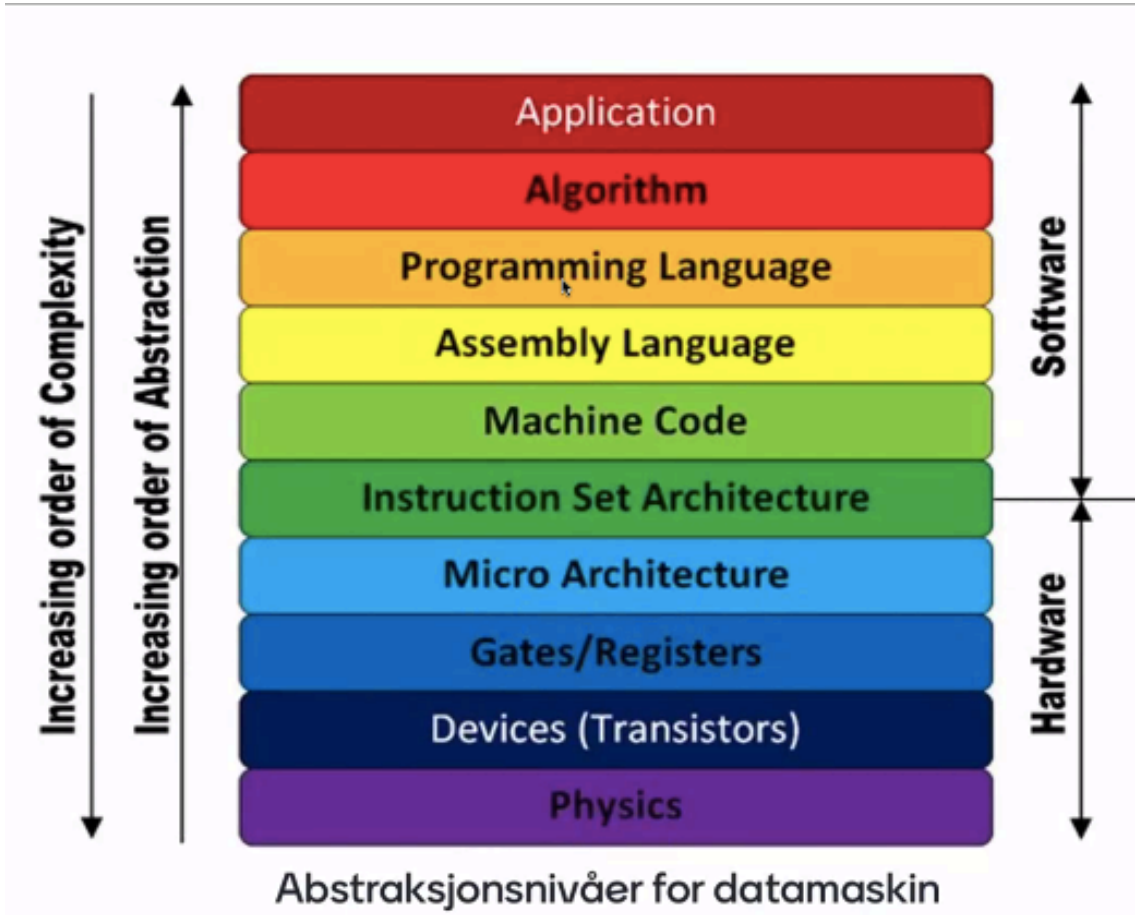
Introduksjon: Maskinvare

Abstraksjonsnivå



Abstraksjonsnivåer for datamaskin

Software: Forteller hva datamaskinen skal gjøre.

Hardware: Relatert til lavtnivå, utfører de instruksene som er blitt bedt av software.

