INF 1400 - Oppsummering

Dawid Kuleczko

December 6, 2016

Contents

1 Digital representasjon og binær logikk:

1.1 Tallsystemer:

I dette kurset skal vi legge spesielt vekt på binære tall, men heksadesimale tall og octale tall, samt tall fra andre tallsystemer, kan også forekomme.

Til daglig bruker vi desimal, altså 10-tallsystemet. Et desimalt tall er representert ved symbolene fra 0 til 9.

eks. $(951)_{dec} = 9 \cdot 10^2 + 5 \cdot 10^1 + 1 \cdot 10^0$

Et binært tall er representert ved symbolene 0 og 1.

eks. $(101)_{bin} = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

Et heksadesimalt tall er representert ved symbolene fra 0 til F der tallene etter 9 er: 10 = A 11 = B 12 = C 13 = D 14 = E 15 = F

eks. $(5BF)_{hex} = 5 \cdot 16^2 + 11 \cdot 16^1 + 15 \cdot 16^0$

Et oktalt tall er representert ved symbolene fra 0 til 8.

eks. $(853)_{oct} = 8 \cdot 8^2 + 5 \cdot 8^1 + 3 \cdot 3^0$

1.2 Konvertering:

1.2.1 Konvertering fra grunntall r til desimaltall:

Generelt: $(...a_2 a_1 a_0, a_{-1} a_{-2}...) = ... + a_2 \cdot r^2 + a_1 \cdot r^1 + a_0 \cdot r^0, a_{-1} \cdot r^{-1} a_{-2} \cdot r^{-2} + ..$

eks. $(1A5, 1C)_{16} = 1 \cdot 16^2 + A \cdot 16^1 + 5 \cdot 16^0 + 1 \cdot 16^{-1} + 12 \cdot 16^{-2} = (421, 109375)_{10}$

1.2.2 Konvertering av desimal til binær:

Eks. $(23)_{10}$

$$23/2 = 11 + 1/2 \qquad a_0 = 1$$

$$11/2 = 5 + 1/2 \qquad a_1 = 1$$

$$5/2 = 2 + 1/2 \qquad a_2 = 1$$

$$2/2 = 1 + 0/2 \qquad a_3 = 0$$

$$1/2 = 0 + 1/2 \qquad a_4 = 1$$

Vi leser det binære tallet nedfra $(10111)_2$

1.2.3 Konvertering fra desimal til grunntall "r":

Samme måte som over bare med "r" isteden for 2.

Eks. $(12)_{10}$ til 3-tallsystemet:

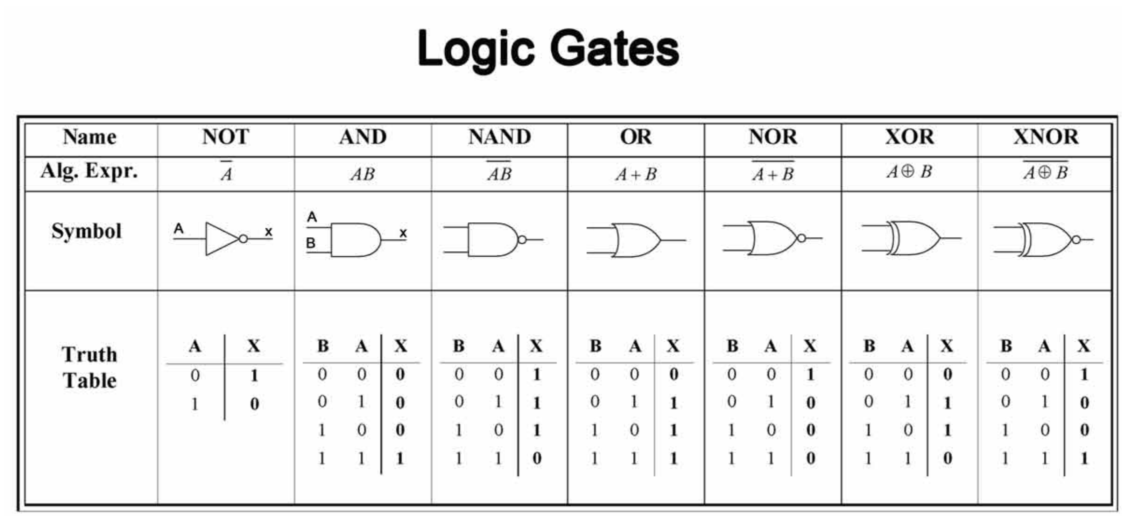
$$12/3 = 4 + 0/3 \qquad a_0 = 0$$

$$4/3 = 1 + 1/3 \qquad a_1 = 1$$

$$1/3 = 0 + 1/3 \qquad a_2 = 1$$

Tallet blir da $(110)_3$ i 3 tallsystemet.

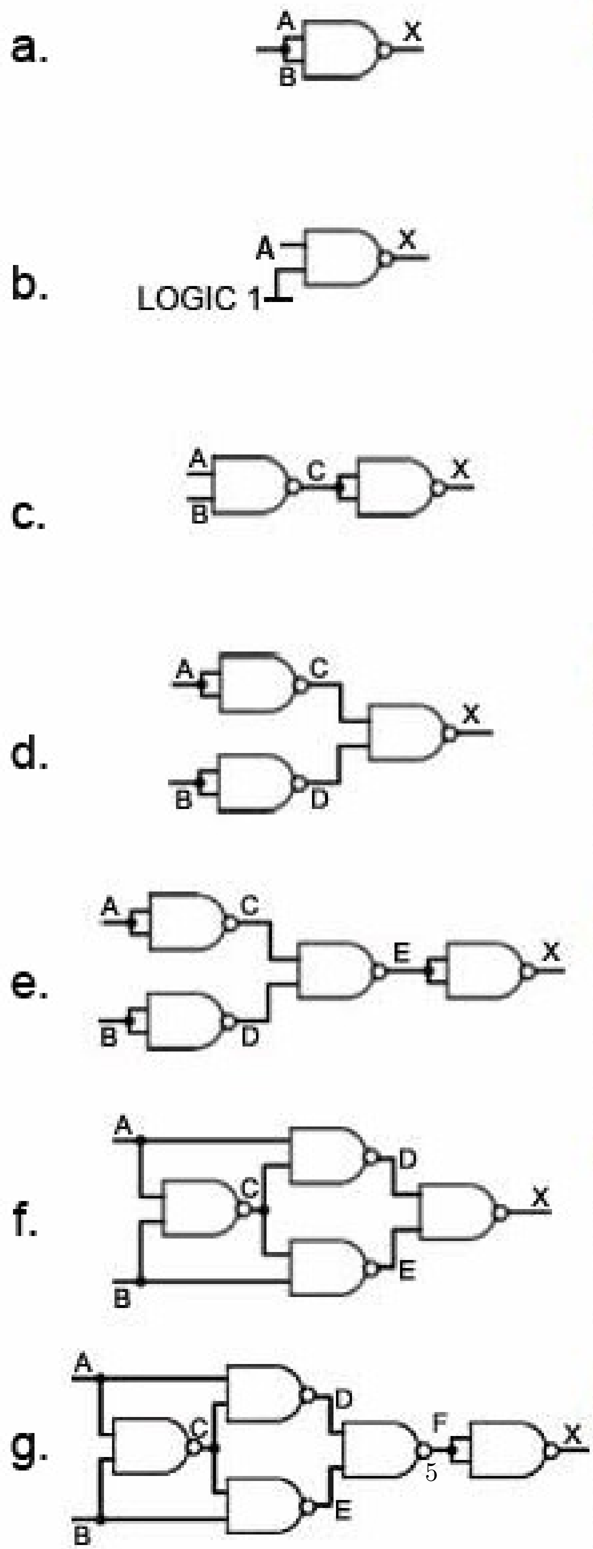
1.2.4 Sannhetstabell:



Logic Gates

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
Symbol	A x	A B x					
Truth Table	A X 0 **1** 1 **0**	B A X 0 0 **0** 0 1 **0** 1 0 **0** 1 1 **1**	B A X 0 0 **1** 0 1 **1** 1 0 **1** 1 1 **0**	B A X 0 0 **0** 0 1 **1** 1 0 **1** 1 1 **1**	B A X 0 0 **1** 0 1 **0** 1 0 **0** 1 1 **0**	B A X 0 0 **0** 0 1 **1** 1 0 **1** 1 1 **0**	B A X 0 0 **1** 0 1 **0** 1 0 **0** 1 1 **1**

Figure 1: Logiske porter



NOT

A	B	X
0	0	1
1	1	0

NOT

A	B	X
0	1	1
0	1	1
1	1	0
1	1	0

AND

A	B	C	X
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

OR

A	B	C	D	X
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

NOR

A	B	C	D	E	X
0	0	1	1	0	1
0	1	1	0	1	0
1	0	0	1	1	0
1	1	0	0	1	0

XOR

A	B	C	D	E	X
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	0

XNOR

A	B	C	D	E	F	X
0	0	1	1	1	0	1
0	1	1	1	0	1	0
1	0	1	0	1	1	0
1	1	0	1	1	0	1

Figure 2: Logiske porter som 2 inputs NAND-porter

1.3 Huntington's postulater

Huntington postulater

(P0)	Mengden {0,1} er lukket under "+" og "•"		lukket
(P1)	x + x = x	x • x = x	
(P2)	x + 0 = x	x • 1 = x	ident.el.
(P2b)	x + 1 = 1	x • 0 = 0	
(P5)	x + x´ = 1	x • x´= 0	komplem.
(P6)	0 ≠ 1		minst 2 el.
(P3)	x + y = y + x	x • y = y • x	kommutativ
(P4)	x • (y + z) = x • y + x • z	x+(y • z) = (x+y) • (x+z)	distributiv
(P5)	(x´)´ = x		

Dualitet for postulatene:

Kan bytte "•" med + hvis man bytter "0" med "1"

Presedens:

Først utføres "0", så "´", så "•" og til slutt "+"

Figure 3: Huntington's postulater

2 Boolsk algebra:

DeMorgans teorem

(x • y)´ = x´+ y´

(x + y)´ = x´ • y´

På invertert form:

x• y = (x´+ y´)´

x + y = (x´• y´)´

Figure 4: DeMorgans teorem

Komplement av funksjon

Inverterer begge sider og bruker DeMorgan

Eksempel:

$F = x'yz'+x'y'z$

$F' = (x'yz'+x'y'z)'$

$F' = (x'yz')'(x'y'z)'$

$F' = (x+y'+z)(x+y+z')$

Eksempel:

$F = x(y'z'+yz)$

$F' = (x(y'z'+yz))'$

$F' = x'+(y'z'+yz)'$

$F' = x'+(y'z')'(yz)'$

$F' = x'+(y+z)(y'+z')$

Figure 5: Komplement av funkson

Regneregler - oversikt

$x + 0 = x$	$x \bullet 1 = x$
$x + x' = 1$	$xx' = 0$
$x + y = y+x$	$xy = yx$
$x + (y+z) = (x+y) + z$	$x\,(yz) = (xy)z$
$x(y+z) = xy + xz$	$x + (yz) = (x+y)(x+z)$
$x + x = x$	$x \bullet x = x$
$x + 1 = 1$	$x \bullet 0 = 0$
$x + xy = x$	$x(x+y) = x$
$(x+y)' = x'y'$	$(xy)' = x' + y'$

Figure 6: Regne regler for boolsk algebra

2.1 Boolske funksjoner med sannhetstabell

En boolsk funksjon kan visualiseres i en sannhetstabell. En gitt funksjon har kun en sannhetstabell Men, en gitt sannhetstabell har uendelig mange funksjonsuttrykk.

Eksempel: F = x + y' z

XYZ	F
000	0
001	1
010	0
011	0
100	1
101	1
110	1
111	1

2.2 Forenkling av uttryk

En funksjon kan forenkles ved regneregler for å gjøre den lettere å håndere og implementere.

Forenklingseksempler

Eksempel:

F = x(x'+y)

F = xx'+xy

F = 0+xy

F = xy

Eksempel:

F = x+x'y

F = (x+x')(x+y)

F = 1(x+y)

F = x+y

Eksempel:

F = (x+y)(x+y')

F = x + (yy')

F = x + 0

F = x

Figure 7: Forenklingseksempler av noen funksjoner

Forenklingseksempler II

Eksempel:

$F = xy+x'z+yz$

$F = xy+x'z+yz(x+x')$

$F = xy+x'z+xyz+x'yz$

$F = xy(1+z)+x'z(1+y)$

$F = xy+x'z$

Eksempel:

$F = (x+y)(x'+z)(y+z)$

$F = (x+y)(x'+z)$ Dualitet

Figure 8: Forenklingseksempler av noen funksjoner

2.3 Maksterm/Minterm

Variable			Minterm		Maxterm	
x	y	z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x+y+z$	M_0
0	0	1	$x'y'z$	m_1	$x+y+z'$	M_1
0	1	0	$x'yz'$	m_2	$x+y'+z$	M_2
0	1	1	$x'yz$	m_3	$x+y'+z'$	M_3
1	0	0	$xy'z'$	m_4	$x'+y+z$	M_4
1	0	1	$xy'z$	m_5	$x'+y+z'$	M_5
1	1	0	xyz'	m_6	$x'+y'+z$	M_6
1	1	1	xyz	m_7	$x'+y'+z'$	M_7

Figure 9: Tabell for minterm og maksterm

Mintermer har notasjon m_x

$F(x, y, z) = \sum(m_3, m_6) = \sum(3, 6) = x'yz' + xyz'$

Makstermer har notasjon M_x

$F(x, y, z) = \Pi(M_3, M_6) = \Pi(3, 6) = (x + y' + z')(x' + y' + z)$

2.3.1 Minterm:

I en funksjon kan en binær variabel x opptre som x eller x'. En funksjon kan være gitt på "sum av produkt" form. Eksempel: F = xy + xy' + x

Hvert "produktledd" som inneholder alle variablene kalles en minterm. For to variable finnes det 4 forskjellige mintermer: xy + xy' + x'y + x'y' For 3 variable finnes det 2^3 forskjellige mintermer.

Hvis man generer en funksjon ut i fra sannhetstabellen får man en sum av mintermer

Eksempel: F = x'y'z + xy'z' + xyz'

XYZ	F
000	0
001	1
010	0
011	0
100	1
101	0
110	1
111	0

En sannhetstabell kan sees på som en liste av mintermer.

2.3.2 Maksterm:

En funksjon kan være gitt på "produkt av sum" form.

Eksempel: F = (x+y)(x+y')y Hvert "summeledd" som inneholder alle variablene kalles maksterm.

For to variable finnes det 4 forskjellige makstermer: (x'y)(x+y')(x'+y)(x'+y')

For n variable finnes det 2^n forskjellige makstermer.

2.4 Designprosedyre

Det er ikke alltid at det enkleste funksjonsuttrykket resulterer i den enkleste port-implementasjonen.

