Agenda

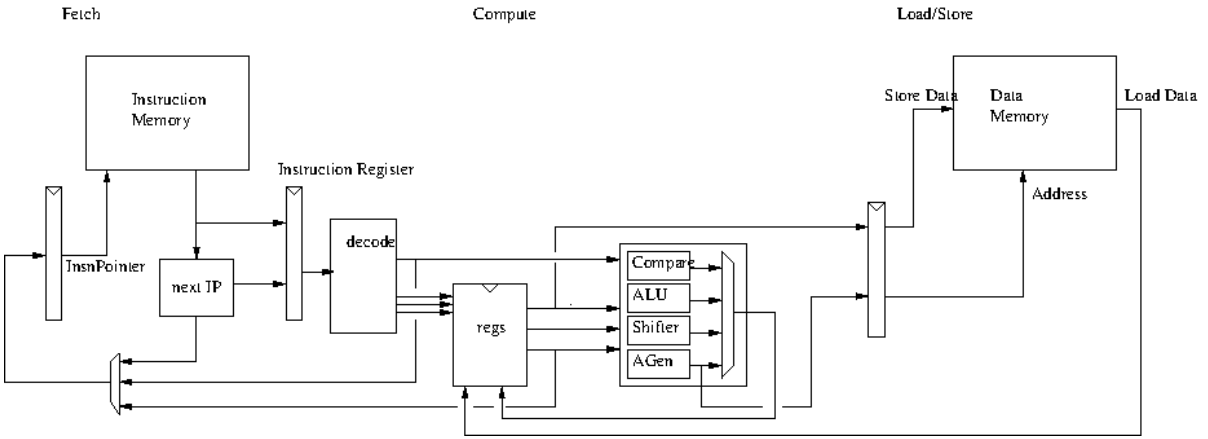
Første forelæsning:

- Yderligere forklaringer og uddybning af A2
- Nedstigning! Fra store til små byggeklodser

Anden forelæsning:

- Nedstigning til CMOS transistorer, Fremstillingsprocess, Moores lov

A2 mikroarkitektur



Spørgsmål? Klart som blæk?

Mere om A2

- Udleverede byggeklodser. Hvordan ser de ud i C ?
- Pipeline registre
- Afkoder(e)
- Debugging - nu med bug!
- Spørgsmål og Svar

Fra store til små byggeklodser

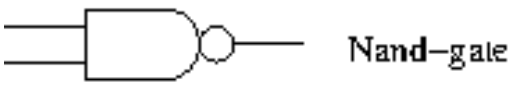
Abstraktionsniveuaer

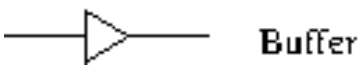
1. Højniveau programmeringssprog: Erlang, OCaml osv
2. Maskinnære programmeringssprog: C
3. Assembler / Symbolsk Maskinsprog: x86, ARM, MIPS
4. Arkitektur (ISA): Maskinsprog - ordrer indkodet som tal
5. Mikroarkitektur: ting som lager, registre, regneenheder, afkodere og hvordan de forbindes så det bliver en maskine
6. Standard celler: Simple funktioner af få bit (1-4) med et eller to resultater. Lagring af data (flip-flops)
7. Transistorer
8. Fysik. Eller noget der ligner

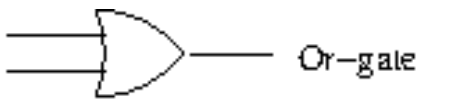
Porte. De simpleste byggeklodser

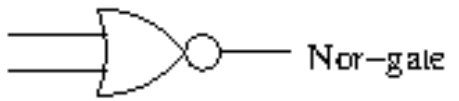
A	B	(A and B)	(A nand B)	(A or B)	(A nor B)	(A xor B)
0	0	0	1	0	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	0	1	0	0

