1.5 Busser

# Introduksjon

* Busser er nødvendig for at PC-ens komponenter skal kunne kommunisere med hverandre.
* En buss består av ledninger som overfører elektriske signaler, og dermed knytter komponentene sammen.

# Parallell og seriell overføring

* Informasjonen som sendes over bussen sendes i form av bitmønstre. Det vil si at bussens oppgave er å overføre bit-verdier mellom komponentene. Hver enkelt bit-verdi overføreres som et binært spennings-signal over en ledning.

* Hvis bitene overføres etter hverandre i tid, kalles det en seriell overføring. Eks: om vi skal overføre 32 bits på denne måten, sendes de 32 bitene etter hverandre over samme ledning.
* Om vi bruker flere ledninger for å overføre flere bits samtidig kaller vi dette parallell overføring. Har vi for eksempel 32 ledninger kan vi overføre 32 bits samtidig.
* Serielle og parallelle busser har ulike egenskaper og ulike virkemåter.

# Port kontra buss

* Om vi har nøyaktig to komponenter som skal utveksle informasjon med hverandre kan vi bruke en buss for å koble de sammen. Siden det kun er to komponenter er det aldri noe tvil om hvem som skal mota informasjonen. Den ene sender informasjonen mens den andre mottar. Dette kalles gjerne for port istedenfor buss.
* Med en gang vi får flere komponenter må vi fortelle hvilke komponenter som skal ha hvilken informasjon, dette gjør at vi får behov for adressering av de ulike enheter slik at vi kan skille mellom dem på en enkel måte.

# Båndbredde

* Båndbredden er overføringskapasiteten en buss har. Denne oppgis i antall bits som kan overføres pr sekund.
* Når man oppgir båndbredde til en buss så er den potensielt maksimale overføringskapasiteten vi oppgir. I praksis oppnår man aldri så høyt båndbredde, i iallfall ikke over lengre tid.
* Det sendes mer enn bare data over bussen. Det må sendes signaler som synkroniserer og kontrollerer det som foregår på bussen.

# Arbitrering. Sentralisert/distribuert buss

* Som tidligere nevnt består en buss av et sett med ledninger som binder de ulike komponentene sammen. Enhetene kobles til bussen på forskjellige steder langs ledningene.
* Bussen er i seg selv passiv, og har ingen annen funksjon enn å overføre elektriske signaler
* Bussen er en delt ressurs, og bare en komponent kan skrive på bussen til et hvert tidspunkt.
* Siden bussen er en delt ressurs vil signalene som en komponent sender ut kunne leses av alle andre som er koblet på bussen. En form for én-skriver-og-alle-lytter.
* Når bit-verdier overføres vil det være en komponent som legger spenning ut på ledningene på bussen. Vi kaller dette for at komponenten skriver på bussen.
* De andre komponentene registrerer spenningsverdiene, vi sier at de leser fra bussen.
* For å unngå konflikt om hvem som skal skrive til bussen (for det er bare en komponent som kan gjøre dette om gangen, for å unngå kaos) skiller vi mellom to hoved metoder; sentraliserte og distribuerte busser.
* Forskjellen på sentraliserte og distribuerte busser er teknikken som brukes for å håndtere slike konflikter. Dette kalles ofte for arbitrering (som betyr å dømme/avgjøre).
* På en sentralisert buss finnes det en busskontroller som tildeler bussen til enheter ved behov. Denne busskontrolleren er en elektrisk krets som er tilknyttet bussen. Den er en del av brikkesettet.
* På en distribuert buss kan alle enheter reservere bussen så fremst den er ledig. I dette tilfellet må hver enhet inneholde mer elektronikk for tilgangskontroll.

# Overføringsretning

* Følgene begreper brukes i forbindelse med datakommunikasjon mellom to komponenter:
  + **Simpleks:** Bare den ene enheten kan sende informasjon. Den andre enheten kan bare motta.
  + **Halv dupleks:** Begge enheter kan både lese og skrive, men ikke samtidig. Når den ene skriver, må den andre vente til den første er ferdig.
  + **Full dupleks:** Begge enheter kan både lese og skrive samtidig. Det kan altså gå trafikk begge veier samtidig.

