UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i:	**INF1400 Digital teknologi**
Eksamensdag:	**3. desember 2008**
Tid for eksamen:	**14:30 – 17:30**
Oppgavesettet er på	**5 sider**
Vedlegg:	**1**
Tillatte hjelpemidler:	**Alle trykte og skriftlige samt kalkulator**

Eksamen med løsningsforslag. Løsningsforslaget er for noen deloppgaver bare en skisse, og har man laget en mer utfyllende forklaring er det bra.

Oppgave 1. Boolsk algebra, Karnaughdiagram (20%)

a) (6%) Vis skritt for skritt hvordan man kan forenkle følgende uttrykk maksimalt (bruk postulater og eventuelt teoremer fra toverdi boolsk algebra):

$$F = a \cdot (a + b)$$
$$= a.a + a.b$$
$$= a + ab$$
$$= a \,.\, (1 + b)$$
$$= a \,.\, 1$$
$$= a$$
$$F = a$$

Her var det også mulig å bruke teorem 6(b) (absorption theorem) fra boka.

b) (7%) Bruk karnaughdiagram for å forenkle følgende funksjon:
F = A'B'C'D + A'BC'D' + A'BC'D + A'BCD' + A'BCD + AB'C'D + ABCD' +ABCD

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

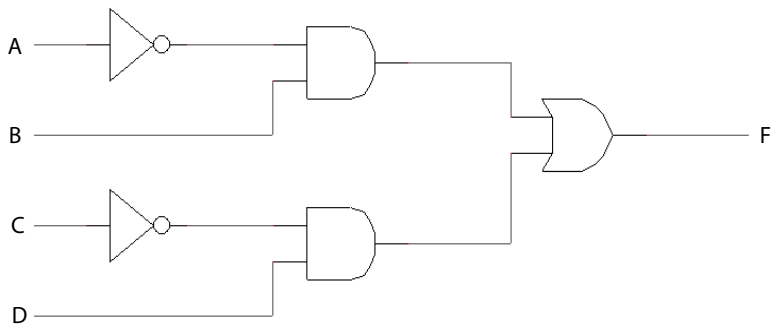
F=A'B + BC + B'C'D

c) (7%) Finn et maksimalt forenklet uttrykk for F i tabellen under. X betyr "don't care".
Tegn så opp en krets som utfører funksjonen med porter.

