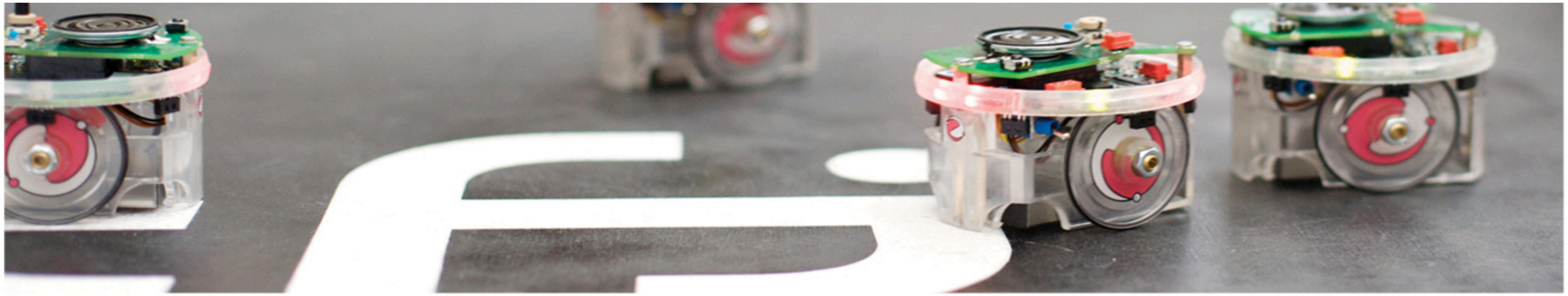


UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

INF2270

Sekvensiell Logikk

Hovedpunkter

- Definisjoner
- Portforsinkelse
- Shift register
- Praktiske Eksempler
- Latch
  - SR
  - D
- Flip-Flop
  - D
  - JK
  - T
- Tilstandsmaskiner
- Tilstandsdiagrammer
- Reduksjon av tilstand
- Ubrukte tilstander
- Eksempler

Definisjoner

- Kombinatorisk logikk

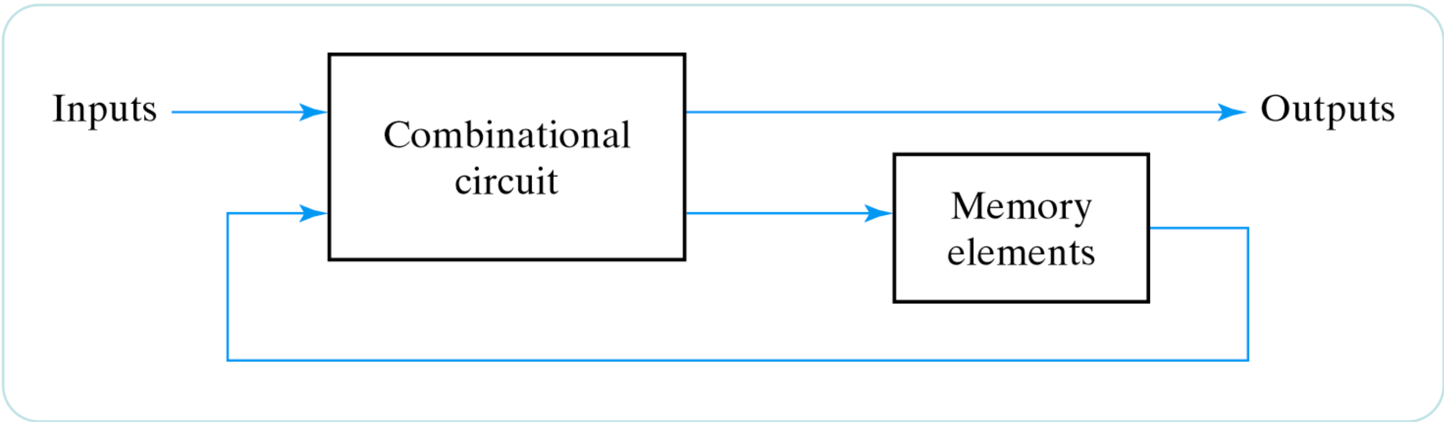
    Utgangsverdiene er entydig gitt av nåværende kombinasjon av inngangsverdier.

- Sekvensiell logikk

    Inneholder hukommelse (låsekretser).

    Utgangsverdiene er gitt av nåværende kombination av inngangsverdier, samt sekvensen (tidligere inngangs-/utgangsverdier)

Sekvensiell Logikk

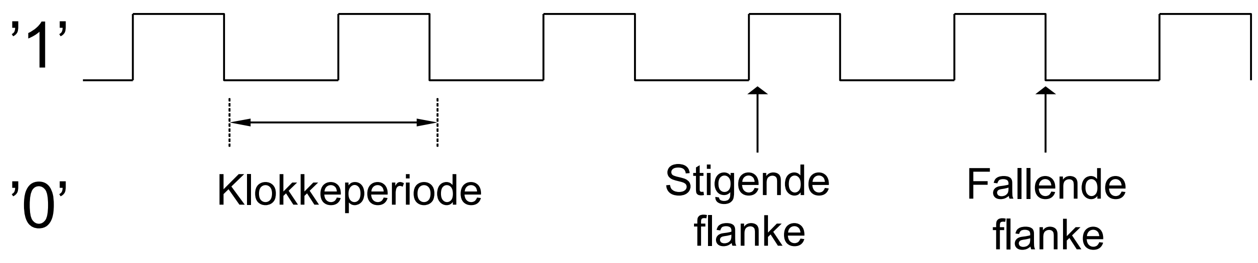


- I *synkrone* sekvensielle kretser skjer endringen(e) i output **samtidig** med endringen i et *klokkesignal.*

- I *asynkrone* sekvensielle kretser skjer endringen(e) i output **uten** noe *klokkesignal.*

- Nesten alle kretser er synkrone.

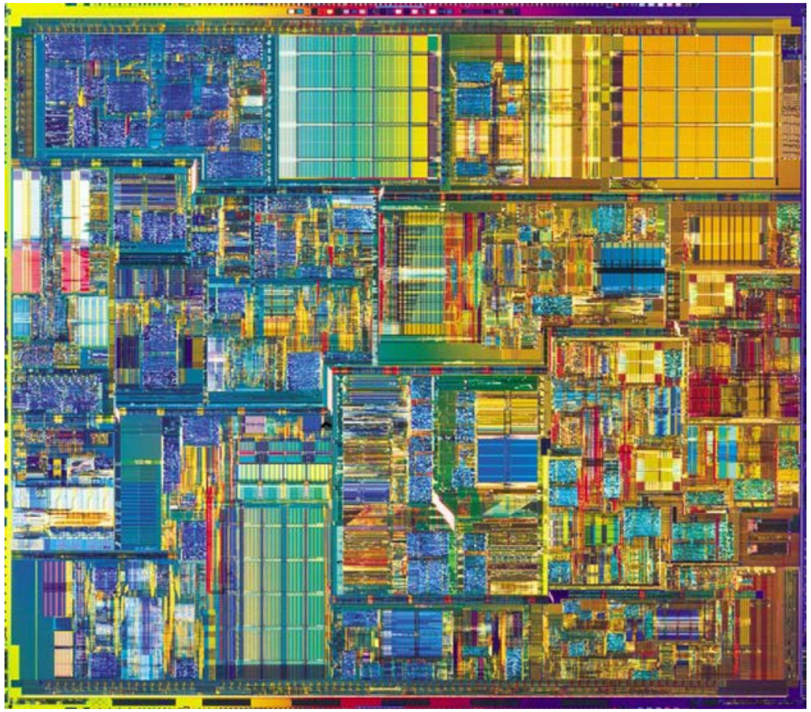
- Et klokkesignal er et digitalt signal som veksler mellom ´0´ og ´1´ med fast takt.



Synkron logikk

- I større digitale system har man behov for å synkronisere dataflyten. Til dette bruker vi et globalt klokkesignal

- Uten global synkroniskering ville det vært total kaos



- Den omvendte av klokkeperioden kalles (klokke)frekvensen, altså

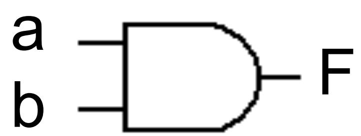
$$frekvens = \frac{1}{klokkeperioden}$$

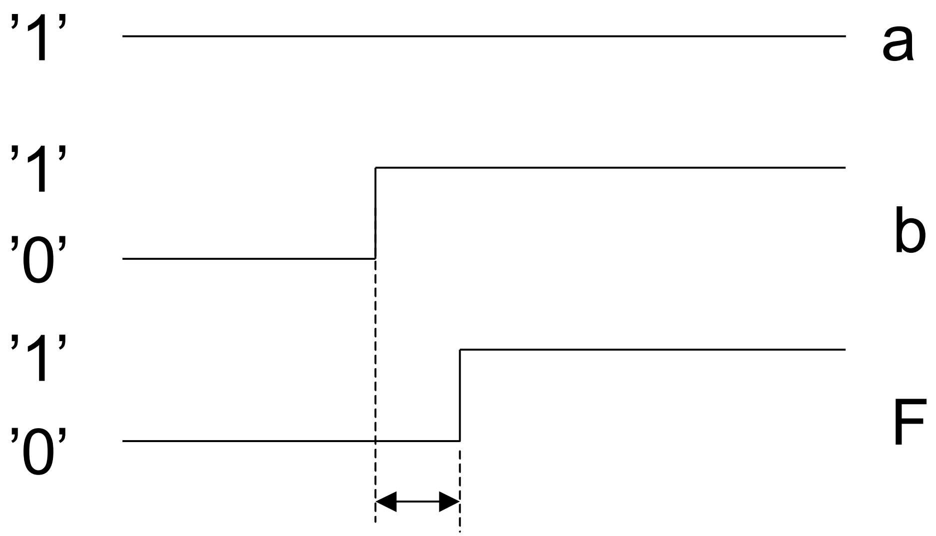
- Ønsker så høy klokkefrekvens som mulig, fordi hver enkelt operasjon da bruker så kort tid som mulig.

- Maksimal klokkefrekvens bestemmes av flere faktorer, blant annet:
  - Lengde på signalveiene
  - Last
  - Forsinkelse gjennom porter (delay)
  - Teknologi.

- NB: Hastighet er ikke direkte proporsjonal med klokkefrekvens.

Portforsinkelse / tidsforsinkelse



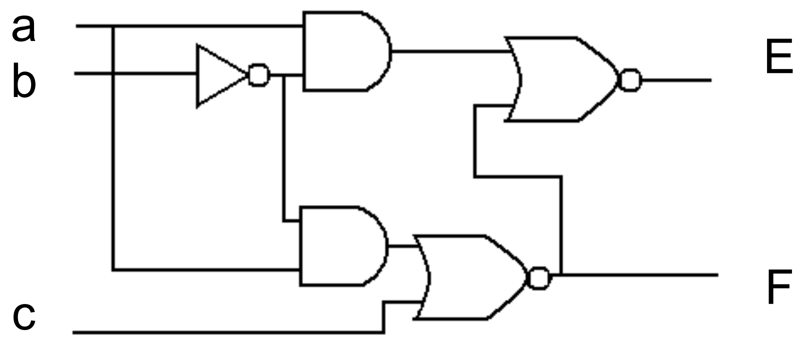


Logisk dybde

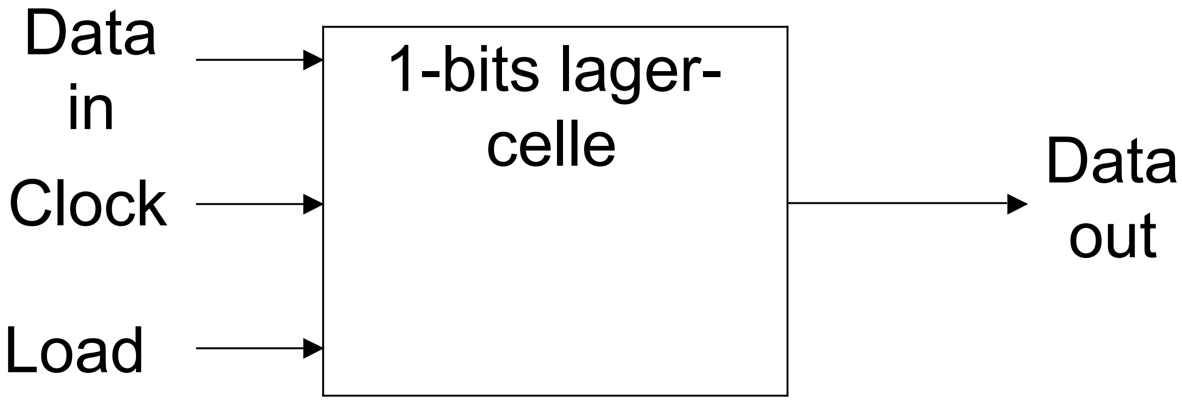
- Logisk dybde: Antall porter et signal passerer fra inngang til utgang.

- Ved å redusere logisk dybde reduseres forsinkelsen gjennom kretsen.