# Bruk av flere busser – bussherarki

* Siden de ulike komponentene i en PC har ulik hastighet på overføringen av data, egner en enkel buss seg dårlig. Vi har derfor flere busser med ulike overføringshastighet.
* Vi snakker om et buss-hierarki der de hurtigste bussene er på toppen, og så blir båndbredden lavere etter hvert som vi går nedover i hierarkiet.
* Hovedårsakene til at det er fornuftig å spre datatrafikk over flere busser:
  + Når vi sprer trafikken over flere busser blir det totale båndbredden større.
  + Vi kan ha ulik båndbredde på de ulike bussene.
  + Vi kan tilpasse bussen til spesielle behov.
* Buss-hierkiet:
  + **Minnebussen:**
    - Dette er den hurtigste bussen (sett bort ifra de interne bussene inne i prosessoren) er bussen mellom prosessor og minnet. Det stilles store krav til båndbredde siden prosessor og minne er de hurtigste komponentene i datamaskinen. Derfor er det hensiktsmessig å isolere denne trafikken på en egen buss.
  + **Hovedbuss:**
    - Den nest hurtigste bussen kobler sammen alle kontrollere (IO-moduler). Eksempler på slike kontrollere (IO-moduler) er diskkontoller, nettverkskort, grafikk-kort og mange andre.
  + **IO-bussene:**
    - Disse bussene danner forbindelsen mellom IO-utstyr og kontroller (IO-modul). Et eksempel på IO-buss er USB.

1.3 Parallelle busser

* En parallell buss består av et sett av parallelle ledninger, som gjør at mange bits overføres samtidig (i parallell) i hver sin ledning.
* En typisk parallellbuss i en datamaskin består av 50-100 ledninger. Disse deles inn i tre grupper som kalles delbusser. Disse overfører hver sin type signal:
  + **Databuss:**
    - Her overføres de binære data. Typisk består en databuss av 16, 32 eller 64 linjer. Antallet kalles databussens bredde og er en nøkkelfaktor for maskinens totale hastighet siden dataflytting ofte er en begrensende faktor i datamaskiner.
  + **Adressebuss:**
    - Her overføres (binære) adresser. Når CPU for eksempel skal lese en lokasjon i minnet, så legges adressen til minnelokasjonen på adressebussen. Bredden til adressebussen avhenger av minnekapasiteten til datamskinen. Den må være bred nok for en hvilken som helst adresse kan legges på adressebussen. Den brukes også til å adressere de forskjellige I/O-modulene.
  + **Kontrollbuss:**
    - Her overføres signaler som styrer og synkroniserer overføringen. Kontrollinjene brukes til å synkronisere og kontrollere utveksling av informasjon. Forskjellige buss-standarder kan ha ulike kontrollinjer.
* **Eksempel på hvordan delbussene fungerer:**
  + Når CPU skal lese en minnelokasjon vil den først reservere bussen slik at andre ikke kan skrive samtidig. Deretter legger CPUen ut adressen til minnelokasjonen på adressebussen samtidig som den setter et kontrollsignal som betyr les minne ut på kontrollbussen. Siden alle komponentene lytter på bussen vil minnet oppfatte at dette er en henvendelse til seg. Minneelektronikken leter opp den rette minnelokasjonen, og legger innholdet på databussen slik at CPUen kan lese det.

# Datakommunikasjon på en parallell buss

* De aller fleste parallelle busser er synkrone busser. Det betyr at de bruker et klokkesignal for å styre aktivitetene på bussene. Et klokkesignal er et signal som hopper mellom høy og lav spenning med helt faste mellomrom
  + En ny handling kan kun starte ved en ny klokkepuls
* Frekvensen på moderne parallelle busser kan være svært høye (glere hundre millioner pulser pr sekund).
* Antall pulser pr sek kalles klokkefrekvensen.
  + Oppgis i antall Hz (Hertz).

# Båndbredde på en parallell buss

* På en parallell buss er båndbredden en overføring pr klokkepuls.
* Vi oppnår aldri denne maksimale ytelsen.
  + Dette skyldes at enkelte klokkepulser må brukes til å overføre adresser og til å vente på at dataoverføringen skal klargjøres.
  + For å finne båndbredde må man altså multiplisere klokkefrekvens og bredden av databussen.

