

# Digitallogik

Finn Schiermer Andersen

Baseret mestendels på figurer fra COD, App A.

## Agenda

CMOS Implementation

Simple byggeklodser (logiske porte)

Mere komplicerede byggeklodser

Tilstandselementer og Clocking

Simple byggeklodser. Vi kan bygge alt funktionalitet med de her primitiver.

Faktisk kan vi nøjes med NOR eller NAND



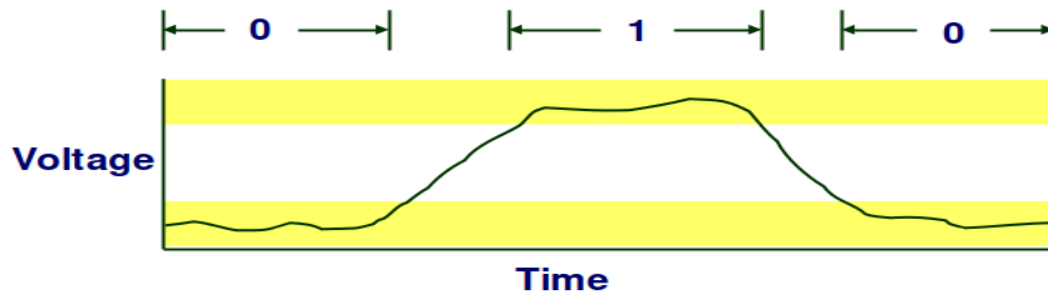
**Figure A.2.1 Standard drawing for an AND gate, OR gate, and an inverter, shown from left to right.** The signals to the left of each symbol are the inputs, while the output appears on the right. The AND and OR gates both have two inputs. Inverters have a single input.

Et lille indskud om CMOS

# Signaler og data

(fra "Computer Systems, a programmers perspective")

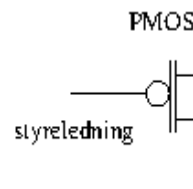
## Digital Signals



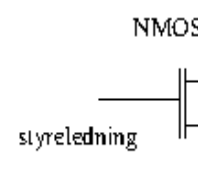
- Use voltage thresholds to extract discrete values from continuous signal
- Simplest version: 1-bit signal
  - Either high range (1) or low range (0)
  - With guard range between them
- Not strongly affected by noise or low quality circuit elements
  - Can make circuits simple, small, and fast

