Løsningsforslag INF1400 – H04

Oppgave 1 – Sannhetstabell og forenkling av Boolske uttrykk (vekt 18%)

I figuren til høyre er det vist en sannhetstabell med 4 variable A, B, C og D. Finn et forenklet Boolsk uttrykk for F.

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

***Løsningsforslag*:**

Fra sannhetstabell:

F = A'B'CD + A'BCD + AB'CD + ABCD

Karnaughdiagram:

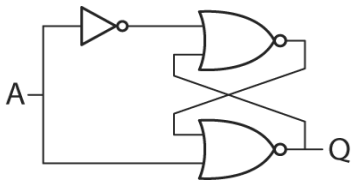
		CD			
		00	01	11	10
AB	00			1	
	01			1	
	11			1	
	10			1	

Leser ut av tabell:

F = CD

Oppgave 2 – SR latch (vekt 17%)

Beskriv oppførselen til kretsen i figuren under ved hjelp av et enkelt Boolsk uttrykk.

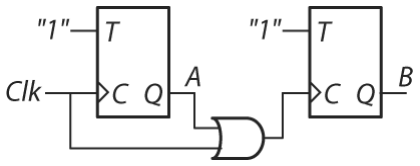


***Løsningsforslag*:**

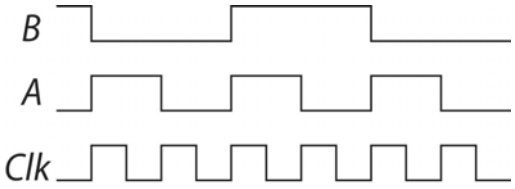
Setter A = 0 får Q = 1. Setter A = 1 får Q = 0. Andre inngangskombinasjoner finnes ikke og kretsen oppfører seg identisk med en inverter Q = A'.

Oppgave 3 – Sekvensiell krets (vekt 18%)

a) Angi hvilken sekvens kretsen i figuren til høyre gjennomløper. Gjør dette ved å skissere signalene *A* og *B* sammen med signalet *Clk* over 5 perioder slik som antydet i figuren nederst. La kretsen starte opp med *A = 0* og *B = 1*. Anta *T = 1* under hele forløpet.



***Løsningsforslag*:**



b) Er dette en binær synkronteller eller noe annet? Begrunn svaret.

***Løsningsforslag*:**

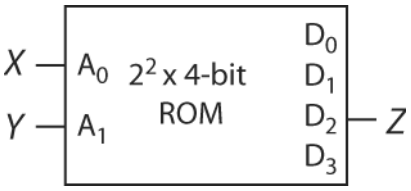
Dette er ikke en binær synkronteller. **Grunn:** Da klokkesignalet ikke går direkte til alle klokkeinngangene er det ikke en synkronteller. Men det er en asynkron (rippel) binær teller som teller fra 3 til 0 hvis man ser bort i fra oppstartstilstanden i oppgaven og regner A som minst signifikante bit og B som mest signifikante bit.

Oppgave 4 – Minne (vekt 15%)

Anta at vi har en ferdigprogrammert ROM. Dette minne kan lagre fire ord, hver på fire bit. Innholdet i minnet er vist i figuren til høyre. D_0 er minst signifikante data bit. Adressebittene kaller vi A_0 og A_1. Bit A_0 er minst signifikante adressebit og A_1 er mest signifikante adressebit. Vi ønsker å lage en enkel 2-inputs NAND port kun ved hjelp av dette minne. Skissér eller forklar hvordan dette kan gjøres.

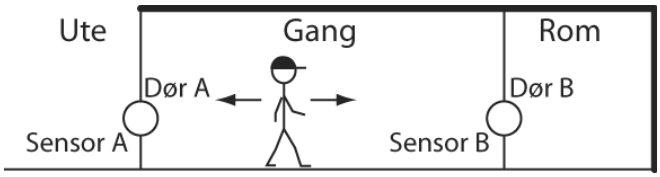
Adresse nr.	D_3	D_2	D_1	D_0
0	1	1	0	0
1	0	1	0	0
2	1	1	0	1
3	0	0	1	1

***Løsningsforslag*:**



$Z = \overline{XY}$

Oppgave 5 – Tilstandsmaskin (vekt 16%)



Vi skal overvåke et område bestående av et utendørsareal, en gang og et rom som illustrert i figuren til høyre. Det er mulig for en person å gå utenfra, gjennom dør A, inn i gjennom gangen, gjennom dør B og inn i rommet. Rommet har bare en dør. I dette området går det en vaktmann fram og tilbake om natta. Vaktmannen kan stoppe opp og snu hvor som helst, men ikke midt i en dør. Ved hver dør er det montert en sensor som registrerer om vaktmannen passerer. Disse sensorene er synkrone med et globalt klokkesignal ”*Clk*”, og gir i fra seg en ”1”er som varer en klokkeperiode hver gang vaktmannen passerer. Ellers er utgangssignalet fra sensorene ”0”. Det vil si at en sensor kan i seg selv ikke registrere hvilken vei vaktmannen passerer, bare om han passerer eller ikke.

Vi ønsker å designe en tilstandsmaskin som kun ved hjelp av disse to sensorsignalene, gjør oss i stand til å vite om vaktmannen befinner seg ute, i gangen eller i rommet. Tilstandsmaskinen skal derfor ha tre binære utgangssignaler ”U”, ”G” og ”R” som skal signalisere hvor vaktmannen befinner seg.

