Norges teknisk–naturvitenskapelige universitet
Institutt for elektronikk og telekomunikasjon

TFE4101 Krets- og Digitalteknikk Høst 2015

Løsningsforslag — Øving 6

1 Binær multiplikasjon og divisjon

a) $A = 10101_{(2)}$ (2's komplement.), $B = 10011_{(2)}$ (2's komplement.)
Se også eksempel 2.4 i Gajski. Dette er helt vanlig multiplikasjon, med to unntak:

- De skiftede multiplikandene summeres *etterhvert*, istedenfor å summere dem tilslutt.
- Man tar toer-komplementet av den siste skiftede multiplikanden. Dette gjøres helt konsekvent. Årsaken til dette er at fortegnsbittet har negativ posisjonsvekt.

Disse to punktene gjør det lett å realisere multiplikasjon i maskinvare.

`Multiplikand`	* Multiplikator
`10101`	* 10011
`000000`	Partielt produkt (med fortegns-forlengelse) før vi starter.
`+110101`	Multiplikand (med fortegns-forlengelse).
`1110101`	Første partielle produkt (med fortegns-forlengelse).
`+110101`	Skiftet (1 posisjon) multiplikand med (fortegns-forlengelse).
`11011111`	Andre partielle produkt (med fortegns-forlengelse).
`+000000`	Skiftet (2 posisjoner) multiplikand (med fortegns-forlengelse).
`111011111`	Tredje partielle produkt (med fortegns-forlengelse).
`+000000`	Skiftet (3 posisjoner) multiplikand (med fortegns-forlengelse).
`1111011111`	Fjerde partielle produkt (med fortegns-forlengelse).
`+001011`	2's kompl av skiftet (4 posisjoner) multiplikand (mff).
`0010001111`	$= 143_{(10)}$ (Ignorer mente ut).

mff = «med fortegns-forlengelse».

> OBS: For å forstå oppgave 5a fullt ut, bør man først sette seg inn i:
>
> - Fortegnsforlengelse.
> - Hvorfor man tar toer-komplementet av den siste skiftede multiplikanden.

b) $C = 10100000_{(2)}$, $D = 111_{(2)}$
Dette er helt vanlig divisjon. Se også kapittel 2.8 i Gajski. Merk at tallene her er uten fortegn (positive).

	10100000	: 111 =	10110
-	111	Skiftet divisor går opp	(MSB = 1)
	110000	Ny dividend	
-	000	Skiftet divisor går ikke opp	(0)
	110000	Ny dividend	
-	111	Skiftet divisor går opp	(1)
	10100	Ny dividend	
-	111	Skiftet divisor går opp	(1)
	110	Ny dividend	
-	000	Skiftet divisor går ikke opp	(LSB = 0)
	110	Rest = $6_{(10)}$	

Det vil si at $10100000_{(2)} : 111_{(2)} = 10110_{(2)} + 110_{(2)}/111_{(2)} = 22_{(10)} + 6/7_{(10)}$

2 Design og analyse av kombinatoriske kretser

a) Utvikling av logisk funksjon

Hver linje i spesifikasjonen gir et produktledd som deretter må summeres sammen. Dette gir oss:

$$Lys = A\bar{B}C + AB + B\bar{C}$$

Denne kan vi forenkle ved manipulasjon. Vi starter med å utvide de to siste leddene slik at vi får funksjonen på kanonisk form:

$$Lys = A\bar{B}C + AB\bar{C} + ABC + \bar{A}B\bar{C} + AB\bar{C}$$

Andre og fjerde ledd er like, så en av disse kan vi fjerne.

$$Lys = A\bar{B}C + ABC + \bar{A}B\bar{C} + AB\bar{C}$$

Nå kan vi trekke ut AC fra de to første leddene og B$\bar{\text{C}}$ fra de to siste:

$$Lys = AC(\bar{B} + B) + (\bar{A} + A)B\bar{C}$$

Parentesene kan nå fjernes og vi står igjen med:

$$Lys = AC + B\bar{C}$$

Dette er funksjonen til en multiplekser (selector) med C som valgsignal (se figur 5.13 i Gajski).

b) Adderer

I en CLA-adderer benyttes ekstra porter til å forhåndsberegne menteforplantningen. Den har derfor høyere kompleksitet enn en ripple-carry adderer. Den er imidlertid vesentlig raskere.