Eksempel:

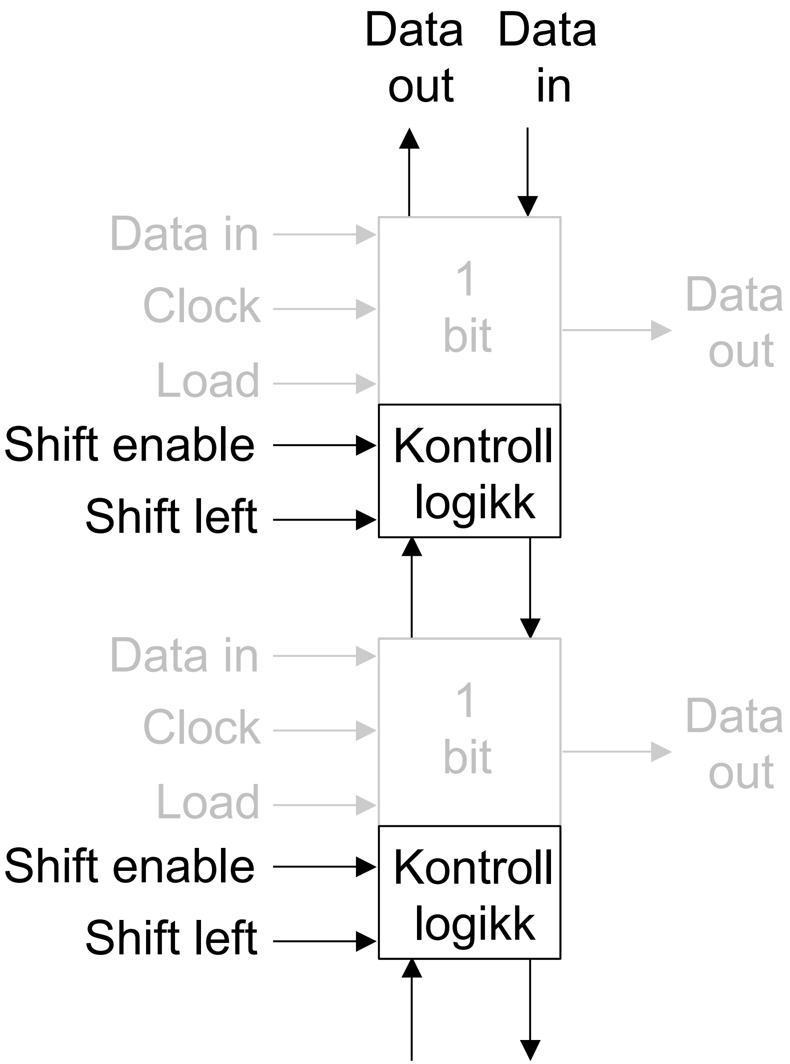


- Internt i en CPU trengs en fleksibel type lagercelle som kan lagre et bit.



- Som regel trenger man å lagre hele byte, halvord eller ord, og gjøre samme operasjon på alle bitene.
- Load-signalet bestemmer om ny verdi skal lastes inn eller ikke.
- 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.

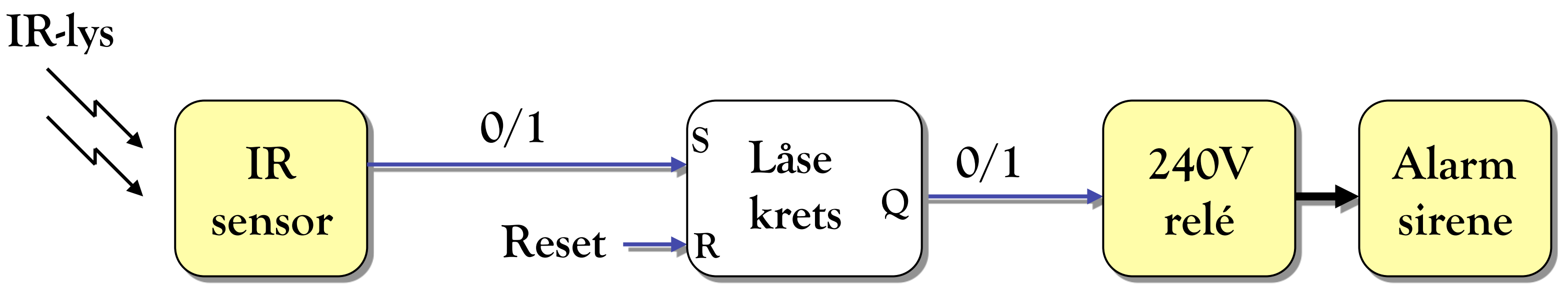
Shiftregister - enkel



Load	Shift Enable	Shift Left	
'0'	'X'	'X'	Ingen endring
'1'	'0'	'X'	Data out := Data in
'1'	'1'	'0'	Shift right
'1'	'1'	'1'	Shift left

Praktiske eksempler

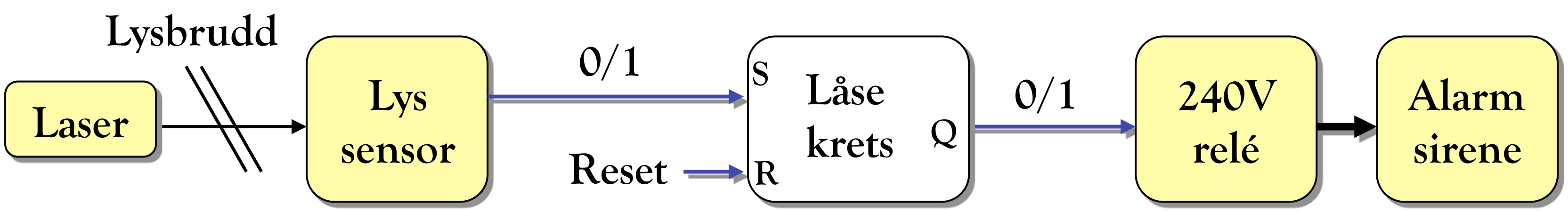
- Logikk som behandler signaler fra fysiske sensorer:
  - Varmefølende persondetektor



Når IR-lyset varierer mottat logikken et “ras” av kortvarige ´1´er pulser (msek). Logikken skal sette sirenen permanent på første mottatte puls.

Praktiske eksempler

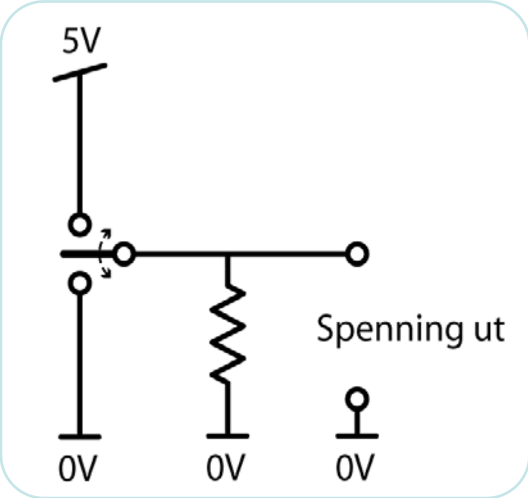
- Logikk som behandler signaler fra fysiske sensorer:
  - Laserbasert tyveridetektor

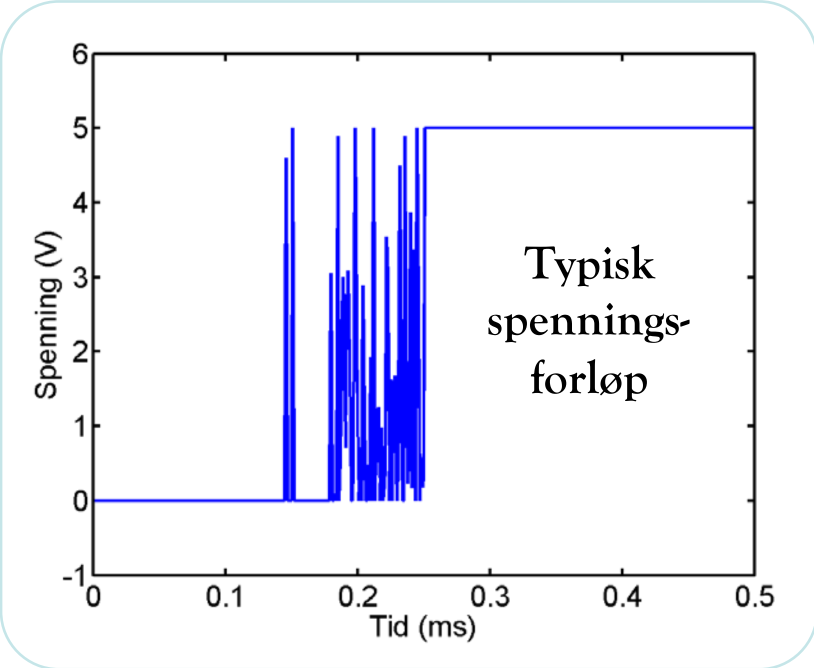


Når laserlyset blir brutt mottar logikken en eller flere ´1´er pulser. Logikken skal sette sirenen permanent på første mottatte puls.

Praktiske eksempler

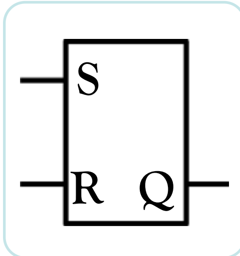
- Kontaktprelling fra mekanisk bryter.
- Mekaniske brytere gir ikke “rene” logiske nivå ut i overgangsfasen. Slike signaler må ofte “renses” ved bruk av låsekretser.





SR-latch – funksjonell beskrivelse

1) Kretsen skal sette Q til ”1” hvis den får ”1” på inngang S. Når inngang S går tilbake til ”0” skal Q forbli på ”1”



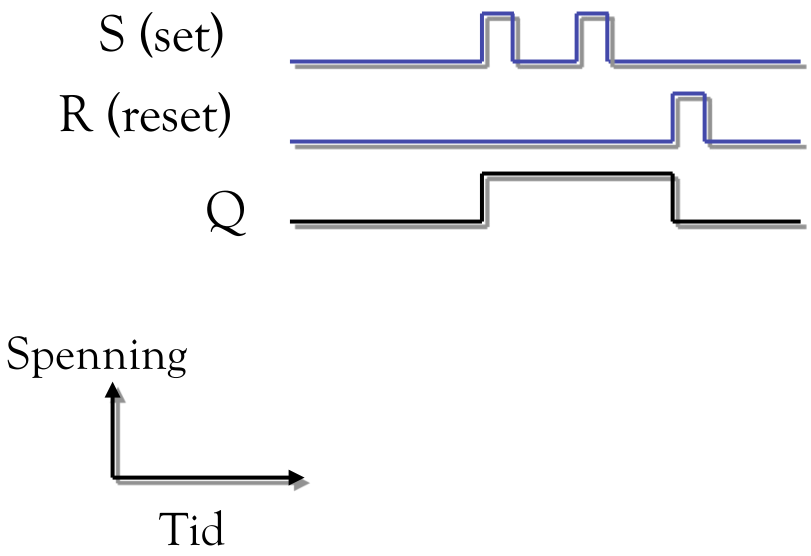
2) Kretsen skal resette Q til ”0” når den får ”1” på inngang R. Når inngang R går tilbake til ”0” skal Q forbli på ”0”

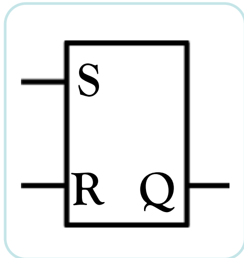
3) Tilstanden ”1” på både S og R brukes normalt ikke

SR

S	R	Q	
0	0		låst
0	1	0	
1	0	1	
1	1	0	

SR-latch – funksjonell beskrivelse

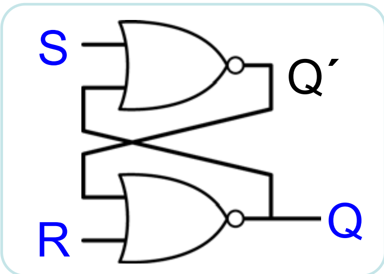




SR

S	R	Q	
0	0		låst
0	1	0	
1	0	1	
1	1	0	

SR-latch – Portimplementasjon NOR



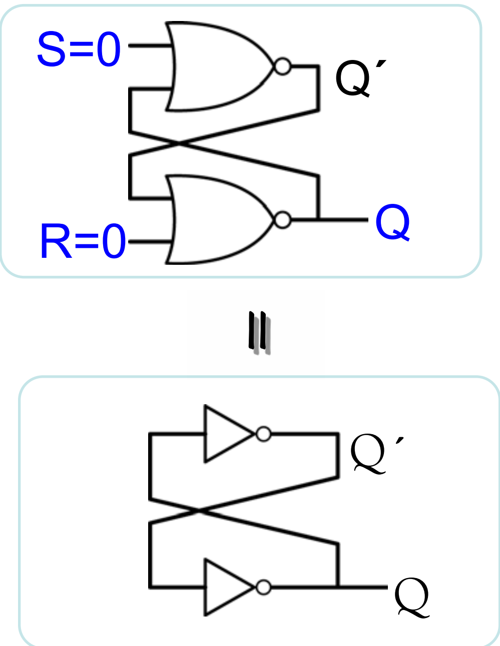
Øvre NOR	
S Q	Q´
0 0	1
0 1	0
1 0	0
1 1	0

Nedre NOR	
Q´R	Q
0 0	1
0 1	0
1 0	0
1 1	0

*Signalet Q´ er ikke invertert av Q for tilstand S=1, R=1

SR-latch – Analyse

- Tilstand S=0, R=0: En NOR port med fast "0" inn på en av inngangene er ekvivalent med NOT