## CMOS Implementation af digitallogik

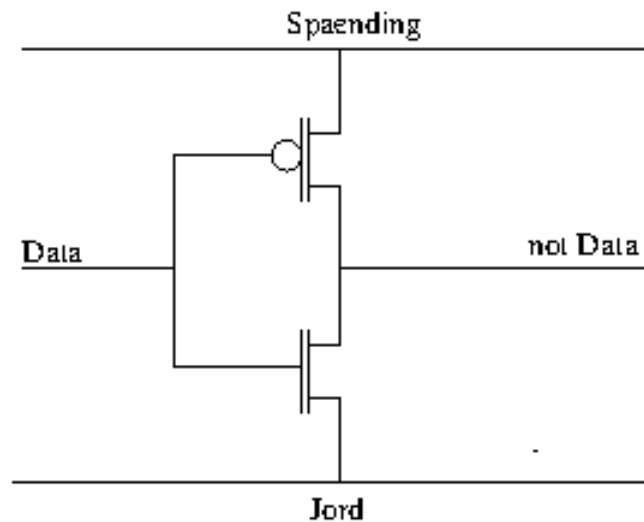
PMOS leder v lavt input



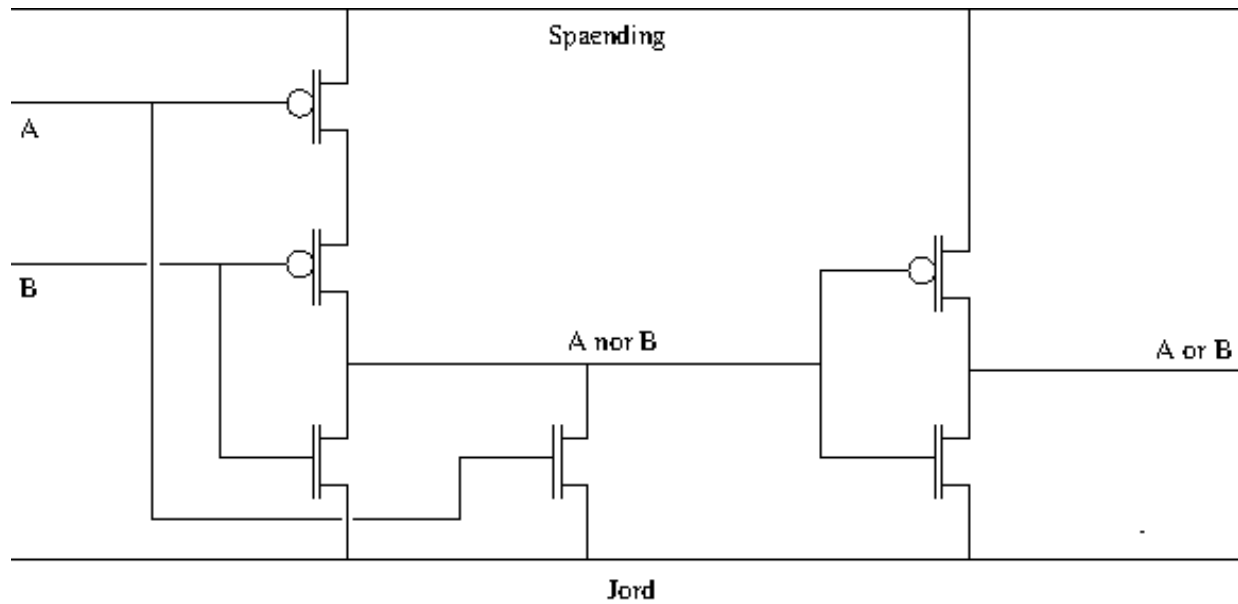
NMOS leder v højt input



Inverter



## CMOS-implementation af Or-port



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

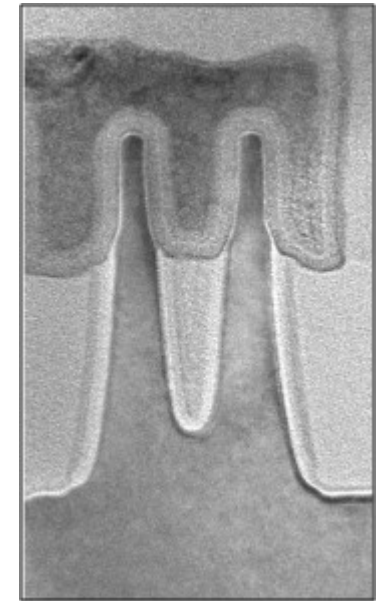
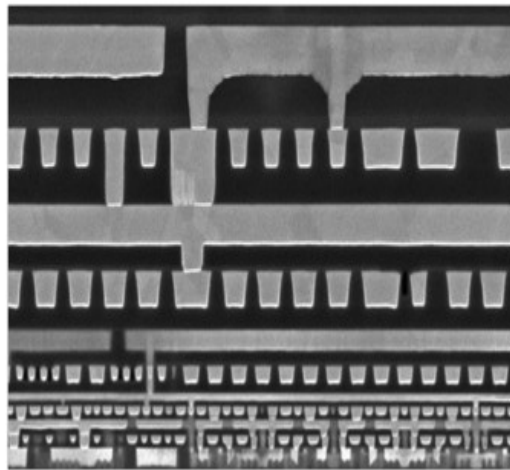
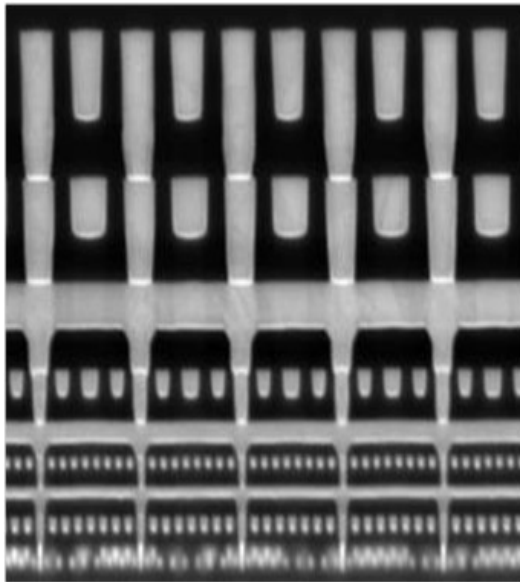
## Lidt om CMOS produktionsprocessen

1. Reducer hele dit design til logiske porte
2. Placer dine porte i et plan og forbind dem.  
Også kaldet "place and route". Der er kun 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](https://en.wikipedia.org/wiki/Semiconductor_device_fabrication)



Transistorer udgøres af flere lag med forskellige elektriske egenskaber. Lagene føres på et efter et via en kompliceret litografisk process.

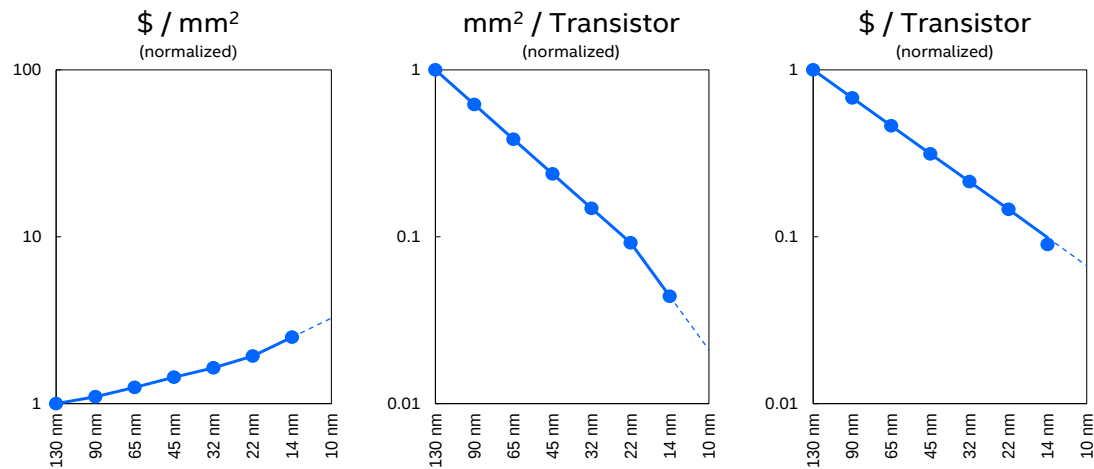


Transistor  
Ca 20nm bred

Eksempler fra Intel 22nm og 14nm CMOS

## Moore's law for dummies:

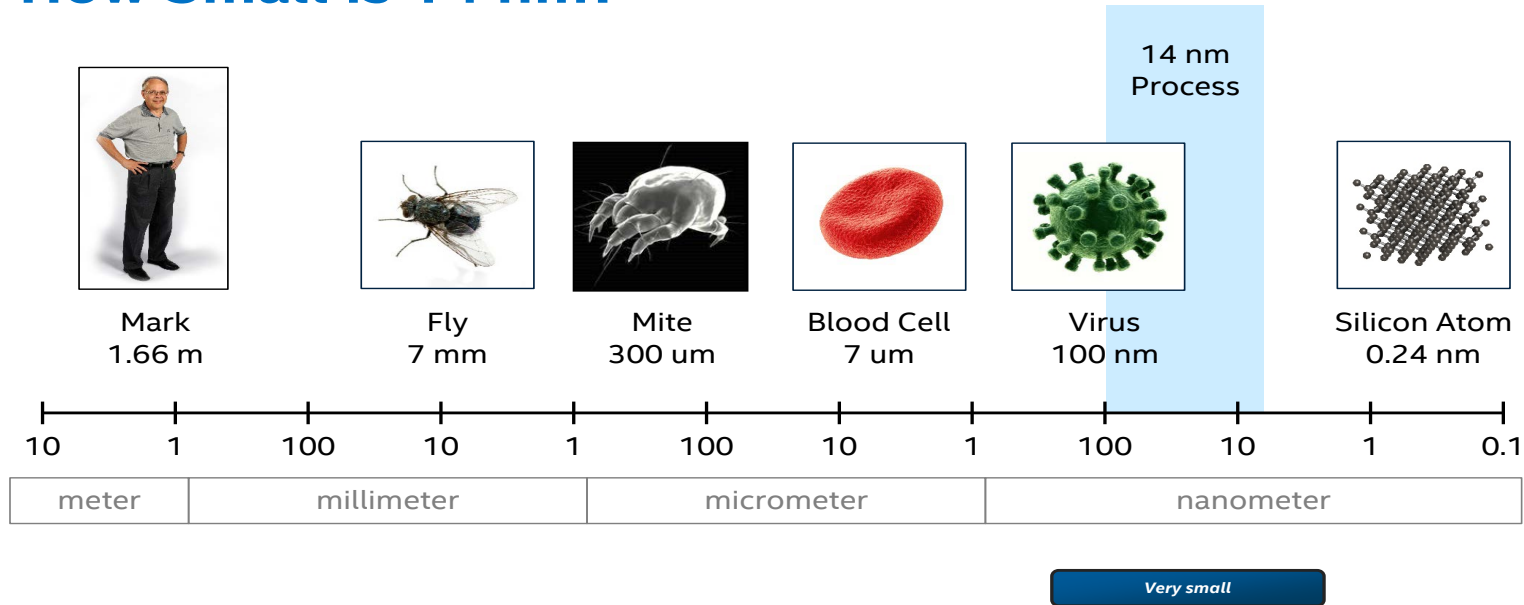
### Cost per Transistor



*Intel 14 nm continues to deliver lower cost per transistor*

IDF14

## How Small is 14 nm?



IDF14

Fra Intel Developer Forum 2014 – beskriver Intels 14nm proces  
Seneste processorer er lavet i 7 eller 5nm processer

# Udviklingen indenfor CMOS teknologi

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

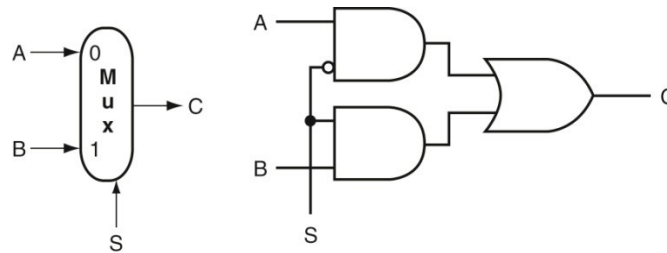
Denne her teknologiudvikling (af CMOS fremstillingsprocessen) har stået på i 50 år. Den har drevet/muliggjort alt mulig anden teknologi. Den har forandret samfundet.