Transistorer: En bryter som skrus av og på, tilfører strøm

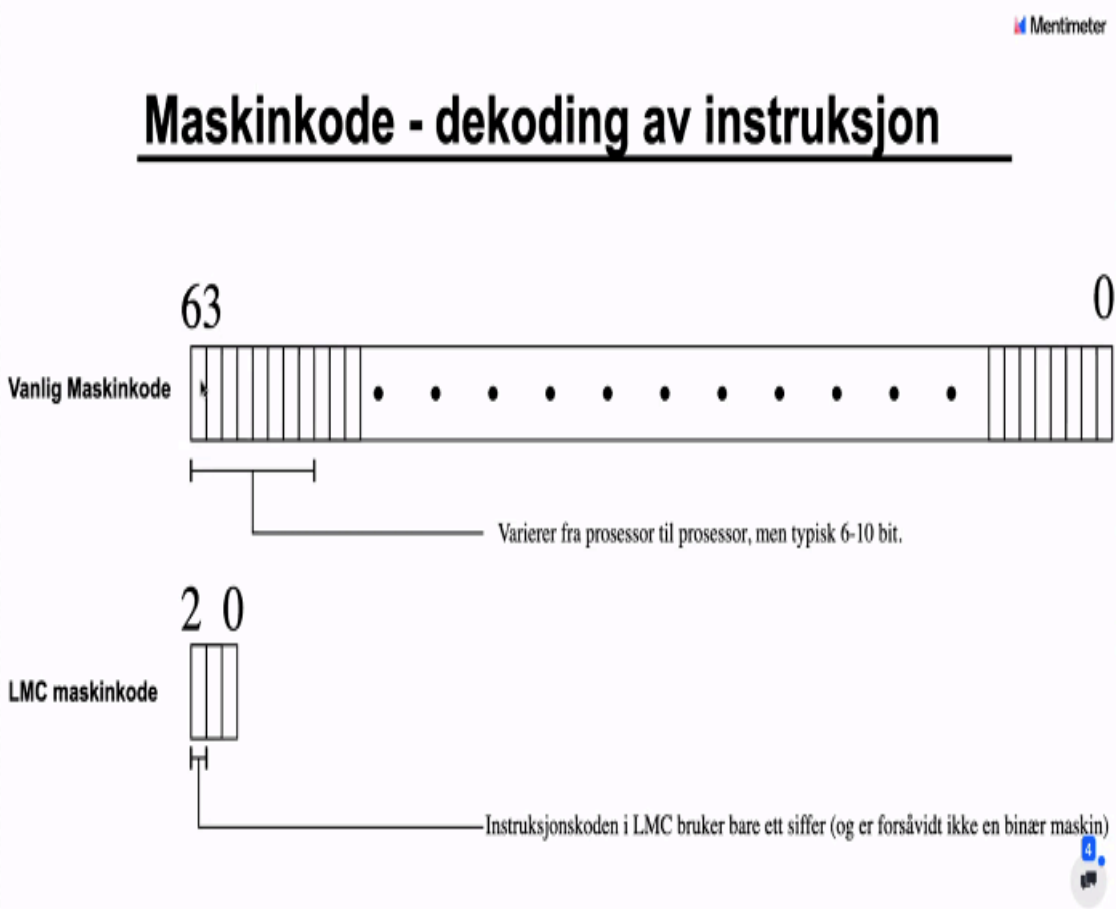
Høynivå: Programmeringspråk som er ansvarlig for å kontrollere/overstyre brukergrensesnittkomponenter

Middelsnivå: Hvordan maskinen tolker det som blir programmert

Lavtnivå: CPU og kjernen, ALU, logiske porter,transistorer