SR

S R	Q
0 0	
0 1	
1 0	
1 1	

Øvre NOR

S Q	Q´
0 0	1
~~0 1~~	~~0~~
~~1 0~~	~~0~~
1 1	0

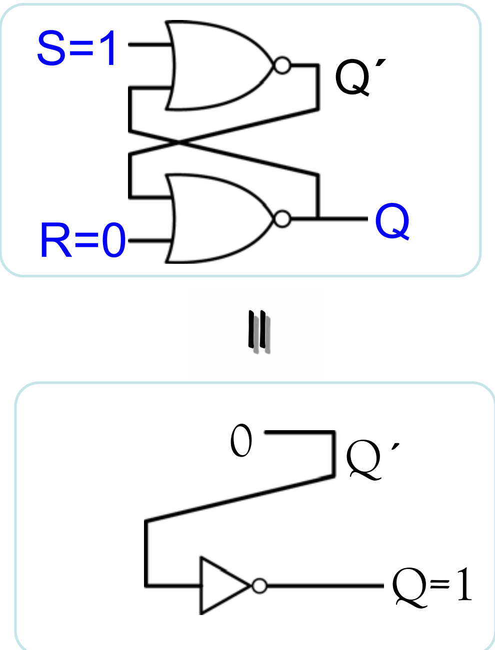
Nedre NOR

Q´R	Q
~~0 0~~	~~1~~
0 1	0
~~1 0~~	~~0~~
1 1	0

Ser bort i fra tilstand S=1 og R=1

SR-latch – Analyse

- Tilstand S=1, R=0: En NOR port med fast "1" inn på en av inngangene gir alltid ut "0"



SR

S	R	Q
0	0	
0	1	
1	0	
1	1	

Øvre NOR

S	Q	Q´
~~0~~	~~0~~	~~1~~
~~0~~	~~1~~	~~0~~
1	0	0
1	1	0

Nedre NOR

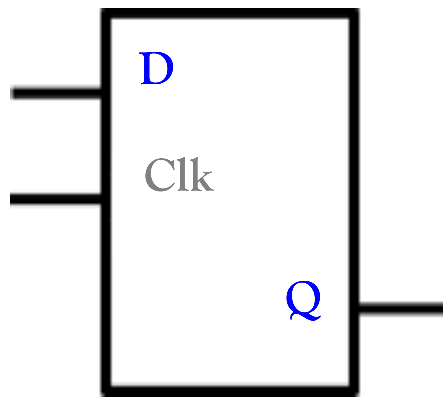
Q´	R	Q
0	0	1
~~0~~	~~1~~	~~0~~
1	0	0
~~1~~	~~1~~	~~0~~

D-Latch

- Dataflyten gjennom en D-latch kontrolleres av et klokkesignal

1) Slipper gjennom et digital signal så lenge klokkeinngangen er “1” (transparent)

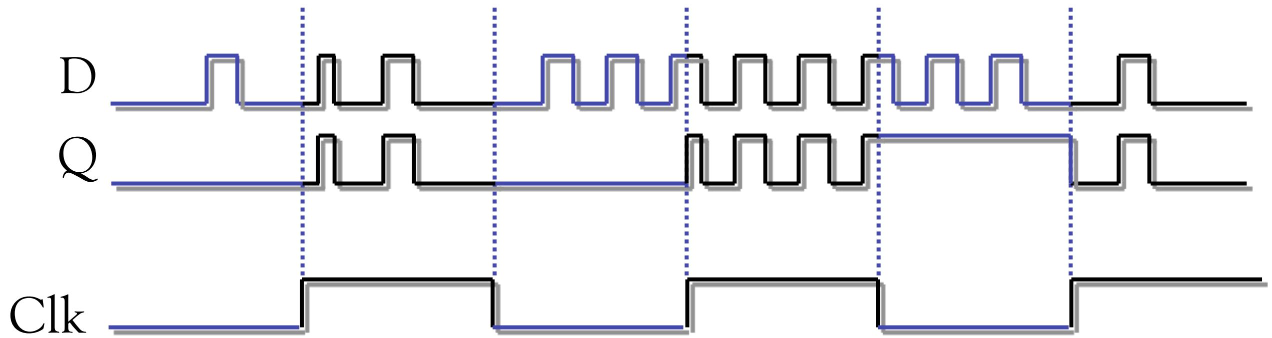
2) I det øyeblikket klokkeinngangen går fra “1” til “0” låser utgangen seg på sin nåværende verdi. Forandringer på inngangen vil ikke påvirke utgangsverdien så lenge klokkesignalet er “0”

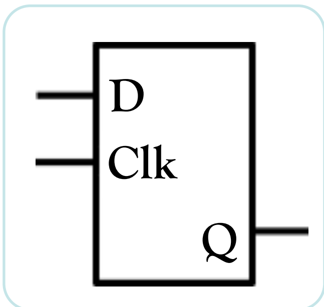


D-Latch

Clk = 1 : kretsen slipper gjennom signalet

Clk = 0 : kretsen holder (låser) utgangssignalet





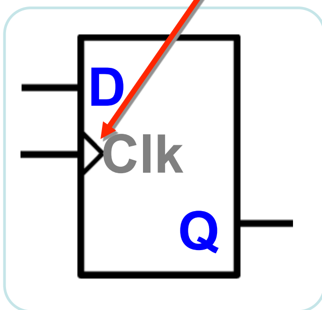
Logisk verdi på D i det øyeblikk Clk går i fra "1" til "0" bestemmer verdien som holdes på Q

Flip-flop

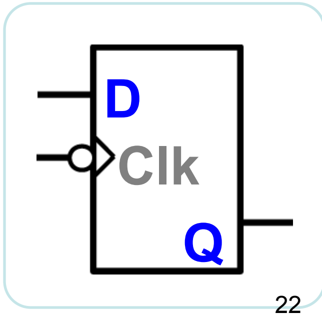
Flip-Flop'er kommer i to varianter:

- Positiv flanketrigget
- Negativ flanketrigget

Hakk, indikerer flanketrigget



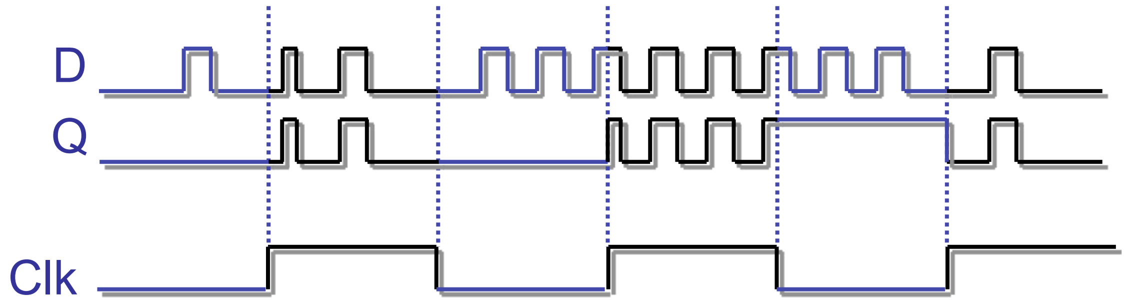
På en positiv flanketrigget Flip-Flop kan utgangen kun skifte verdi i det øyeblikk klokkesignalet går fra "0" til "1".

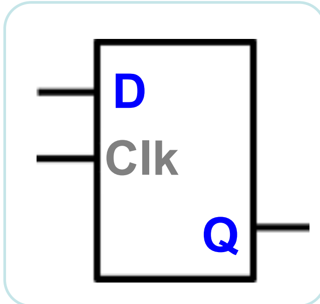


På en negativ flanketrigget Flip-Flop kan utgangen kun skifte verdi i det øyeblikk klokkesignalet går fra "1" til "0".

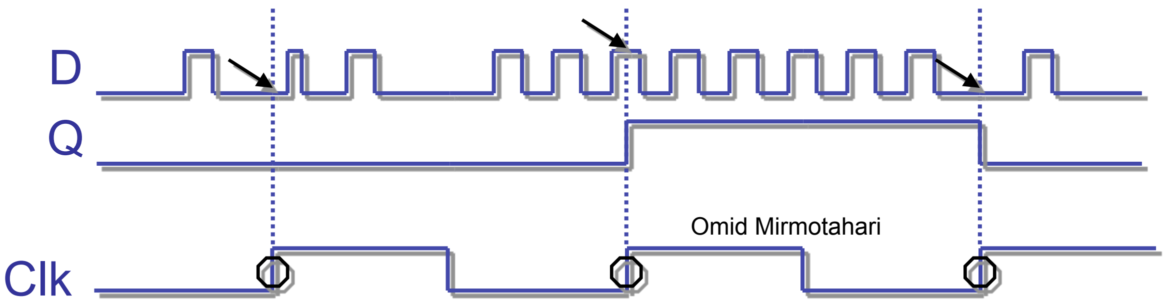
D-Flip-Flop

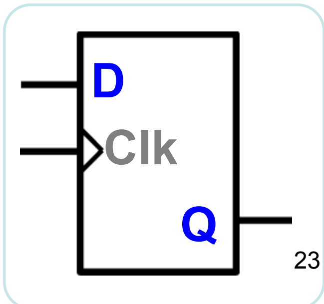
En D latch er transparent for Clk=1





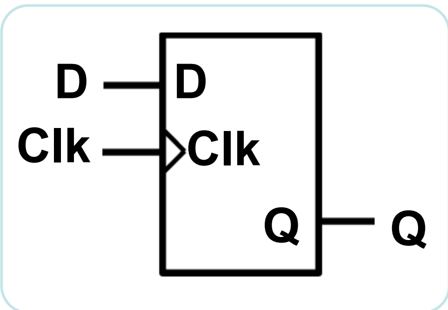
En positiv flanketrigget D flip-flop sampler verdien på D i det øyeblikk Clk går fra ”0” til ”1” (positiv flanke). Denne verdien holdes fast på utgangen helt til neste positive flanke





Karakteristisk tabell/ligning

For flip-flop'er kan man generelt beskrive neste utgangsverdi Q(t+1) som funksjon av nåværende inngangsverdi(er), og nåværende utgangsverdi Q(t)



Karakteristisk tabell for D flip-flop

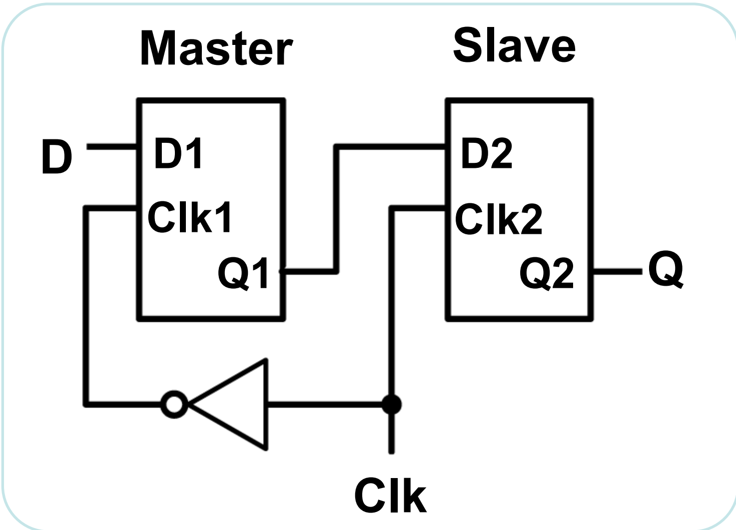
D	Q(t+1)
0	0
1	1

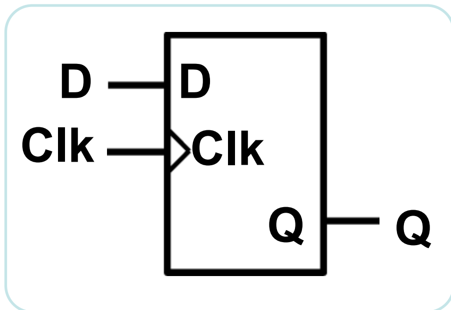
Karakteristisk ligning for D flip-flop

$$Q(t+1) = D$$

D-Flip-Flop

En positiv flanketrigget D flip-flop kan lages av to D-latcher (Master-Slave)