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.

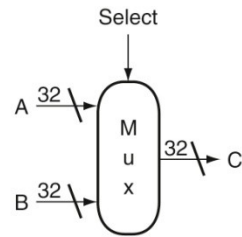
CMOS udflugt slut – tilbage til det digitallogiske niveau

En mux bruges til at vælge mellem mulige resultater.  
Det er det tætteste vi kommer på en if-then-else I hardware

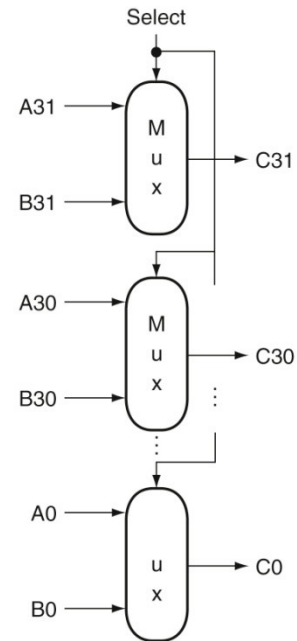


**Figure A.3.2 A two-input multiplexor on the left and its implementation with gates on the right.** The multiplexor has two data inputs (*A and B*), which are labeled *0 and 1*, and one selector input (*S*), as well as an output *C*.

Bemærk den lille “boble”...negation af input-signal



a. A 32-bit wide 2-to-1 multiplexor

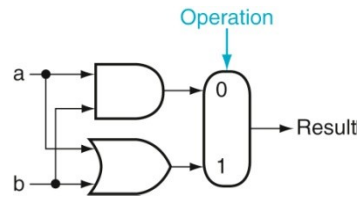


b. The 32-bit wide multiplexor is actually an array of 32 1-bit multiplexors

**Figure A.3.6 32-bit multiplexor. A multiplexor is arrayed 32 times to perform a selection between two 32-bit inputs. Note that there is still only one data selection signal used for all 32 1-bit multiplexors.**

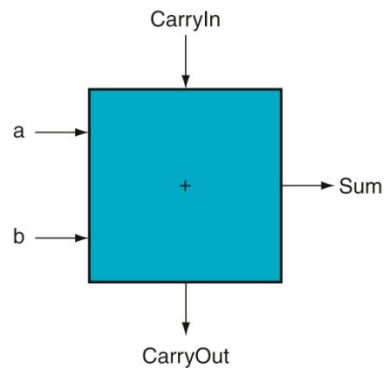
En proto-ALU :-) “bitslice”

Vi kan udvide med flere funktioner hvis resultater  
derpå udvælges af en større mux



**Figure A.5.1 The 1-bit logical unit for AND and OR.**

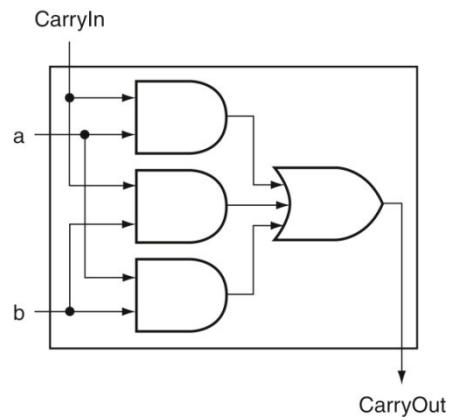




**Figure A.5.2 A 1-bit adder.** This adder is called a full adder; it is also called a (3,2) adder because it has three inputs and two outputs. An adder with only the a and b inputs is called a (2,2) adder or half-adder.

Inputs			Outputs		Comments
a	b	CarryIn	CarryOut	Sum	
0	0	0	0	0	$0 + 0 + 0 = 00_{\text{two}}$
0	0	1	0	1	$0 + 0 + 1 = 01_{\text{two}}$
0	1	0	0	1	$0 + 1 + 0 = 01_{\text{two}}$
0	1	1	1	0	$0 + 1 + 1 = 10_{\text{two}}$
1	0	0	0	1	$1 + 0 + 0 = 01_{\text{two}}$
1	0	1	1	0	$1 + 0 + 1 = 10_{\text{two}}$
1	1	0	1	0	$1 + 1 + 0 = 10_{\text{two}}$
1	1	1	1	1	$1 + 1 + 1 = 11_{\text{two}}$

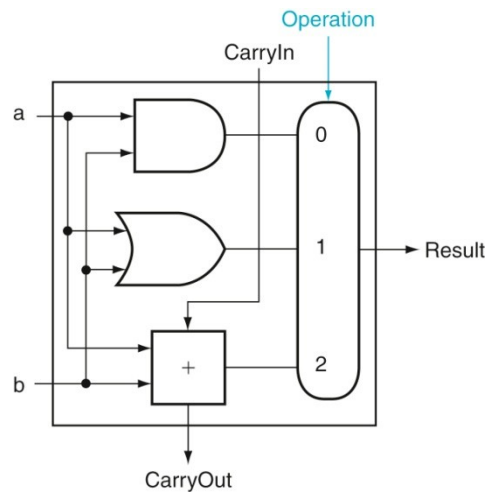
**Figure A.5.3 Input and output specification for a 1-bit adder.**



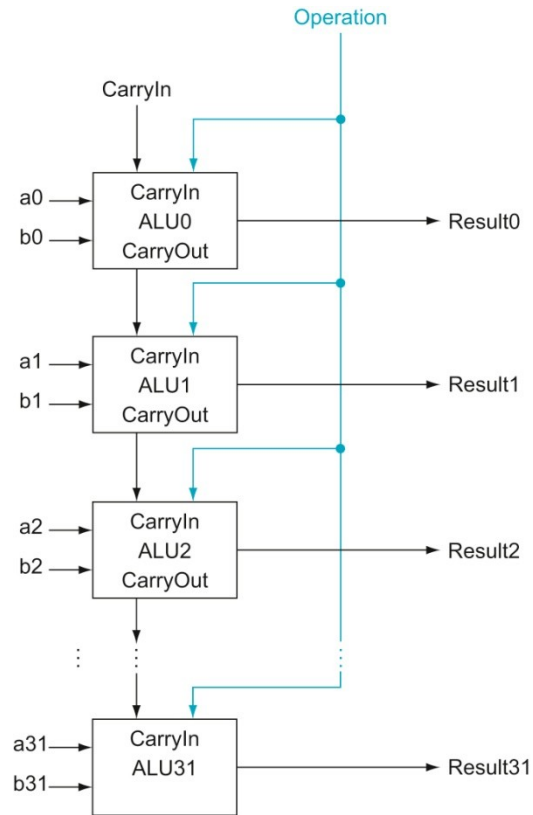
**Figure A.5.5 Adder hardware for the CarryOut signal.** The rest of the adder hardware is the logic for the Sum output given in the equation on this page.

