Binær addisjon av 1+1+1
1 i sum og 1 i mente

		1
+		1
=	1	0
+		1
=	1	1

OPPGAVE 15
Binær addisjon av 1+1+1+1
100

			1
+			1
=		1	0
+			1
=		1	1
+			1
=	1	0	0

Prøvde å sette det opp i en seriekoblet adder men usikker på hvor rett det ble

