UNIVERSITETET
I OSLO

Dagens tema

- Dagens tema hentes fra kapittel 3 i “Computer Organisation and Architecture”
- Sekvensiell logikk
- Flip-flop’er
- Tellere og registre
- Design av sekvensielle kretser
- (Tilstandsdiagram)

2009

Sekvensiell logikk

- Hvis output fra en krets kun er avhengig av nåværende input, kalles den ***kombinatorisk***
- Hvis output fra en krets er avhengig av nåværende og tidligere input, kalles kretsen ***sekvensiell***. Den må da inneholde minne eller hukommelse
- Alle datamaskiner inneholder en blanding av kombinatorisk logikk og hukommelse.
- Hukommelse finnes i mange varianter avhengig av hva de skal brukes til:
  - I RAM brukes spesialisert teknologi basert på lagring av elektriske ladinger (kondensatorer)
  - Inne i CPU'en brukes hukommelse basert på logiske porter.
  - En mengde andre typer som Hard-disk, DVD, CD, FPGA, Flash, (E)PROM etc. (Kommer tilbake til disse senere i kurset)

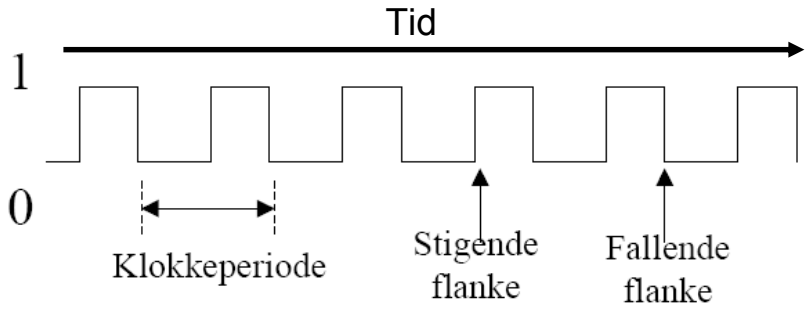
Sekvensiell logikk

- I ***synkrone*** sekvensielle kretser skjer endringen(e) i output samtidig med endringen i et såkalt ***klokkesignal***, eller når klokkesignalet enten er '0' eller '1'

- I ***asynkrone*** sekvensielle kretser skjer endringene i output med en gang uten noe klokkesignal.

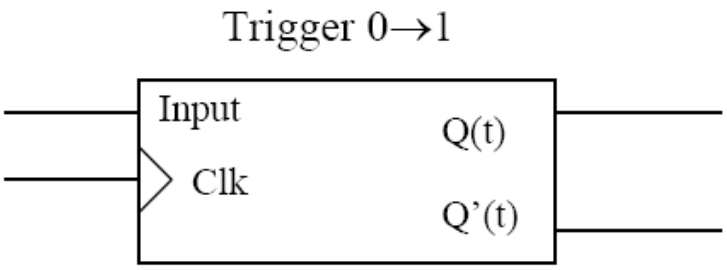
- Asynkrone kretser er som oftest raskere, men benyttes sjelden fordi de er mye vanskeligere å designe og teste

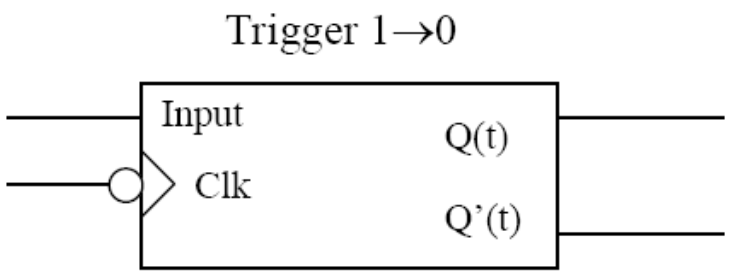
- Klokkefrekvensen f = 1/klokkeperioden
  - Høyere frekvens betyr kortere tid mellom hver gang en endring kan skje
  - Pentium 4 kjører på over 3 GHz
  - Eksperimentelle transistorer kan skifte mellom '0' og '1' med en frekvens på 507 GHz!