Ved forenkling på portnivå må man vite hvilke porter man har til rådighet, og så justere funksjonsuttrykket mot dette. (håndverk)

Generell design prosedyre

1.Bestem hvilke signal som er innganger og utganger

2.Sett opp sannhetstabell for alle inngangskombinasjoner

3.Generer funksjonsuttrykket som sum av mintermer

4.Tilpass / forenkle funksjonsuttrykket mot aktuelle porter

3 Karnaugh diagram:

Grafisk metode for forenkling av Boolske uttryk

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

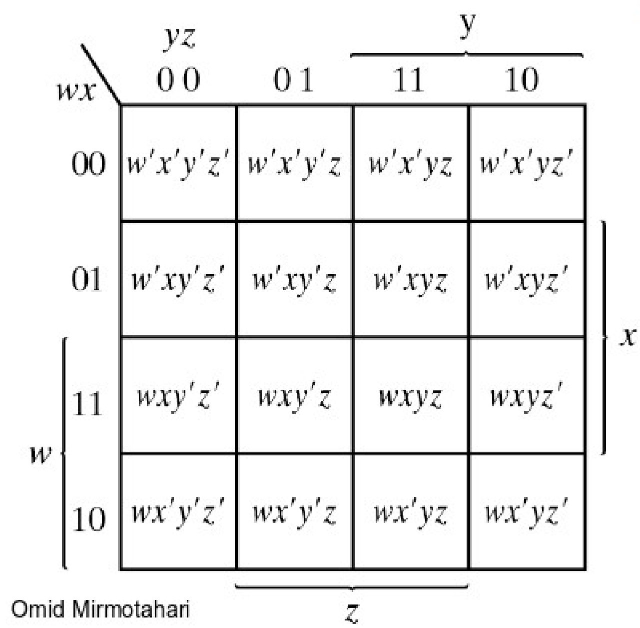


Figure 10: Karnaugh diagram med mintermene

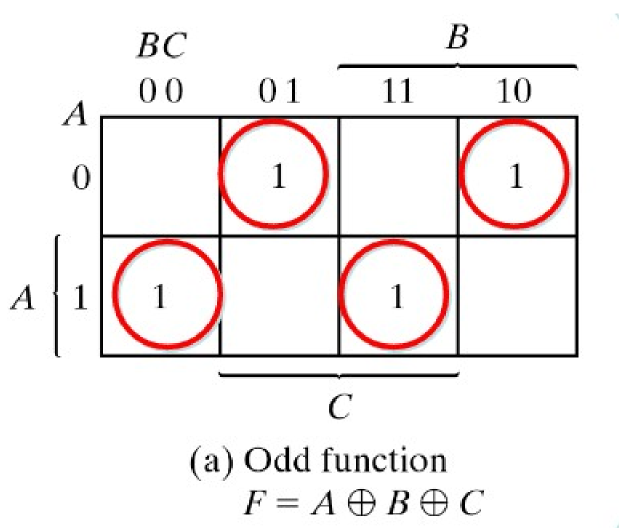


Figure 11: XOR

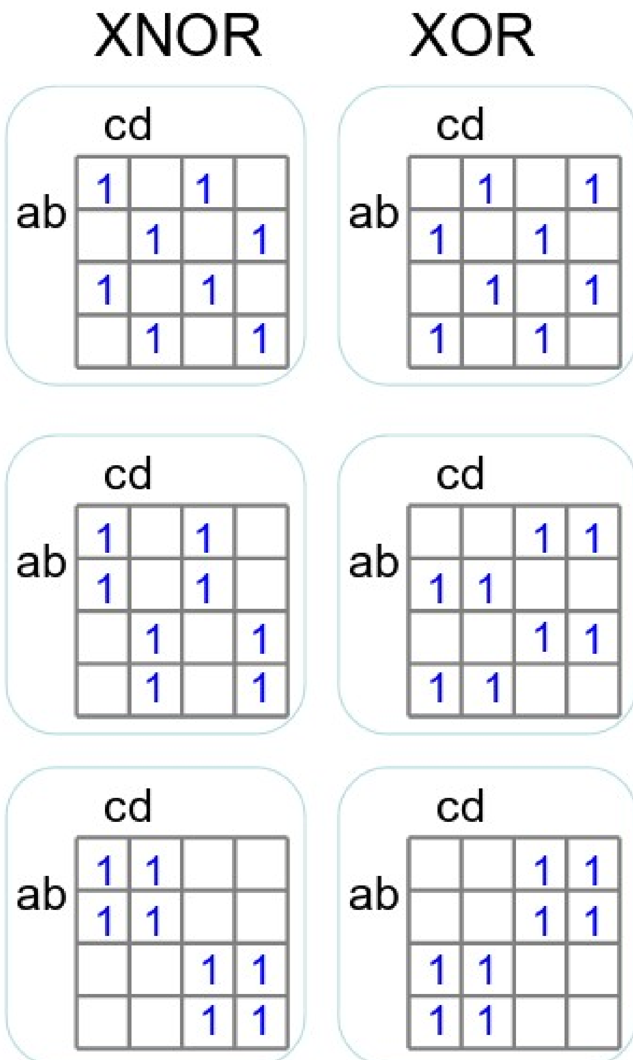


Figure 12: XNOR og XOR

4 Kobinatorisk logikk:

4.1 Binær addisjon:

Prosedyren for binær addisjon er identisk med prosedyren for desimal addisjon:

$$\text{Eks:}\quad \begin{bmatrix} 0101 & \\ 1011 & + \\ -------- & \\ 10000 & \end{bmatrix} 5 + 11 = 16$$

4.2 Binær substraksjon:

7	0 1 1 1
6	0 1 1 0
5	0 1 0 1
4	0 1 0 0
3	0 0 1 1
2	0 0 1 0
1	0 0 0 1
0	0 0 0 0
-1	1 1 1 1
-2	1 1 1 0
-3	1 1 0 1
-4	1 1 0 0
-5	1 0 1 1
-6	1 0 1 0
-7	1 0 0 1
-8	1 0 0 0

Figure 13: Representasjon av binære negative tall

For å substrahere negative binære tall, bruker man toerkomplement metoden.

Det tallet som skal substraherers med må inverteres og plusses på 1, deretter skal det plusses med den andre tallet. Tallet til overs går ut.

Eks: $\begin{bmatrix} 1101 & \\ 1011 & - \\ -------- & \end{bmatrix} \begin{bmatrix} 1101 & \\ 0101 & + \\ -------- & \\ (1)0010 & \end{bmatrix}$ 13-11 = 2

13+5=18 = 1 0 0 1 0

4.3 Binære addere:

4.3.1 Halvadder:

Halvaddere tar ikke mente inn.

Halvadder implementasjon

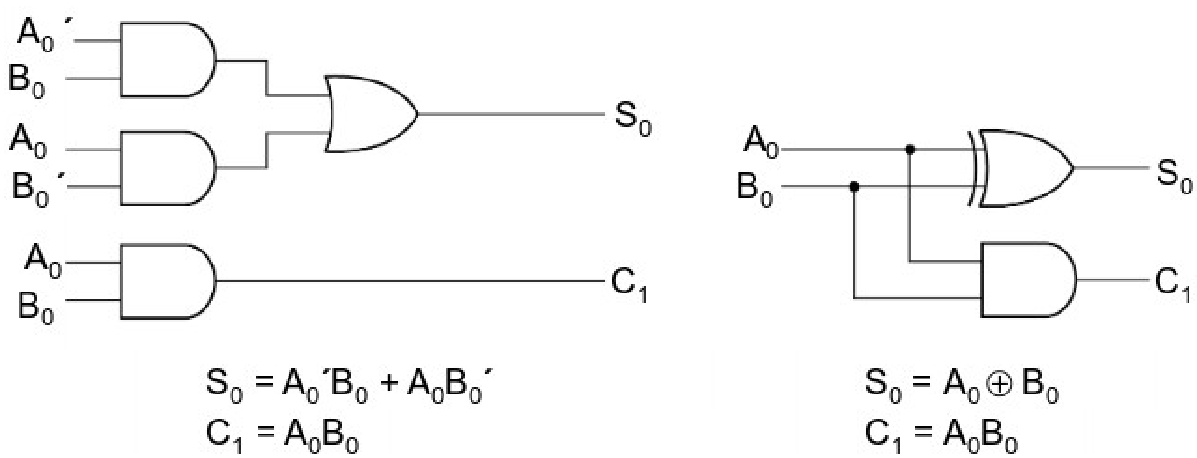


Figure 14: Halvadder implementasjon

4.3.2 Fulladder:

Fulladder tar mente inn.