Abstraksjonsnivåene er ment for å gi en forenkling over datamaskinen og hjelper oss med å forstå hvilket nivå vi befinner oss i og hvorvidt nivået er software eller hardware.

Maskinkode



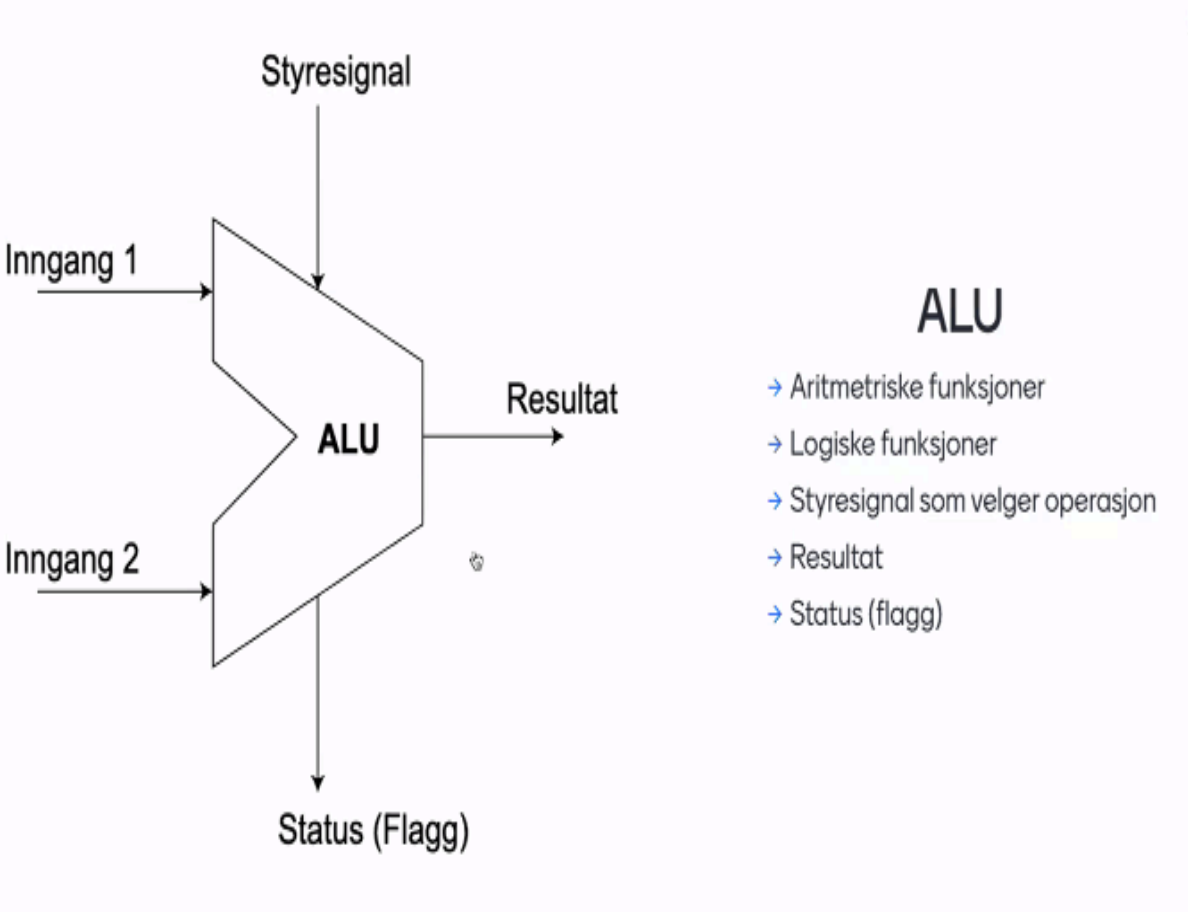
Maskinkode: Sett med instruksjoner
som blir direkte utført i CPU'en
dekoding av bitene er maskinkodens prinsipp.