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

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

F=A'B + C'D

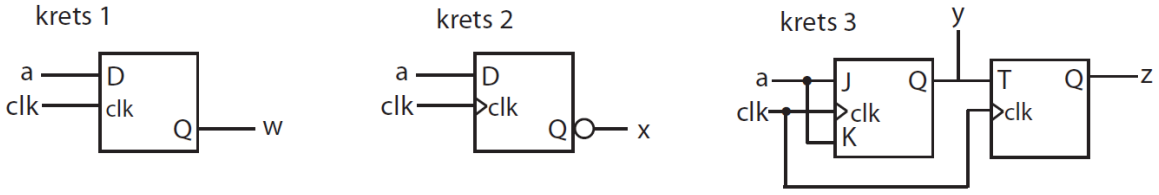


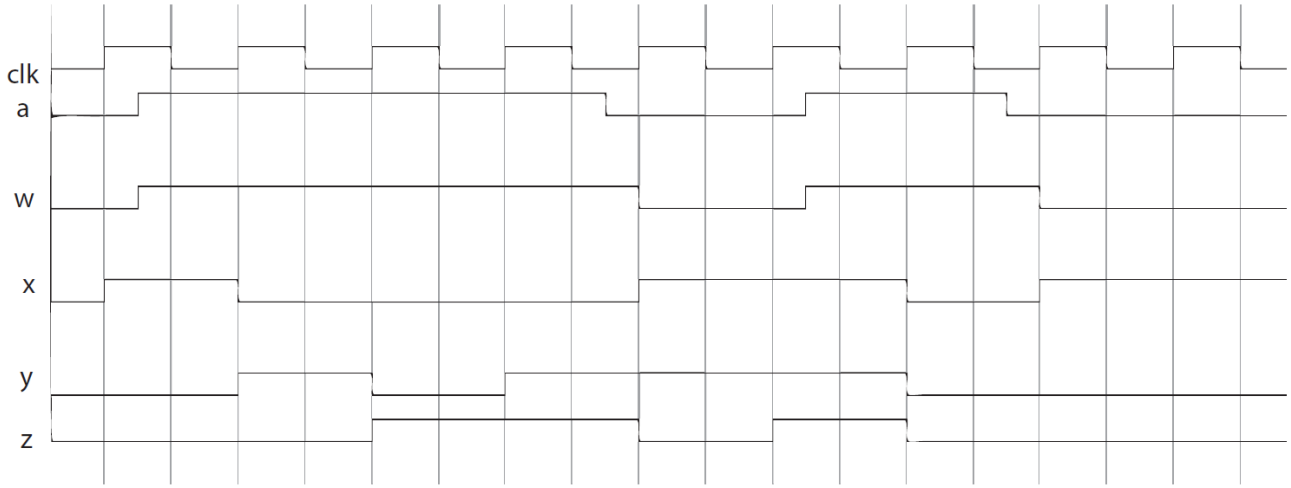
Oppgave 2. Sekvensiell logikk. (20%)

Hver av de 3 kretsene under tar inn signalene `clk` og `a`. Anta at `w`, `x`, `y` og `z` er 0 i utgangspunktet og at `clk` og `a` har verdier som vist i diagrammet under.

Svarene på oppgave 2 tegnes på vedleggsarket!

a) (6%)Tegn opp tidsforløpet til signalet `w` fra krets 1
b) (6%) Tegn opp tidsforløpet til signalet `x` fra krets 2
c) (8%) Tegn opp tidsforløpet til signalene `y` og `z` fra krets 3





Krets 1 er en D-latch (ikke noe triangel ved klokkeinngang).
Krets 2 er en D-flip-flop som toggler på stigende klokkeflanke og med utlesing av det inverterte signalet.
Krets 3 er ekvivalent med 2 T-flip-floper der a bestemmer om den første skal toggle på stigende clk og utgangen fra den første bestemmer om den andre skal toggle på stigende clk.

Oppgave 3. Tilstandsmaskin. (21%)

Vi skal designe en tilstandsmaskin som kontrollerer et lyskryss ved to kryssende veier: *nord-sør* og *øst-vest*.

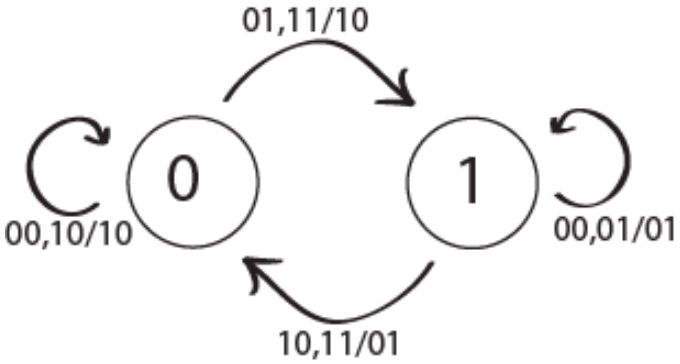
Vi setter systemets klokkesignal, `clk`, til 0,05 Hz (slik at vi ikke får lysskifte oftere enn hvert 20. sekund).

Vi ser for oss at trafikklysene skal skifte mellom rødt og grønt. Se bort fra gult lys. Når det er grønt for den ene retningen er det rødt for den andre. Vi kaller utgangene som styrer disse lysene `NSlight` (nord-sør) og `EWlight` (øst-vest). Når et av disse signalene er i høy tilstand blir det grønt lys i den tilhørende retningen. Når det er lavt blir det rødt lys.

Vi har sensorer i begge retninger som detekterer om det er biler som venter, disse inngangene kalles `NScar` og `EWcar`.

