UNIVERSITETET
I OSLO

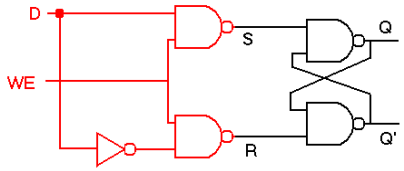
Dagens temaer

- Dagens temaer er hentet fra kapittel 3 i læreboken.
- Mer om latcher
- Design av sekvensielle kretser
- Tilstandsdiagram
- Registre
- Kort om 1. obligatoriske oppgave
- Demo av 'Digital Works'

UNIVERSITETET
I OSLO

Forbedring av RS-latch

- RS-latch fra forrige forelesning (leser inn D kun når WE=1)

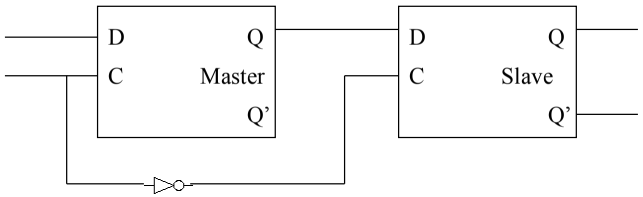


- Ønsker å kontrollere bedre når endringer skjer
  - Går over fra nivå-trigging til kant-trigging, dvs når WE (CLK) går fra 0 til 1, eller fra 1 til 0
  - En latcher som er kant-trigget kalles også for en **flip-flop**

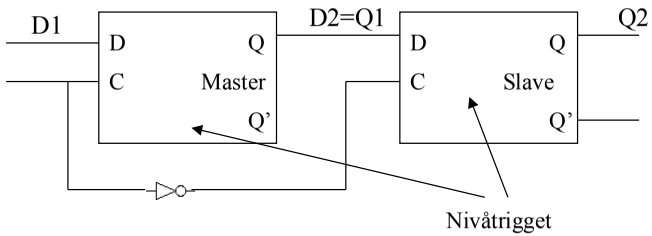
UNIVERSITETET
I OSLO

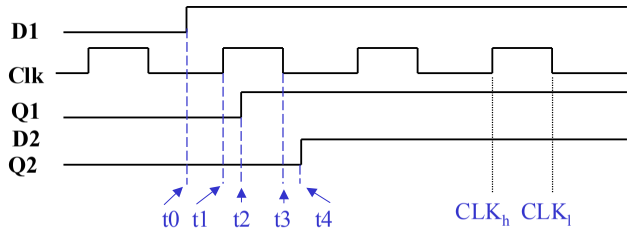
Forbedring av RS-latch: D-flipflop

- Kobler 2 i serie (den ene leser mens den andre er låst)



UNIVERSITETET
I OSLO

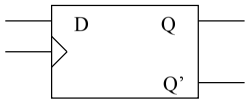




- **t0**: D1 går fra lav til høy, men FF-D1 er låst
- **t1:** Clk endrer nivå fra lav til høy
- **t2**: Q1 og D2 endrer verdi fra lav til høy. FF-D2 er fortsatt låst
- **t3:** Klokkeinngangen på FF-D2 endrer nivå fra lav til høy
- **t4:** Q2 endrer verdi fra lav til høy
- (t2-t1) + (t4-t3) er den totale forsinkelsen gjennom flip-flop'en
- Lengden på høy klokkepuls må være større enn forsinkelsen gjennom flip-flop'en, dvs CLK_h-CLK_l > (t2-t1)
- t1-t0 > tx (tx kalles setup time)
- D1 kan ikke gå lav før etter en viss tid (hold time)
- Kravene setter begrensninger til maksimum tillatte klokkefrekvens
- Finnes også andre krav som må overholdes

Flere flipflop-typer (1)

D-flipflop

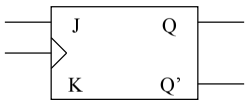


Karakteristisk tabell og ligning

D	Q(t+1)
0	0
1	1

Q(t+1) = D

JK-flipflop



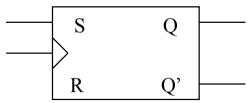
Karakteristisk tabell og ligning

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

Q(t+1) = JQ'(t) + K'Q(t)

Flere flipflop-typer (2)