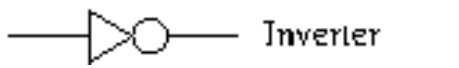
A	not A
0	1
1	0



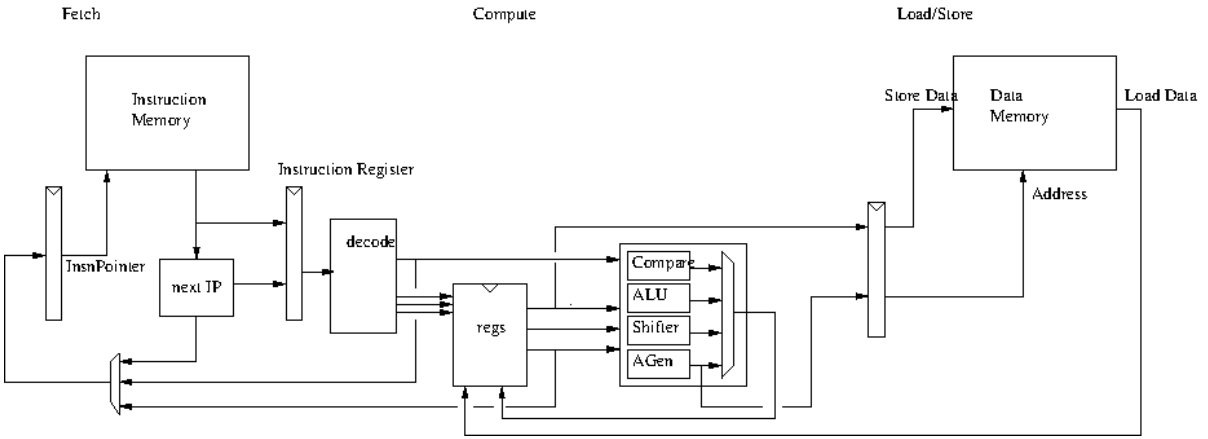






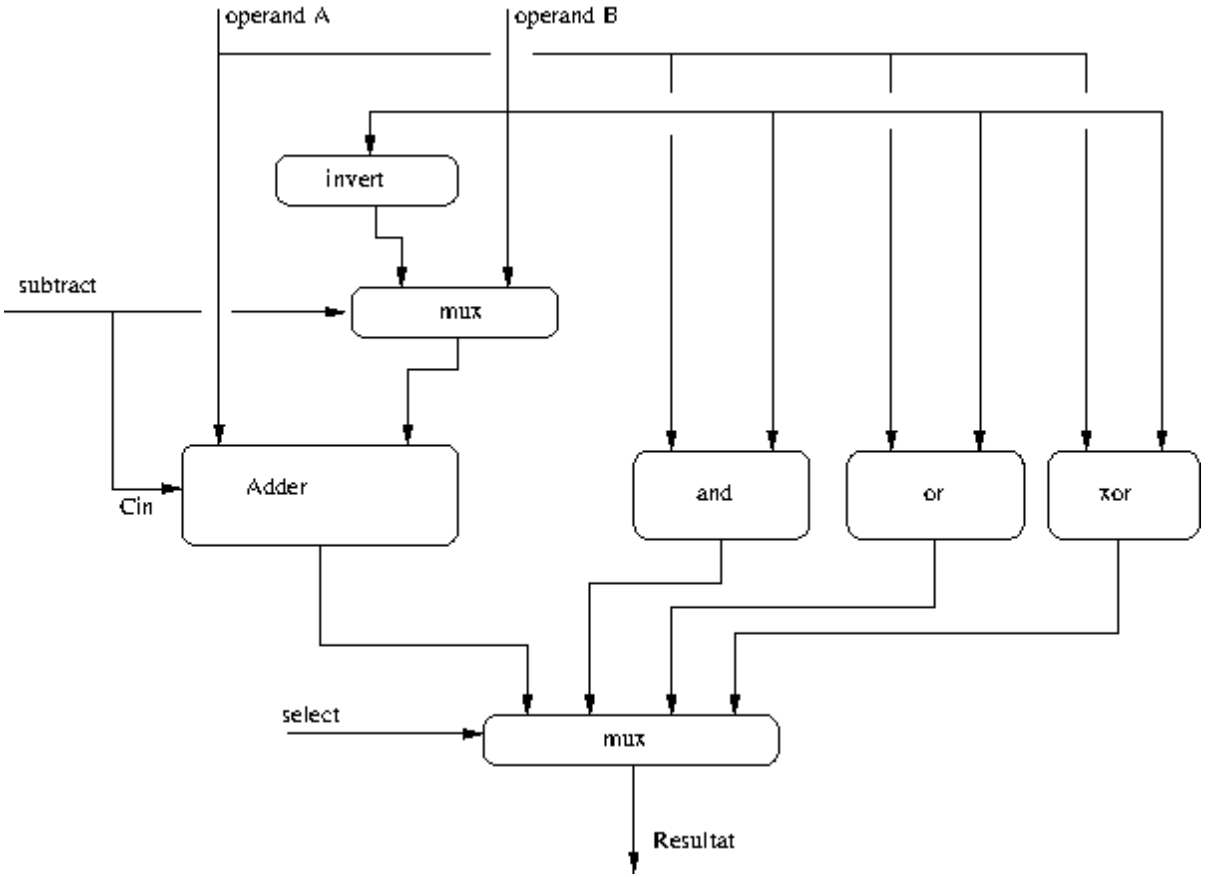


Store klodser laves af små klodser



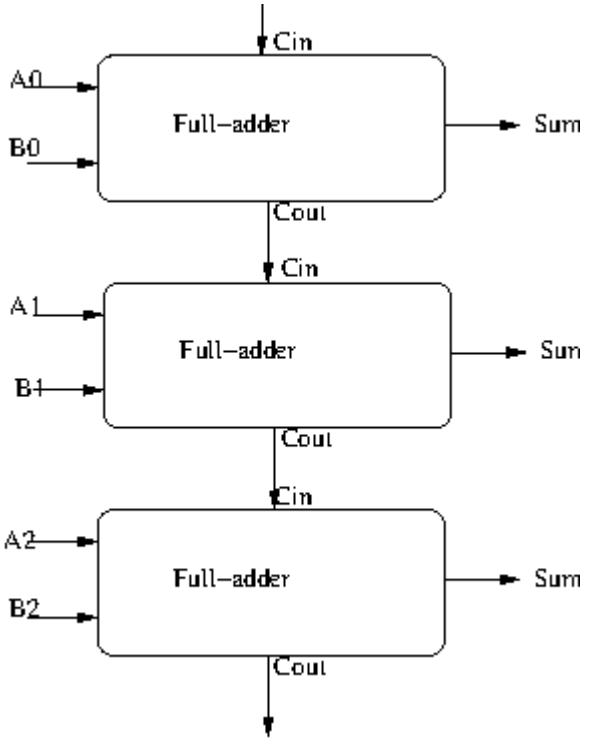
Store klodser laves af små klodser (II)

Aritmetisk-Logisk enhed (ALU)



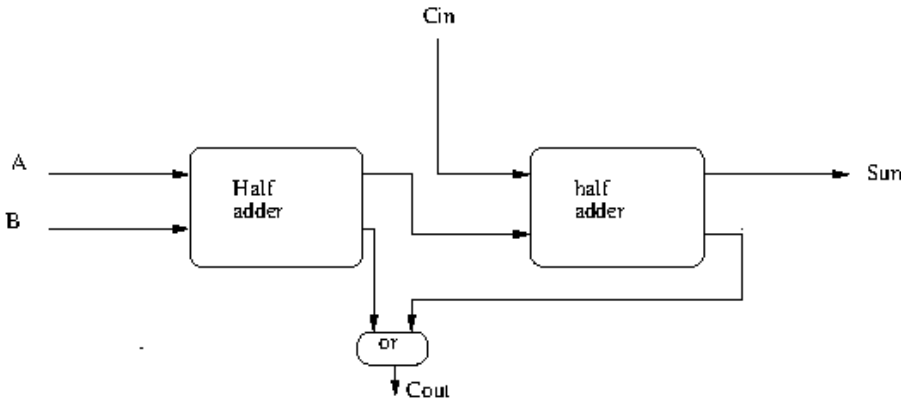
Store klodser laves af små klodser (III)

Ripple-Carry-adder (3 bit), addition bit-by-bit



Store klodser laves af små klodser (IV)

Full-adder (3-bit adder)

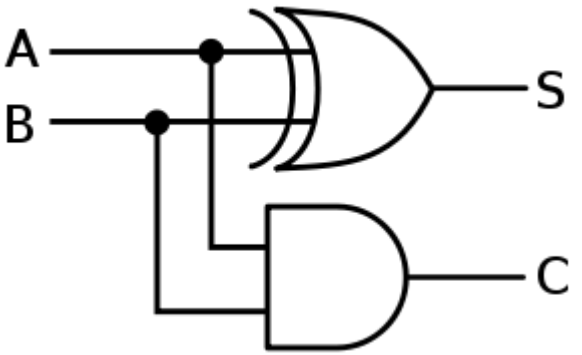


Halfadder:

A	B	Sum	Mente
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Store klodser laves af små klodser (V)