Fulladder implementasjon

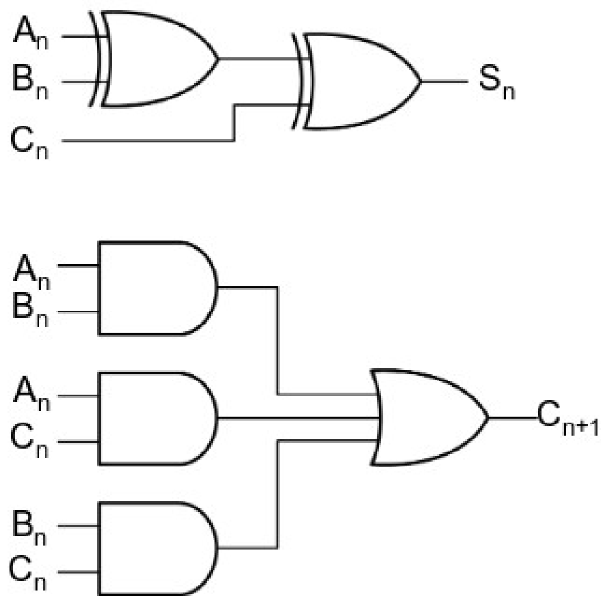


Figure 15: Fulladder implementasjon

4.3.3 Et adder system

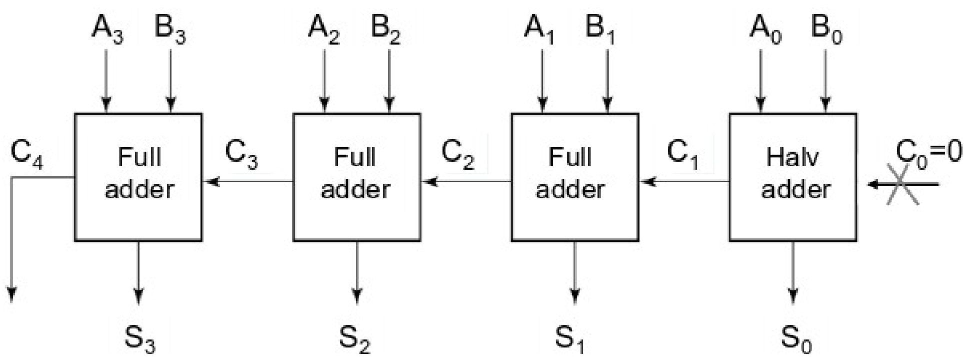


Figure 16: Et system av halv og fulladdere

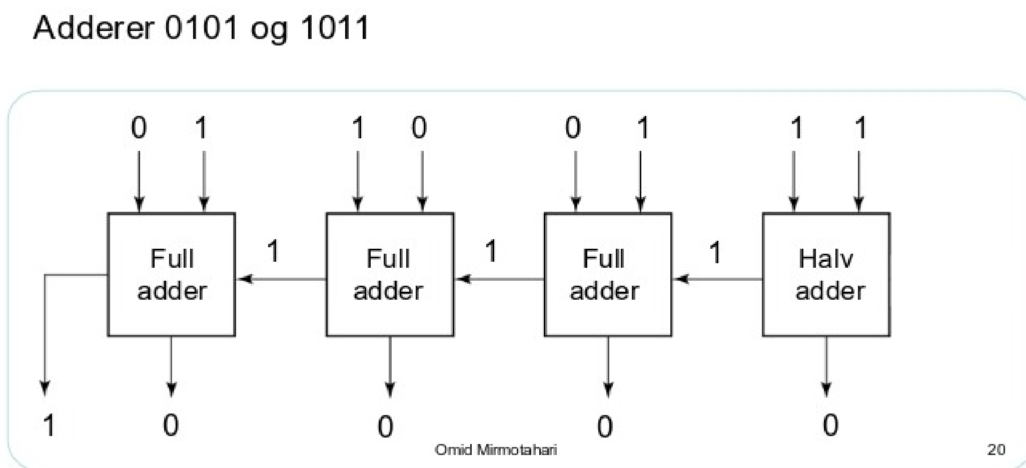


Figure 17: Et eksempel på addisjon av to binære 4-bits tall

4.4 "Carry Lookahead"

Ønsker å unngå menteforplantning – gir økt hastighet

G_i – generate: brukes i menteforplantningen P_i – propagate: påvirker menteforplantningen

4.5 Komparator:

Komparator – sammenligner to tall A og B

$3utganger : A = B, A > BogA < B$

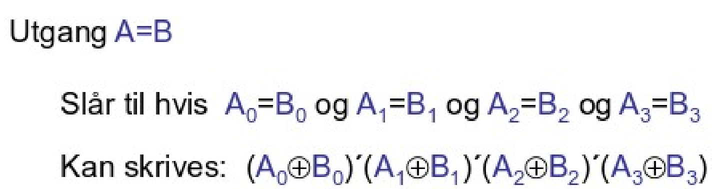


Figure 18: A=B

Utgang $A>B$ slår til hvis:

($A_3>B_3$) eller

($A_2>B_2$ og $A_3=B_3$) eller

($A_1>B_1$ og $A_2=B_2$ og $A_3=B_3$) eller

($A_0>B_0$ og $A_1=B_1$ og $A_2=B_2$ og $A_3=B_3$)

Kan skrives:

$$(A_3B_3') + (A_2B_2')(A_3\oplus B_3)' + (A_1B_1')(A_2\oplus B_2)'(A_3\oplus B_3)' +$$
$$(A_0B_0')(A_1\oplus B_1)'(A_2\oplus B_2)'(A_3\oplus B_3)'$$

Figure 19: $A > B$

Utgang $A<B$ slår til hvis:

($A_3<B_3$) eller

($A_2<B_2$ og $A_3=B_3$) eller

($A_1<B_1$ og $A_2=B_2$ og $A_3=B_3$) eller

($A_0<B_0$ og $A_1=B_1$ og $A_2=B_2$ og $A_3=B_3$)

Kan skrives:

$$(A_3'B_3) + (A_2'B_2)(A_3\oplus B_3)' + (A_1'B_1)(A_2\oplus B_2)'(A_3\oplus B_3)' +$$
$$(A_0'B_0)(A_1\oplus B_1)'(A_2\oplus B_2)'(A_3\oplus B_3)'$$

Figure 20: $A < B$

4.6 Dekoder:

Dekoder – tar inn et binært ord og gir ut alle mintermer Kan generere generelle logiske funksjoner direkte fra mintermene på utgangen Eksempel: 3bit inn/8bit ut

Eksempel: 3bit inn

Innganger			Utganger							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Figure 21: Sannhetstabell til en 3 bits decoder

4.7 Enkoder:

Enkoder er motsatt av dekoder

Eksempel: 8x3 enkoder

Innganger								Utganger		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Figure 22: Sannhetstabell for en 8x3 enkoder

Prioritets-enkoder:

Hvis flere "1"ere inn - ser kun på inngang med høyst indeks (prioritet)