Hvordan beregner vi “sum” signalet med brug af digitallogiske porte?

## En lidt mere ALU-agtig ALU (nu med “plus”)



**Figure A.5.6 A 1-bit ALU that performs AND, OR, and addition (see Figure A.5.5).**

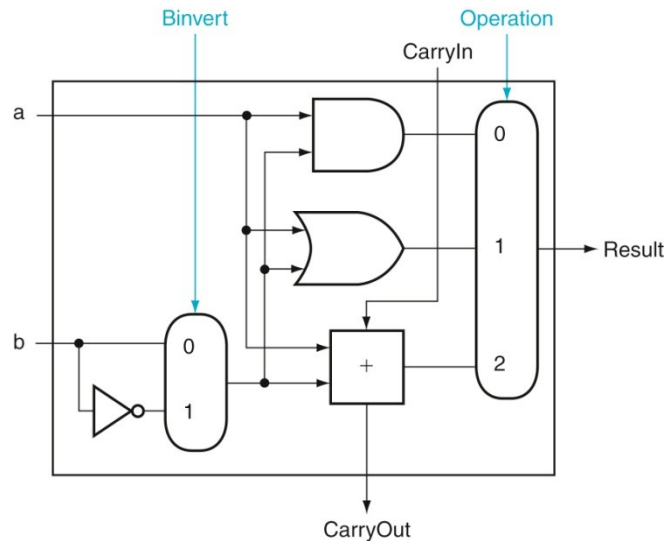


**Figure A.5.7 A 32-bit ALU constructed from 32 1-bit ALUs.** CarryOut of the less significant bit is connected to the CarryIn of the more significant bit. This organization is called ripple carry.

Vi skal også kunne trække fra

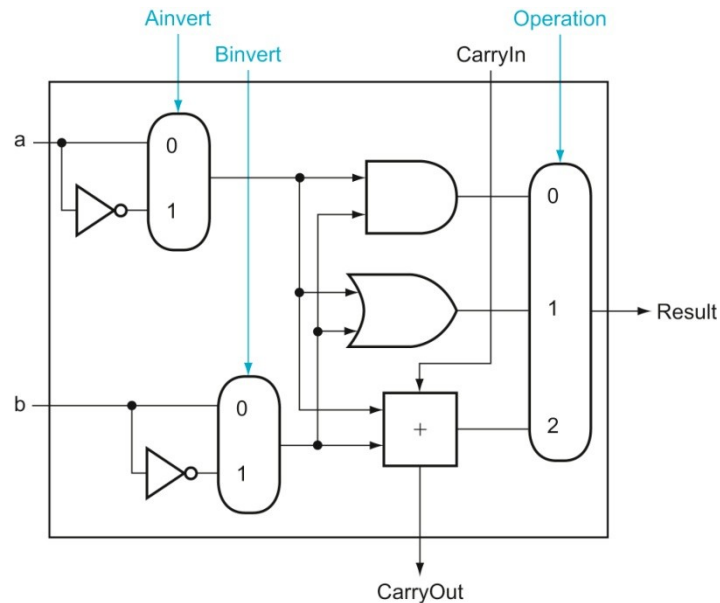
Det gør vi ved at negere og addere

Hvordan er det nu vi negerer et 2-komplement tal?



**Figure A.5.8 A 1-bit ALU bitslice that performs AND, OR, and addition on *a* and *b* or *a* and *b*.** By selecting *b* (*Binvert* = 1) and setting *CarryIn* to 1 in the least significant bit of the ALU, we get two's complement subtraction of *b* from *a* instead of addition of *b* to *a*.

Vi kan også beregne bitvist “NOR” – men bruger det vist ikke?  
(det er et levn fra MIPS udgaven af bogen)



**Figure A.5.9 A 1-bit ALU that performs AND, OR, and addition on *a* and *b* or *a* and *b*.** By selecting *a* (*Ainvert* = 1) and *b* (*Binvert* = 1), we get a NOR *b* instead of *a* AND *b*.



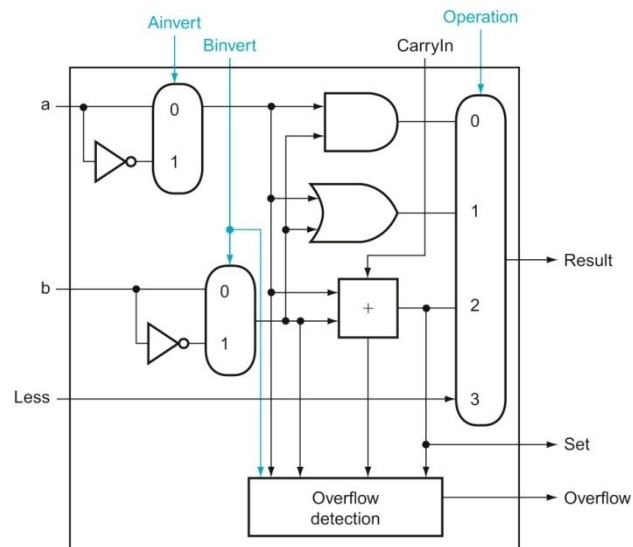
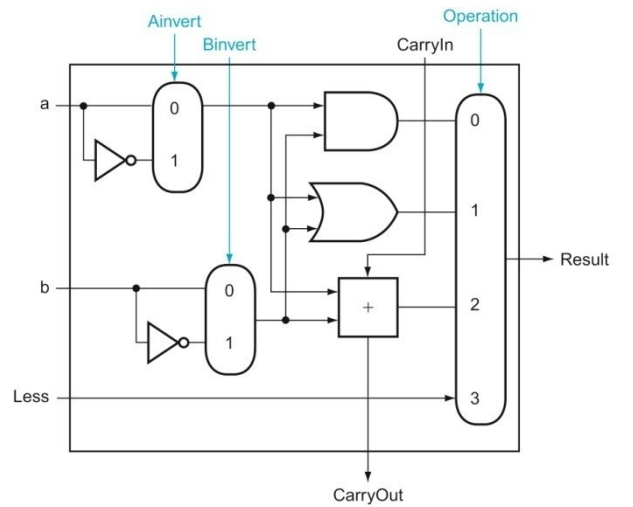
Hvordan kan vi sammenligne tal?