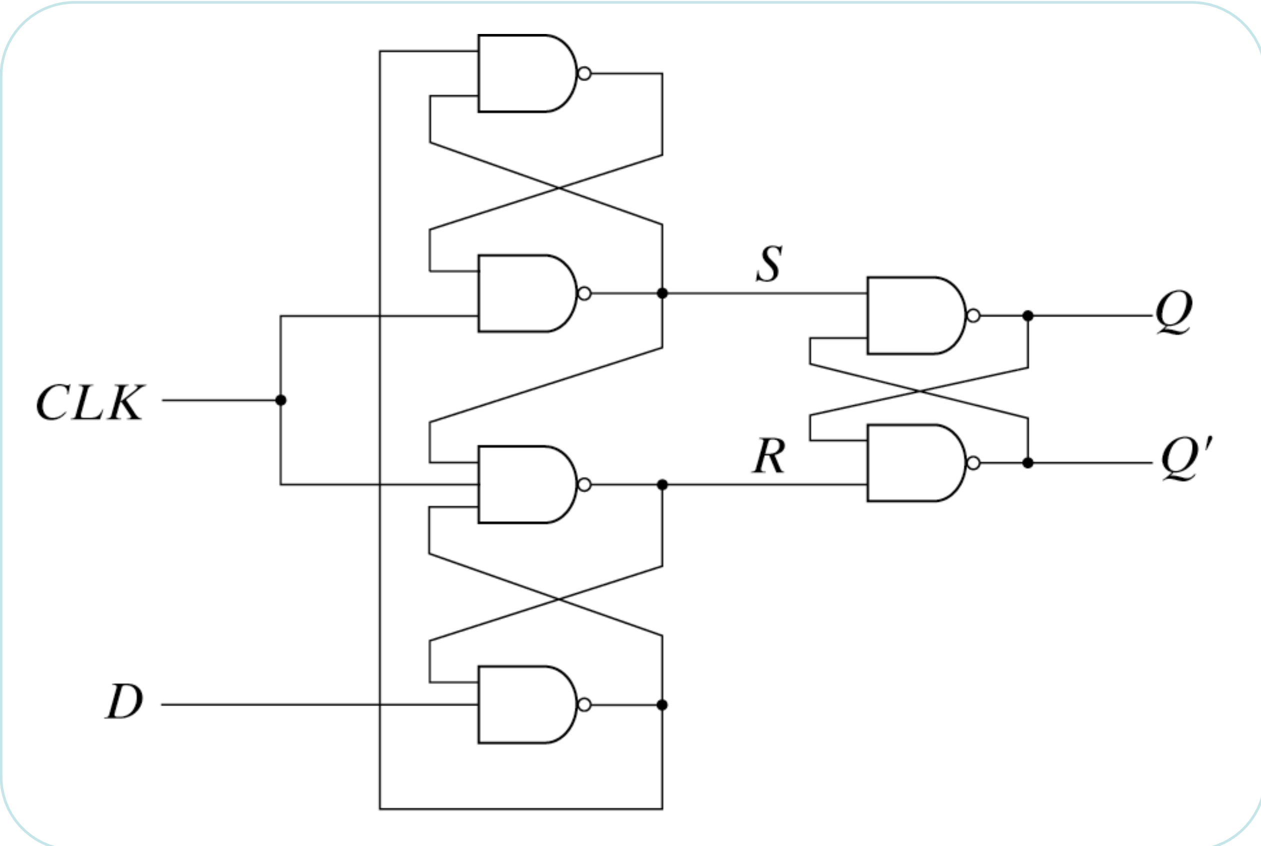




Under Clk=0 er første D latch (master) transparent

Under Clk=1 er siste D latch (slave) transparent

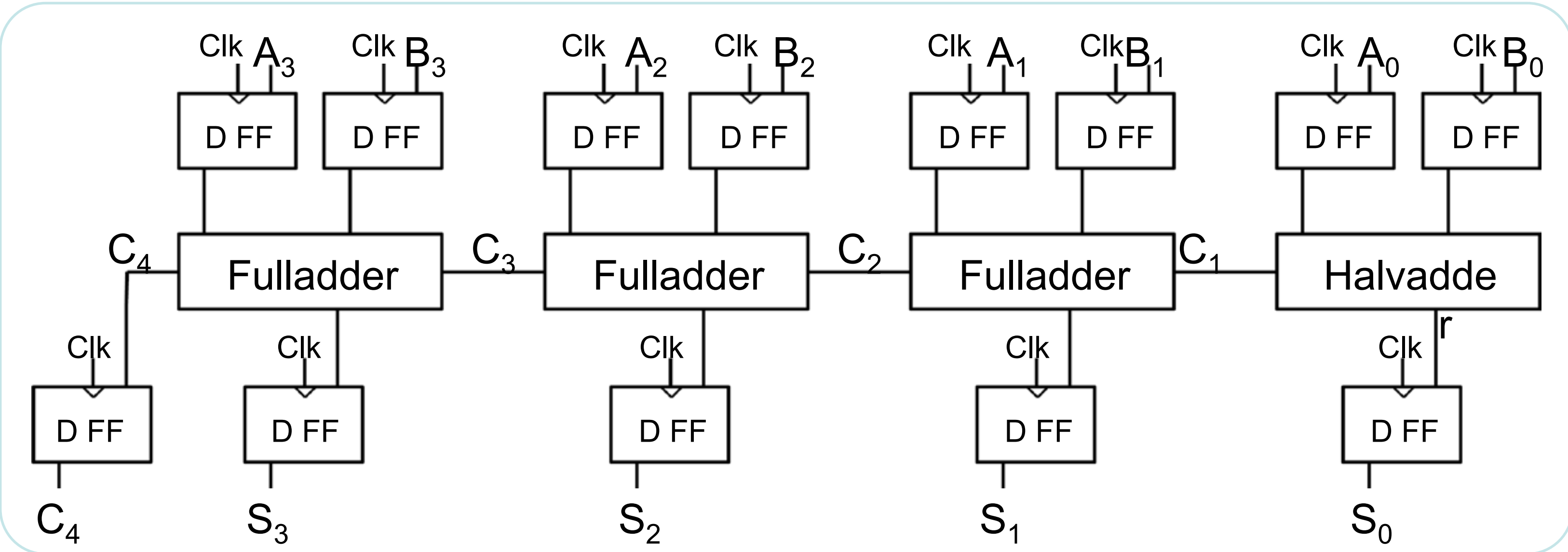
D-Flip-Flop – kompakt versjon



D-Flip-Flop, eksempel

En rippeladder vil i et kort tidsrom gi gal sum ut.

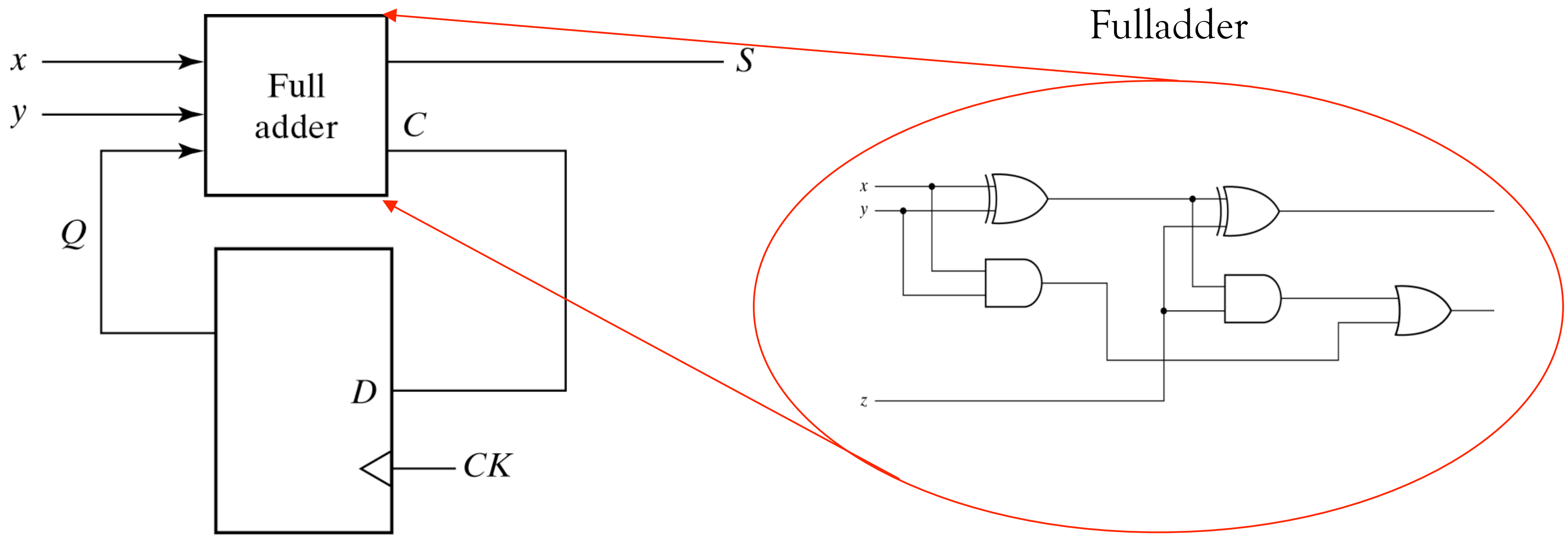
Styring av signalflyt med D flip-flops kamuflerer dette



På positiv Clk flanke kommer nye data inn til adderen. I samme øyeblikk leses forrige (stabiliserte) sum ut.

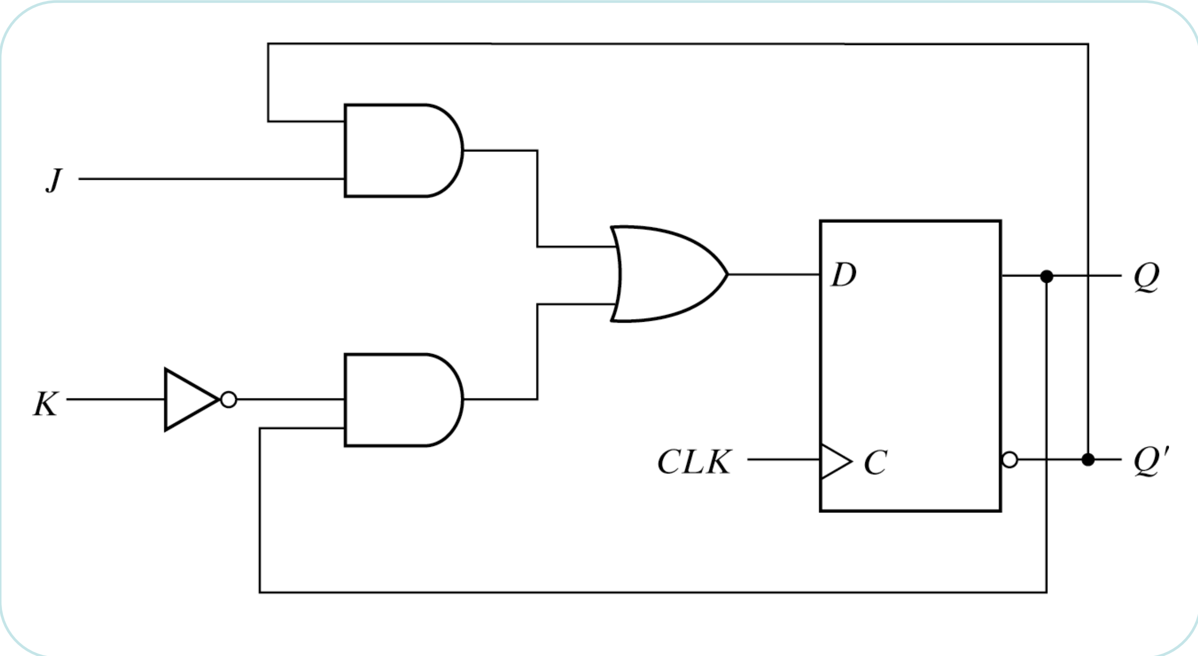
D-Flip-Flop, eksempel

Seriell adder

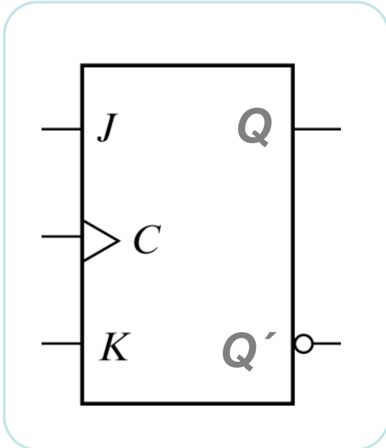


JK Flip-Flop

Kretsoppbygging



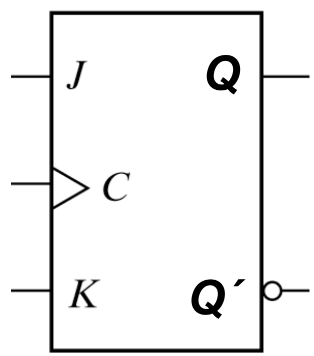
Grafisk symbol



JK Flip-Flop

En JK flip-flop har følgende egenskaper

J=0, K=0:	Utgang låst
J=0, K=1:	Resetter utgang til ”0”
J=1, K=0:	Setter utgang til ”1”
J=1, K=1:	Inverterer utgang Q→Q´



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

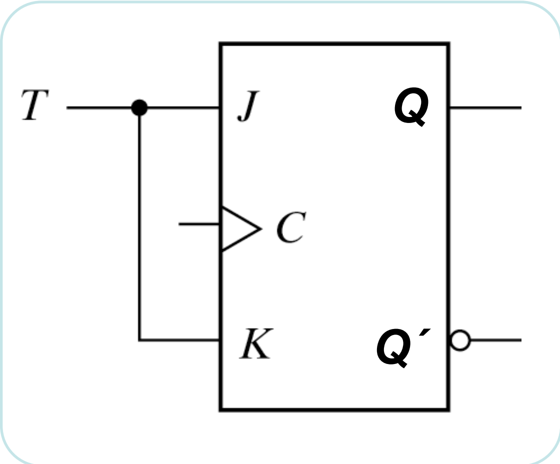
Utgangen kan kun forandre verdi på stigende klokkeflanke