Flip-Flop

- Et 1-bits hukommelseselement som kan lagre ett bit kalles en ***flip-flop***
- En ny verdi kan bare leses inn og lagres når klokkesignalet går fra 0→1 (eller 1→0, avhengig av konstruksjonen)
- En flip-flop'er har enten en eller to innganger (pluss klokkesignal) og en utgang
- Finnes flere ulike typer flip-flop'er, og typen bestemmer hva som lagres gitt inngangs-signalet/ene, og hva som er lagret fra før
- Som regel finnes det et ekstra output-signal som gir den inverterte verdien av det andre output-signalet



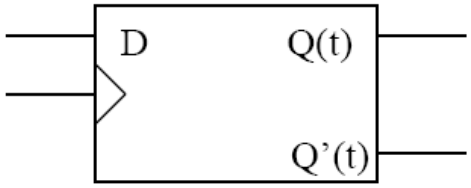


Karakteristisk tabell og karakteristisk ligning

- Virkemåten til en flip-flop beskrives ved ***karakteristisk tabell***
- En karakteristisk tabell er en sannhetsverditabell hvor tidsaspektet er representert
- Output er bit som lagres etter at klokkesignalet har endret verdi 0→1 (***neste tilstand***)
- Input er både utvendige signaler inn til flip-flop'en, og bit-verdien som er lagret før klokkesignalet endrer verdi 0→1 (***nåværende tilstand***)
- Hvis virkemåten beskrives med et boolsk uttrykk eller en likning, kalles denne for en ***karakteristisk likning***.
- Hvis vi ønsker å vite hvilke input vi må ha får å få en bestemt neste tilstand når vi kjenner nåværende tilstand, brukes en ***eksitasjonstabell***

D flip-flop

Symbol:



Karakteristisk likning: Q(t+1) = D

Karakteristisk tabell:

D	Q(t)	Q(t+1)
0	0	0
0	1	0
1	0	1
1	1	1

Karakteristisk tabell:
(alternativ form)

D	Q(t+1)
0	0
1	1

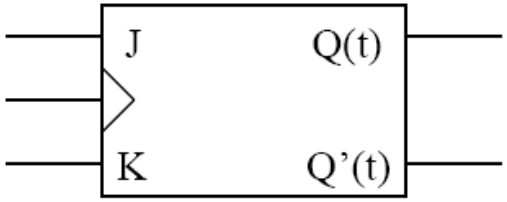
Eksitasjonstabell:

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

2009

JK Flip-flop

Symbol:



Karakteristisk likning: Q(t+1) = JQ'(t) + K'Q(t)

Karakteristisk tabell:

Q(t)	J	K	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Karakteristisk tabell:
(alternativ form)

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q'(t)

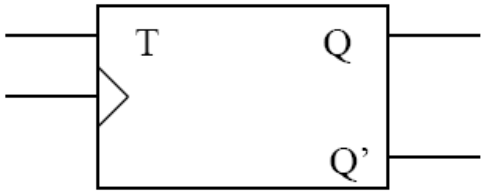
Eksitasjonstabell:

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

2009

T flip-flop

Symbol:



Karakteristisk likning: Q(t+1) = TQ'(t) + T'Q(t)

T flop-flop'en er det samme som en JK flip-flop hvor inngangene er koblet sammen, dvs. J = K

Karakteristisk tabell:

Q(t)	T	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

Karakteristisk tabell: (alternativ form)

T	Q(t+1)
0	Q(t)
1	Q('(t)

Eksitasjonstabell:

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

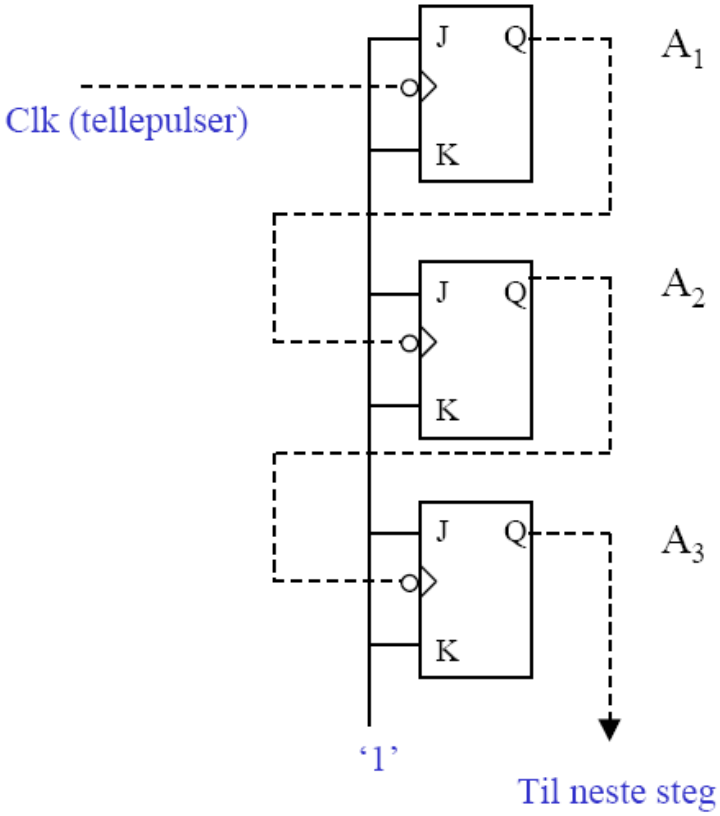
2009

Tellere

- Tellere er svært sentrale komponenter i alle digitale system
- Brukes i klokker, kontroll av programeksekvering, styring av andre komponenter, generering av bit-mønstre, osv.
- Designes ved hjelp av flip-flop'er og ekstra porter
- Finnes i mange varianter:
  - Binære/desimale
  - Opp/ned
  - Rippel(asynkrone)/Synkrone
  - Med/uten parallel innlasting av startverdi
  - Med/uten nullstilling

Rippelteller

- I en rippelteller trigges bare den første (=minst signifikante bit) flip-flop'en av et klokkesignal (tellepulser); de andre trigges av output fra forrige flip-floper