Udvidet ALU der kan implementere SLT  
(Set on Less Than instruktion)

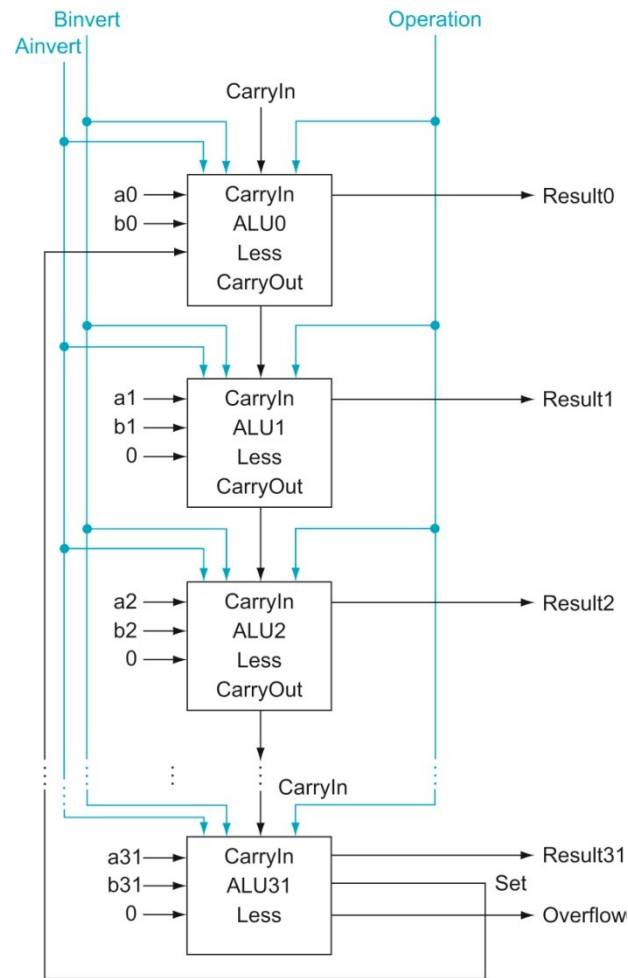
SLT:

$$A = (B < C) ? 1 : 0$$

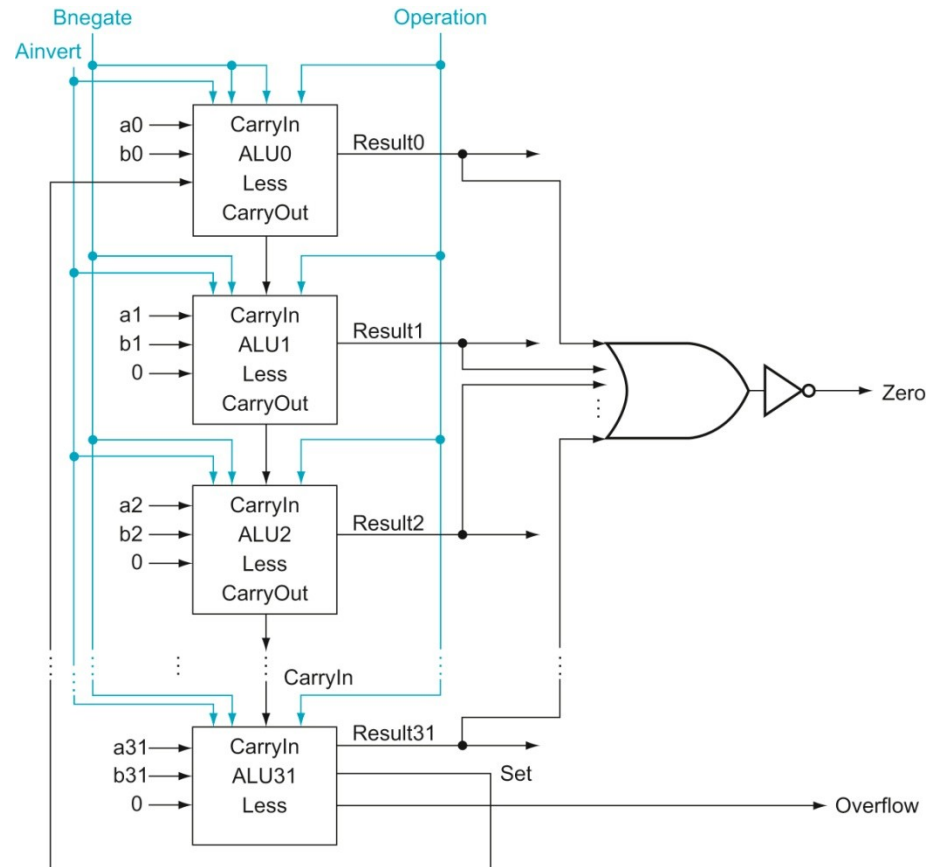
Og afgøre om  $A == B$   
(til f.eks. BEQ)



**Figure A.5.10 (Top) A 1-bit ALU that performs AND, OR, and addition on *a* and *b* or *b*, and (bottom) a 1-bit ALU for the most significant bit.** The top drawing includes a direct input that is connected to perform the set on less than operation (see Figure A.5.11); the bottom has a direct output from the adder for the less than comparison called Set. (See Exercise A.24 at the end of this appendix to see how to calculate overflow with fewer inputs.)



**Figure A.5.11 A 32-bit ALU constructed from the 31 copies of the 1-bit ALU in the top of Figure A.5.10 and one 1-bit ALU in the bottom of that figure.** The Less inputs are connected to 0 except for the least significant bit, which is connected to the Set output of the most significant bit. If the ALU performs  $a - b$  and we select the input 3 in the multiplexor in Figure A.5.10, then Result = 0 ... 001 if  $a < b$ , and Result = 0 ... 000 otherwise.

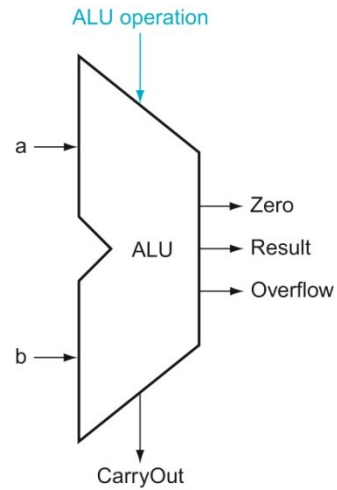


**Figure A.5.12 The final 32-bit ALU.** This adds a Zero detector to Figure A.5.11.

## Styring af ALUen

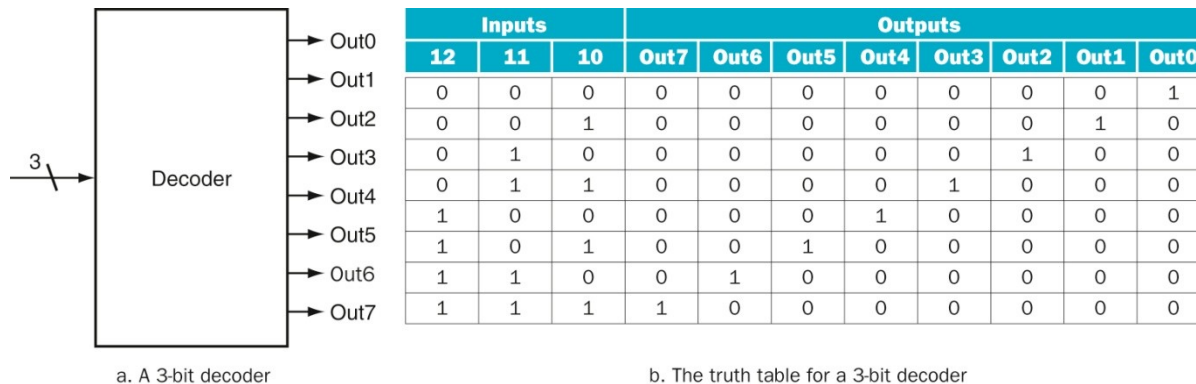
ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract

**Figure A.5.13** The values of the three ALU control lines, Ainvert, Bnegate, and Operation, and the corresponding ALU operations.



