UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i	IN 240 — Digital systemkonstruksjon
Eksamensdag:	13. desember 1994
Tid for eksamen:	9.00 – 15.00
Oppgavesettet er på 5 sider.	
Vedlegg:	Ingen
Tillatte hjelpemidler:	Ingen

Kontroller at oppgavesettet er komplett
før du begynner å besvare spørsmålene.

1 Vekt 15%

1-a

Gitt funksjonen:

f(A,B,C,D) = ABCD + ACD' + A'BCD + BC'D + ACD + AB'C'D' + A'B'D'

Forenkle funksjonen mest mulig på formen sum av produkter ved å tegne Karnaugh-diagram.

1-b

Tegn skjema for port-implementasjon av den forenklede funksjonen.

1-c

Forenkle funksjonen mest mulig på formen produkt av summer.

1-d

Skriv funksjonen som en liste av maxtermer.

(Fortsettes side 2.)

2 Vekt 15%

Gitt funksjonen:

$$f(x,y,z) = xy + x'z$$

Hvis dette funksjonsuttrykket implementeres direkte i AND-OR-porter gir det opphav til en logisk hasard. Beskriv hva som menes med dette, hva som forårsaker hasarden i dette tilfellet, og hva dette kan medføre av konsekvenser. Vis så hvordan hasarden kan unngås.

3 Vekt 15%

3-a

Gitt funksjonen:

$$f(A,B,C,D) = AB' + AC + B'D' + BD + C'D$$

Anvend Quine-McCluskey algoritmen på denne funksjonen, og beskriv i form av tabeller hva du gjør for å komme frem til et forenklet funksjonsuttrykk.

3-b

Hva er hovedforskjellene på Quine-McCluskey-algoritmen og algoritmen i Espresso?

4 Vekt 15%

4-a

Tegn diagrammer for en 3-inputs NOR-port i NMOS og CMOS og forklar hvordan de virker ved forskjellige input-verdier. (Formler for dimensjonering av transistorer er ikke ønsket, kun en enkel forklaring på "bryter"-nivå, dvs switch-level)

4-b

Hvilken teknologi (NMOS eller CMOS) bruker mest strøm, og hvorfor?

4-c

Hvilken implementasjon vil ha kortest tidsforsinkelse i bruk, og hvorfor?

5 Vekt 15%

Denne oppgaven handler om asynkron logikk.

5-a

Forklar hva 'fundamental mode' er.

(Fortsettes side 3.)

5-b

En asynkron krets har en input x, to eksitasjonsvariable Y_1 og Y_2, sekundærvariable y_1 og y_2 og er beskrevet ved følgende ligninger:

$$Y_1 = xy_1 + xy_2'$$
$$Y_2 = y_1y_2 + xy_1$$

Tegn skjema for kretsen og sett opp transisjonstabell.

5-c

Finn ut om kretsen har kappløp, forklar eventuelt hva de går ut på og om de er kritiske. Beskriv hvordan kretsen virker.

6 Algoritmiske tilstandsmaskiner (vekt 25%)

Figuren viser en registerblokk og en adressegenerator. Registerblokken består av 16 registere. Inneholdet av registrene beskriver hvilke adresser som skal genereres. Adressegeneratoren leser innholdet av registerblokken, og ut fra dette, genererer den adresser på en 12 bits adresse-buss, *Adr*. Adressegeneratoren er rask i den forstand at den er istand til å generere en ny adresse for hver klokkpuls som den mottar på klokke-signalet, *Clk*.

Adressegeneratoren virker på følgende vis. Den må først gjennom en initialisering. Vi antar at input-signalene, *Reset* og *Start*, er lave, dvs. de har logisk verdi lik '0'. Det første som gjøres, er at adressegeneratoren resettes. Det gjøres ved at *Reset* trekkes høyt, dvs. til logisk verdi '1'. Når *Clk* så går høy, får vi 0 ut på *RegblokkAdr*-bussen. Dette er en peker (adresse) til registerblokken. Vi peker dermed til første lokasjon, lokasjon 0, i registerblokken. Ganske snart etterpå blir innholdet at denne lokasjonen i registerblokken tilgjengelig på *Data*-bussen. På *Data*-bussen står det nå følgende informasjon: bitene 0-11 inneholder en start-adresse, SA_0, bitene 12-19 inneholder en lengde, L_0, og bitene 20-23 inneholder en neste-peker, NP_0, som er en peker til neste lokasjon i registerblokken. *Reset* trekkes så lav igjen og vi er klare til å begynne.

Adressegeneratoren venter så på at *Start* skal gå høy. På den første stigende klokkeflanken etter at dette skjer, skal adressegeneratoren lese *Data*-bussen. Startadressen, SA_0, sendes ut på *Adr*-bussen. På den neste stigende klokkeflanken, skal adressen på *Adr*-bussen inkrementeres slik at adressen blir SA_0+1. Det samme gjentar seg på de neste stigende klokkeflankene, d.v.s. *Adr*-bussen inkrementeres med 1 på hver stigende klokkeflanke. Antall inkrementeringer er gitt av lengden, L_0, minus 1, d.v.s. L_0-1. La oss ta et eksempel. Anta at $L_0 = 5$. Dette betyr at vi skal ha 4 inkrementeringer og dermed følgende 5 adresser ut på *Adr*-bussen: SA_0, SA_0+1, SA_0+2, SA_0+3 og SA_0+4.

Når vi er ferdig med alle inkrementeringene, skal adressegeneratoren lese en ny lokasjon fra registerblokken. Denne lokasjonen er gitt av neste-pekeren, NP_0. De samme operasjonene, som er beskrevet overfor, gjentar seg så med de nye verdiene for startadresse, lengde og neste peker. Merk: Det at vi går fra en lokasjon til en annen i registerblokken, skal ikke medføre et opphold i adressegenereringen. Vi skal også da overholde kravet om at en ny

(Fortsettes side 4.)

adresse skal genereres på *hver* klokkeflanke.

Adressegenereringen avsluttes på følgende vis. Hver gang vi leser en ny lokasjon fra registerblokken, skal lengden, L, sjekkes. Hvis den er lik 0, betyr det at vi skal avslutte, og hoppe til tilstanden vi var i før vi startet initialiseringen. *End* skal da trekkes høy.

På alle stigende klokkeflanker skal *Reset* sjekkes. Dersom *Reset* er aktivisert høy, skal adressegeneratoren resettes. For enkelhetsskyld kan *End* være høy når adressegeneratoren er resatt.

6-a

Anta at vi har følgende verdier i registerblokken ($ betyr at tallet er hexadesimalt):

$SA_0 = \$100$
$L_0 = \$04$
$NP_0 = \$4$
$SA_4 = \$600$
$L_4 = \$03$
$NP_4 = \$7$
$L_7 = \$00$

Anta videre at *Start* og *Reset* er lave, d.v.s. har logisk verdi lik '0'. Lag et tidsdiagram av hva som skjer med kretsen når adressegeneratoren først resettes ved at *Reset* trekkes høy i en klokkeperiode, og så en klokke periode senere, startes ved at *Start* trekkes høy i en klokkeperiode. Tidsdiagrammet skal vise de logiske verdiene på følgende signaler og busser: *Clk*, *Reset*, *RegblokkAdr*, *Data*, *Start*, *Adr* og *End* i tiden fra adressegeneratoren resettes og startes til den har gjort seg ferdig med å utføre 'programmet' i registerblokken.

6-b

Lag et utvidet ASM kart av adressegeneratoren.

6-c

Lag et blokk-diagram av adressegeneratoren. Skill kontroll-del fra data-del. Kontroll-delen tegnes som en boks med input- of output-signaler. Vi er ikke interessert i innmaten av boksen her, men kun hvilke input- og output-signaler vi behøver. Det samme gjelder data-delen, som utfører register operasjonene i ASM kartet. Her vil det riktignok være behov for flere register-deler, men hver del skal kun tegnes som en boks med input- og output-signaler. Gi passende navn på kontroll-signalene som kontroll-delen må generere.

6-d

Tegn et tilstandsdiagram av kontroll-delen.

(Fortsettes side 5.)

6-e

Implementer kontroll-delen med D-flip-floper; først ved å bruke så få flip-floper som mulig, så ved å bruke 'one-hot code' metoden (en flip-flop pr tilstand).