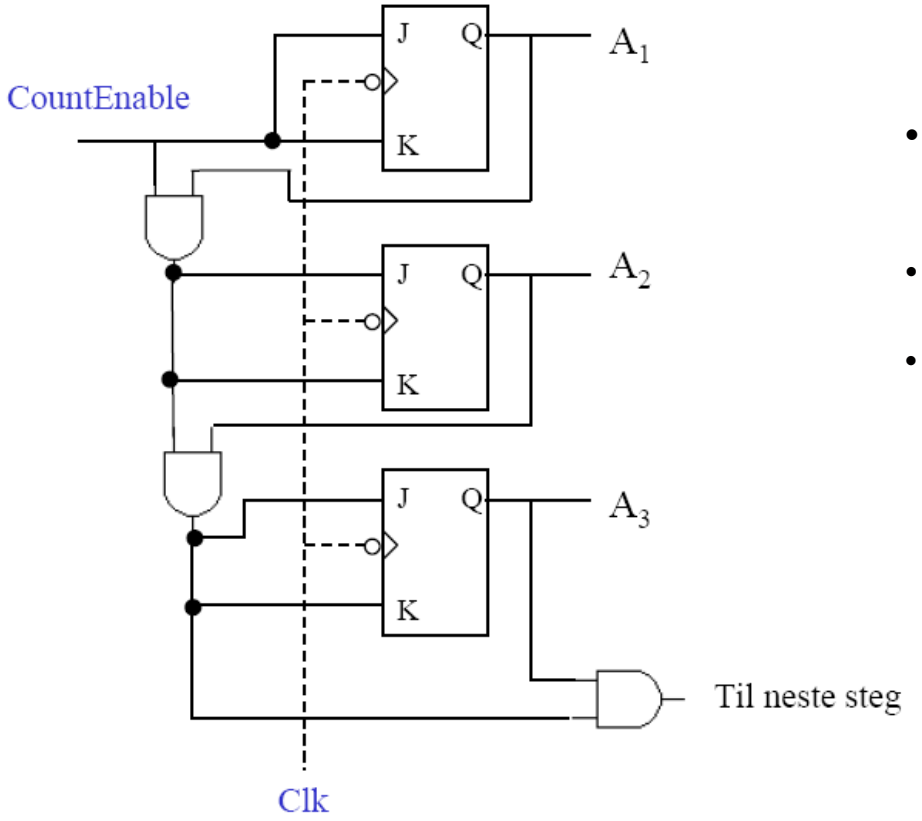


A3	A2	A1	Operasjon	Forklaring
0	0	0	Inverter A1	
0	0	1	Inverter A1	A1 går fra 1 til 0 og inverterer A2
0	1	0	Inverter A1	
0	1	1	Inverter A1	A1 går fra 1 til 0 og inverterer A2
				A2 går fra 1 til 0 og inverterer A3
1	0	0	Inverter A1	
1	0	1	Inverter A1	A1 går fra 1 til 0 og inverterer A2
1	1	0	Inverter A1	
1	1	1	Inverter A1	A1 går fra 1 til 0 og inverterer A2
				A2 går fra 1 til 0 og inverterer A3
				A3 går fra 1 til 0

2009

Synkronteller

- I en synkronteller trigges ***alle*** flip-flop'ene av samme klokkesignalet, og ikke bare den første som i en rippelteller