Prosessor/CPU: Ansvarlig for å utføre instrukser i ett dataprogram og er en «must» for å gjennomføre datamaskinens funksjoner. I tillegg til å delegere arbeid til andre enheter i hardwaren.

64 eller 32 bit: Prosessoren er designet etter en gitt arkitektur,
Enten INTEL x86-64 eller x32 og andre ulike arkitekturen.
Hvis det er INTEL X86-64 så kan vi lagre opptil 2^65 =17 000 000
som gjør at vi kan utføre flere instrukser og bevare mer data i mir
For maskinkode, betyr det flere instrukser og flere bit som må dekodes, noe som gjør prosessen mer effektivt

LMC baseres ikke på en slik arkitektur hvor vi kan utføre få instru
og er begrenset til 100 celler, som vil si at det er færre data som kan lagres.

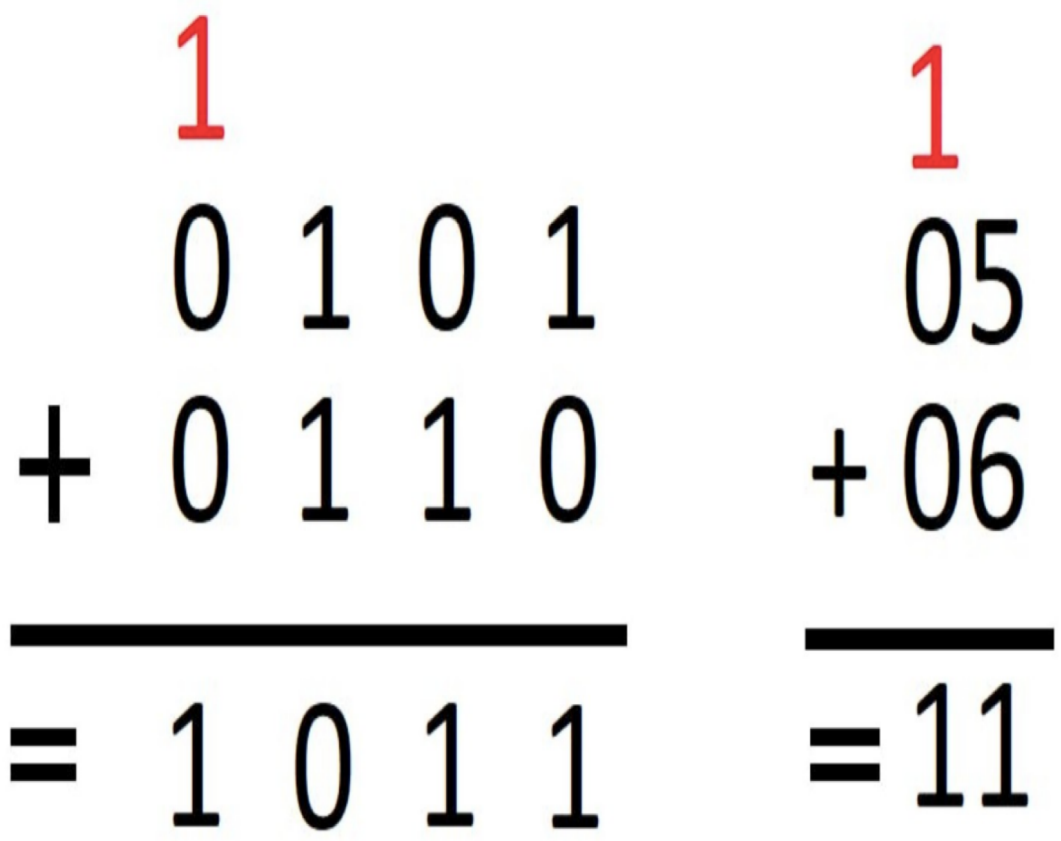
ALU – Aritmetrisk logisk enhet



ALU er en del av CPUen og er ansvarlig for å utføre aritmetriske og logiske operasjoner.

Aritmetriske funksjoner: Subtraksjon, addisjon, deling og ganging
Logiske funksjoner: Funksjoner som består av boolean(True or False).
Status (flagg): Skulle noe galt oppstå så går verdien gjennom flaggene.
Styresignal: Det som styrer ALU til hva den skal gjøre, altså er det addisjon, eller subtraksjon som ALU skal gjøre.

Binær Addisjon



Maskinkode sin prinsipp: Dekoding av bitene.

Addisjonskretsen: Tar to inngangsbit og returnerer hverken 1 eller 0.

Sannhetsverditabell



Begreper

- Port: Har alltid en logisk funksjon, men kan også ha en matematisk funksjon. AND port har både en logisk og matematisk funksjon.

- Kretser: En samling av porter som utfører en gitt funksjon. Adder-port, halv-adder-port.

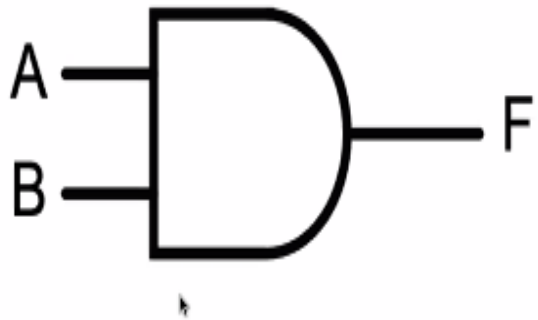
- Funksjon: Kan være en logisk funksjon, kan være en matematisk funksjon og inneholder en eller flere porter.