**Figure A.5.14** The symbol commonly used to represent an ALU, as shown in Figure A.5.12. This symbol is also used to represent an adder, so it is normally labeled either with ALU or Adder..

En afkoder (også kaldet en “one-hot encoder”)



**Figure A.3.1 A 3-bit decoder has three inputs, called 12, 11, and 10, and 23 = 8 outputs, called Out0 to Out7.** Only the output corresponding to the binary value of the input is true, as shown in the truth table. The label 3 on the input to the decoder says that the input signal is 3 bits wide.

Hvordan beregnes outputs ved brug af digitallogiske porte?

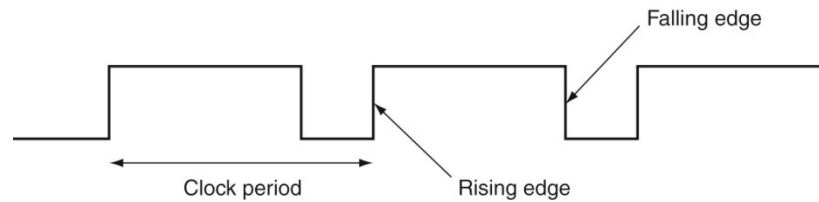
## Tilstandselementer

De byggeklodser vi har set indtil nu er “kombinatoriske”.  
De genberegner kontinuerligt output ud fra input

Nu skal vi se på byggeklodser der kan “holde” eller “huske”  
Data. De har en “tilstand” og kaldes tilstandselementer

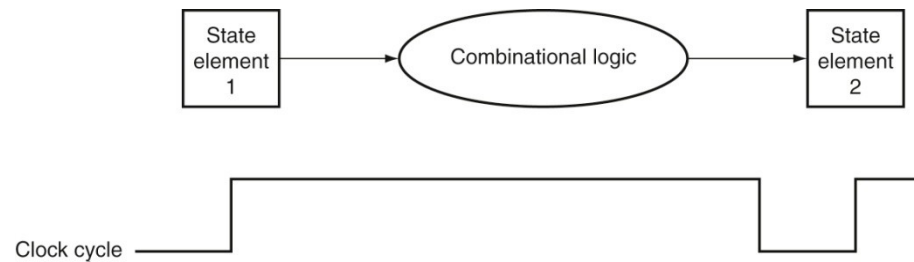


## Tilstandselementer styres af en “clock”



**Figure A.7.1 A clock signal oscillates between high and low values.** The clock period is the time for one full cycle. In an edge-triggered design, either the rising or falling edge of the clock is active and causes state to be changed.

En “clock” bestemmer hvornår tilstandselementer skal opdateres



**Figure A.7.2** The inputs to a combinational logic block come from a state element, and the outputs are written into a state element. The clock edge determines when the contents of the state elements are updated.

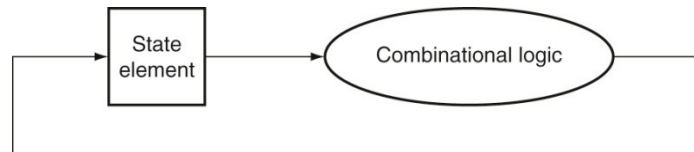
En vigtig figur:

Sammenhængen mellem clock og digitallogisk design

Signalet skal kunne nå rundt i hver clock-cycle

Længere signalvej → lavere clock frekvens

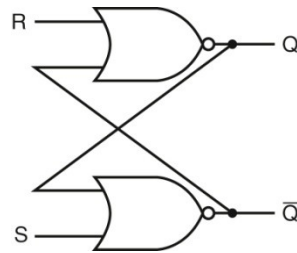
Kortere signalvej → højere clock frekvens



**Figure A.7.3 An edge-triggered methodology allows a state element to be read and written in the same clock cycle without creating a race that could lead to undetermined data values.** Of course, the clock cycle must still be long enough so that the input values are stable when the active clock edge occurs.

## En SR-latch

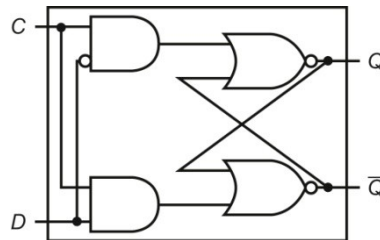
Det simpleste tilstandselement bygget af digitallogiske porte



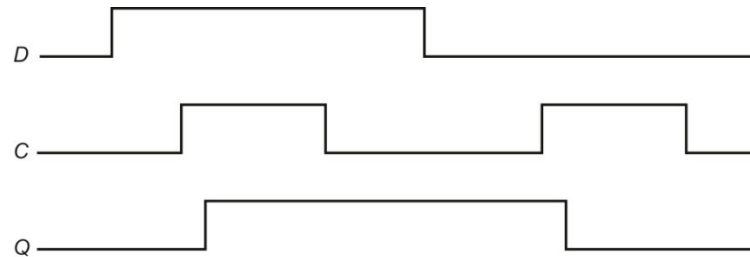
**Figure A.8.1** A pair of cross-coupled NOR gates can store an internal value. The value stored on the output  $Q$  is recycled by inverting it to obtain  $\sim Q$  and then inverting  $\sim Q$  to obtain  $Q$ . If either  $R$  or  $Q$  is asserted,  $\sim Q$  will be deasserted and vice versa.

## En D-Latch

Vi udvider SR-latchen med styring af S og R



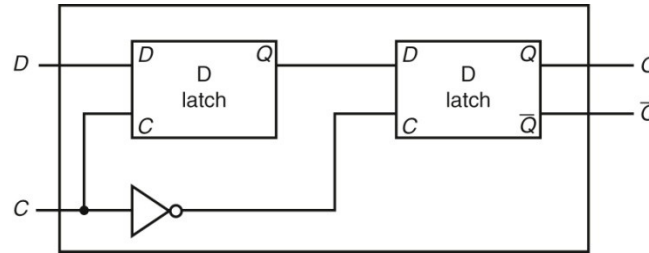
**Figure A.8.2 A D latch implemented with NOR gates.** A NOR gate acts as an inverter if the other input is 0. Thus, the cross-coupled pair of NOR gates acts to store the state value unless the clock input,  $C$ , is asserted, in which case the value of input  $D$  replaces the value of  $Q$  and is stored. The value of input  $D$  must be stable when the clock signal  $C$  changes from asserted to deasserted.