4.8 Multiplekser (MUX):

Velger hvilke innganger som slippes ut

Hver inngang kan bestå av ett eller flere bit

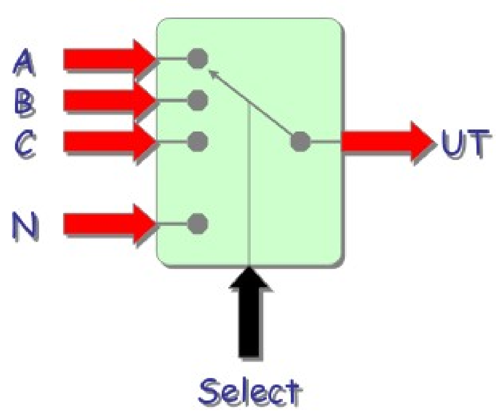


Figure 23: MUX

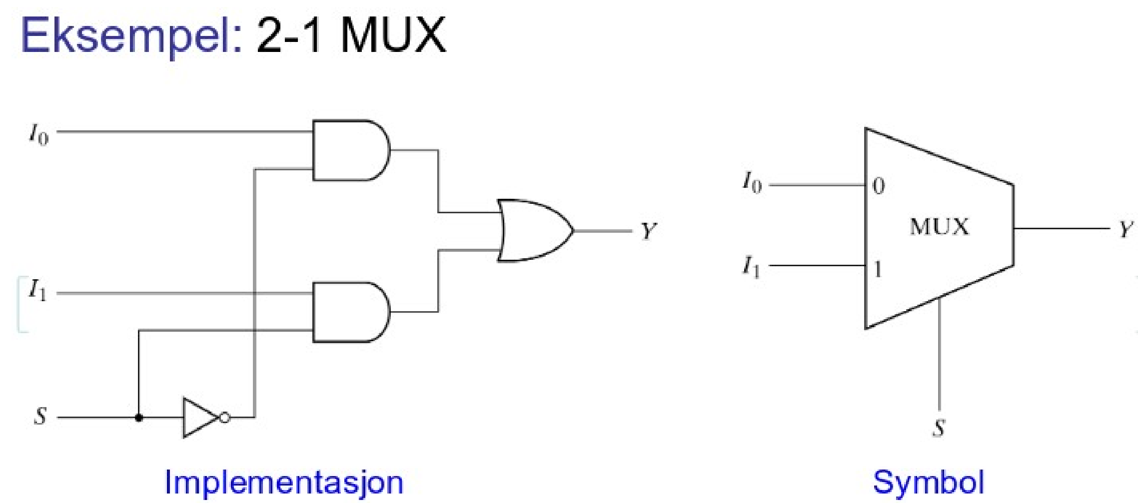


Figure 24: En 2-1 MUX

4.9 Demulitplekser:

Motsatt av MUX, velger hvilke utganger som slippes ut

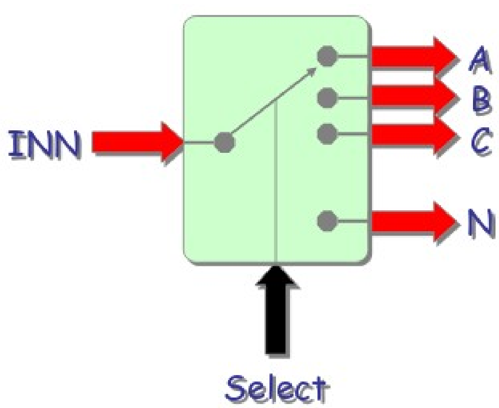


Figure 25: Demultiplekser

4.10 Aritmetisk logisk enhet (ALU):

En elektronisk krets som utfører aritmetiske og logiske operasjoner

5 Sekvensiell logikk:

Kombinatorisk logikk: Utgangsverdiene er entydig gitt av nåværende kombinasjon av inngangsverdier.

Sekvensiell logikk: Inneholder hukommelse (låsekretser). Utgangsverdiene er gitt av nåværende kombinasjon av inngangsverdier, samt sekvensen av tidligere inngangs-/utgangsverdier.

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

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.

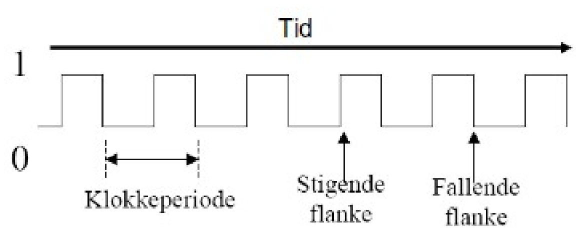


Figure 26: Flanker

Ønsker så høy klokkefrekvens som mulig, for at hver enkelt operasjon bruker så kort tid som mulig

Maksimal klokkefrekvens bestemmes av: - Lengde på singalveiene - Last - Forsinkelsene gjennom porter(delay) - Teknologi

5.1 Logisk dybde

Antall proter et singal passerer fra inngang til utgang, ved å redusere dette minsker forsinkelsen gjennom kretsen.

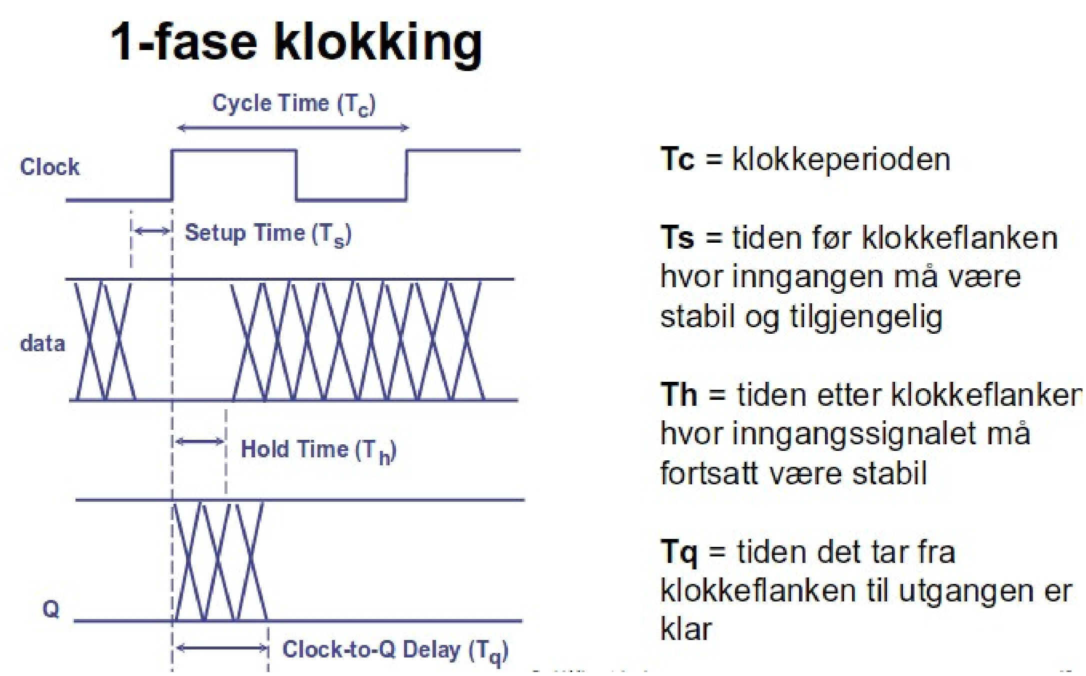


Figure 27: Klokking

5.2 Latcher(låsekretser):

De ulike latchene:

SR - Latch

Set Reset - Latch

Setter Q til "1" hvis den får "1" på inngang S.

Når inngang S går tilbake til "0" skal Q forbli på "1"

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"

Tilstanden "1" på både S og R brukes normalt ikke

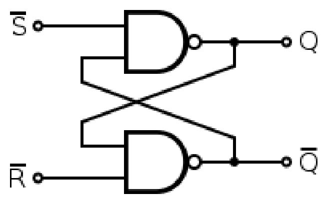
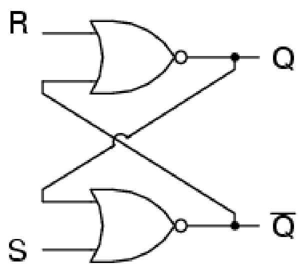


Figure 28: SR-Latch med NAND



S	R	Q	$\overline{Q}$
0	0	latch	latch
0	1	0	1
1	0	1	0
1	1	0	0

Figure 29: SR-Latch med 2 NOR

D - Latch

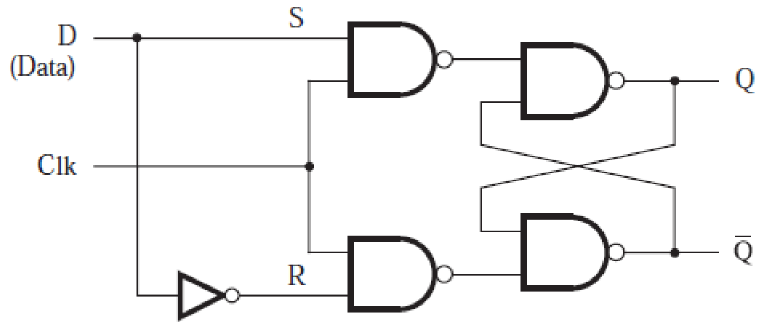
Data - Latch

Dataflyten gjennom en D-latch kontrolleres av et klokkesignal

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

I det øyeblikket klokkeinngangen går fra "1" til "0" låser utgangen seg på sin nåværende verdi.