AND-port

AND - port

$F = AB$

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

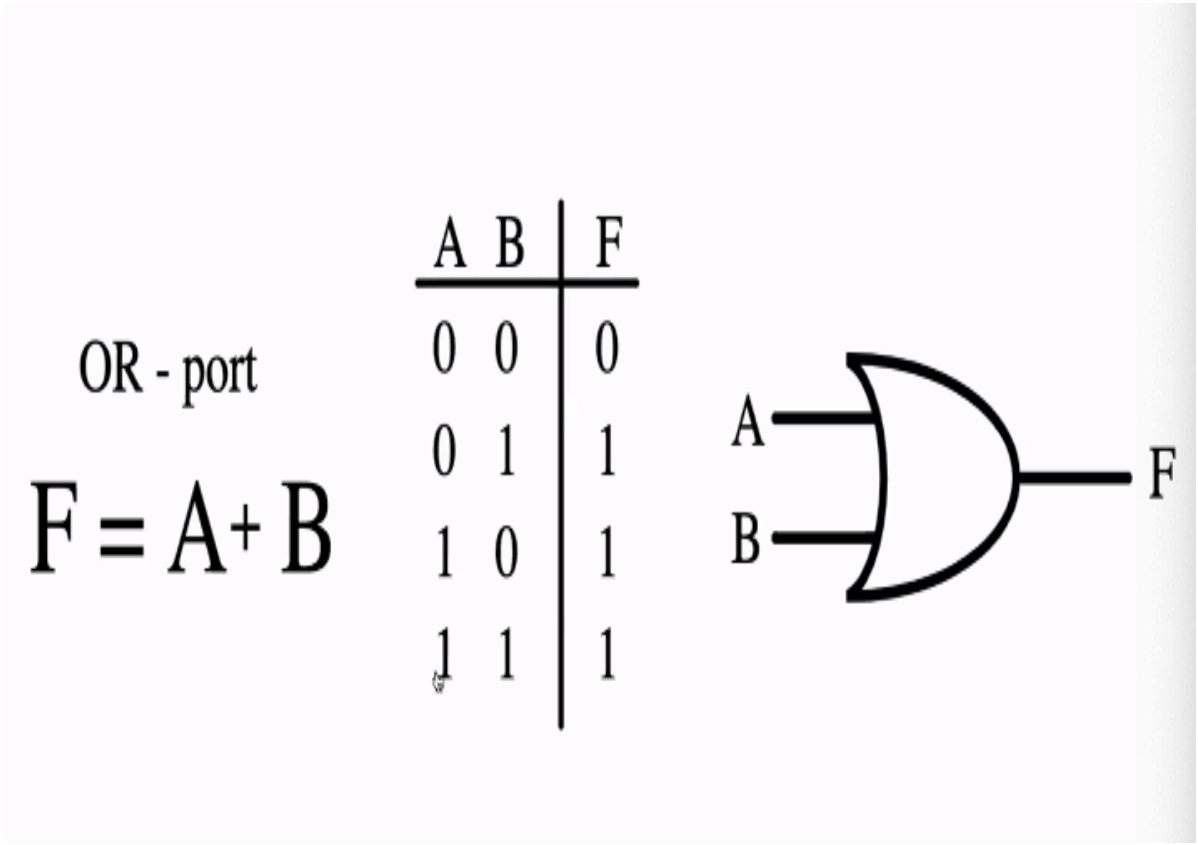


Tegningen til høyre er hvordan porten ser ut. Kjennetegn hvis det er snakk om AND—port er hvis porten ser ut som bokstaven «D»

Kan tenke på «AND» i if-sjekk for python, bruker samme prinsipp.

En AND-port benyttes for å kunne skrive “mente” ut. Kan sees igjennom inngangsbit (1 og 1)

OR-port



Kjennetegn hvis det er snakk om OR—port er hvis porten ser ut som «Bue»

Kan tenke på «OR» i if-sjekk for python. Oppfylles hvis ett av inngangsbitene er 1, hvis begge er 0 så får vi 0 ut.

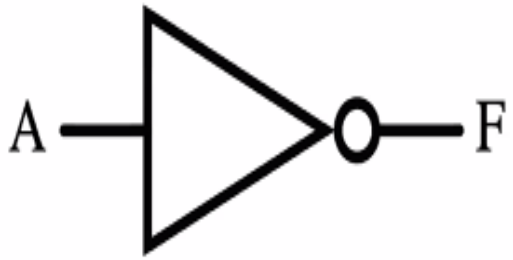
INVERTER

Hvis det er en sirkel etter porten, har vi en inverter. Inngangsbit 0 gir 1 ut, Inngangsbit 1 gir 0 ut. Gir ut det motsatte

INV - port

$F = A'$

A	F
0	1
1	0

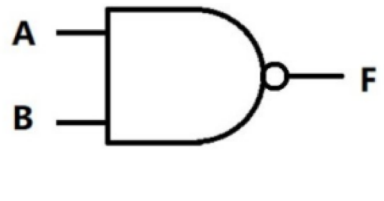


NAND-, NOR- og XOR-port

NAND - port

$F = (AB)`$

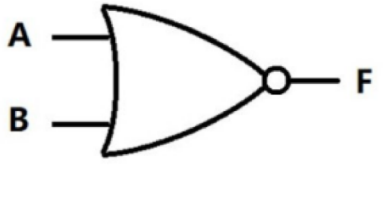
A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