SR-flipflop

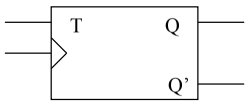


Karakteristisk tabell og ligning

S	R	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Udefinert

Q(t+1) = SQ'(t) + R'Q(t)

T-flipflop



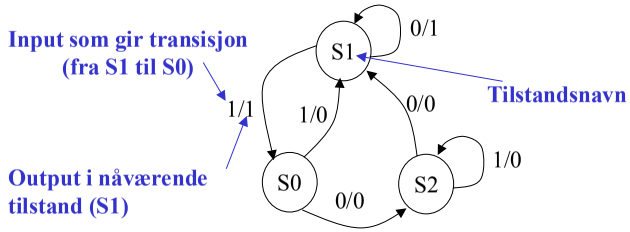
Karakteristisk tabell og ligning

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

Q(t+1) = TQ'(t) + T'Q(t)

Design av sekvensielle kretser

- Trenger metoder for design av sekvensielle kretser.
- 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



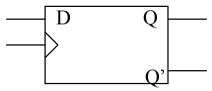
- Tilstandstabellen gis samme informasjon som tilstandsdiagrammet, dvs sammenhengen mellom **nåværende tilstand**, **neste tilstand**, **input** og **output**

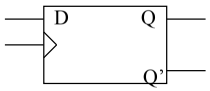
Nåværende tilstand	Neste tilstand		Output (nå)	
	Input=0	Input=1	Input=0	Input=1
S0	S2	S1	0	0
S1	S1	S0	1	1
S2	S1	S2	0	0

- Gir binærverdi til tilstandene: S0=00, S1=01 og S2=10
- Trenger 2 flip-floper for å lagre tilstandsinformasjon
- Må bestemme ligningene for input til flip-flop'ene og for output.
- Kaller de 2 tilstandsbitene 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	Udefinert tilstand; skal aldri befinne seg der Kan brukes til å forenkle
1	1	1	X	X	X	

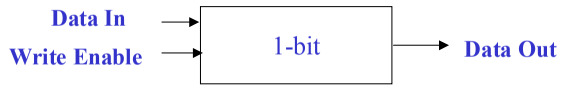
- Velger å bruke D flip-floper (enklest)
- Må bestemme inngangene til slik at neste-tilstanden blir riktig, dvs hvilken kombinasjon av nåværende tilstand og input som gir en '1' i nestetilstand:
  - DA = A'B'y'+ AB'y
  - DB = A'B'y + A'By'+ AB'y' = Ay'+ By'+ A'B'y
- Output bestemmes tilsvarende:
  - F = A'By' + A'By = A'B



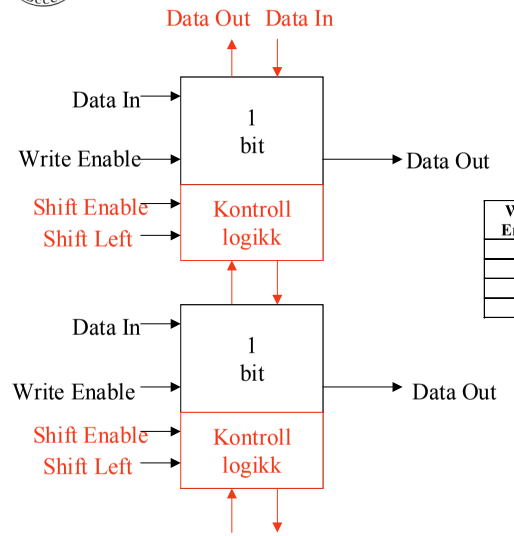


Registre

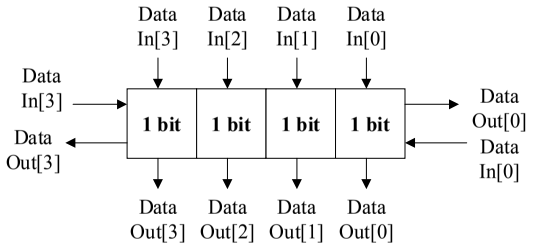
- Internt i en CPU trengs lagerceller som kan lagre bit, byte, halvord eller ord
- En en-bits lagercelle består som regel av en D-flipflop og ekstra logikk for å styre innlasting av data til lagercellen (se læreboka side 36).
- Implementasjonen varierer avhengig av tiltenkt bruk
- Eksempel:



- 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



- 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.