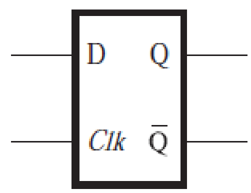
Forandringer på inngangen vil ikke påvirke utgangsverdien så lenge klokkesignalet er "0"



(a) Circuit

Clk	D	$Q(t+1)$
0	x	$Q(t)$
1	0	0
1	1	1

(b) Characteristic table



(c) Graphical symbol

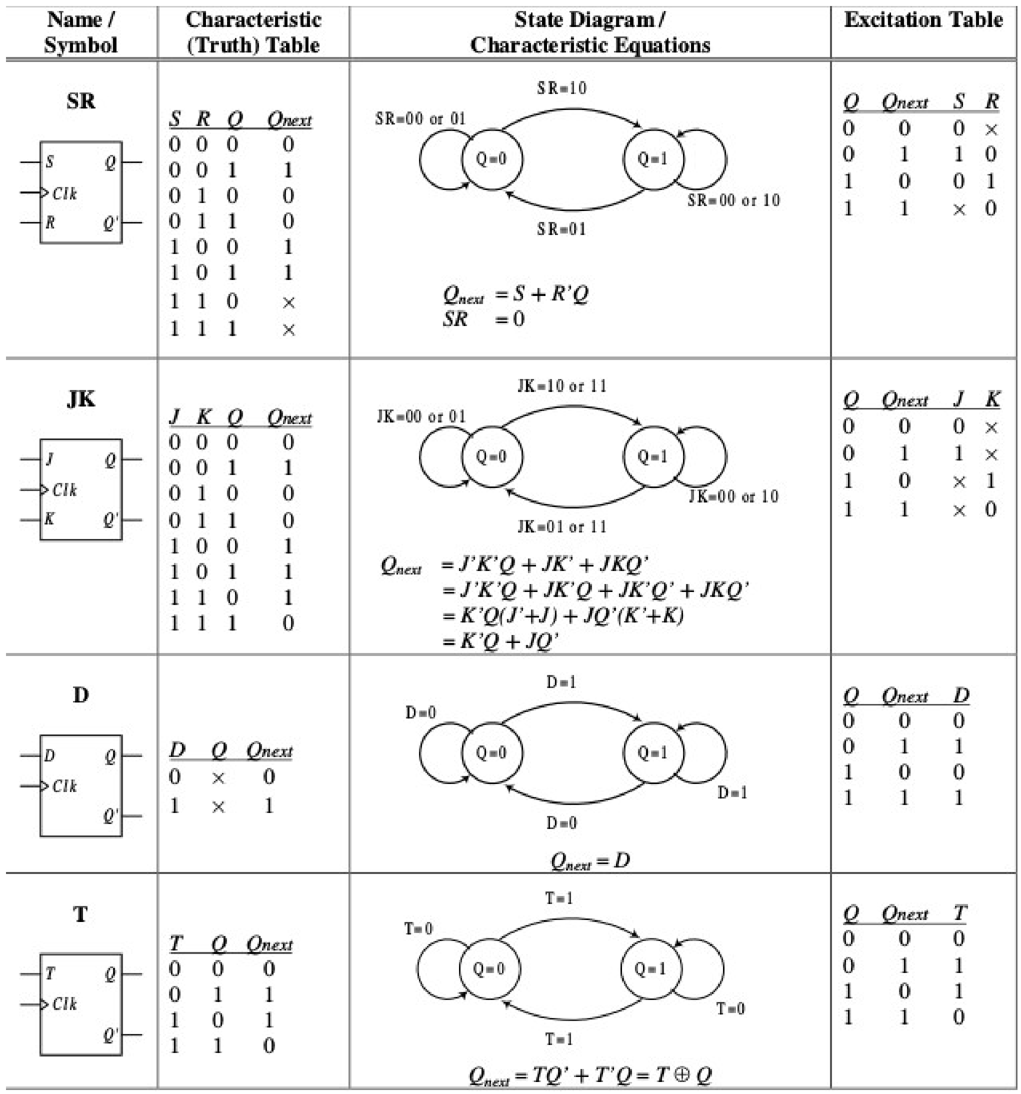
Figure 30: D-Latch

5.3 Flip-Floper:

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 til 1 (eller 1 til 0, avhengig av konstruksjonen)

En flip-flop'er har enten en eller to innganger (pluss klokkesignal) og en utgang



Name / Symbol	Characteristic (Truth) Table	State Diagram / Characteristic Equations	Excitation Table
SR	S R Q Q_{next} 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0 × 1 1 1 ×	SR=10 SR=00 or 01 Q=0 Q=1 SR=00 or 10 SR=01 $Q_{next} = S + R'Q$ $SR = 0$	Q Q_{next} S R 0 0 0 × 0 1 1 0 1 0 0 1 1 1 × 0
JK	J K Q Q_{next} 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1 0	JK=10 or 11 JK=00 or 01 Q=0 Q=1 JK=00 or 10 JK=01 or 11 $Q_{next} = J'K'Q + JK' + JKQ'$ $= J'K'Q + JK'Q + JK'Q' + JKQ'$ $= K'Q(J'+J) + JQ'(K'+K)$ $= K'Q + JQ'$	Q Q_{next} J K 0 0 0 × 0 1 1 × 1 0 × 1 1 1 × 0
D	D Q Q_{next} 0 × 0 1 × 1	D=1 D=0 Q=0 Q=1 D=1 D=0 $Q_{next} = D$	Q Q_{next} D 0 0 0 0 1 1 1 0 0 1 1 1
T	T Q Q_{next} 0 0 0 0 1 1 1 0 1 1 1 0	T=1 T=0 Q=0 Q=1 T=0 T=1 $Q_{next} = TQ' + T'Q = T \oplus Q$	Q Q_{next} T 0 0 0 0 1 1 1 0 1 1 1 0

Figure 31: Typer Flip-floper

Flip-Flop'er kommer i to varianter:

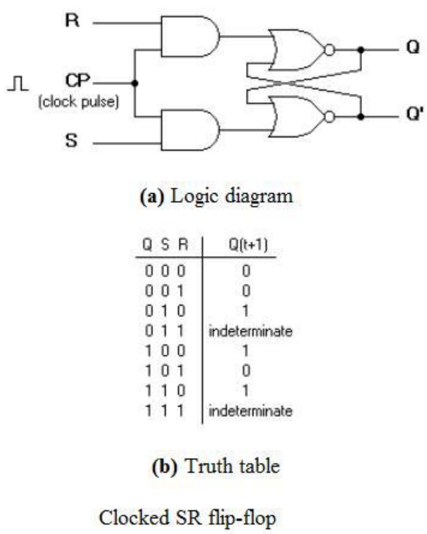
- Positiv flanketrigget
- Negativ flanketrigget

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".