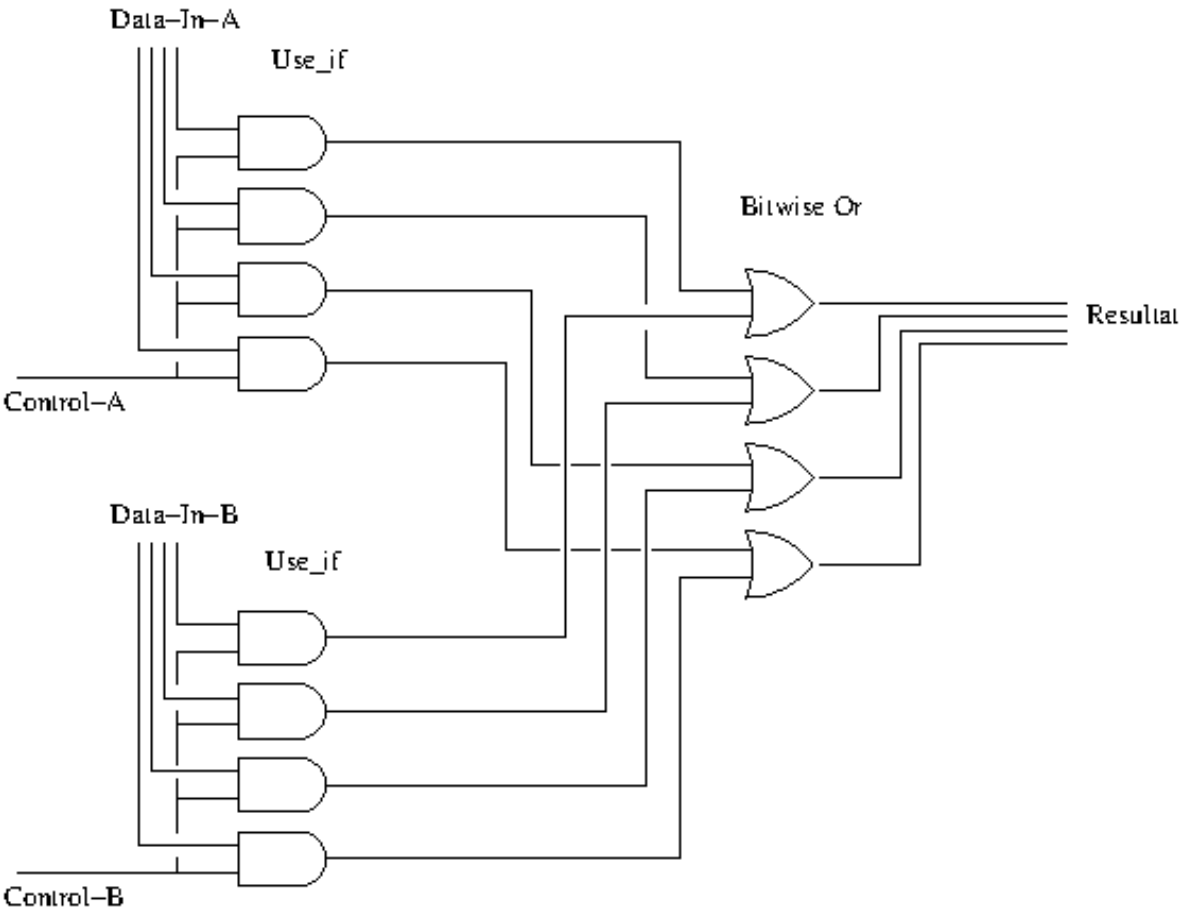
Half-adder (2-bit adder)



A	B	Sum	Mente
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

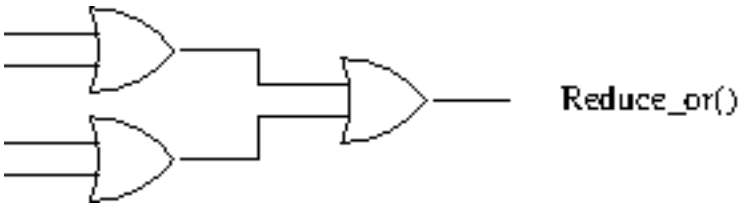
Sådan laves en multiplexor

Ved hjælp af use_if() og or()



Sådan laves en "er lig med"

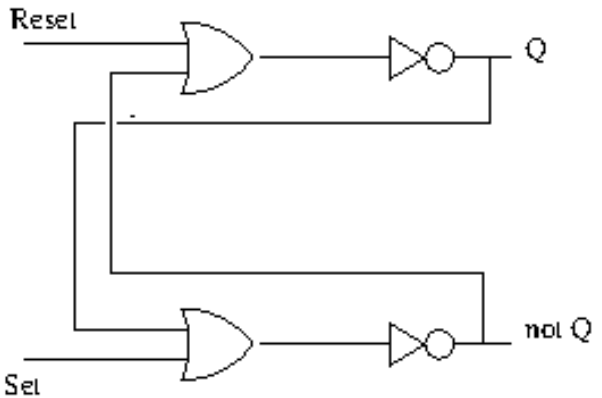
```
val is_equal(val a, val b) {
    val bitwise_diff = xor(a, b);
    bool equal = ! reduce_or(bitwise_diff);
    return equal
}
```



