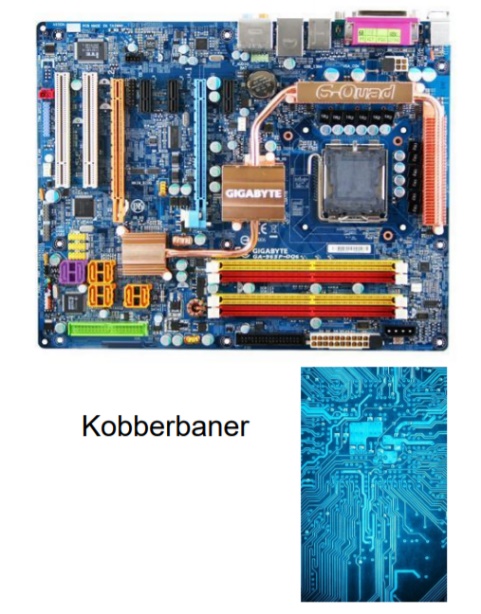
Busser notater

**Buss VS Port**

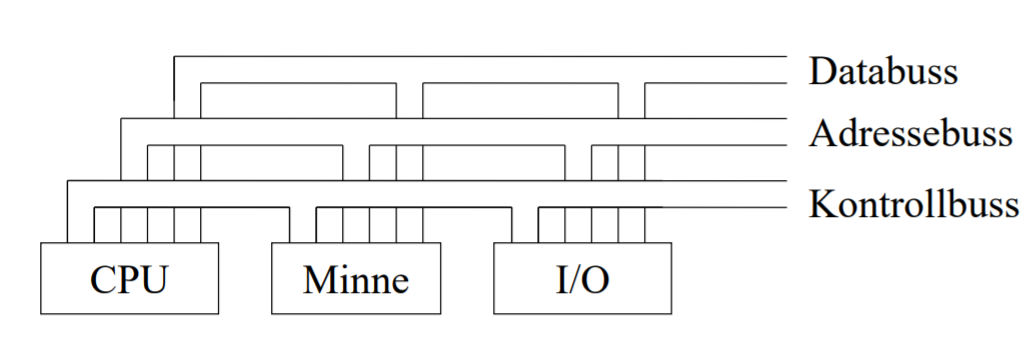
**Port**: da er det kun en mottaker og sender, det er oppgitt hvor dataene skal.

**Buss**: har flere komponenter på samme ledning/buss, som gjør at det er flere potensielle mottakere for en sender.

**Utforming**

 Busser kan være på kretskort med kobberbaner eller som eksterne kabler som SATA kabler eller USB.

**Parallelle busser**



**Adressebussen**: forteller hvilken komponent som skal motta dataene som sendes. Overføres binære adresser.

**Databuss**: Her overføres binære data

**Kontrollbuss**: Signaler som styrer og synkroniserer overføringen

Den teoretiske hastigheten til busser vil man aldri oppnå i virkeligheten siden en del av bussen brukes for å angi adressen. Derfor vil datahastigheten bli mindre enn teoretisk hastighet.

**Synkron Buss**

*Alle busser har en klokkefrekvens Hz, alle hendelser skjer kun etter en puls. Den kan starte etter pulsen, ikke imellom. Hz angir derfor hastigheten til en buss.*

