Dagens temaer

- Dagens temaer hentes fra kapittel 3 i "Computer Organisation and Architecture"
- Kort repetisjon fra forrige gang
- Kombinatorisk logikk
- Analyse av kretser
- Eksempler på byggeblokker
- Forenkling vha. Karnaugh-diagram

Boolsk algebra

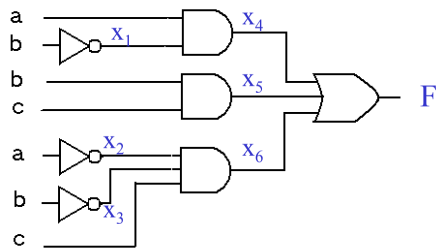
- Brukes for å beskrive funksjoner i digitale kretser
- Tre grunnleggende funksjoner: AND, OR og NOT
- Andre funksjoner som NAND, NOR, XOR og XNOR kan avledes fra AND, OR og NOT
- En Boolsk funksjon kan beskrives enten vha.
  - Sannhetsverditabell eller
  - Funksjonsforskrift
- To funksjoner er ekvivalente hvis de for alle input-kombinasjoner gir samme output

Design og analyse av logiske kretser

- En *kombinatorisk* krets har ingen hukommelse, dvs at output-verdiene kun er avhengig av nåværende input-verdier.
- I en *sekvensiell* krets er output-verdiene avhengige av både nåværende og tidligere input-verdier, mao har den hukommelse
- *Design* er prosessen med å sette sammen mindre blokker til større moduler, mens *analyse* er å finne ut hva en krets gjør.
- For å kontrollere at et design implementerer den ønskede funksjonen må man teste og analysere kretsen etter at den er ferdig.
- Som regel ikke mulig å teste fullstendig

Eksempel: kretsanalyse

Steg 1: Sett symboler på utgangen(e) og mellomsignaler, dvs signaler mellom porter:



Steg 2: Utled likningene for mellomsignalene og sett inn inngangssignalene:

$x_1 =$

$x_2 =$

$x_3 =$

$x_4 =$

$x_5 =$

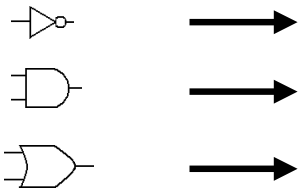
$x_6 =$

Steg 3: Utled tilslutt uttrykket for utgangssigalet F ved å sette inn verdiene for x_4, x_5 og x_6 som funksjon av inngangssignalene a, b og c

$F = x_4 + x_5 + x_6 =$

Kretsdesign

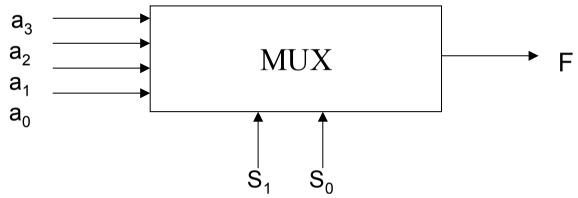
- Vanligvis brukes bare NAND eller bare NOR-porter for å implementere boolske funksjoner.
- NAND (og NOR) funksjonen er universell: Den kan implementere enhver boolsk funksjon.
- Bruker følgende regler:
  de Morgans teorem (ab)' = a' + b'
  (a')' = a



- Trenger ofte større byggeblokker enn bare AND, OR og NOT-porter (eller NAND/NOR) når man designer kretser.
- Mange mye brukte moduler har egne navn:
  - Multiplexer
  - Dekoder
  - Enkoder
  - Adder
  - Latch (1-bits minnekrets)
  - Flip-flop (1-bits minnekrets)
  - Skiftregister (Fler-bits minnekrets)

Multiplekser

- **Multiplekseren** sender ett av mange input-signaler ut på en output-linje.
  - Hvilken inputlinje som velges bestemmes av **select**-signalene
  - Antall input-linjer er alltid en potens av 2



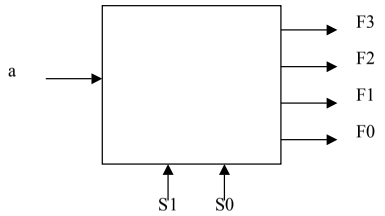
- Select-linjene er innsignaler, og utgangssignalet lik ett av de fire inngangssignalene:

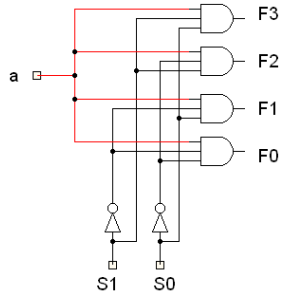
S_1	S_0	F
0	0	a_0
0	1	a_1
1	0	a_2
1	1	a_3

- Kan designes f.eks slik:

Demultiplekser

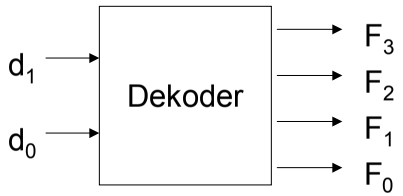
- **Demultiplekseren** gjør det motsatte av multiplekseren og sender ett input-signal ut på en av mange output-linjer.
  - Hvilken outputlinje som velges bestemmes av **select**-signalene
  - Antall output-linjer er alltid en potens av 2





Dekoder

- En **dekoder** setter én av 2^n outputlinjer til 1, avhengig av input-verdien på n input-linjer (konverterer fra 2-talls- til 1-tallssystemet)