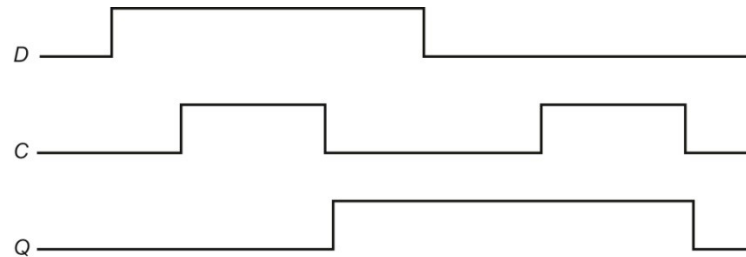
**Figure A.8.3 Operation of a D latch, assuming the output is initially deasserted.** When the clock, *C*, is asserted, the latch is open and the *Q* output immediately assumes the value of the *D* input.

## En D-Flip-flop

Vi går fra “level-triggered” til “edge-triggered” opdatering  
Det giver mere præcis styring fordi en “signalkant” varer  
meget kort tid.

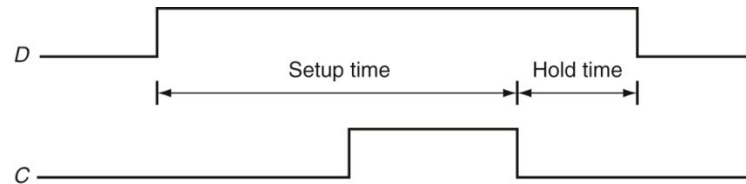


**Figure A.8.4 A D flip-flop with a falling-edge trigger.** The first latch, called the master, is open and follows the input  $D$  when the clock input,  $C$ , is asserted. When the clock input,  $C$ , falls, the first latch is closed, but the second latch, called the slave, is open and gets its input from the output of the master latch.



**Figure A.8.5 Operation of a D flip-flop with a falling-edge trigger, assuming the output is initially deasserted.** When the clock input (*C*) changes from asserted to deasserted, the *Q* output stores the value of the *D* input. Compare this behavior to that of the clocked *D* latch shown in Figure A.8.3. In a clocked latch, the stored value and the output, *Q*, both change whenever *C* is high, as opposed to only when *C* transitions.





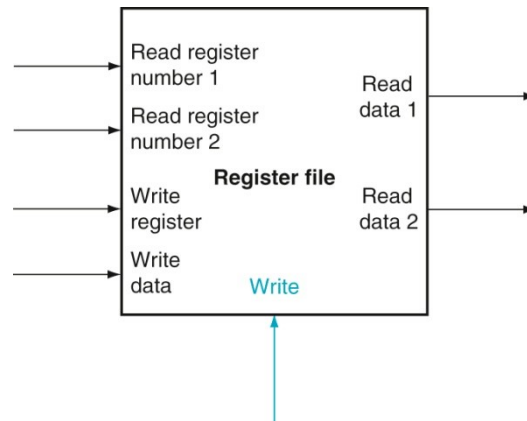
**Figure A.8.6 Setup and hold time requirements for a D flip-flop with a falling-edge trigger.**

The input must be stable for a period of time before the clock edge, as well as after the clock edge. The minimum time the signal must be stable before the clock edge is called the setup time, while the minimum time the signal must be stable after the clock edge is called the hold time. Failure to meet these minimum requirements can result in a situation where the output of the flip-flop may not be predictable, as described in Section A.11. Hold times are usually either 0 or very small and thus not a cause of worry.

Større (sammensatte) tilstandselementer

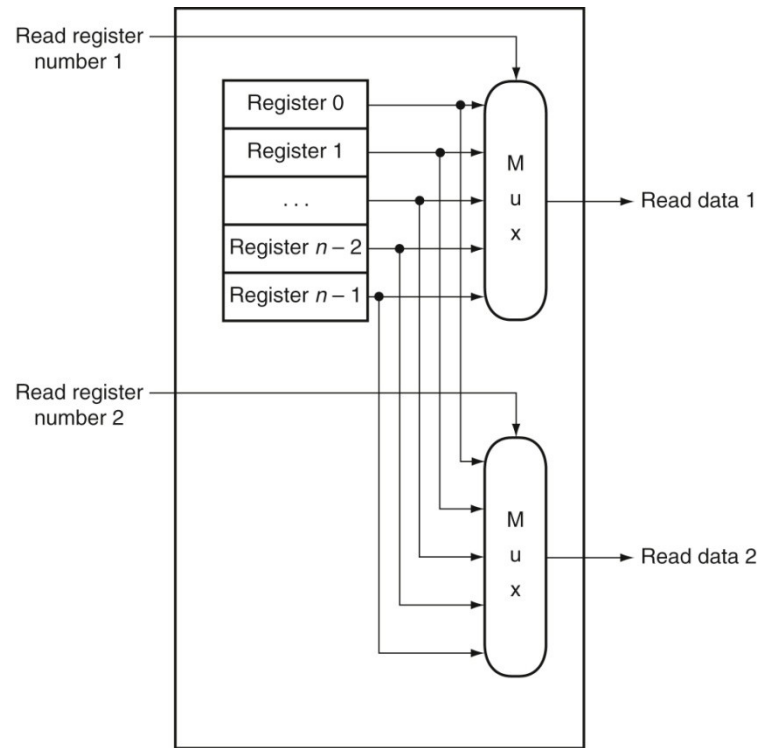
Registre og SRAM

## Registerbank



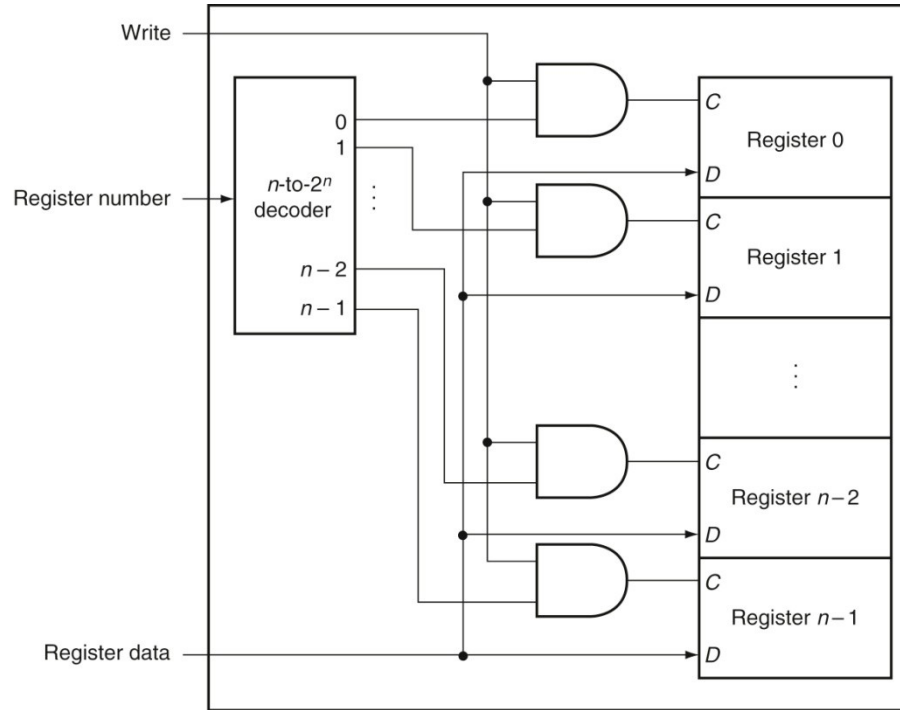
**Figure A.8.7** A register file with two read ports and one write port has five inputs and two outputs. The control input Write is shown in color.