Vi antar at det kun eksisterer en vaktmann og ingen andre personer i systemet. Vi kan også anta at under oppstart av tilstandsmaskinen er alle flip-flop-utgangene lik ”0” og vaktmannen er utendørs. Vi antar også et ideelt system uten mulighet for feil i sensorene. Gangen er lang og sensor A kan ikke gi ut ”1” samtidig som at sensor B gir ut ”1”.

- Definer tilstandene.
- Tegn opp tilstandsdiagram.
- Tegn opp tilstandstabell.
- Tegn opp selve tilstandsmaskinen basert på D flip-flopper.

***Løsningsforslag*:**

Lar tilstand ”00” representere at vaktmannen er utendørs
Lar tilstand ”01” representere at vaktmannen er i gangen
Lar tilstand ”10” representere at vaktmannen er i rommet

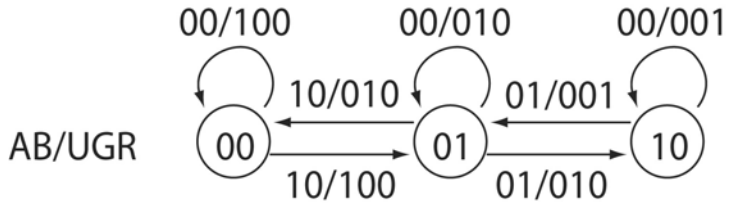
Siden det er spesifisert at maskinen garantert starter opp i tilstand ”00” ser vi bort i fra, og unngår tilstand ”11”.

Innganger: A : (”1” når vaktmann passerer sensor A, ellers ”0”)
B : (”1” når vaktmann passerer sensor B, ellers ”0”)

Utganger: R : (”1” når vaktmann er i rommet, ellers ”0”)
G : (”1” når vaktmann er i gangen, ellers ”0”)
U : (”1” når vaktmann er i ute, ellers ”0”)

Klokkesignal: Clk

Tilstandsdiagram (Det finnes flere løsninger på denne oppgaven)



Tilstandstabell (Det finnes flere løsninger på denne oppgaven)

Fra antagelsene ser man at i tilstand ”00” kan ikke B være høy (da vaktmannen er ute). Setter derfor inn x-don’t care i denne situasjon. I tilstand ”10” kan ikke A være høy (da vaktmannen er i rommet). Setter ”x” i denne situasjon. Basert på antagelsene i teksten har vi satt inn ”x” for alle situasjoner der både A og B er høye samtidig da dette ikke kan forekomme. Tilstand ”11” vil heller ikke opptre slik at ”x” er satt in her og.

Q1	Q2	A	B	D1	D2	U	G	R
0	0	0	0	0	0	1	0	0
0	0	0	1	x	x	x	x	x
0	0	1	0	0	1	1	0	0
0	0	1	1	x	x	x	x	x
0	1	0	0	0	1	0	1	0
0	1	0	1	1	0	0	1	0
0	1	1	0	0	0	0	1	0
0	1	1	1	x	x	x	x	x
1	0	0	0	1	0	0	0	1
1	0	0	1	0	1	0	0	1
1	0	1	0	x	x	x	x	x
1	0	1	1	x	x	x	x	x
1	1	0	0	x	x	x	x	x
1	1	0	1	x	x	x	x	x
1	1	1	0	x	x	x	x	x
1	1	1	1	x	x	x	x	x

Neste tilstand:

D_1	AB				
Q_1Q_2		00	01	11	10
	00		x	x	
	01		1	x	
	11	x	x	x	x
	10	1		x	x

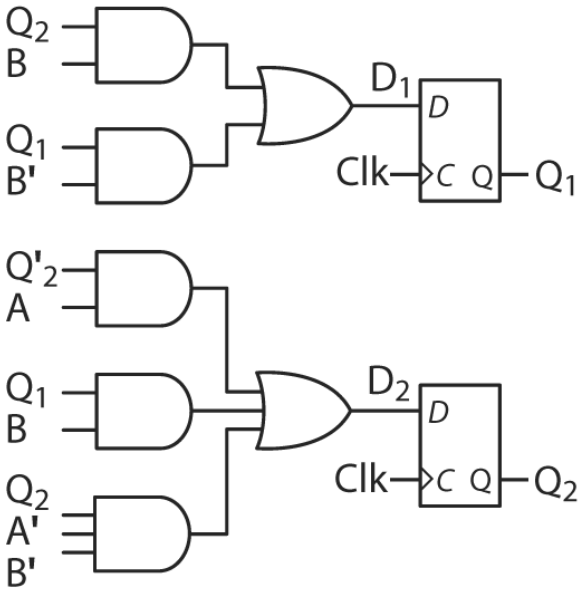
D_2	AB				
Q_1Q_2		00	01	11	10
	00		x	x	1
	01	1		x	
	11	x	x	x	x
	10		1	x	x

$D_1 = Q_2B + Q_1B'$

$D_2 = Q_2A'B' + Q_1B + Q_2'A$

Utganger:
Kun gitt av tilstand

$U = Q_1' Q_2'$
$G = Q_1' Q_2$
$R = Q_1 Q_2'$



Oppgave 6 – VHDL (vekt 16%)

VHDL koden i figuren til høyre beskriver en enkel logisk krets. Vis med et skjema (tegning) hvordan denne kretsen ser ut.

```
entity krets is
    Port ( clk : in std_logic;
           A   : in std_logic;
           B   : in std_logic;
           Y   : out std_logic);
end krets;

architecture Behavioral of krets is

signal X : std_logic;
begin

   process(clk)
   begin

      if (clk='1' and clk'event) then
         X <= A;
      end if;

   end process;

   Y <= X xor B;

end Behavioral;
```