NOR - port

$F = (A+B)`$

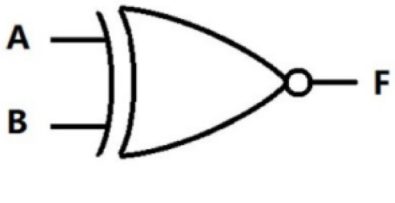
A	B	F
0	0	1
0	1	0
1	0	0
1	1	0



XNOR - port

$F = (A \oplus B)`$

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

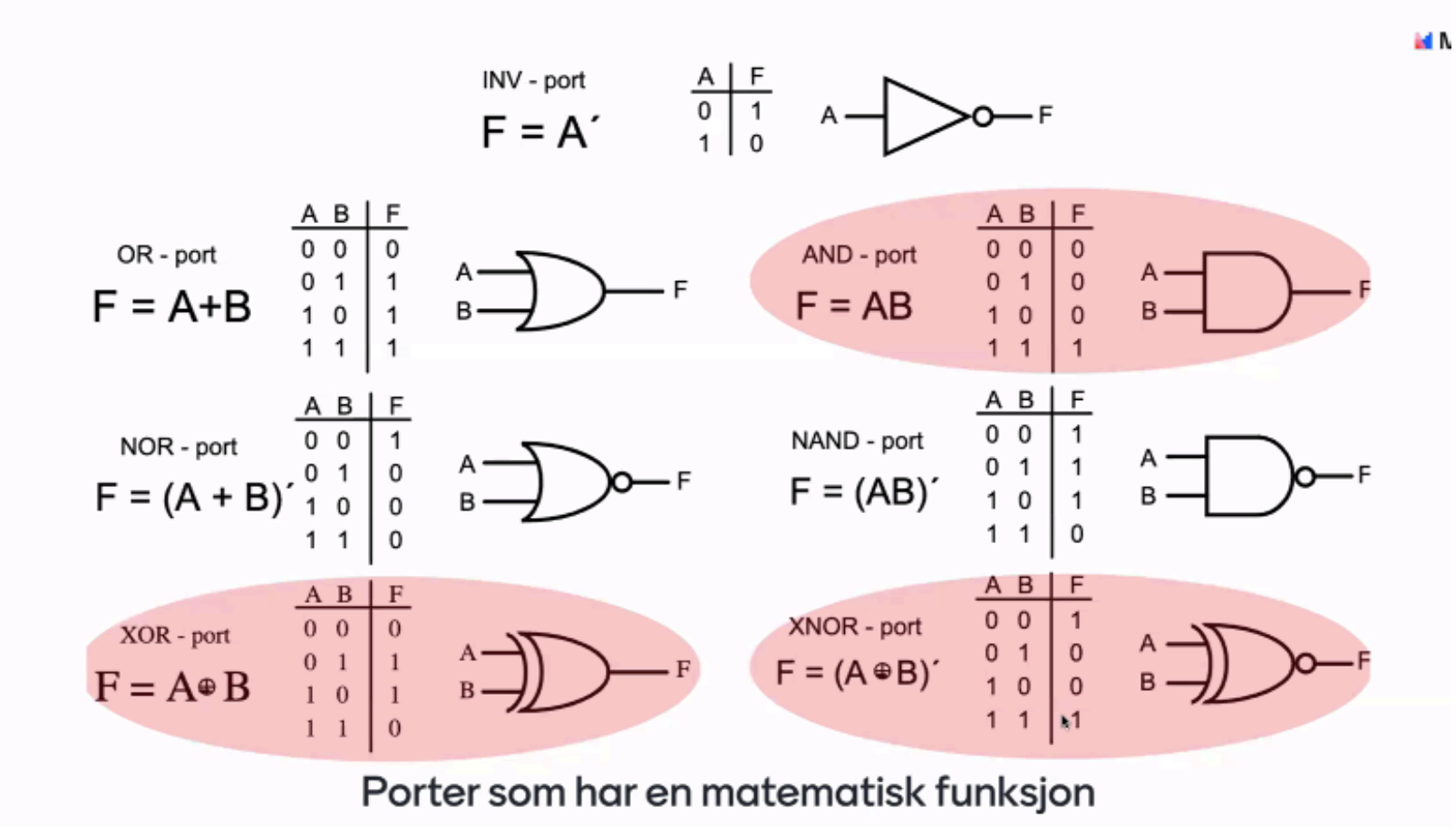


NAND: Kjennetegnes ved at der en AND-port men med en INVERTER. Gir ut motsatt for hva en AND-port opprinneli skal gi.

NOR: Kjennetegnes ved at der en OR-port men med en INVERTER. Gir ut motsatt for hva en OR-port opprinnelig skal gi.