Vi vil at lyset skal skifte "grønn" retning hvis det er en bil som venter i den andre kjøreretningen. Hvis det ikke er ventende biler i den andre retningen skal lyset fortsette å vise grønt i samme retningen som den siste bilen som krysset. (Hvis det er biler som venter i begge retninger skal altså lyset skifte til den andre retningen.)

a) (7%) Definer tilstander og tilstandskoder og tegn et tilstandsdiagram for systemet
To tilstander: Grønt lys NS (kode: 0) / Grønt lys EW (kode:1)



b) (7%) Sett opp og fyll inn tilstandstabellen

Nåværende tilstand	Innganger		Neste tilstand	Utganger	
Q	Nscar	Ewcar	Q	Nslight	Ewlight
0	0	0	0	1	0
0	0	1	1	1	0
0	1	0	0	1	0
0	1	1	1	1	0
1	0	0	1	0	1
1	0	1	1	0	1
1	1	0	0	0	1
1	1	1	0	0	1

c) (7%) Forenkle om mulig funksjonsuttrykkene og implementer systemet med port(er) og D-flip-flop(er).

Neste tilstand:

D = Q'Ns'Ew + Q'NsEw + QNs'Ew' + QNs'Ew

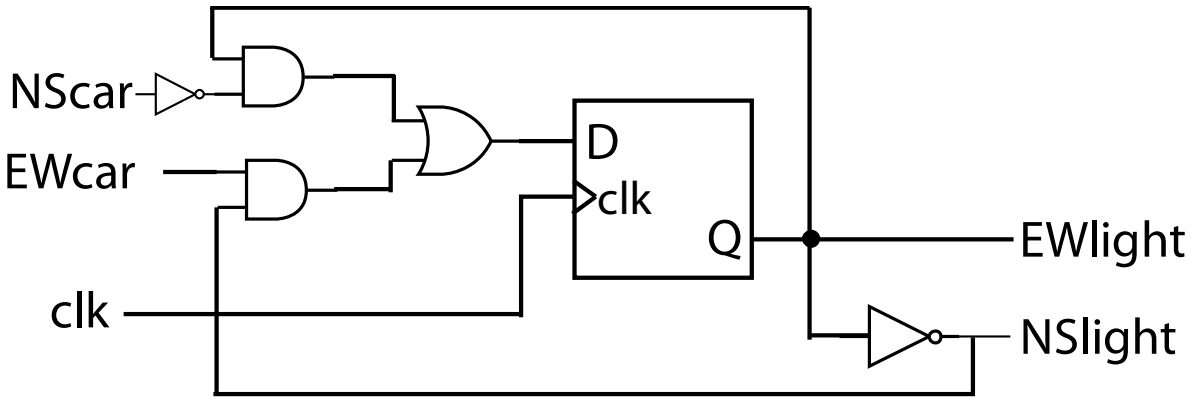
D = Q'(Ns'Ew + NsEw) + Q(Ns'Ew' + Ns'Ew)

D = Q'Ew + QNs'

(OK å bruke Karnaughdiagram også)

NSlight = Q'