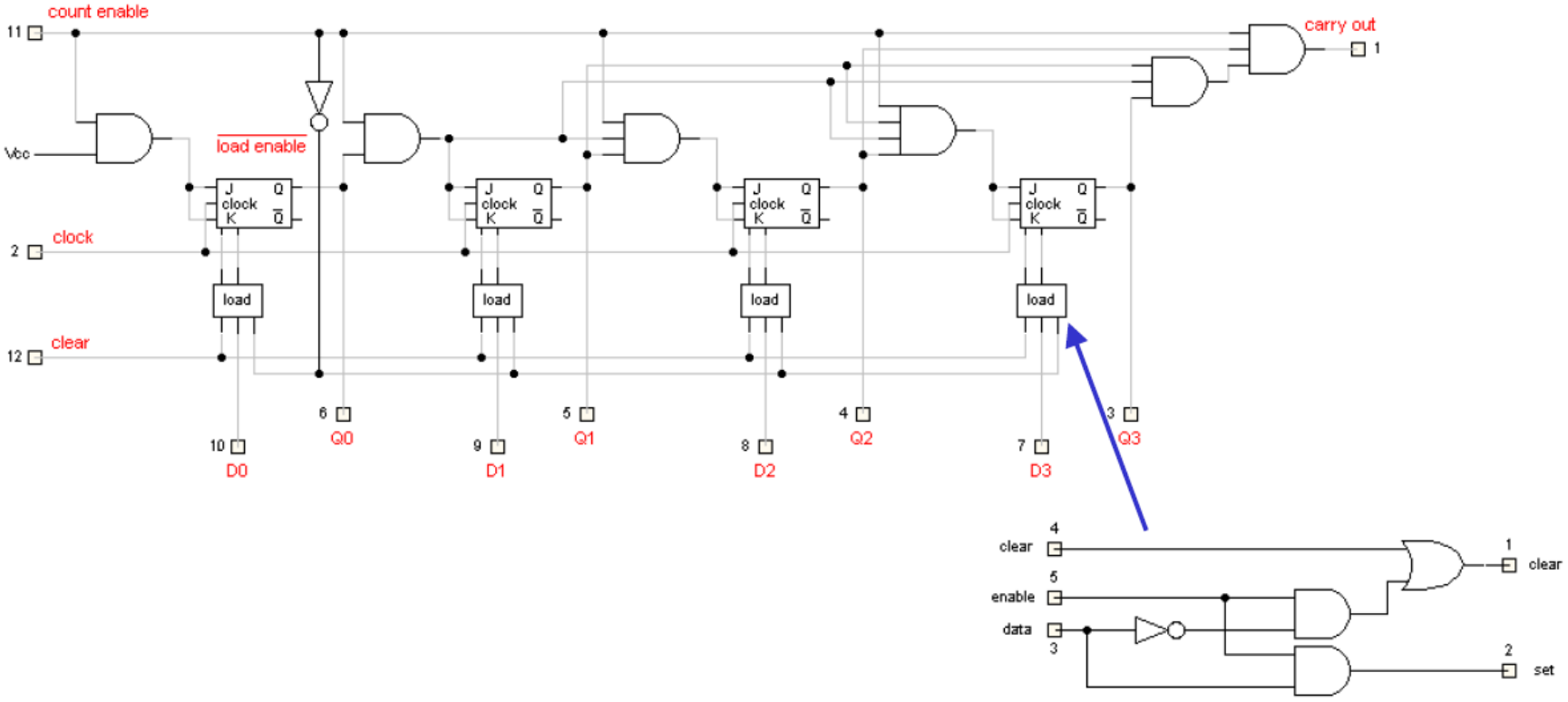


- Så lenge CountEnable = 1 vil A1 inverteres for hver klokkeperiode
- A_2 inverteres bare hvis A_1 = 1
- A_3 inverteres bare hvis A_1 = 1 **og** A_2 = 1

2009

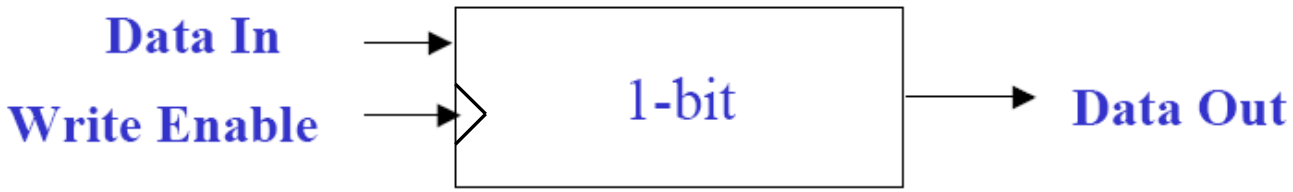
Synkronteller med innlasting og parallell nullstilling

- Så lenge CountEnable=1, vil kresten telle. Tellingen stopper når CountEnable=0
- Når Clear=1, vil alle flip-flop'ene nullstilles
- Data lastes inn gjennom D0 til D3, mens utgangsverdiene er på Q0 til Q3