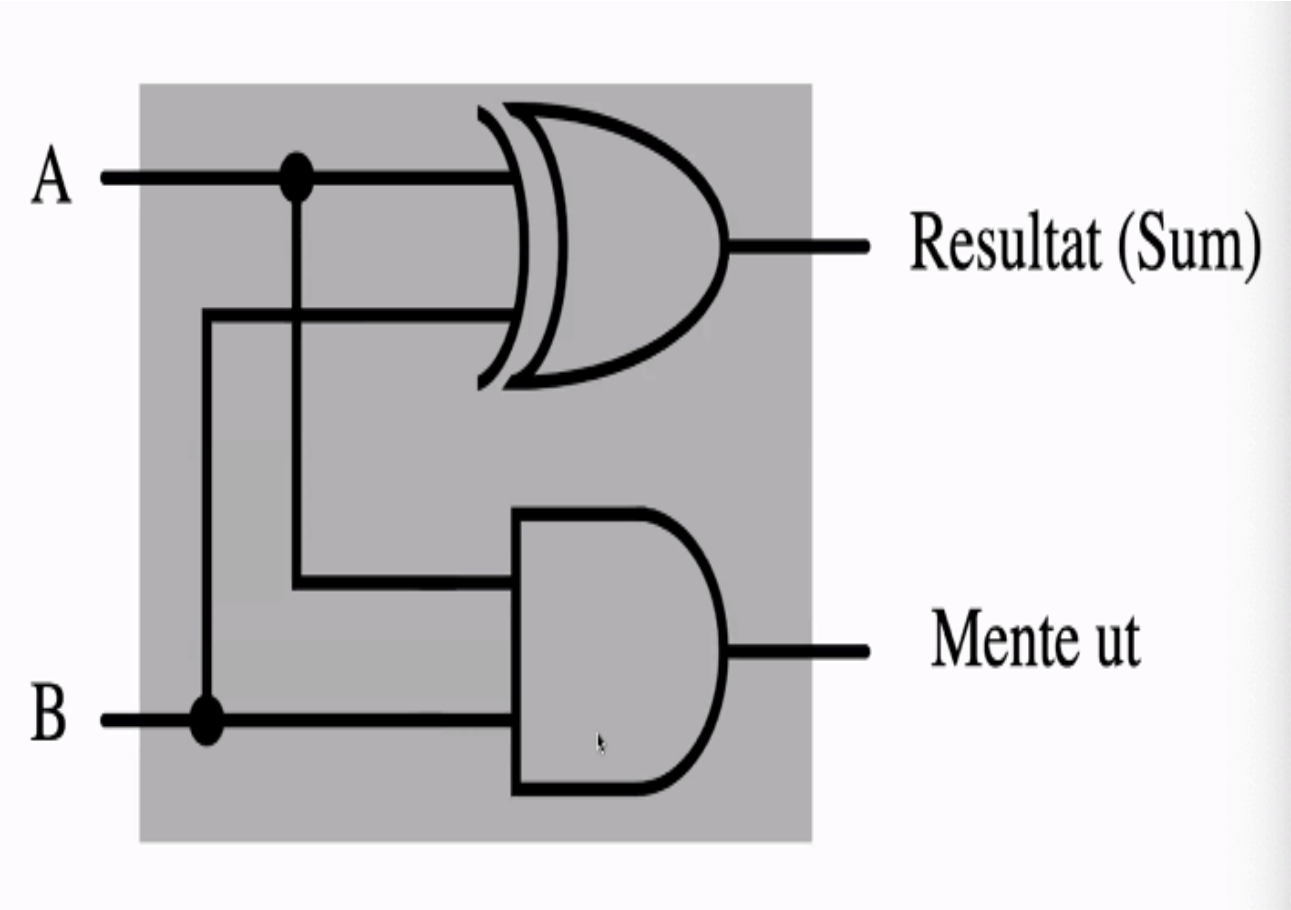
XOR: Kjennetegnes ved at der en OR-port men med en INVERTER og en bue før. Godtar ikke enten eller, må være 0 i inngangsbit eller 1 i inngangsbit får få ut 1. Viktigste ved binær addisjon siden den gjør det lett å føre videre menteverdien (1+1 = 1 i mente).

Alle portene



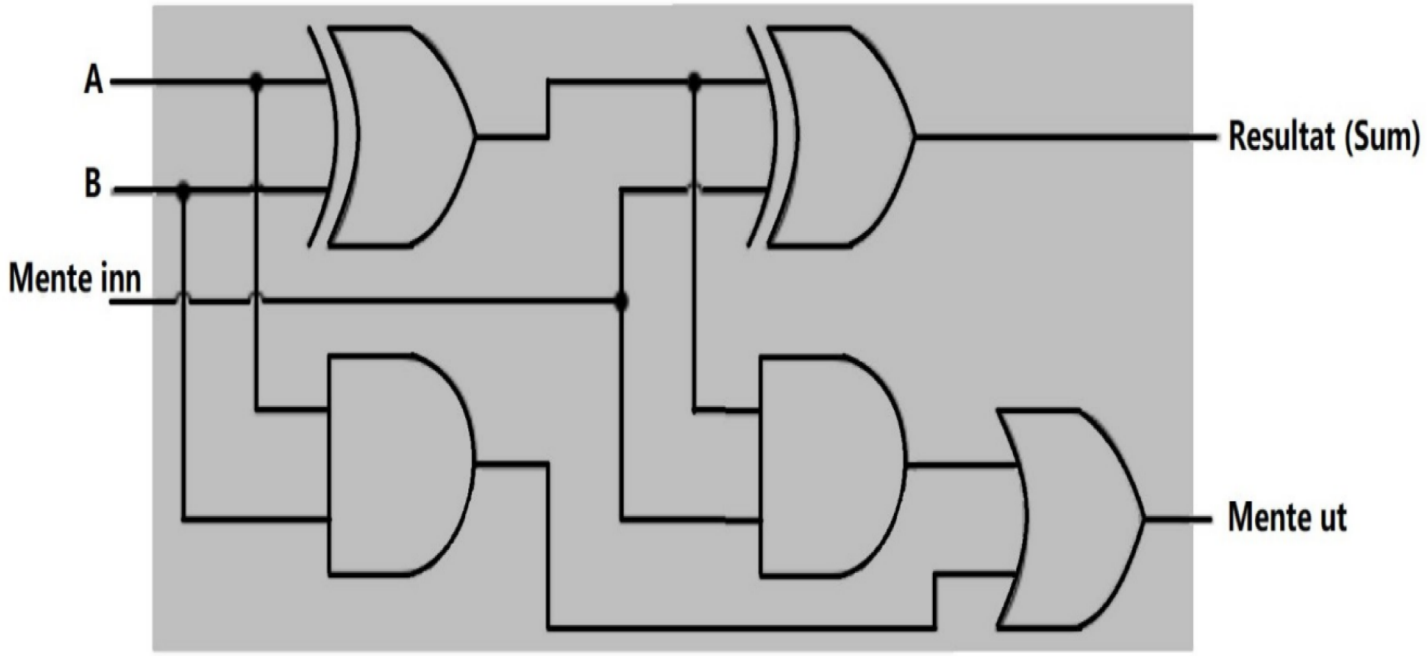
De røde markerer de som er matematiske operasjoner