De ulike Flip-Flopene:

SR - Flip-flop Set Reset - Flip-Flop



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

Figure 32: Kretsoppbygging til en SR-Flip-Flop

JK - Flip-flop J(Set) K(Reset) - Flip-Flop

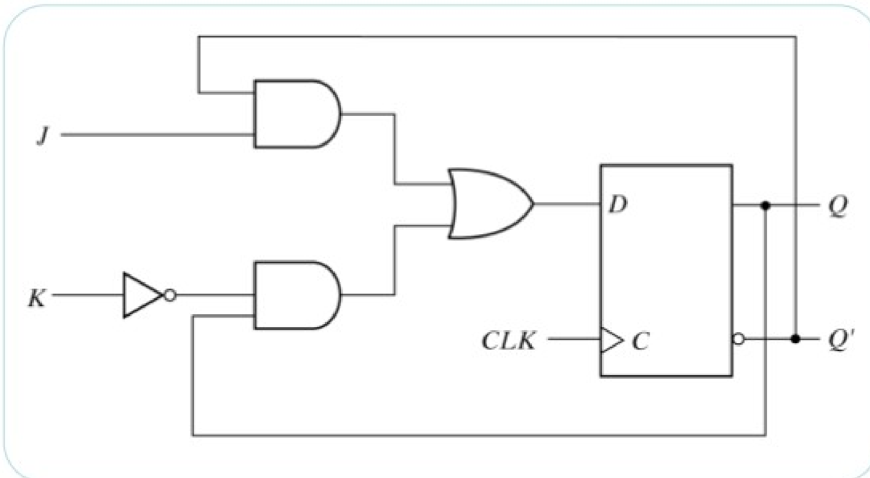
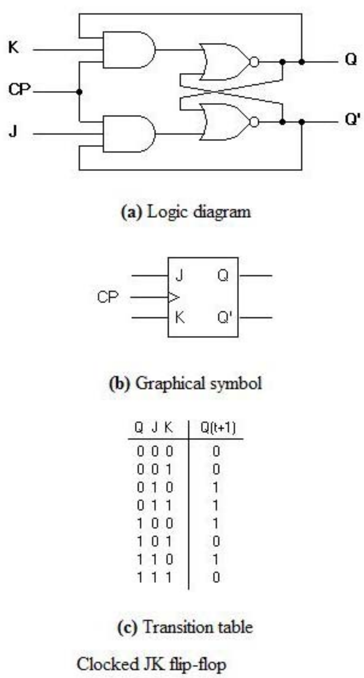


Figure 33: Kretsoppbygging til en JK-Flip-Flop



Q 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

Figure 34: Typer Flip-floper

D - Flip-flop Data - Flip-Flop

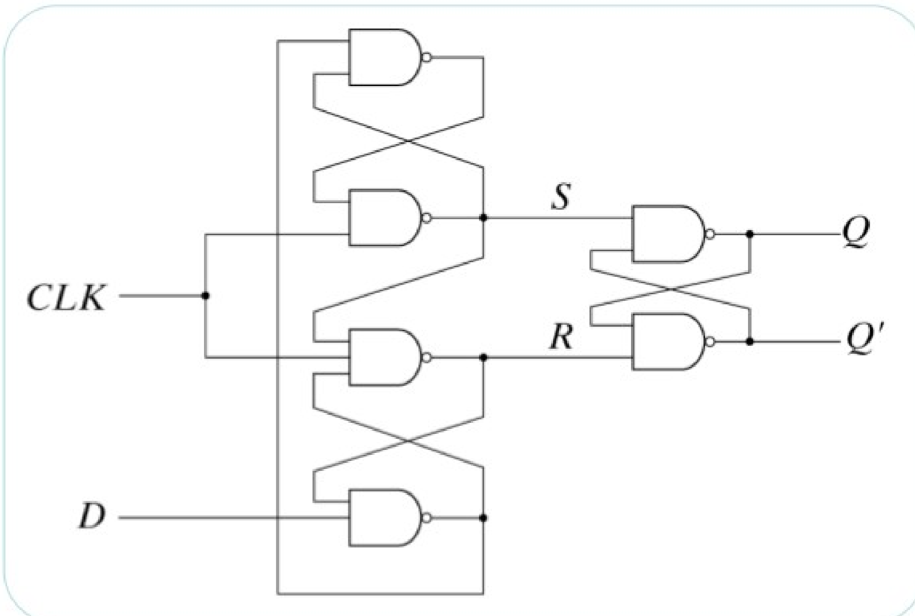


Figure 35: Kretsoppbygning til en D-Flip-Flop

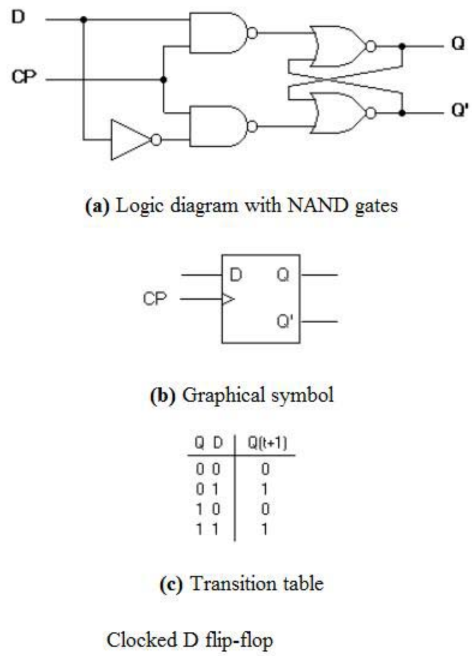
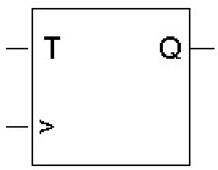


Figure 36: Kretsoppbygning til en D-Flip-Flop

T - Flip-flop Toggle - Flip-Flop

T flip-flop

- Full name: Toggle flip-flop
  - Common in counter design
  - Can construct a T from a D and mux
- When CLK↑
  - The output toggles if the input is a "1"
  - …and holds its present value if the input is a "0"



$T(t)$	$Q(t)$	$Q(t+\Delta t)$
0	0	0
0	1	1
1	0	1
1	1	0

Figure 37: T - Flip-Flop

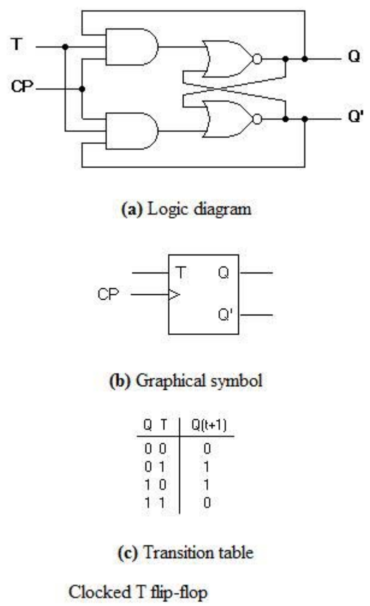


Figure 38: T - Flip-Flop

6 Tilstandsmaskiner:

Tilstandsmaskiner er en metode til å beskrive systemer med logisk og dynamisk (tidsmessig) oppførsel.

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

Brukes mye innen:

–Logiske/digitale styresystemer

–Sanntidssystemer

–Telekommunikasjon

–Kompilatorteknikk

–Digitalteknikk

Modellen av en tilstandsmaskin:

-Tilstander

er et begrep som benyttes til å beskrive systemets tilstand.

er et verdisett/attributter som beskriver systemets egenskaper

- Hendelser

endrer systemet fra en tilstand til en annen

er et begrep som benyttes om innganger/påvirkninger på systemet

kan beskrives som en plutselig og kortvarig påvirkning av systemet.