# Hva begrenser båndbredden

* For å øke båndbredden må vi øke databussbredde og/eller klokkefrekvensen.
* Hvis vi øker databussbredden, øker vi også de fysiske dimensjonene på bussen og alt tilkoblingsutstyr.
  + Parallelle busser har vært 8-bits, 16-bits,32-bits og 64-bits.
    - Av praktiske årsaker er det ikke gunstig å øke dden ytterligere.
  + Jo bredere bussen er (jo flere parallelle ledninger) jo større vil de individuelle forskjellene bli
  + Jo lengre ledningen til bussen er jo større blir ulikhetene i gangtid.
* Men vi kan heller ikke øke klokkefrekvensen i det evige.
  + Signalet bruker en viss tid gjennom ledningen, og siden alle elektriske signaler påvirkes av omgivelsene (og hverandre), vil det være små individuelle forskjeller på tiden som hver av signalene i en parallellbuss bruker.
  + Når vi øker klokkefrekvensen vil bitene komme så tett at tiden mellom dem nærmer seg forskjellene i gangtid.

# Hvor bruker vi parallelle busser?

* Tidligere var parallelle busser svært mye brukt, men nå brukes de primært som minnebuss mellom primærminnet og prosessoren.
* Minnebussen er den aller hurtigste bussen vi har på en datamaskin (bortsett fra interne busser som er innebygget i prosessoren).
  + På moderne PC er minnebussen en 64 bits parallell buss med klokkefrekvens i GHz-området.
  + Ser vi på et moderne hovedkort ser vi at prosessoren og primærminnet står med noen cm mellomrom.
    - Dette er for at minnebussen skal være kortest mulig, og dermed kunne ha høyest mulig klokkefrekvens.

1.4 Serielle busser

# Båndbredde på serielle busser

* Man skulle tro at det er mye smartere å bruke parallelle busser fordi i prinsippet bør jo en 16-bits parallell buss være 16 ganger raskere enn en seriell buss av samme størrelse.
  + Men siden en seriellbuss kun overfører en bit i slengen slipper man å tenke på vekselvirkningen mellom parallelle signaler.
    - Man har derfor ikke den samme begrensningen på klokkefrekvens og ledningslengde.
    - Man kan bruke høyere frekvenser og lengre ledninger uten å få problemer.
      * I tillegg har elektronikken blitt både god og billig som gjør at moderne seriebusser har høy båndbredde, lange ledninger og små dimensjoner på kontakter og annet koplingsmatriell.
  + De senere årene har serielle busser blitt mer og mer populær, og tatt over der det tidligere er brukt parallelle busser.

# Hvor bruker vi serielle busser

* IO-busser er busser for IO-utstyr (tastatur, mus, skanner, skriver, eksterne harddisker, kamera, videokamera osv.)
* Tidligere ble parallelle busser som IO-busser, men etter at kravet for IO-busser endret seg ble disse bussene for dårlig.
  + Det ble krav om økt båndbredde og krav om fleksibilitet og brukervennlighet økt
  + Dette førte til at de gamle bussene ble erstattet med nye og langt bedre IO-busser.
    - Det mest kjente eksemplet på en slik buss er USB (Universal Serial Bus).
* Tre ting som skiller de moderne IO-bussene fra de mer tradisjonelle bussene:
  + De erstattet parallelle busser med serielle busser.
    - Dette er fordi det er en stor fordel med små dimensjoner på koblingsmatriell, lange ledninger og god båndbredde.
  + Hot swap
    - Hot swap er når du kan koble til/fra en IO-buss uten å skru av datamaskinen. Hot swap er ikke eksklusivt for seriellebusser, og kan også gjøre med parallelle busser.
    - Hot swap forutsetter at elektronikken er laget for det og at programvaren håndterer det.
  + Tettere samarbeid mellom bussen og operativsystemet enn før.
    - På grunn av hot swap er operativsystemet mye mer involvert i bussoverføringene enn før. For at vi skal kunne bruke hot swap må operativsystemet inneholde kode som styrer IO-utstyret og får beskjed når noe kobles til og fra, slik at den kan laste inn rett driver og sørge for at utstyret kan tas i bruk.