En JK flip-flop er den mest generelle flip-floppen vi har

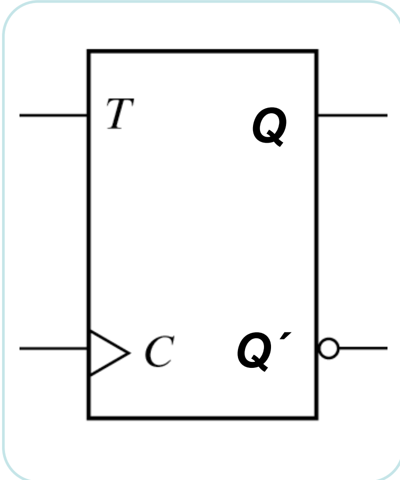
$$Q(t+1) = JQ'(t) + K'Q(t)$$

T Flip-Flop

Kretsoppbygging



Grafisk symbol



T Flip-Flop

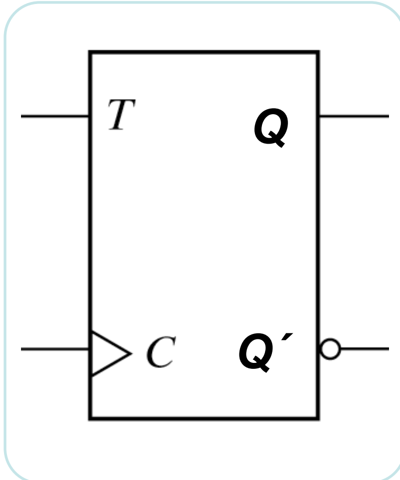
En T flip-flop har følgende egenskaper

T=0, Utgang låst

T=1, Inverterer utgang Q→Q´

Utgangen kan kun forandre verdi på stigende klokkeflanke

Det er lett å lage tellere av T flip-flop'er



T	Q(t+1)
0	Q(t)
1	Q´(t)

$$Q(t+1) = T \oplus Q(t)$$

Tilstandsmaskin

En tilstandsmaskin er et sekvensielt system som gjennomløper et sett med tilstander styrt av verdiene på inngangssignalene

Tilstanden systemet befinner seg i, pluss evt. inngangsverdier bestemmer utgangsverdiene

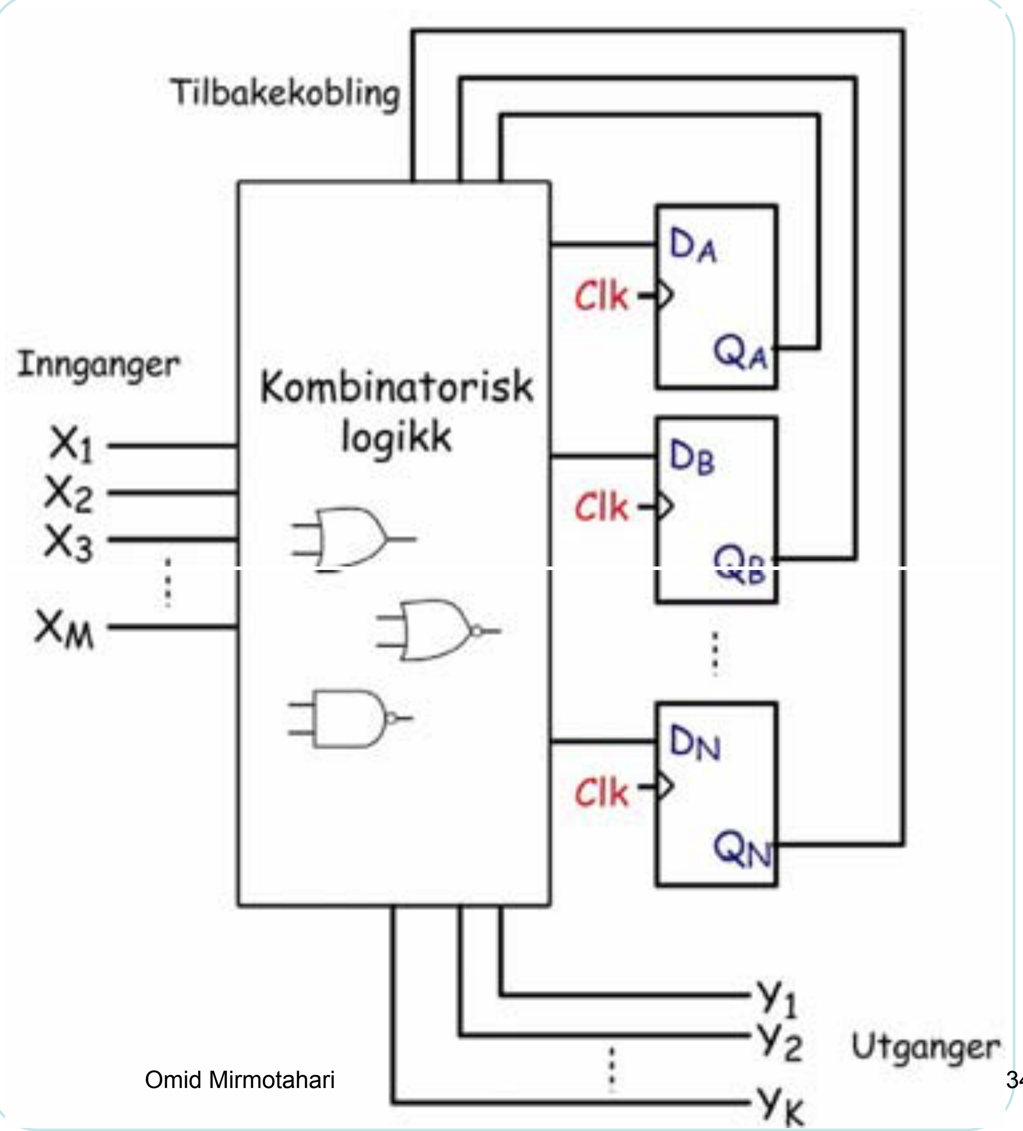
Tilstandsmaskins-konseptet gir en enkel og oversiktlig måte å designe avanserte system på

Tilstandsmaskin

Generell tilstands-maskin basert på D flip-flops

N-stk flip-flops gir 2^N forskjellige tilstander

Utgangssignalene er en funksjon av nåværende tilstand pluss evt. inngangsverdier



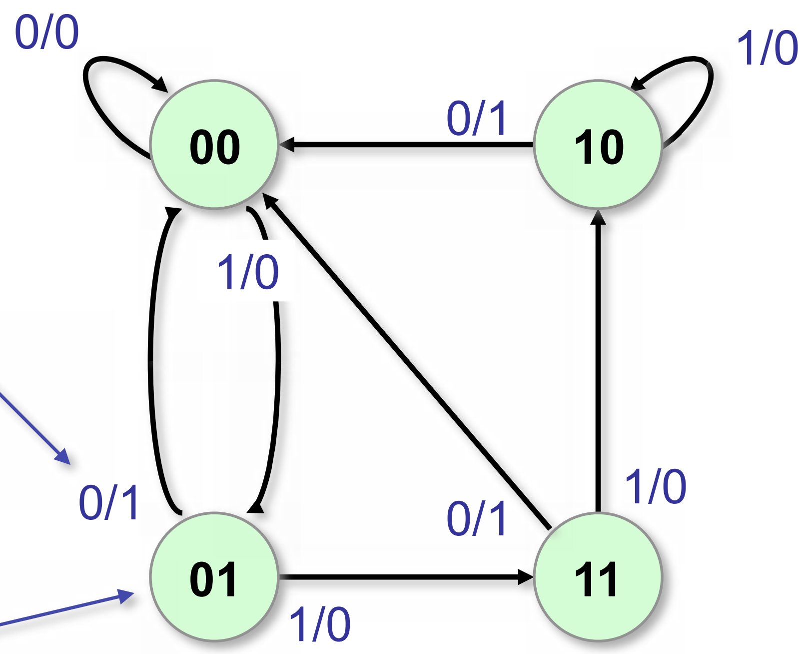
Tilstandsdiagram

Tilstandsdiagram = grafisk illustrasjon av egenskapene til en tilstandsmaskin

Eksempel nr.1:

Inngangsverdi x som medfører ny tilstand, samt utgangsverdi y for opprinnelig tilstand med inngangsverdi x

x / y



Tilstand

$Q_A Q_B$

Tilstandstabell

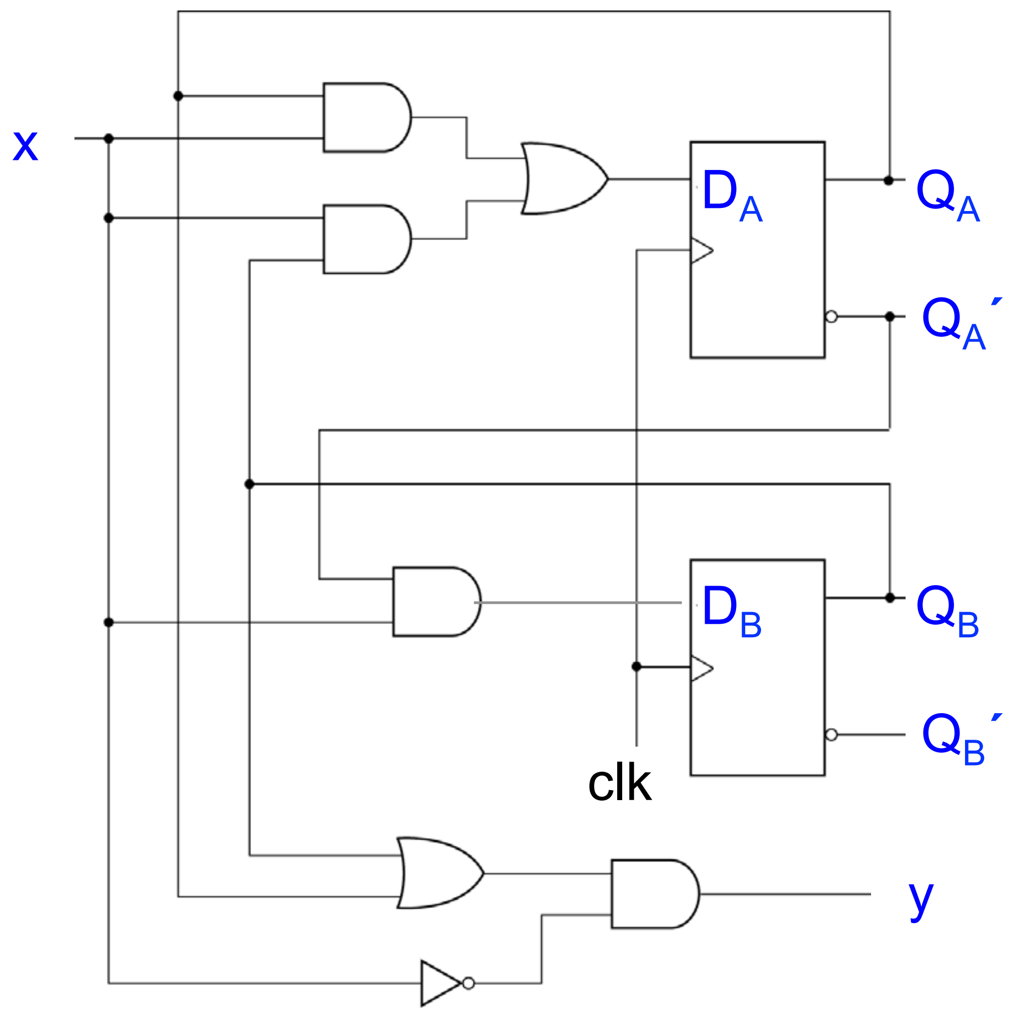
Tilstandstabell = sannhetstabell for tilstandsmaskin

Eksempel nr.1: En inngang, en utgang og 2 stk. D flip-flops

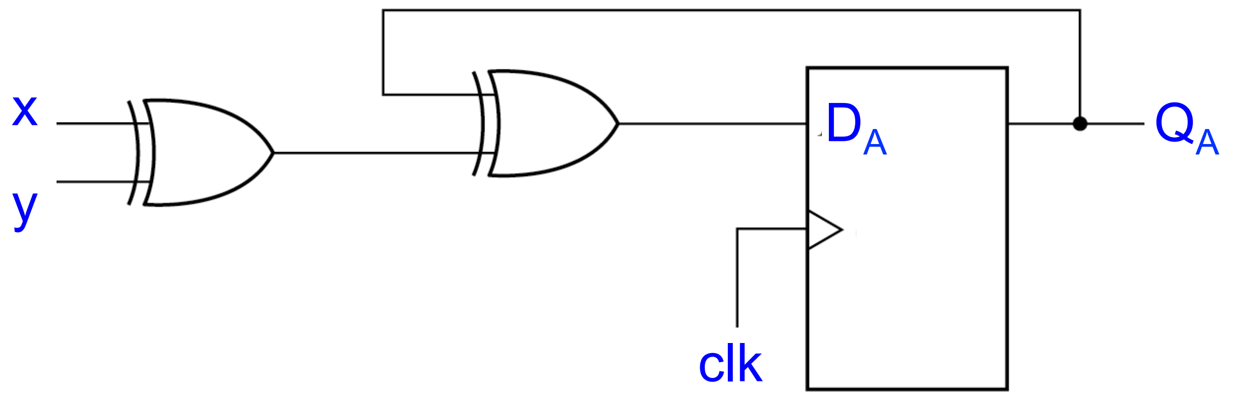
Nåværende tilstand Q_A	Q_B	Inngang x	Neste tilstand Q_A	Q_B	Utgang for nåværende tilstand y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Eksempel nr.1

Tilstandsmaskin der utgang y er en funksjon av tilstanden gitt av verdiene til Q_A og Q_B, samt inngangen x



Eksempel nr.2

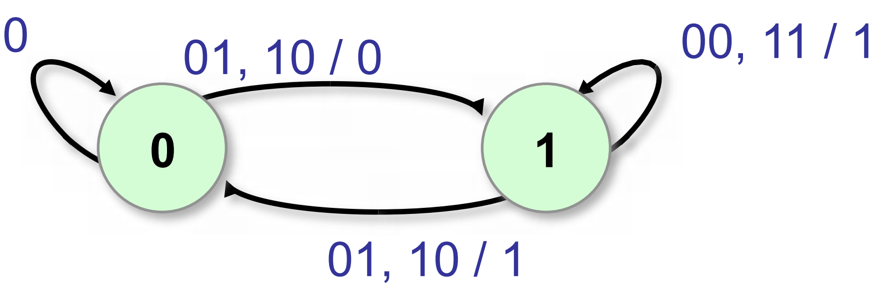


To innganger x og y, en utgang som bare er gitt av tilstanden Q_A

Nåværende tilstand Q_A	Innganger x	Innganger y	Neste tilstand Q_A	Utgang for nåværende tilstand Q_A
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Eksempel nr.2