c) Dekoder

I en 2-4 dekoder er utgang C0 høy når inngangene tar verdien 00 og lav ellers. Tilsvarende er C1 høy når inngangene tar verdien 01 og lav ellers, osv. For en komplett løsning må vi sette opp sannhetstabell for hver enkelt funksjon i hver

enkelt krets, og se hvilken krets som tilsvarer en 2-4 dekoder. For løsning b1 ser vi imidlertid at C3 ikke er høy når inngangene tar verdien 11, følgelig kan dette ikke være en 2-4 dekoder. Av de to gjenstående er det bare realiseringen av C1 som er forskjellig. Denne skal være høy når A1=0 og A0=1. Dette er tilfelle for b3 men ikke for b2.

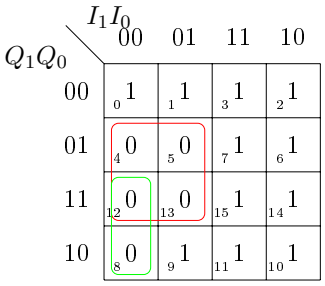
d) PLA

PLA-realisering av følgende likninger:

$$U_1 = \bar{Q}_1\bar{Q}_0 + Q_0I_1 + Q_1\bar{Q}_0I_0 + Q_1\bar{Q}_0I_1$$
$$U_0 = \bar{Q}_1\bar{Q}_0\bar{I}_1 + \bar{Q}_1Q_0I_1 + \bar{Q}_1\bar{Q}_0I_0 + Q_1\bar{Q}_0\bar{I}_1\bar{I}_0$$

Vi lager Karnaughdiagram for de to likningene. Ettersom målet er å benytte færrest mulig OG- termer, må vi her vurdere hvilken forenkling som gir dette. Vanligvis grupperer vi 1-mintermer (enere i diagrammet), men siden vi har tilgjengelig invertering på utgangen, så kan vi også gruppere 0-mintermer (nullere i diagrammet). Invertering av en utgangsfunksjon oppnås ved å koble den andre inngangen på tilhørende XOR-port til 1 (se sannhetstabell for XOR-port).

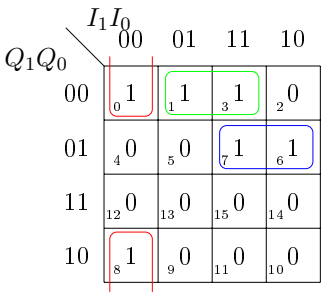
U_1:



Her ser vi at implementering med 0-mintermer (makstermer) gir to OG-termer. Den alternative løsningen med 1-mintermer ville gitt tre OG-termer. Vi får altså:

$$U_1 = \overline{(Q_0\bar{I}_1 + Q_1\bar{I}_1\bar{I}_0)}$$

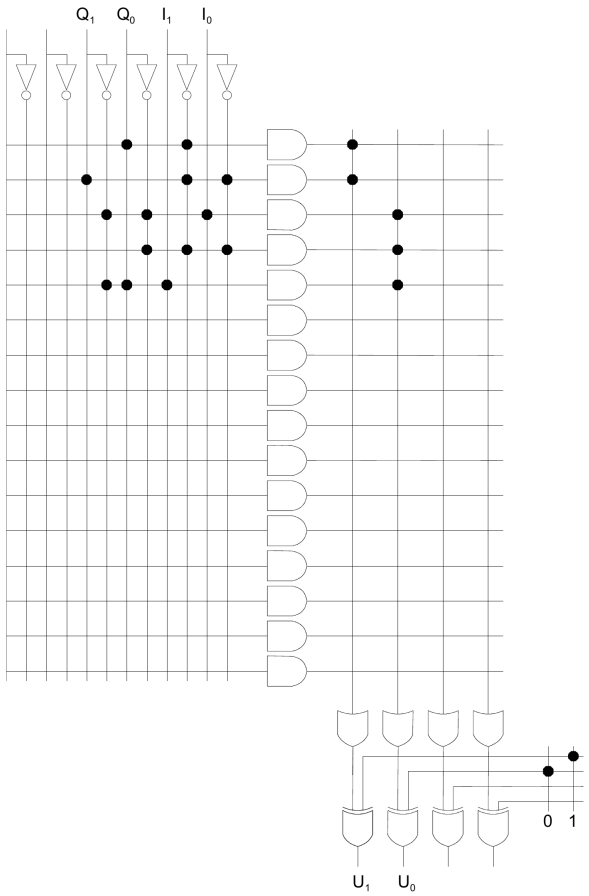
U_0:



Her ser vi at implementering med 1-mintermer gir tre OG-termer. Den alternative løsningen med 0-mintermer (makstermer) ville gitt fire OG-termer. Vi får altså:

$$U_0 = \bar{Q}_0\bar{I}_1\bar{I}_0 + \bar{Q}_1\bar{Q}_0\bar{I}_0 + \bar{Q}_1Q_0I_1$$

I Figur 1 er de ulike OG-termene merket av i den øvre venstre delen av figuren. ELLER-termene som inngår i den enkelte funksjon er avmerket i den øvre høyre delen av figuren. Legg også merke til at vi inverterer U1 men ikke U0 ved å koble henholdsvis en ener og en nuller inn på XOR- portene nederst til høyre i figuren.



Figur 1: PLA med realisering av kombinatorikk

e) Kombinatorisk kretsdesign på blokknivå

Vi kan starte designprosessen med å dele oppgaven i mindre deloppgaver. Kretsen vår skal kunne

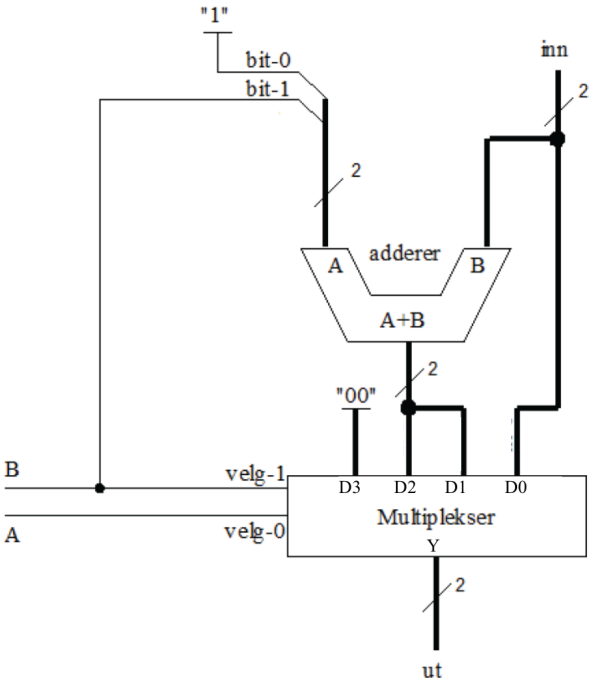
- Trekke 1 fra inn ved hjelp av en 2s-komplement addisjonskrets.
- Legge 1 til inn ved hjelp av en 2s-komplement addisjonskrets.
- Velge en av fire mulige svar ved hjelp av en multiplekser

Det å trekke fra 1 er det samme som å legge til -1. Siden vi opererer med 2 bit brede 2s-komplement tall så svarer det til å legge til 11. Det vil altså si at vi enten skal legge til 01 (inn + 1) eller legge til 11 (inn – 1). Det er med andre ord bare det mest signifikante av de to bittene som er forskjellige i de to addisjonene. Samtidig har vi at styresignalet B=0 når vi skal legge til '01' og B=1 når vi skal legge til '11'. Følgelig kan vi la B være verdien av det mest signifikante bittet som skal legges til, og koble det minst signifikante bittet direkte til 1.

Videre kan vi la styresignalene A og B styre multiplekseren slik at den slippe signalet inn rett igjennom, slipper resultatet av addisjonen igjennom, eller slipper verdien '00' igjennom. Resultatet av addisjonen må vi rute inn på to av

inngangene på multiplekseren siden det er to kombinasjoner av styresignalene som benytter seg av dette resultatet.

Figur 2 viser en mulig kretsløsning på blokknivå.



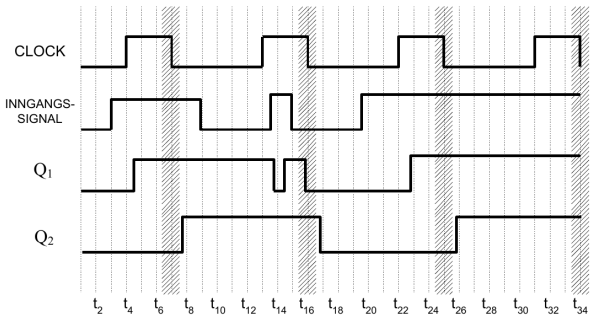
Figur 2: Kombinatorisk krets på blokknivå

3 Flanke- og nivåstyrte vipper

a) Vi ser fra figuren at pulstog Q_1 skifter fra høy til lav, og tilbake til høy, mens klokken er høy. Dette pulstoget tilhører derfor en D-latch.

Pulstog Q_2 skifter nivå kun i tilknytning til den fallende klokke-flanken, og hører derfor til en D-vippe (flankestyrt).

Plasseringen av invertereren gjør at slave-låsen leder når CLOCK er lav. Følgelig vil utgangen Q endre seg i det CLOCK skifter fra høy til lav, og vi får en vippe som er styrt av den fallende (eller negative) flanken.



Figur 3: De skraverte områdene er tidsintervallet $T_{oppsett}$-T_{holde}

Q	$Q_{(next)}$	D
0	0	0
0	1	1
1	0	0
1	1	1

Tabell 1: Eksitasjonstabell for D-vippe

Q	$Q_{(next)}$	T
0	0	0
0	1	1
1	0	1
1	1	0

Tabell 2: Eksitasjonstabell for T-vippe

b) $T_{oppsett}$-T_{holde} er et tidsintervall som omslutter den aktive flanken til klokken. Hvis man endrer inngangssignalet i dette tidsrommet har man ikke kontroll på hvordan det vil innvirke på utgangsverdien til kretsen. Dette er vist i Figur 3.

c) Tabell 1: Vi ser at nestetilstanden til D-vippen er lik inngangsignalet til vippen.

Tabell 2: Hvis inngangssignalet T er 1, vil vippen skifte tilstand ved neste aktive klokkeflanke (toggle-vippe).

Tabell 3: Dersom vippen er i tilstanden 0, er det S som bestemmer om vippen skal settes til 1 eller forbli 0. S kalles derfor set-inngangen. Dersom vippen er i tilstand 1, er det R som bestemmer om vippen skal resettes (settes til 0). R kalles derfor reset-inngangen til vippen. Merk at det ikke finnes noen rad i tabellen der både S og R er 1, siden dette er ulovlig for SR-vippen.

Tabell 4 Dersom vippen er i tilstanden 0, er det J som bestemmer om vippen skal settes til 1 eller forbli 0. J kalles derfor set-inngangen. Dersom vippen er i tilstand 1, er det K som bestemmer om vippen skal resettes (settes til 0). K kalles derfor reset-inngangen til vippen. Merk at dersom både J og K er 1, så vil vippen skifte tilstand ved neste aktive klokkeflanke (toggle).

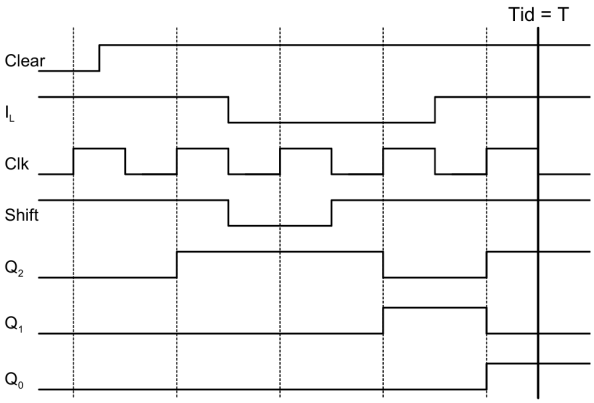
d) Ved tiden T er Q2=1, Q1=0 og Q0=1. Skiftregisteret svarer til figur 7.4 i Gajski. Se Figur 4 for oppførsel.

Q	$Q_{(next)}$	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Tabell 3: Eksitasjonstabell for SR-vippe

Q	$Q_{(next)}$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Tabell 4: Eksitasjonstabell for JK-vippe



Figur 4: Skiftregister med inngangsverdier