Tilstandselementer

Lagring af en enkelt bit

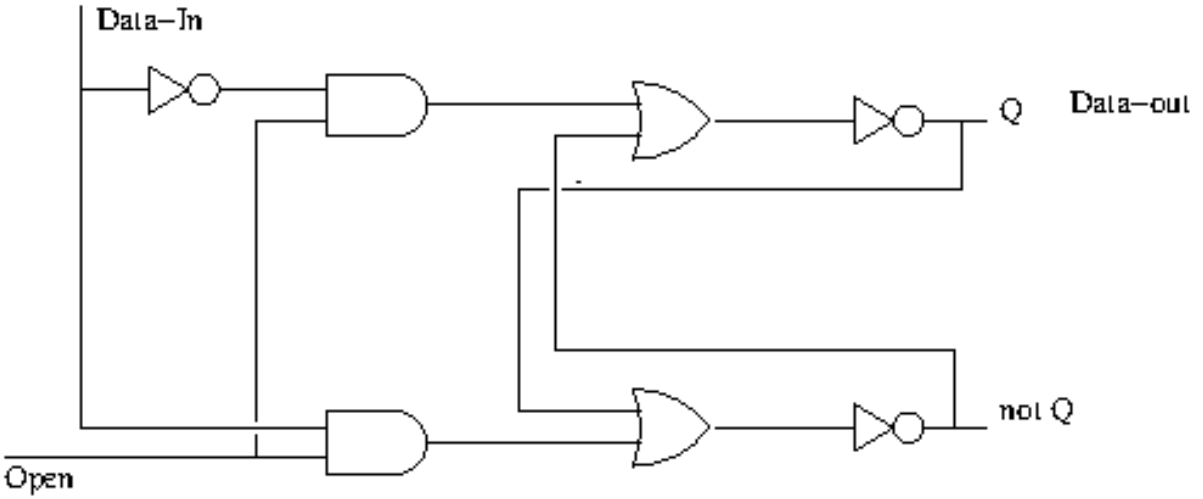
En RS latch (R=reset, S=Set)



Reset	Set	Funktion
0	0	Ingen ændring
0	1	Q -> 1
1	0	Q -> 0
1	1	Forbudt!

Lagring af en enkelt bit (II)

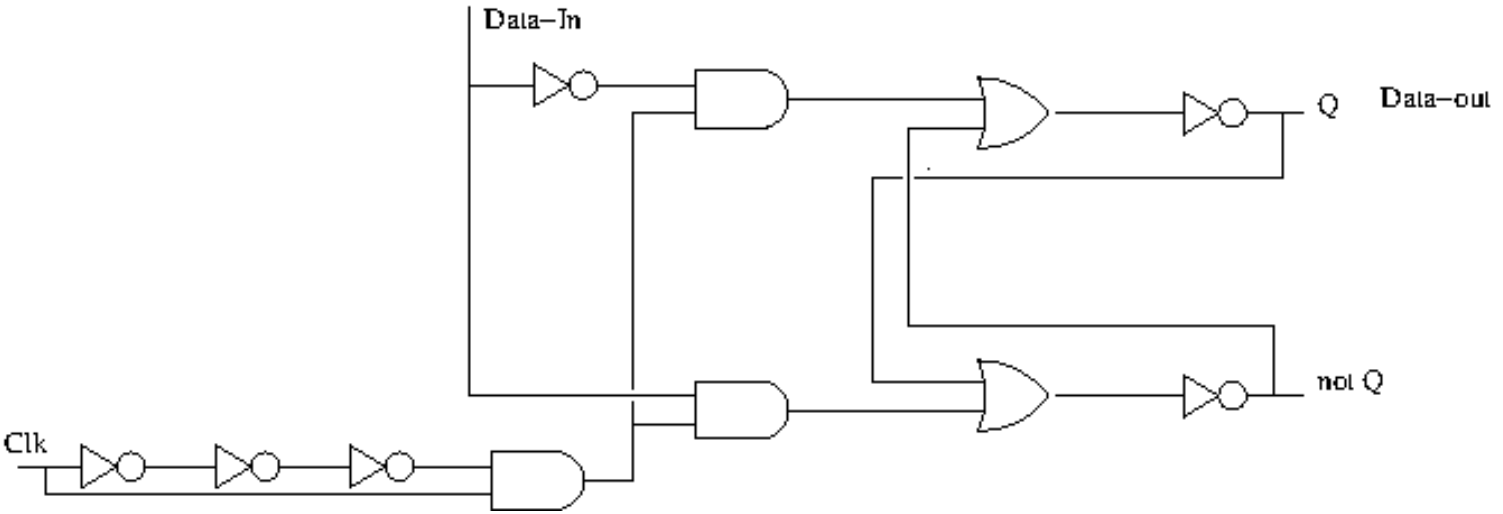
En D-latch (D for Data). Opdateres sålænge 'open' er høj



Undertiden kaldes en D-latch også for en SRAM-celle.

Lagring af en enkelt bit (III)

En Kant-trigget D-latch. Opdateres på den stigende flanke af 'clk'

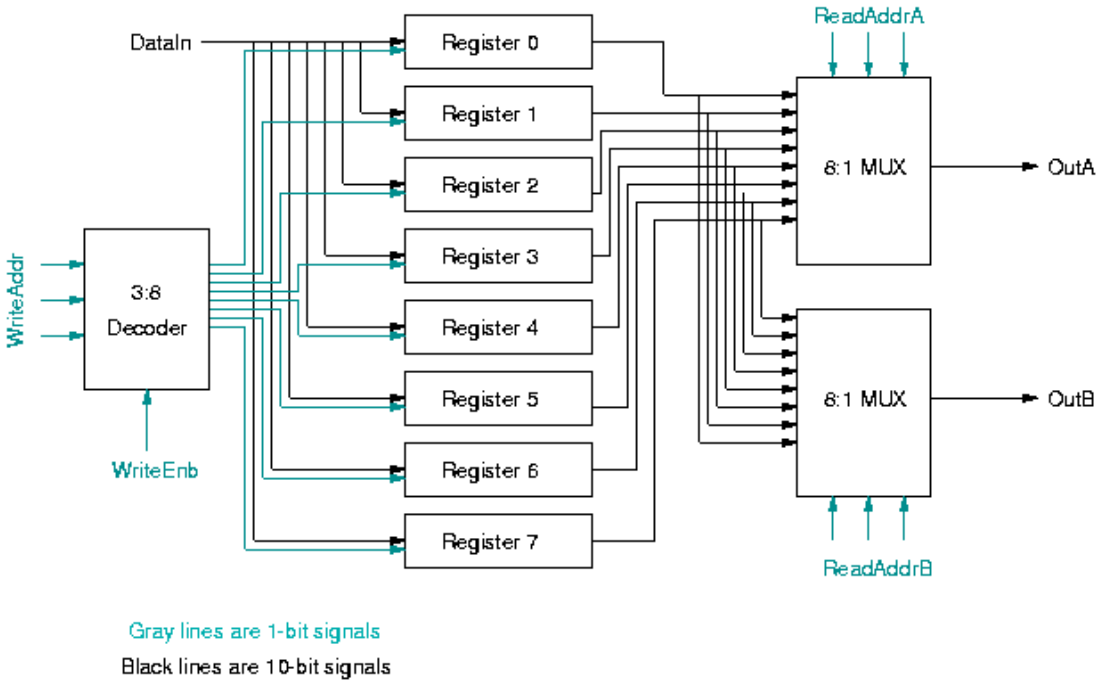


Et helt register kan bygges af D-latche. En til hver bit.

Registrene (eksempel på lille lagerblok)

En lagerblok består af et antal registre. Data på data indgangen føres frem til samtlige registre.

En addresse dekoder udvælger hvilket register der skal skrive. Multiplexore udvælger hvilke 2 registre der udlæses



Fra digitallogik til transistorer

Implementation af porte

Den grundlæggende funktionalitet kan implementeres på et utal af måder

Man kan f.eks. bruge tryk (luft eller vand) til at repræsentere 1 og mangel på tryk til at repræsentere 0. Eller omvendt. Og så bygge en hel computer op omkring det. Blot man kan lave en ventil der kan styres af tryk, så er den ged barberet.

Man kan også finde nogle klodser i Minecraft og sætte dem sammen.

Men vi vil interessere os for implementationer i CMOS: Complementary Metal Oxide Semiconductors.

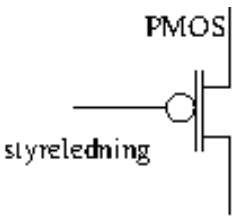
Transistorer

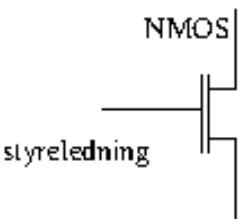
En transistor i CMOS er en kontakt som kan styres af en spænding.

Der er to slags transistorer.

- PMOS: kontakten er åben, når der ikke er spænding på styreledningen
- NMOS: kontakten er åben, når der er spænding på styreledningen

I CMOS bruges de altid parvis for at minimere strømforbrug.

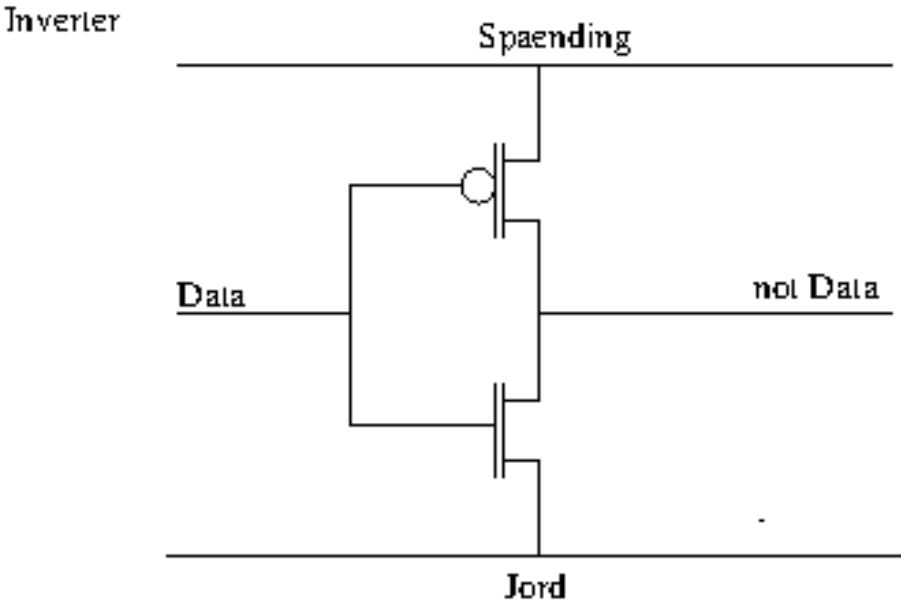




I CMOS skal man altid bruge PMOS transistorer til at løfte spændingen (dvs forbinde direkte eller indirekte til strømforsyningen), mens NMOS skal bruges til at sænke spændingen (dvs forbinde direkte eller indirekte til "jord").

Implementation af en inverter

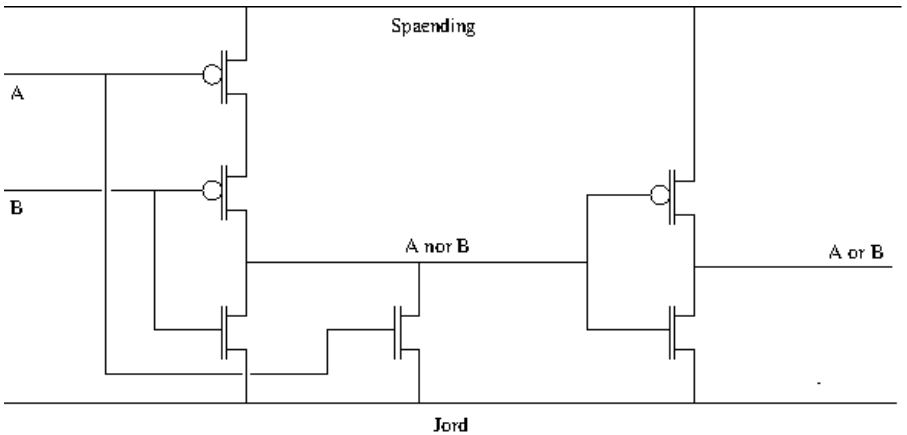
Den simpleste digitallogiske byggeklods:



Ved input tæt på 0 eller tæt på 1, bruges minimalt strøm. Men ved input i området midt imellem kan der løbe mere strøm. Det giver varme. Så tilstandsskift skal helst være hurtige.

Implementation af en Or-port

Og en Nor-port.



A	B	Nor	Or
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

En Or-port laves af en Nor-port med en inverter bagefter

Transistorer

Bedre end ventiler og vandslanger!

- Holdbare: Ingen bevægelige dele (bortset fra elektroner)
- Produceres ved en litografisk process.
- Kan laves ekstremt små
- Kan arbejde utroligt hurtigt (få pico-sekunder om at skifte, måske 5-20ps når de føder andre transistorer tæt på)
- i CMOS omsættes tilstrækkelig lidt energi til at man kan integrere *mange* transistorer på meget lidt plads (over en milliard på en kvadrat-centimeter)

Wikipedia har en fornuftig artikel, hvis du vil grave:
https://en.wikipedia.org/wiki/CMOS

Langt ude: http://visual6502.org/JSSim/expert-6800.html

Transistorer (II)

- Effekt er cirka proportional med skifte-frekvens, kvadratet på spændingen og kredsløbets kapacitans
- Vi kan påvirke skifte-frekvens på design tidspunktet
  - Design så ledninger skifter værdi så sjældent som muligt
  - F.eks. ved at stoppe for clock-signalet til de dele af chippen der ikke laver noget på et givent tidspunkt
- Maksimal skifte-frekvens er cirka proportional med spænding
- DVFS: Dynamic voltage-frequency scaling. Vi skruer løbende op og ned for både spænding og clock-frekvens efter behov.
- Alle nyere chips bruger en form for DVFS.

Produktionsproces

Sådan laver du en chip!