Tilstandsdiagram



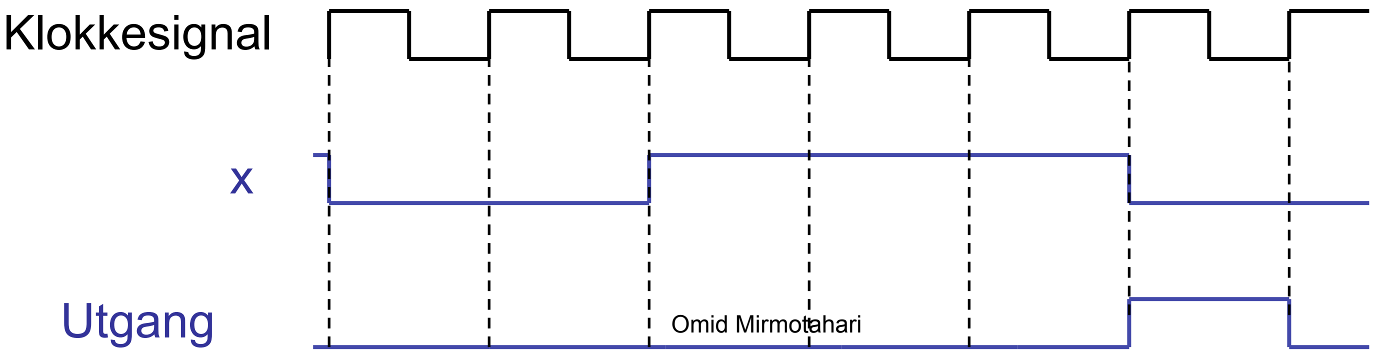
Liste av inngangskombinasjoner som gir ny tilstand / utgangsverdi for nåværende tilstand*

*Merk at i dette tilfelle er utgangsverdien kun avhengig av tilstanden (uavhengig av inngangsverdiene)

Eksempel nr.3 – design av sekvensdetektor

Ønsker å lage en krets som finner ut om det har forekommet tre eller flere "1"ere etter hverandre i en klokket bit-sekvens x

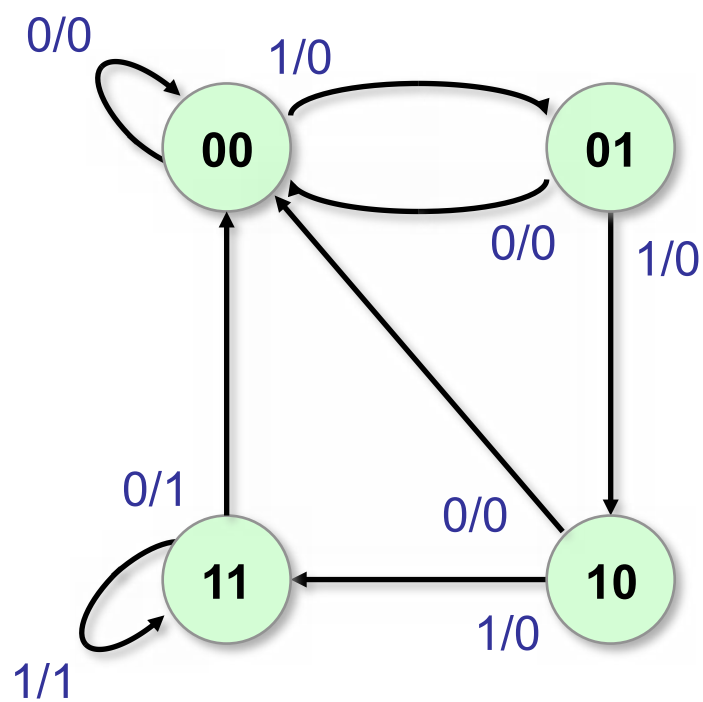
Klokket bit-sekvens: Binært signal som kun kan skifte verdi synkront med et klokkesignal



Eksempel nr.3 – design av sekvensdetektor

Tilstandsdiagram

Velger å ha 4 tilstander. Lar hver tilstand symbolisere antall "1"ere som ligger etter hverandre i bit-sekvensen.



Inngang: bit-sekvens x

Utgang: gitt av tilstanden, "0" for tilstand 0-2, "1" for tilstand 3

Eksempel nr.3

Bruker D flip-flops

D_A og D_B settes til de verdiene man ønsker at Q_A og Q_B skal ha i neste tilstand

Nåværende tilstand		Inngang	Neste tilstand		Utgang for nåværende tilstand
Q_A	Q_B	x	Q_A	Q_B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

$D_A = Q_A'Q_Bx + Q_AQ_B'x + Q_AQ_Bx$

$D_B = Q_A'Q_B'x + Q_AQ_B'x + Q_AQ_Bx$

$y = Q_AQ_B$

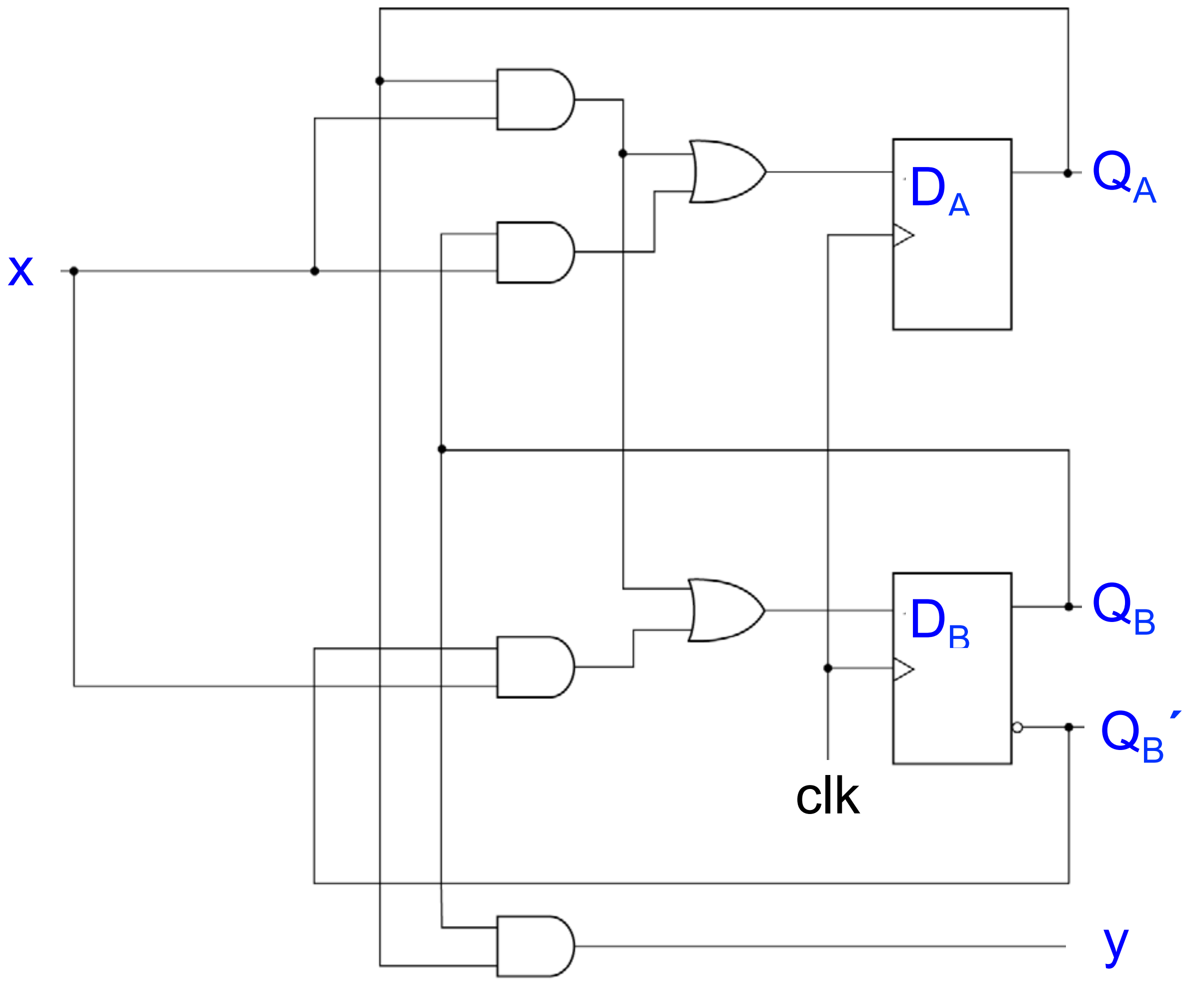
Eksempel nr.3

Forenkler uttrykkene med Karnaugh-diagram

$D_A = Q_A x + Q_B x$

$D_B = Q_A x + Q_B' x$

$y = Q_A Q_B$



Reduksjon av tilstander

En tilstandsmaskin gir oss en eller flere utgangssignal som funksjon av en eller flere inngangssignal

Hvordan dette implementeres internt i maskinen er uinteressant sett utenifra

I noen tilfeller kan man fjerne tilstander (forenkle designet) uten å påvirke inngangs/utgangs-funksjonene

Reduksjon av tilstander

Hvis to tilstander har samme utgangssignal, samt leder til de samme nye tilstandene gitt like inngangsverdier, er de to opprinnelige tilstandene like. En tilstand som er lik en annen tilstand kan fjernes.

Reduksjon av tilstander

Eksempel:

Tilstand G er lik tilstand E

Nåværende tilstand	Inngang	Neste tilstand	Utgang
A	0	B	0
A	1	B	0
B	0	C	0
B	1	D	0
C	0	A	0
C	1	D	0
D	0	E	0
D	1	F	1
E	0	A	0
E	1	F	1
F	0	G	0
F	1	F	1
G	0	A	0
G	1	F	1

Reduksjon av tilstander

Eksempel:

Fjerner tilstand G.
Erstatter hopp til G
med hopp til E

Nåværende tilstand	Inngang	Neste tilstand	Utgang
A	0	B	0
A	1	B	0
B	0	C	0
B	1	D	0
C	0	A	0
C	1	D	0
D	0	E	0
D	1	F	1
E	0	A	0
E	1	F	1
F	0	E	0
F	1	F	1

Reduksjon av tilstander

Eksempel:

Nåværende tilstand	Inngang	Neste tilstand	Utgang
A	0	B	0
A	1	B	0
B	0	C	0
B	1	D	0
C	0	A	0
C	1	D	0
D	0	E	0
D	1	F	1
E	0	A	0
E	1	F	1
F	0	E	0
F	1	F	1

Nå er tilstand F lik tilstand D

Fjerner tilstand F

Reduksjon av tilstander

Eksempel:

Har fjernet tilstand F

Nåværende tilstand	Inngang	Neste tilstand	Utgang
A	0	B	0
A	1	B	0
B	0	C	0
B	1	D	0
C	0	A	0
C	1	D	0
D	0	E	0
D	1	D	1
E	0	A	0
E	1	D	1

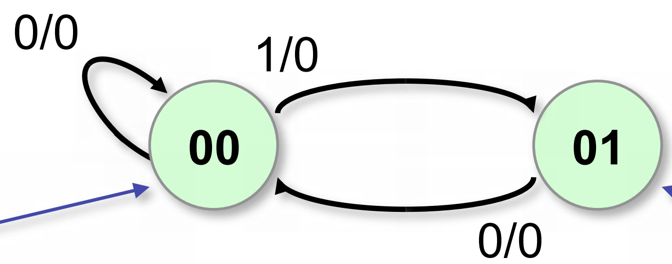
Tilordning av tilstandskoder

I en tilstandsmaskin med M tilstander må hver tilstand tilordnes en kode basert på minimum N bit der $2^N \geq M$

Kompleksiteten til den kombinatoriske delen avhenger av valg av tilstandskode

Anbefalt strategi for valg av kode: prøv-og-feil i tilstandsdiagrammet

Tilstand nr.1
Kode: 00

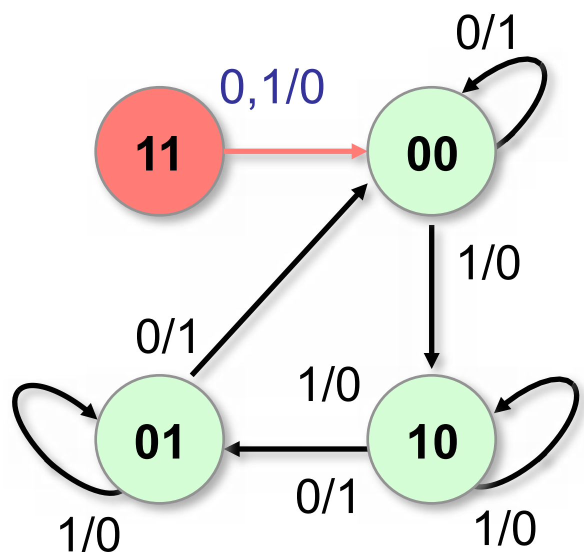


Tilstand nr.2
Kode: 01

Ubrukte tilstander

I en tilstandsmaskin med N flip-flopper vil det alltid finnes 2^N tilstander. Designer man for M tilstander der $M < 2^N$ vil det finnes ubrukte tilstander.