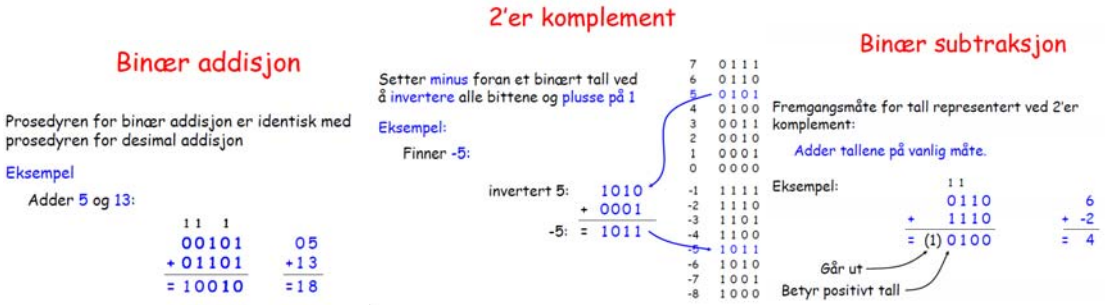
EWlight = Q



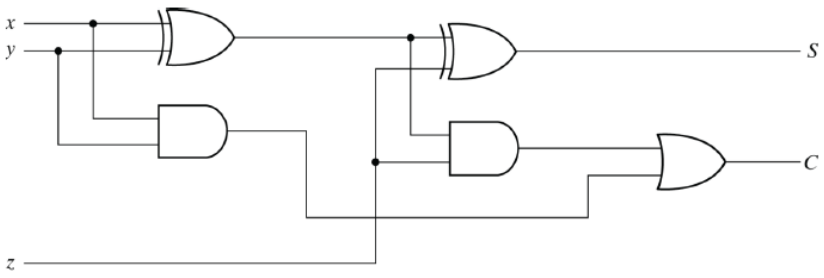
Oppgave 4. Adder, ROM (25%)

a) (4%) Gitt binærtallene A=1001 og B=0100. Hvis hvordan man kan regne ut S=A+B (binær addisjon). Vis deretter hvordan man kan regne ut S=A-B (binær subtraksjon) ved hjelp av 2'er-komplement.

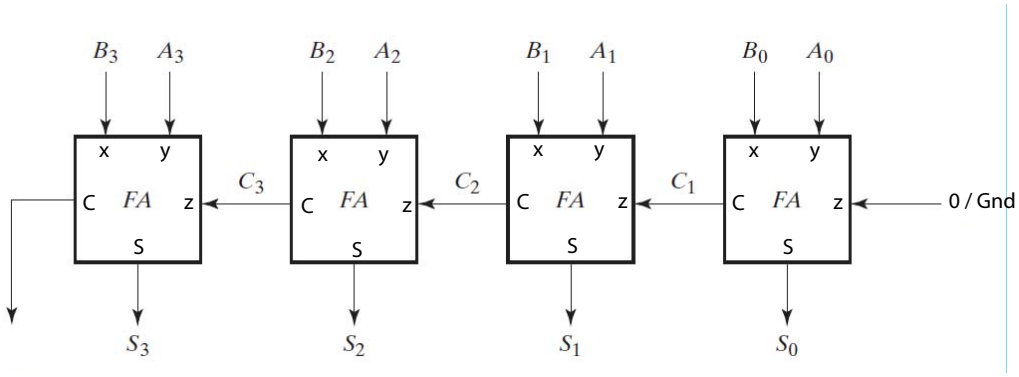
A+B = 1101 A-B = 0101 (Følger samme mønster som under, men andre tall)



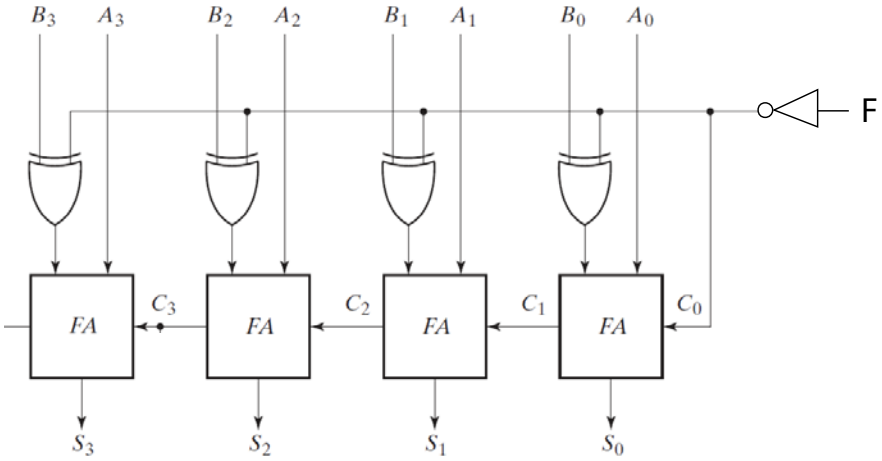
b) (4%) Vis med en figur hvordan man kan lage et adder-system som gjør operasjonen S=A+B (hvor S, A og B er 4 bits) ved å koble sammen et antall av kretsen i figuren under.
Sett navn på inngangene og utgangene. Bruk et symbol for å representere kretsen under, ikke tegn denne på portnivå.



(Kretsen er en fulladder.) Her bør man spesifisere en "oversettelse" mellom inn-/utgangene x,y,z,S,C på kretsen og signalene S,A,B i addersystemet. Eksempel:



c) (4%) Vis med en figur hvordan man kan lage et system som i tillegg til å gjøre operasjonen S=A+B også kan gjøre operasjonen S=A-B, ved å koble sammen et antall av kretsen i figuren over og noen ekstra porter (du har XOR-porter og invertere til disposisjon). Systemet skal ha en inngang F som styrer oppførselen, ved F=1 skal det være addisjon og ved F=0 subtraksjon.
Vi går her ut i fra at A>=B og du trenger ikke å ta hensyn til "overflow". Sett navn på inngangene og utgangene. Bruk kun et symbol for å representere kretsen i figuren over, ikke tegn denne på portnivå.



Også her bør man ha en korrespondanse mellom signalnavn og inn-/utganger på kretsen, se løsning på forrige oppgave. Legg merke til inverteren etter F, denne tegningen er ikke helt som på forelesningsnotatene der man opererer med 0 for addisjon og 1 for subtraksjon

d) (4%) Hva er carry lookahead? Nevn på stikkordsform fordel(er) og eventuell(e) ulempe(r) ved dette.

Carry lookahead er logikk som legges til en vanlig rippeladder for å unngå menteforplantning. Mente regnes ut "direkte" fra inngangene og er ikke avhengig av resultatet fra forrige fulladder.

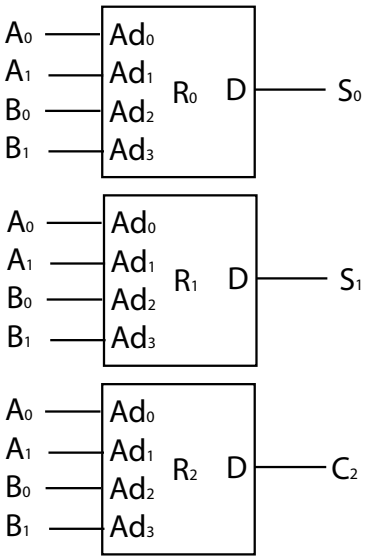
+: Gir raskere addisjon (kan gi høyere klokkefrekvens i et system)

-: Krever mer logikk som følge av mer komplisert design

e) (9%) Gitt at du kun har 16x1 (16 posisjoner med 1 bits ordbredde) ROM-er (eller LUT-er), som adresseres på vanlig måte (binært), til disposisjon:

Vis hvordan man med 3 ROM-er kan lage en 2-bits adder (A og B er 2 bits hver) med mente/carry ut (C_2). Innganger til adderen: $A_0\,A_1\,B_0\,B_1$ Utganger: $S_0\,S_1\,C_2$

Vis først med en figur hvordan ROM-ene skal kobles med innganger og utganger. Spesifiser deretter ROM-innhold. Kan en slik løsning dra nytte av carry lookahead? Begrunn svaret.



Input				Output		
B1	B0	A1	A0	C2	S1	S0
Ad3	Ad2	Ad1	Ad0	ROM2	ROM1	ROM0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Hver ROM har de same inngangene men inneholder hver sin bit av løsningen (ekvivalent med én 16x3-ROM). Denne løsningen kan ikke dra nytte av/trenger ikke carry lookahead fordi det ikke er noen menteforplantning i systemet. Merk at tabellen kan se annerledes ved andre gyldige adresseringsmetoder.

Oppgave 5. VHDL (14%)

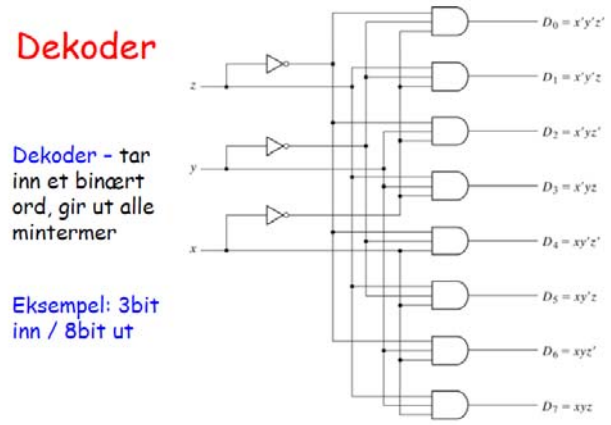
a) (8%) VHDL-koden under beskriver en kjent krets. Hva kalles den? Forklar kort hva den gjør. Vis med en figur (porter og ledninger) hvordan den kan kobles opp.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity foo is
    port(
        p : in std_logic_vector(1 downto 0);
        out0 : out std_logic;
        out1 : out std_logic;
        out2 : out std_logic;
        out3 : out std_logic
    );
end foo;

architecture behavioral of foo is
begin
    out0 <= '1' when p="00" else '0';
    out1 <= '1' when p="01" else '0';
    out2 <= '1' when p="10" else '0';
    out3 <= '1' when p="11" else '0';
end behavioral;
```

Koden beskriver en 2-4-dekoder (ved bruk av when/else utenfor process)
NB: figuren under er en 3-8-dekoder



Kretsen kan ikke kalles en demultiplexer fordi den ikke har enable inn. Demultiplexer trenger en datalinje inn i tillegg til ”select”(=p i dette tilfellet).

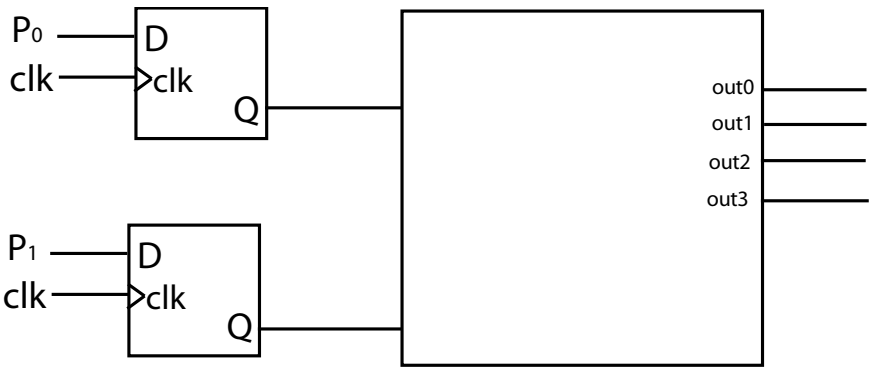
b) (6%) Vi utvider nå kretsen som vist under. Lag en figur som illustrerer utvidelsen. Bruk et symbol for kretsen i a), ikke tegn denne på portnivå. *Merk: man trenger ikke å ha a) riktig for å få poeng her.*

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity foo2 is
    port(
        clk  : in std_logic;
        p : in std_logic_vector(1 downto 0);
        out0 : out std_logic;
        out1 : out std_logic;
        out2 : out std_logic;
        out3 : out std_logic
    );
end foo2;

architecture behavioral of foo2 is
    signal e : std_logic_vector(1 downto 0);
begin
    out0 <= '1' when e="00" else '0';
    out1 <= '1' when e="01" else '0';
    out2 <= '1' when e="10" else '0';
    out3 <= '1' when e="11" else '0';

    process(clk)
    begin
        if (clk='1' and clk'event) then
            e <= p;
        end if;
    end process;
end behavioral;
```



Det har blitt lagt til et register (2 D-flip-floper) på inngangssignalet (select) til dekoderen