1. Reducer hele dit design til porte
2. Placer dine porte i et plan og forbind dem. Også kaldet "place and route". Der er kun ganske få adskildte lag hvori forbindelser kan løbe.
3. Beregn alle forsinkelser for signaler under en række forskellige scenarier (varians i ledningsevne, forsyningsspænding, skifte-aktivitet)
4. Beregn herfra max clock frekvens og strømforbrug
5. Hvis du er utilfreds, start forfra
6. Oversæt dit design til "masker" der kan styre en efterfølgende litografisk process
7. Producer.
8. Test og sorter. Sælg!

https://en.wikipedia.org/wiki/Semiconductor_device_fabrication

Moores lov

Moores lov er ikke en lov i samme forstand som tyngdeloven.

- Moores lov er et løfte. En afstemning af forventninger og mål for den industri der laver produktionsprocessen. Et løfte om løbende at sikre inkrementel forbedring af den litografiske process.
- Hver ny version af processen skal helst give transistorer som fylder halvt så meget som den foregående version
- Hver version kaldes iøvrigt en "process node" og angives ved størrelsen af den mindste detalje der kan laves.
- Vi har p.t. 7nm processen i brug.
- Hver opgradering til næste process-version er dyr. En fabrik koster ca 30 milliarder.

Intel IDF 2014

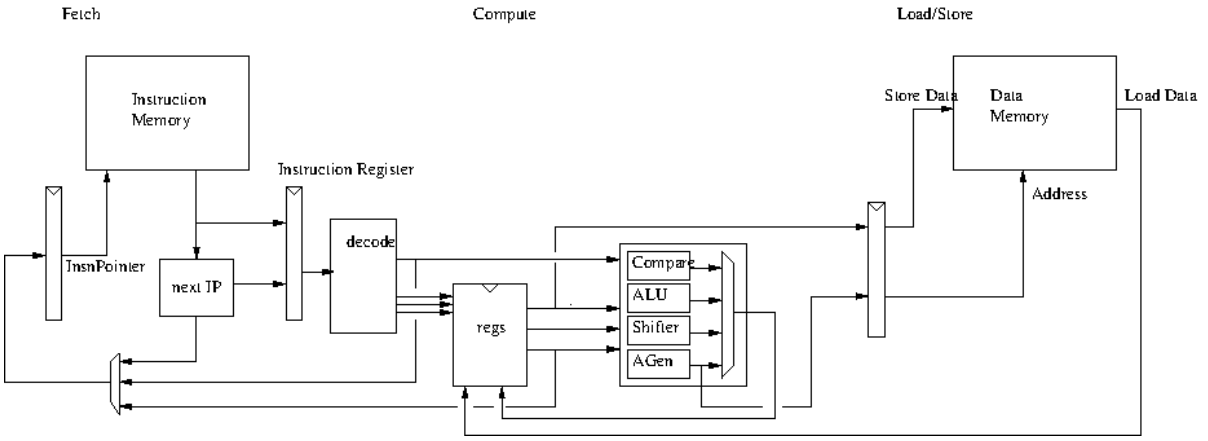
Opsamling

- Mange ting laves i CMOS: CPU'er, GPU'er, Kamera-sensorer, specielle regnemaskiner rettet mod AI, og meget, meget mere
    - Dyre ting er typisk fremstillet på den nyest og smarteste CMOS process.
    - Billigere komponenter er bygget på en ældre process.
- Implementationsteknologien (CMOS) kan udskiftes. Tænk nano-et-eller-andet. Eller tænk vandslanger og tryk-drevne ventiler. Digitallogik kan implementeres på mange måder.
- Hver fremstillingsprocess har sin egen balance mellem beregning og kommunikation. Så selvom mikroarkitekturer kan "portes" fra en fremstillingsprocess til den næste for at udnytte Moores lov, så skal mikroarkitekturen "rebalanceres" for at udnytte potentialet fuldt ud

Opsamling (II)

- Antallet af transistorer er vokset eksponentielt siden en gang i 70'erne
- Tilsvarende fremskridt for regnehastighed (udtrykt ved clock-frekvens) frem til ca 2005.
- Siden 2005 er max clock frekvens kun steget marginalt. Energiomsætning (varme) er nu væsentligste begrænsning
- Denne her teknologiudvikling (af CMOS fremstilingsprocessen) har stået på siden 70'erne. Den har drevet/muliggjort alt mulig anden teknologi. Den har forandret samfundet.
- Det bliver vanskeligere og vanskeligere at opretholde Moores lov. Muligvis er det allerede slut. Det afhænger lidt af den præcise definition.
- Selvom vi nærmer os "end of the road" for CMOS, så er der ikke noget der kan konkurrere. Endnu.

A2 mikroarkitektur - revisited



Spørgsmål og Svar