Problem: Under oppstart (power up) har man ikke full kontroll på hvilken tilstand man havner i først. Havner man i en ubrukt tilstand som ikke leder videre til de ønskede tilstandene vil systemet bli låst.



Løsning: Design systemet slik at alle ubrukte tilstander leder videre til en ønsket tilstand.

Generell designprosedyre basert på D flip-flops

1) Definer tilstandene, inngangene og utgangene
2) Velg tilstandskoder, og tegn tilstandsdiagram
3) Tegn tilstandstabell
4) Reduser antall tilstander hvis nødvendig
5) Bytt tilstandskoder hvis nødvendig for å forenkle
6) Finn de kombinatoriske funksjonene
7) Sjekk at ubrukte tilstander leder til ønskede tilstander
8) Tegn opp kretsen

Design eksempel nr.4

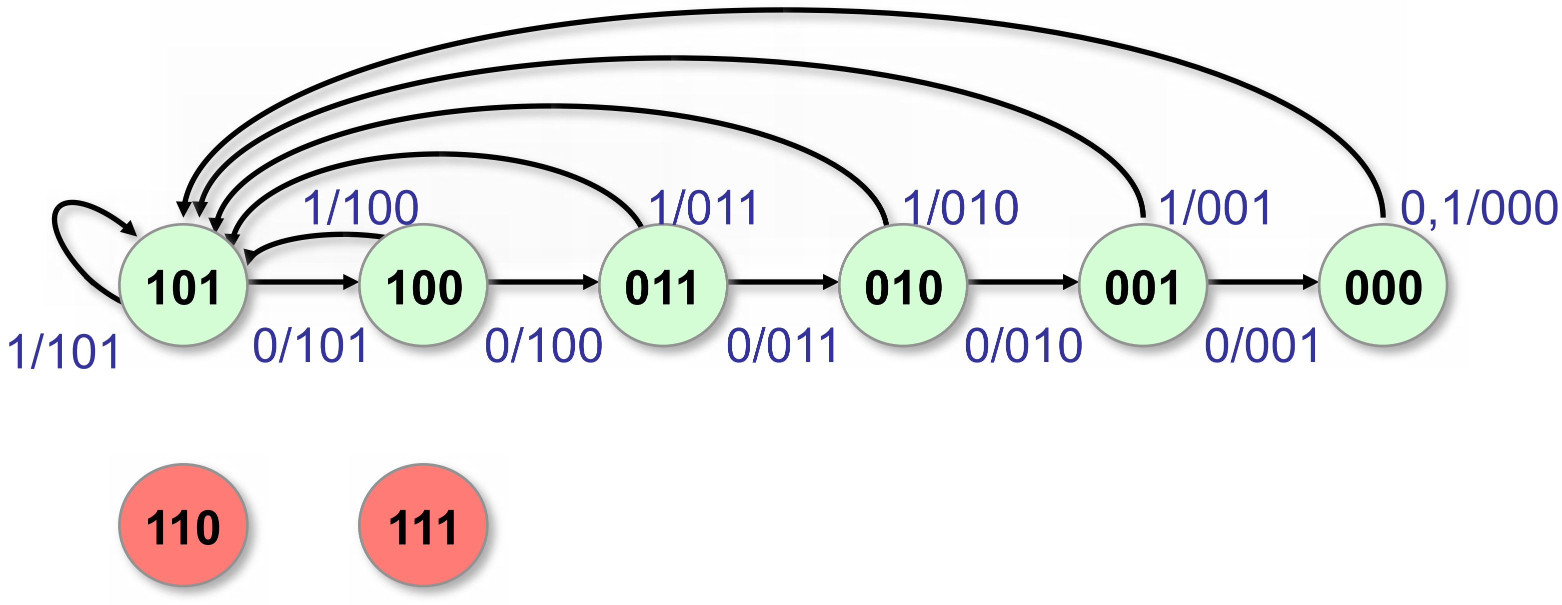
Designer en teller som teller sekvensen 5,4,3,2,1,0. Etter 0 skal telleren gjenta sekvensen (telle rundt). Telleren skal kunne resettes til 5 med ett reset signal.

1) Velger en tilstand for hvert tall ut. Systemet har 1 reset inngang, og trenger 3 utganger for å representere tallene 5 til 0.

2) Velger tilstandskoder som direkte representerer tallene ut. Tallene ut blir gitt av tilstandene

Eksempel nr.4

2) Tegner tilstandsdiagram



Registrerer at vi har to ubrukte tilstander

Eksempel nr.4

3) Tegner tilstandstabell

4) Ingen reduksjons-mulighet

5) Velger å ikke bytte tilstandskoder da utgangene i såfall må omformes

Nåværende tilstand / utgang Q_A	Q_B	Q_C	Inngang R	Neste tilstand Q_A	Q_B	Q_C
0	0	0	0	1	0	1
0	0	0	1	1	0	1
0	0	1	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	0	0	1
0	1	0	1	1	0	1
0	1	1	0	0	1	0
0	1	1	1	1	0	1
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	X	X	X
1	1	0	1	X	X	X
1	1	1	0	X	X	X
1	1	1	1	X	X	X

Ubrukte tilstander

Eksempel nr.4

6) Setter inn i karnaughdiagram og finner forenklede funksjoner

D_A

$Q_A Q_B$ \ $Q_C R$	00	01	11	10
00	1	1	1	0
01	0	1	1	0
11	x	x	x	x
10	0	1	1	1

D_B

$Q_A Q_B$ \ $Q_C R$	00	01	11	10
00	0	0	0	0
01	0	0	0	1
11	x	x	x	x
10	1	0	0	0

D_C

$Q_A Q_B$ \ $Q_C R$	00	01	11	10
00	1	1	1	0
01	1	1	1	0
11	x	x	x	x
10	1	1	1	0

$D_A = R + Q_A'Q_B'Q_C' + Q_A Q_C$

$D_B = Q_B Q_C R' + Q_A Q_C' R'$

$D_C = Q_C' + R$

Eksempel nr.4

6) Sjekker at ubrukte tilstander leder til ønskede tilstander – ok

Nåværende tilstand / utgang Q_A	Q_B	Q_C	Inngang R	Neste tilstand Q_A	Q_B	Q_C
1	1	0	0	0	1	1
1	1	0	1	1	0	1
1	1	1	0	1	1	0
1	1	1	1	1	0	1