## Registerbank – læsesiden – to læseporte



**Figure A.8.8** The implementation of two read ports for a register file with  $n$  registers can be done with a pair of  $n$ -to-1 multiplexors, each 32 bits wide. The register read number signal is used as the multiplexor selector signal. Figure A.8.9 shows how the write port is implemented.

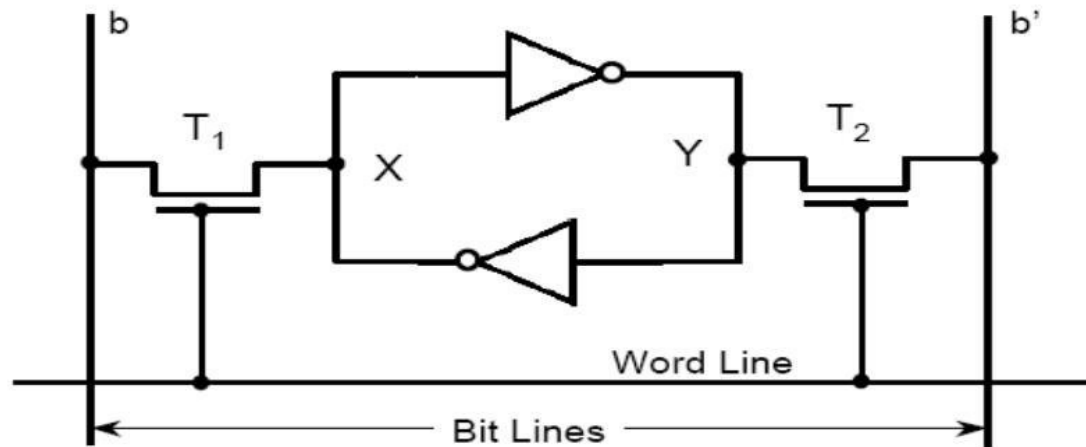
## Registerbank – skrivesiden – en skriveport



**Figure A.8.9** The write port for a register file is implemented with a decoder that is used with the write signal to generate the C input to the registers. All three inputs (the register number, the data, and the write signal) will have setup and hold-time constraints that ensure that the correct data are written into the register file.

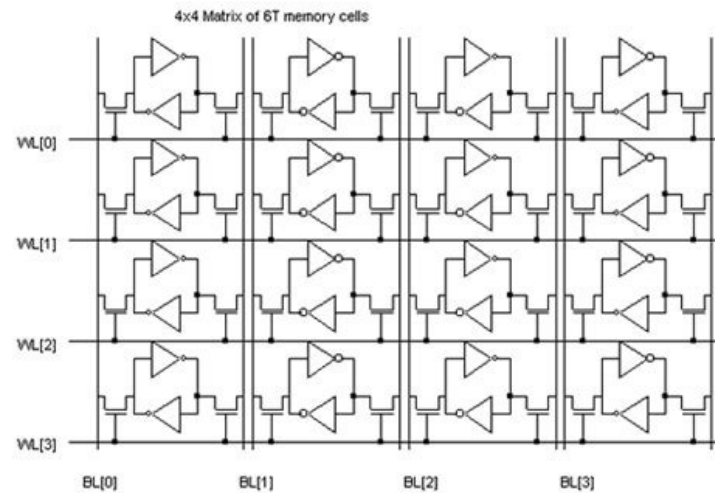
# A Static RAM Cell

A Static RAM Cell



Bit lines: Læsning/skrivning af valgt bit  
Word line: Udvælger "ord" af bits

## SRAM Organisation



BL = Bitlines

WL = Wordlines

DRAM

Dynamic Random Access Memory

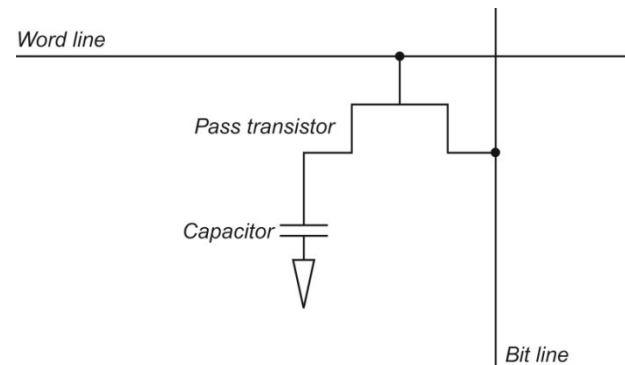
“dynamisk” = tilstand skal “opfriskes” periodisk  
for ikke at gå tabt



DRAM bruger en enkelt-transistor pr bit

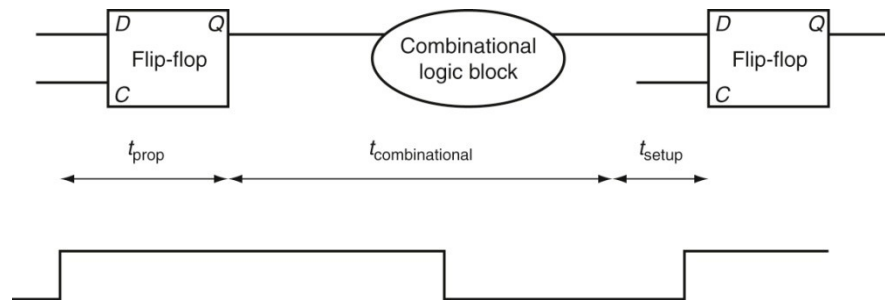
SRAM bruger en D-latch pr bit, hvilket typisk kræver 6 eller 8 transistorer

SRAM er hurtigere og dyrere, DRAM billigere og langsommere



**Figure A.9.5** A single-transistor DRAM cell contains a capacitor that stores the cell contents and a transistor used to access the cell.

## Digitallogisk design og maximal clock-frekvens (igen)



**Figure A.11.1** In an edge-triggered design, the clock must be long enough to allow signals to be valid for the required setup time before the next clock edge. The time for a flip-flop input to propagate to the flip-flop outputs is  $t_{prop}$ ; the signal then takes  $t_{combinational}$  to travel through the combinational logic and must be valid  $t_{setup}$  before the next clock edge.