HALV ADDER



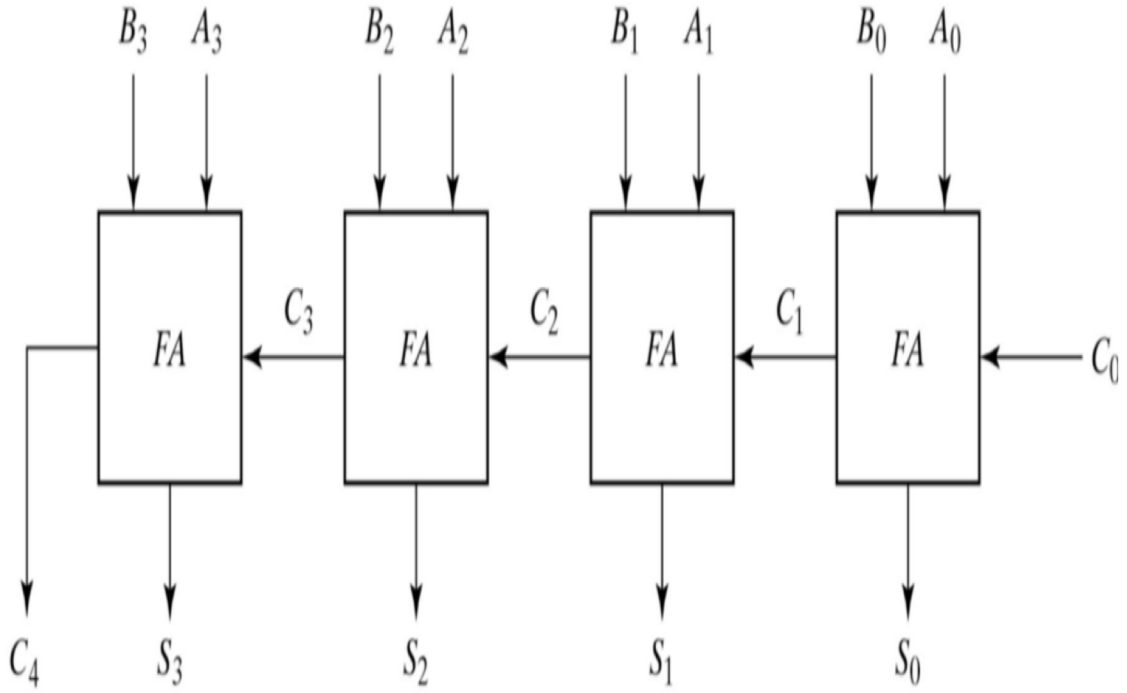
Hva er problemet i en halv-adder?
Det er å kunne hente forrige “mente” verdi i en ny addisjonsprosess. Altså hvordan den forholder seg til den neste addisjonsprosessen. I dette tilfellet så vil vi ikke innhente mente ut i en ny addisjonskre da mente ikke direkte leder til en ny addisjonskrets

FULL ADDER



Nå får vi løst problemet vi hadde i forrige slide med HALV-ADDER hvor vi får hentet inn mente og benytte den i en ny addisjonsprosess.

SERIELL ADDER



En SERIELL ADDER består av mange FULL ADDERE koblet sammen som er koblet i en serie. Mente sendes bortover i seriekoblingen.

Hvorfor brukes ikke seriell adder?
Problemet i en seriell adder er da mente. Årsaken er at svaret i en sum er klar til å skrive ut, men håndteringsprosessen for "mente" er lang. Vi vil at det skal være effektivt, altså at operasjonen tar mye mindre enn et sekund.

Eksamensoppgave

Gates



Hvilke(n) port er avbildet over:
Velg ett eller flere alternativer

- [ ] XOR-port
- [ ] NOR-port
- [x] OR-port
- [ ] NOT-port
- [ ] XNOR-port
- [ ] NAND-port
- [ ] AND-port