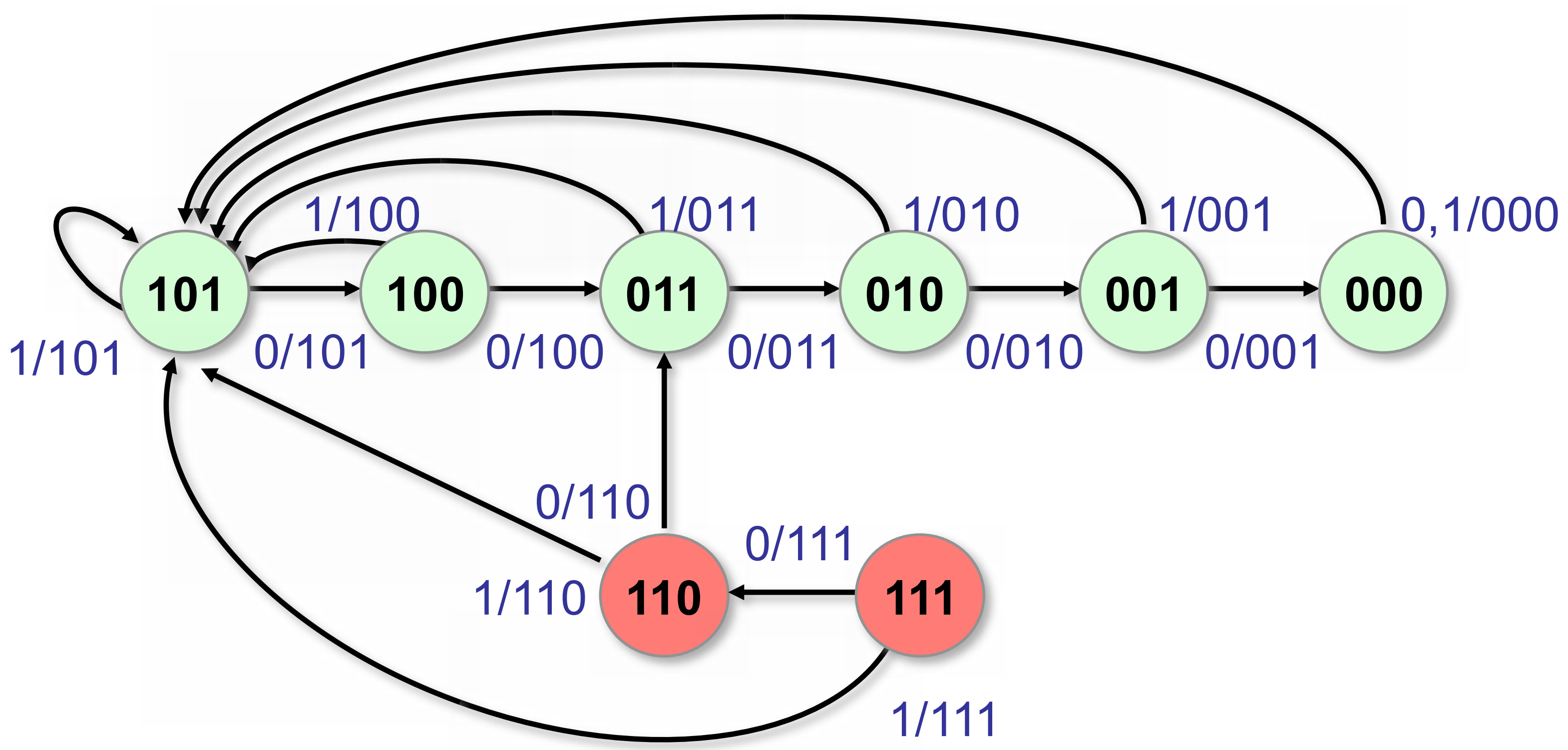
$$D_A = R + Q_A'Q_B'Q_C' + Q_A Q_C$$

$$D_B = Q_BQ_CR' + Q_A Q_C'R'$$

$$D_C = Q_C' + R$$

Eksempel nr.4

6) Alle ubrukte tilstander leder til ønskede tilstander, viser med diagram

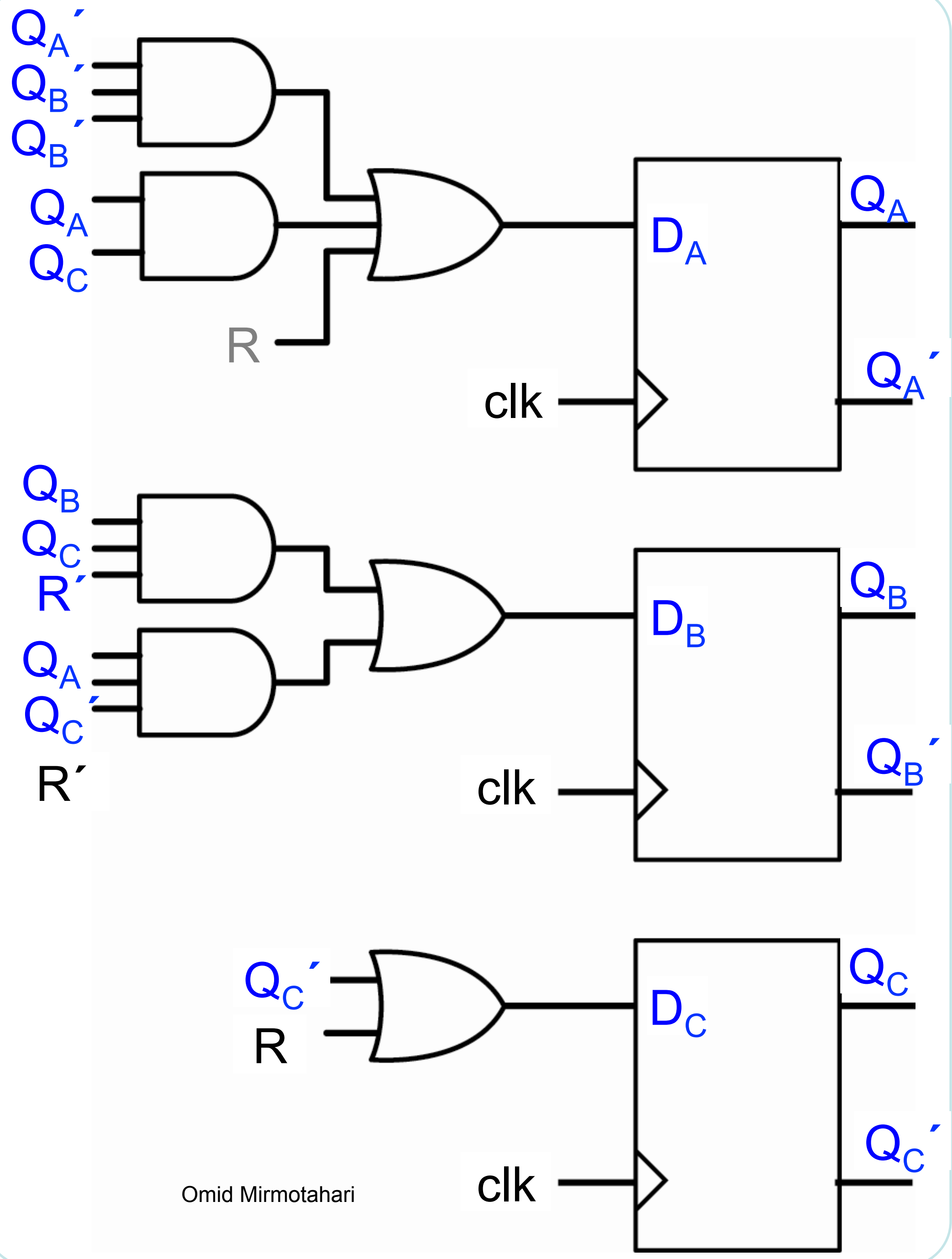


Eksempel nr.4

7) Tegner opp krets

Q_A, Q_B og Q_C blir tellerens utganger

Telleren resettes ved å sette R=1

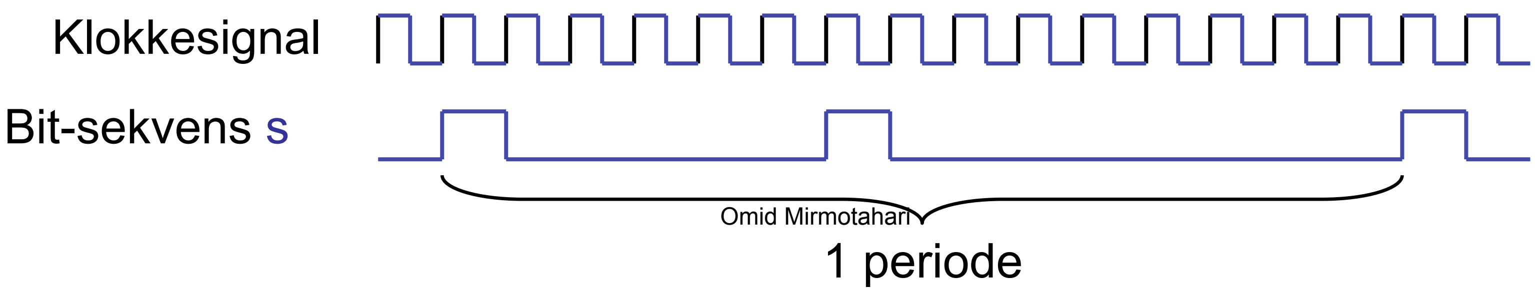


Eksempel nr.5 - trafikklys

Ønsker å bruker tilstandsmaskin for å styre trafikklys

Krysset har to vanlige trafikklys A og B. Disse styres med de binære signalene R_A, G_A, Gr_A samt R_B, G_B, Gr_B. Setter man G_A til "1" lyser det grønt i lys A osv.

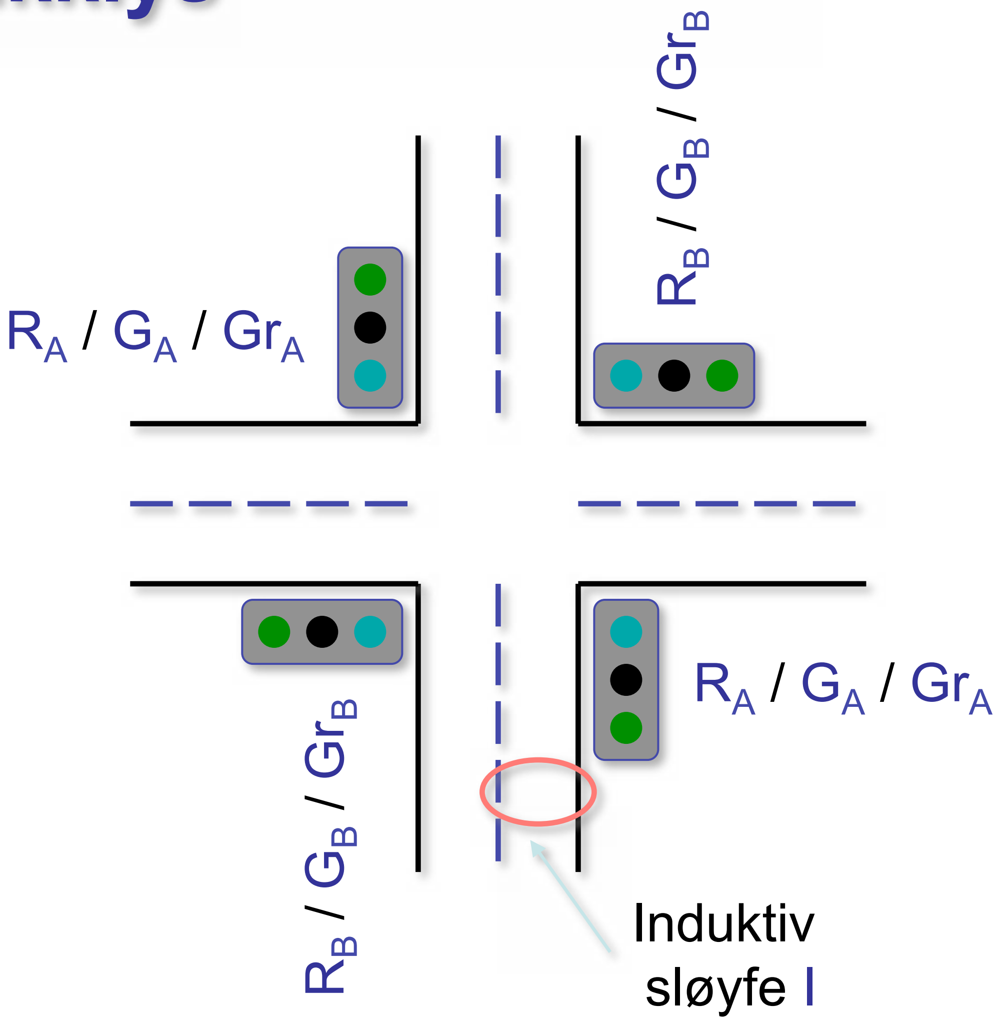
For å generere lyssekvensene bruker vi en repeterende bit-sekvens s som vist under. Avstanden mellom "1"er pulsene gir intervallene mellom skifte fra grønt i lys A til grønt i lys B og motsatt.



Eksempel nr.5 - trafikklys

Systemet har en induktiv sensor i bakken som registrerer biler den ene veien. Bil over sensoren gir I=1 ellers har vi I=0

Vi ønsker at bil registrert av sensoren skal gi grønt lys i A så fort som mulig



Eksempel nr.5

1,2) Velger følgende forenklede tilstander:

00 - Grønt lys i A, rødt lys i B

01 - Gult lys i A og B. Skifter mot grønt lys i B.

10 - Rødt lys A, grønt lys i B

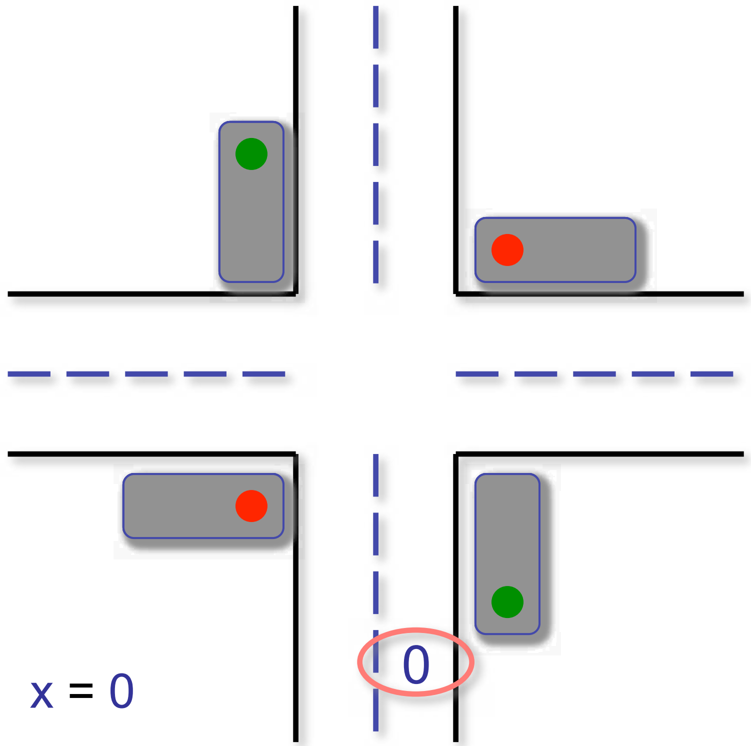
11 - Gult lys i A og B. Skifter mot grønt lys i A.

Innganger: s, l

Utganger: R_A, G_A, Gr_A, R_B, G_B, Gr_B

Lar utgangene kun være en funksjon av tilstanden

Eksempel nr.5

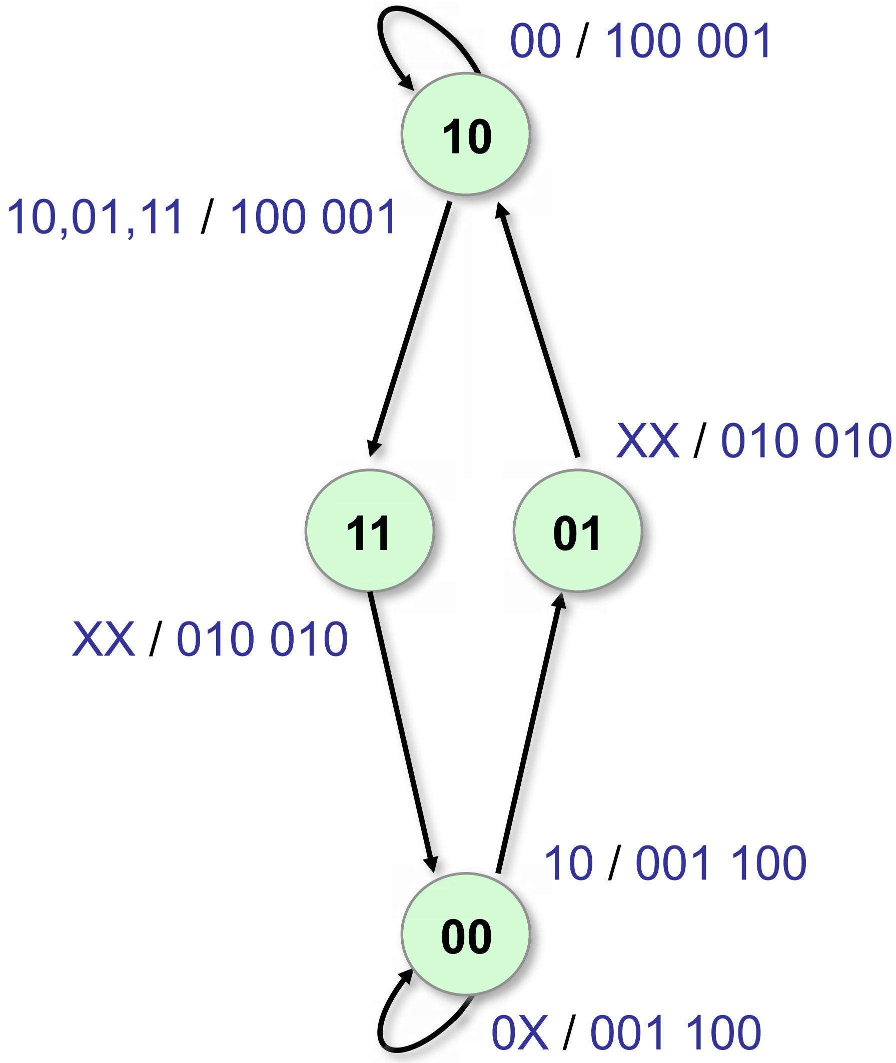


Eksempel nr.5

2) Tilstandsdiagram

X – don't care

sI / $R_A G_A Gr_A \ R_B G_B Gr_B$



6) Finner kombinatoriske funksjoner

$D_A = Q_A \oplus Q_B$

$D_B = Q_A Q_B{'} l + Q_B{'} s l{'}$

$R_A = Q_A Q_B{'}$

$G_A = Q_B$

$Gr_A = Q_A{'} Q_B{'}$

$R_B = Gr_A$

$G_B = G_A$

$Gr_B = R_A$

Nåværende tilstand		Innganger		Neste tilstand		Utganger					
Q_A	Q_B	s	l	Q_A	Q_B	R_A	G_A	Gr_A	R_B	G_B	Gr_B
0	0	0	0	0	0	0	0	1	1	0	0
0	0	0	1	0	0	0	0	1	1	0	0
0	0	1	0	0	1	0	0	1	1	0	0
0	0	1	1	0	0	0	0	1	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0
0	1	0	1	1	0	0	1	0	0	1	0
0	1	1	0	1	0	0	1	0	0	1	0
0	1	1	1	1	0	0	1	0	0	1	0
1	0	0	0	1	0	1	0	0	0	0	1
1	0	0	1	1	1	1	0	0	0	0	1
1	0	1	0	1	1	1	0	0	0	0	1
1	0	1	1	1	1	1	0	0	0	0	1
1	1	0	0	0	0	0	1	0	0	1	0
1	1	0	1	0	0	0	1	0	0	1	0
1	1	1	0	0	0	0	1	0	0	1	0
1	1	1	1	0	0	0	1	0	0	1	0

Eksempel nr.5

7) Tegner opp krets

$D_A = Q_A \oplus Q_B$

$D_B = Q_A Q_B' I + Q_B' s I'$

$R_A = Q_A Q_B'$

$G_A = Q_B$

$Gr_A = Q_A' Q_B'$

$R_B = Gr_A$

$G_B = G_A$

$Gr_B = R_A$