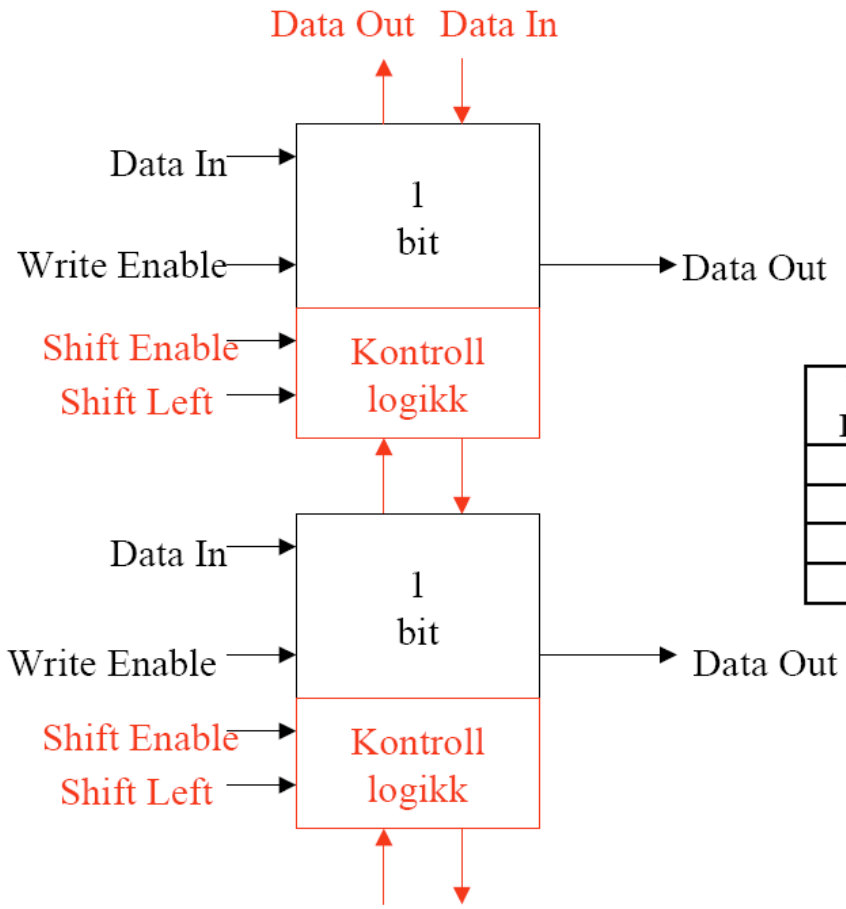


Registre

- Internt i en CPU trengs lagerceller som kan lagre bit, byte, halvord eller ord
- En en-bits lagercelle består som regel av en D-flipflop og ekstra logikk for å styre innlasting av data til lagercellen (se læreboka side 36).
- Implementasjonen varierer avhengig av tiltenkt bruk
- Eksempel:



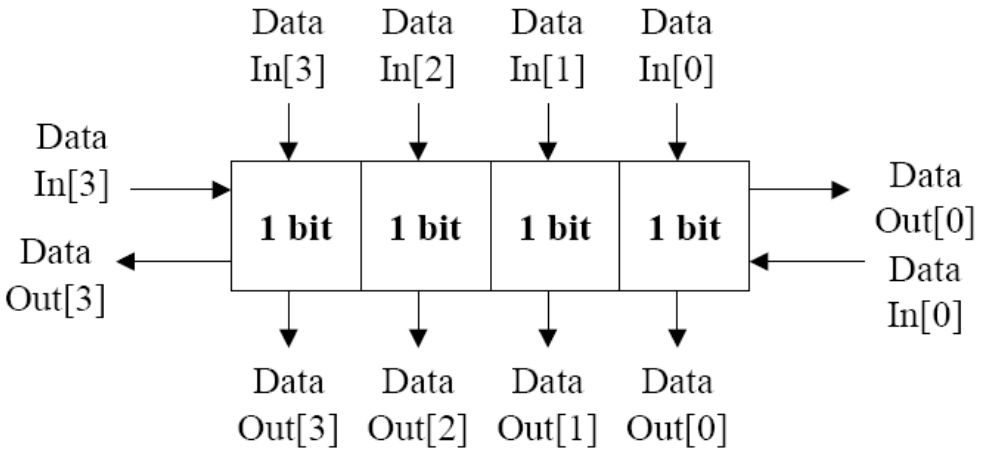
- **Write Enable** bestemmer om ny verdi skal lastes inn.
- Ved å sette sammen 1-bits celler i parallell, får man et register.
- Hvis man i tillegg kan laste data over i nabocellen, kalles det et skiftregister



Write Enable	Shift Enable	Shift Left	Funksjon
0	X	X	Ingen endring
1	0	X	Data Out:= Data In
1	1	0	Kopier mot høyre
1	1	1	Kopier mot venstre