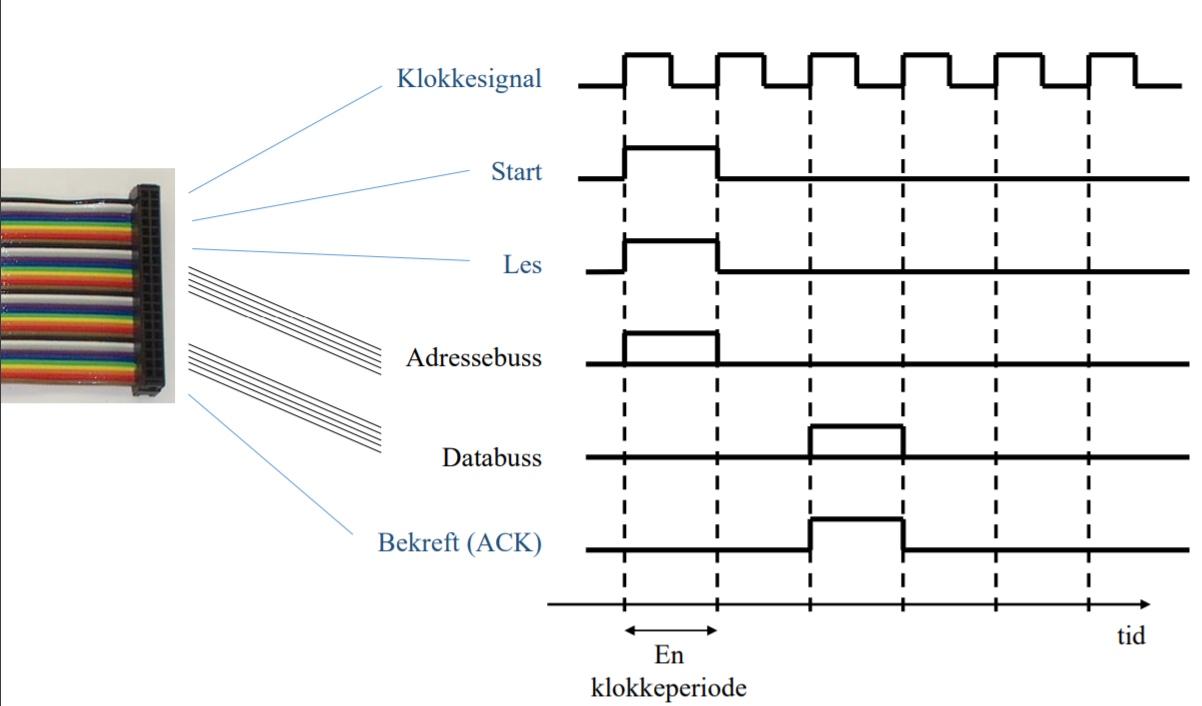
**Start**: skal våkne alle komponentene slik at de kan sjekke om de er mottakeren for signalet.

**Les**: varsler om at vi skal lese, ikke skrive.

**Adressebuss**: denne angir adressen til komponenten som vi ønsker å oppnå kontakt med.

**Databuss**: overfører data tilbake

**Bekreft (ACK):** Et signal som sendes for bekrefter at minnet har funnet det den skal finne. Derfor vet vi at alt er klart, og minnet har funnet fram dataene.

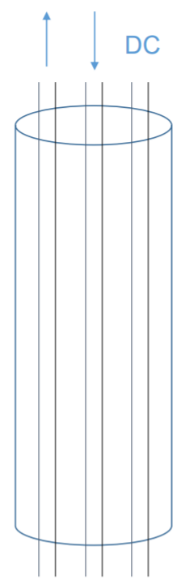


**Serielle busser**

**Eksempler på serielle busser:** USB (Universal Serial Bus), Firewire, sATA (erstatter ATA/IDE), SAS (erstatter parallell SCSI).

**Fordeler**:

* Høy båndbredde selv over lengre avstander.
* Små dimensjoner på koblingsmateriell.
* Mindre ledninger enn ved bruk av parallell buss.



**Eksempel på en seriell buss**. Her med 3 ledningspar som overfører

i ulike retninger. Det er ledningspar fordi overføring av spenning må ha + og -. Spenningen vi har er forskjellen mellom + og ., vi må ha med referanse. F.eks. et punkt er ikke 1 meter, men det kan være 1 meter over f.eks. gulvet.

**Båndbredde**

En parallell buss kan overføre flere bits samtidig ettersom bredden på den, derfor skulle man tro at den er raskere enn en seriell buss siden den må overføre en bit om gangen. Derimot etter hvert som man øker klokkefrekvensen mye så oppstår det problemer med parallelle bussen:

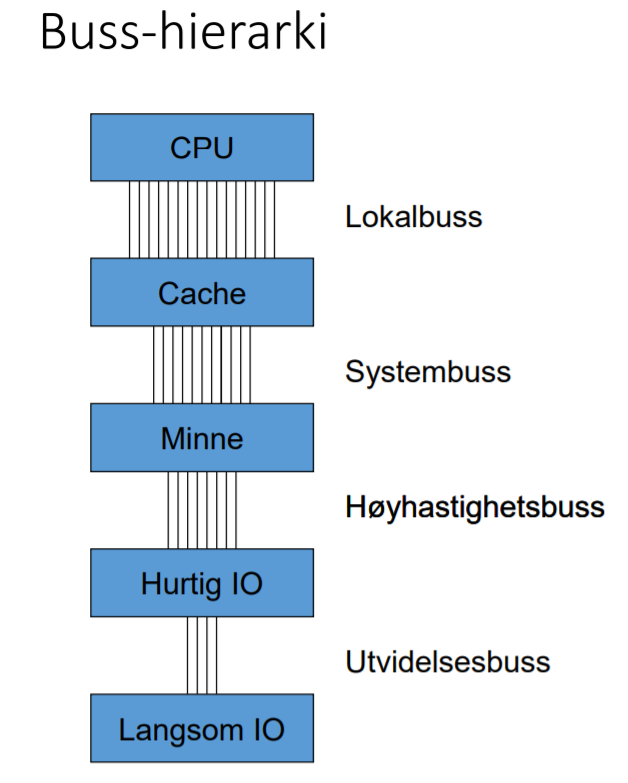
* Når bits sendes i parallell buss kan noen bits komme før noen andre som fører til problemer, altså det er en tidsforskjell. Hvis man sender på nytt før et signal på en annen ledning er kommet fram blir det feil.
* På en seriell buss kan man skru opp klokkefrekvensen mye mer en parallell buss fordi bitene sendes etter hverandre, og vi får ikke noe tidsforskjell problem.

**Forskjell**: Serielle busser raskere ved lengre hastigheter. Parallelle raskere ved korte avstander.

**Moderne busser**

Seriell er mer brukt enn parallell buss i moderne tid grunnet:

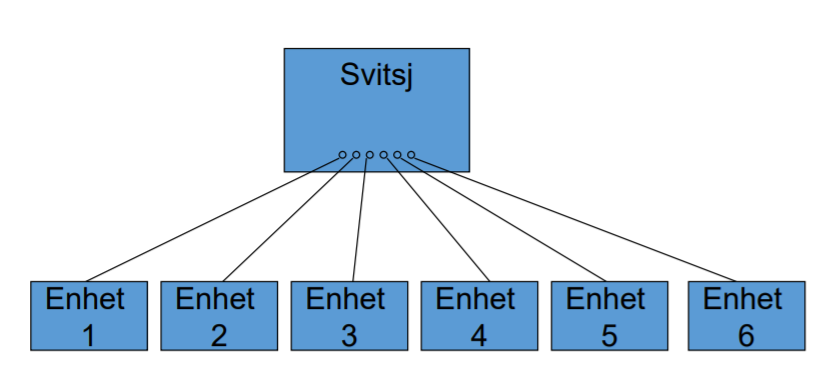
* Den bedre båndbredden
* Mindre fysiske dimensjonen.
* Hot Swap (koble til og fra utstyr når pc står på). Merk: hop swap kan også brukes med parallelle busser, men må være laget for å fungere med det, samt vi må ha programvare som håndterer det.

**Buss hierarki**

Jo høyere opp i hierarkiet jo høyere båndbredde og bredere.

**Svitsje teknologi**

* Kobler sammen enheter parvis.
* Flere samtidige sammenkoblinger kan settes opp.
* Hver sammenkobling er en punkt-til-punkt-forbindelse mellom to enheter som skal kommunisere.
* Denne teknologien er hentet fra teknologi brukt i nettverk.



**PCI VS PCI Express**

PCI Express er en svitsjet buss. Versjon 1 bruker 2,5GHz i hver retning.

Forskjellig utstyr bruker forskjellige hastigheter, f.eks. disker er langsommere enn grafikkort. Altså ulikt utstyr krever ulik båndbredde. Hvis en krever mer båndbredde, da gir vi den flere «lanes» til den enheten, som vi si flere serielle overføringer som overfører samtidig. F.eks. 4 «lanes» gir firedobbel båndbredde enn kun 1 lane.