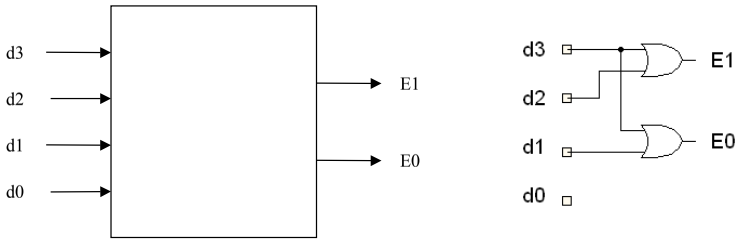
- Sannhetsverditabellen for en dekoder er som følger

d_1	d_0	F_3	F_2	F_1	F_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

- Dekoderen kan implementeres slik:

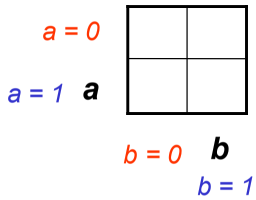
Enkoder

- En **enkoder** utfører motsatt operasjon av en dekoder og setter 2^n outputsignaler som funksjon av verdien på n inputsignaler

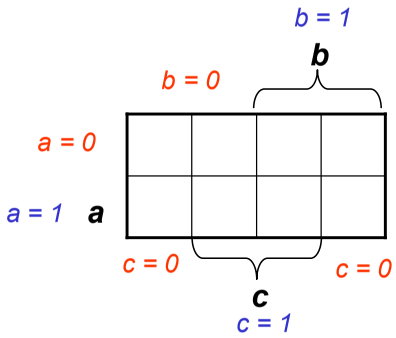


Karnaughdiagram

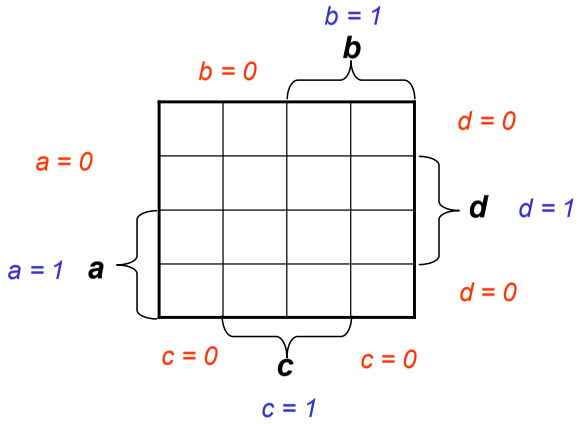
- Spesiell sannhetsverdi-tabell som brukes til å forenkle Boolske funksjoner
- Tabellen tegnes som et rutenett med 2^n ruter for en funksjon med n variable
- Langs sidene merkes de ruter hvor hver variabel er '1' (ikke-invertert) og '0' (invertert).
- Eksempel: Karnaughdiagram for funksjon med 2 variable 'a' og 'b'



Karnaughdiagram for funksjon med 3 variable



Karnaughdiagram for funksjon med 4 variable

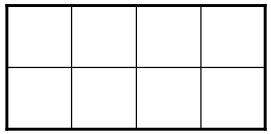


Forenkling av funksjoner

- **Steg 1**: Tegn Karnaugh-diagram av riktig størrelse
- **Steg 2**: Sett et '1' i de rutene der for hvor funksjonen er '1', og '0' ellers
- Eksempel: Karnaugh-diagram for funksjonen F=ab + ac + abc

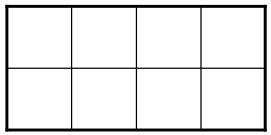
Forenkling av funksjoner (forts.)

- **Steg 3**: Kombinér naboruter med '1' til **så store som** mulig rektangler med 1, 2, 4, 8 osv ruter. Ruter med '1' kan godt deles av flere rektangler for å få dem så store som mulig. Kanter/hjørner er naboer med andre kanter/hjørner
- Eksempel: F=ab + ac + abc



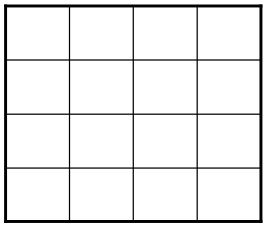
Forenkling av funksjoner (forts.)

- **Steg 4:** For hvert rektangel fra steg 3, finn ut hvilke variable som **ikke** skifter verdi innen rektangelet
- **Steg 5**: De variablene som ikke endrer verdi innenfor et rektangel AND'es sammen og utgjør et ledd i den forenklede funksjonen



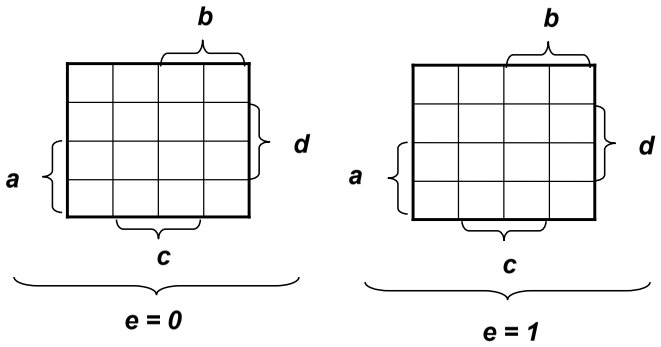
- **Steg 6**: Den forenklede funksjonen består av alle leddene fra steg 5 OR'et sammen

Eksempel: Forenkle funksjonen F= xyz + x'y'z + xzw + xy'zw'



Merknader til Karnaughdiagram (1)

- Karnaugh-diagram brukes sjelden for funksjoner med 5 eller mer variable



Merknader til Karnaughdiagram (2)

- Hvor man plasserer variabelnavnene er det samme, bare man får listet opp alle mulige kombinasjoner