2009

Shiftregister

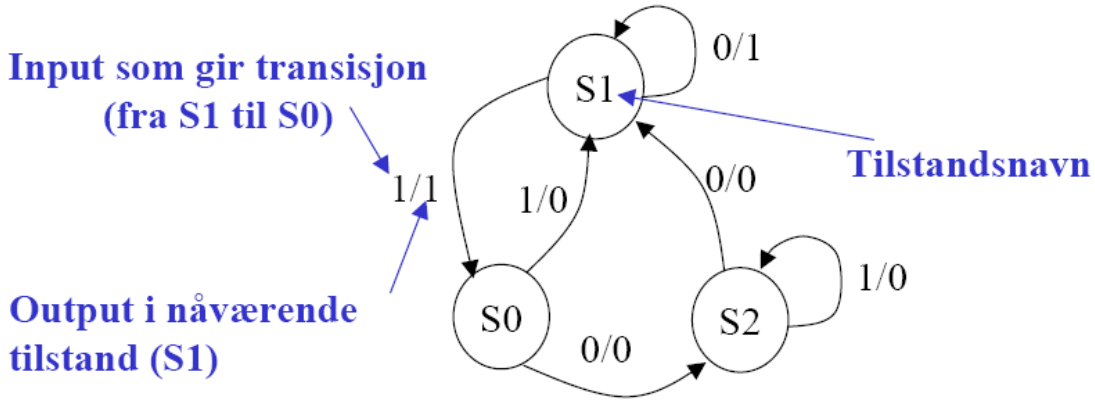


- Ved en skiftoperasjon vil et bit "falle ut" enten ut av posisjon 0 (ved høyreskift), eller posisjon n ved venstreskift.
- Hvis man gjør rotasjon sender man, bit'et fra posisjon 0 inn i posisjon n ved høyrerotasjon, og bit'et fra posisjon n sendes inn i posisjon 0 ved venstrerotasjon.
- Ved å skifte bitmønsteret som representerer et binært tall en plass mot høyre, dividerer man med 2.
- Ved å skifte bitmønsteret som representerer et binært tall en plass mot venstre, multipliserer man med 2.

Design av sekvensielle kretser/system

- Trenger metoder for design av sekvensielle kretser.

- Tilstandsmaskin, sentrale begreper:
  - Tilstand: Innholdet i alle lagerceller (flip-flop'er, registre etc) på et bestemt tidspunkt
  - Tilstandsdiagram: Grafisk fremstilling av tilstandene i et system og overgangene mellom dem
  - Transisjon: Overgang fra en tilstand til en annen



Tilstandstabell

- Tilstandstabellen er en oversikt som gir samme informasjon som tilstandsdiagrammet, dvs. sammenhengen mellom **nåværende tilstand**, **neste tilstand**, **input** og **output**

Nåværende tilstand	Neste tilstand		Output (nå)	
	Input=0	Input=1	Input=0	Input=1
S0	S2	S1	0	0
S1	S1	S0	1	1
S2	S1	S2	0	0

Tilstandsmaskin

- Gir binærverdi til tilstandene: S0=00, S1=01 og S2=10
- Trenger 2 flip-floper for å lagre tilstandsinformasjon
- Må bestemme boolske uttrykk for input til flip-flop'ene (eksitasjonstabell) og for output.
- Kaller de 2 tilstands**bitene** for hhv. **A** og **B**, input for **y** og output for **F**
- Reorganiserer tabellen på forrige side :

Nåværende tilstand		Input	Neste tilstand		Output (nå)
A	B	y	A	B	F
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	X	X	X
1	1	1	X	X	X

} Udefinert tilstand; skal aldri befinne seg der
Kan brukes til å forenkle

2009

Tilstandsmaskin

- Velger å bruke D flip-floper (enklest)

- Må bestemme inngangene til slik at neste-tilstanden blir riktig, dvs hvilken kombinasjon av nåværende tilstand og input som gir en '1' i nestetilstand:

  DA = A'B'y'+ AB'y
  DB = A'B'y + A'By'+ AB'y' = Ay'+ By'+ A'B'y

- Output bestemmes tilsvarende:

  F = A'By' + A'By = A'B (eller bare F = B ved å bruke "don't care"-betingelser)