- Aksjoner

er det som kommer ut av systemet(resultatet) er en respons på en hendelse

6.1 Tilstandstabell

Sannhetstabell for tilstandsmaskiner

Q1	Q0	X2	X1	X0	D1	D0	Out
0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0
0	0	0	1	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	1	0	0	0
0	0	1	1	0	0	0	0
0	0	1	1	1	0	0	0
0	1	0	0	0	0	1	0
0	1	0	0	1	0	1	0
0	1	0	1	0	0	0	0
0	1	0	1	1	1	0	0
0	1	1	0	0	0	0	0
0	1	1	0	1	0	0	0
0	1	1	1	0	0	0	0
0	1	1	1	1	0	0	0
1	0	0	0	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	0	0	0	0
1	0	0	1	1	1	0	0
1	0	1	0	0	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	0	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	0	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	0	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	0	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	0

Figure 39: Eksempel på en tilstandstabell fra oblig 2

6.2 Tilstandsdiagram

For å visualisere oppførselen til systemer brukes gjerne tilstandsdiagramer

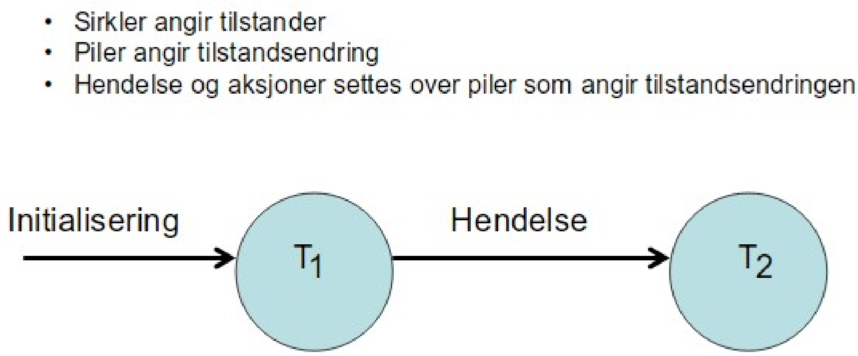


Figure 40: Tilstandsdiagram

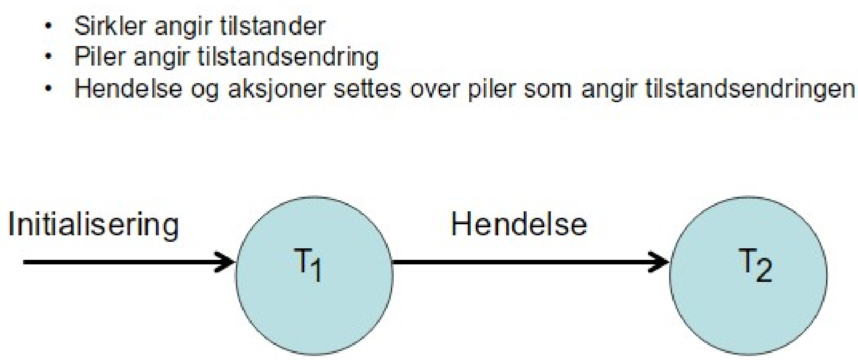


Figure 41: Tilstandsdiagram

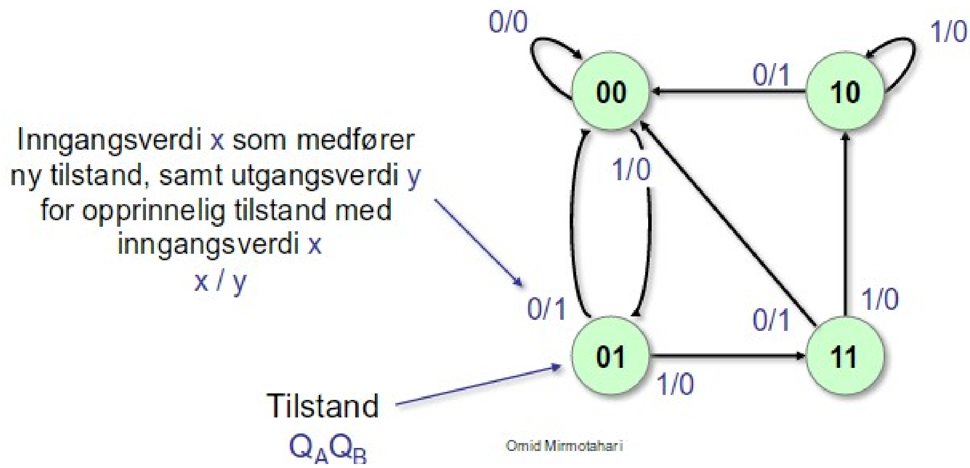


Figure 42: Eksempel på en tilstandsdiagram

6.3 Reduksjon av antall 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

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	G	0
F	1	F	1
G	0	A	0
G	1	F	1

Figure 43: Eksempel på reduksjon av tilstander

Vi ser at tilstand G er lik E

Vi kan fjerne tilstand G og erstatte hopp til G med hopp til E

Etter det ser vi at tilstand F blir lik D, da fjerner vi F

Slik kan vi redusere tilstandene.

6.4 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

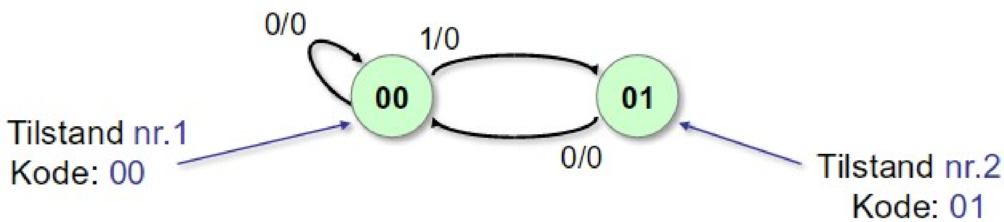


Figure 44: Tilordning av tilstandskoder

6.5 Ubrukte tilstander

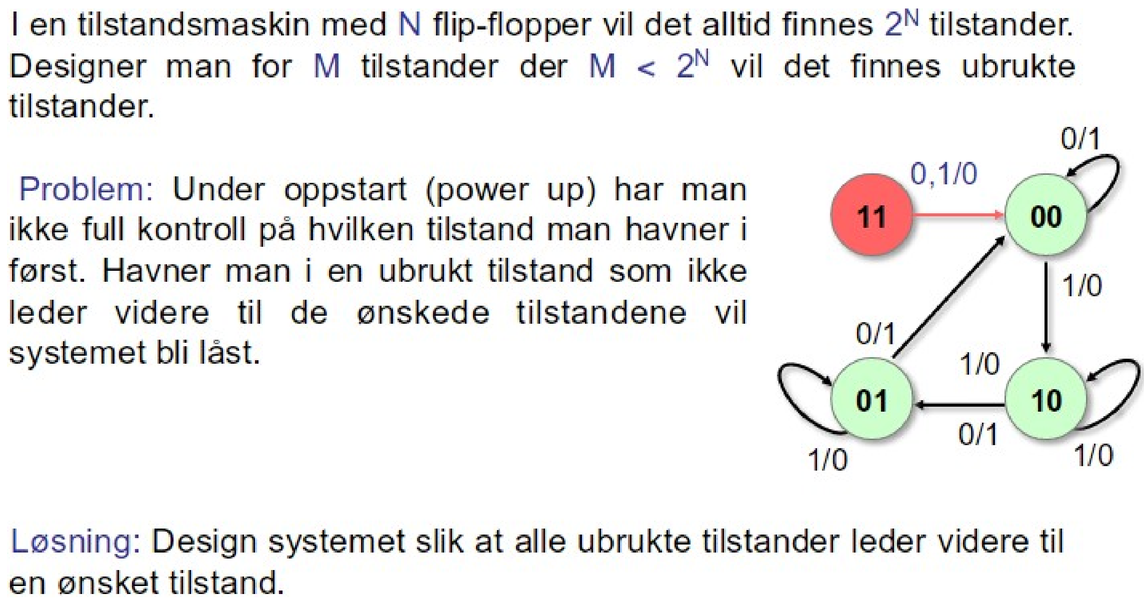


Figure 45: Ubrukte tilstander

6.6 Designprosedyre(basert på D flip-floper)

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

Figure 46: Designprosedyre

6.7 Eksempel

7 Datamaskinarkitektur:

8 CMOS: