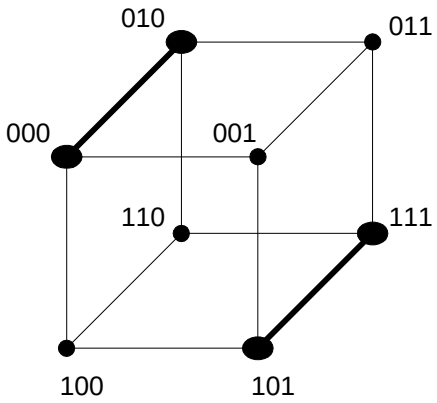
Løsningsforslag øving 2
Digitalteknikk og datamaskiner – TFE4105 – H12

Oppgave 1: Kube og Karnaughdiagram

I) a) Vi summerer mintermene og får:

$$T = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}B\overline{C} + A\overline{B}C + ABC$$



Figur 2.1: Tredimensjonal kube

Vi kaller våre mintermer (hjørnene: 000, 010, 101, og 111) for 0-kuber, og sidekantene 0-0 og 1-1 for 1-kuber ('-' betyr "don't care", eller vilkårlig 1 eller 0). Vi sier også at 0-0 dekker 000 og 010, og at 1-1 dekker 101 og 111. Generelt: en kube som har '-' i n posisjoner er en n-kube.

Mintermene er merket med store sirkler på kuben over. Vi ser at mintermene danner nabohjørner på kuben. Det vil si at det bare er ett bit som forandres mellom dem. Dermed kan de grupperes, og det boolske vil bli forenklet. Gruppering utføres ved å beholde de bit som er felles for nabo-mintermer. For mintermene på øverste venstre kant av kuben ser vi at MSB og LSB er felles, og at begge er 0. Det vil si at vi kan erstatte disse mintermene med primleddet $\overline{A}\,\overline{C}$, som er merket med en tykk strek mellom mintermene. Dette kan også skrives som 0-0. Tilsvarende for mintermene på nederste høyre kant, hvor vi finner vi at disse kan erstattes med primleddet AC, eller 1-1.

Ved å kombinere disse to leddene, får vi det forenklede uttrykket:

$$\underline{\underline{T = \overline{A}\,\overline{C} + AC = \overline{A \oplus C}}}$$

b) Karnaughdiagram for funksjonen T:

A \ BC	00	01	11	10
0	1 (0)	0 (1)	0 (3)	1 (2)
1	0 (4)	1 (5)	1 (7)	0 (6)

$\overline{A}\,\overline{C}$

AC

Ved å gruppere mintermene 000 og 010, finner vi at fellesfaktoren er $\bar{A}\,\bar{C}$ (eller 0-0). Ved å gruppere mintermene 111 og 101 finner vi at fellesfaktoren er AC (eller 1-1). Siden Karnaughdiagrammet gir resultatet på SOP-form skriver vi:

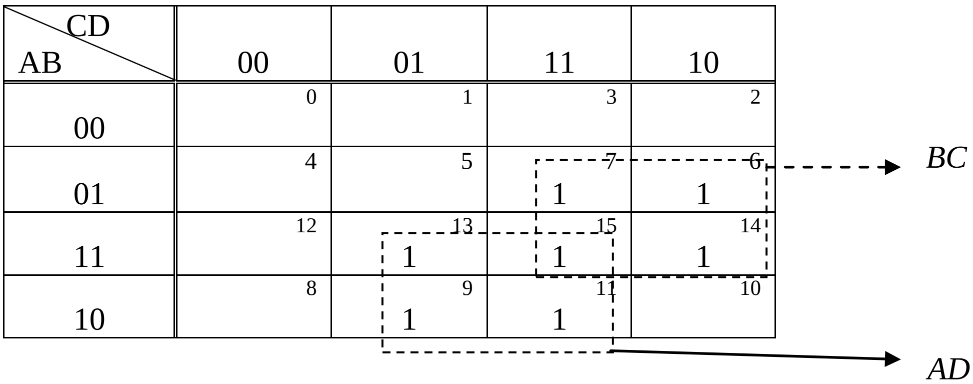
$$\underline{\underline{T = \bar{A}\,\bar{C} + AC}}$$

c) Begge metoder baseres på at det bare er ett bit som forandres mellom naboceller. Ved hjelp av fig 4.3 i Gajski, ser vi at Karnaughdiagrammet bare er en todimensjonal representasjon av kuben fra oppgave I.

d)

T_1: Her brukes samme fremgangsmåte som i oppgave II, men her kan man se etter grupper på 2, 4, 8, og 16 mintermer.

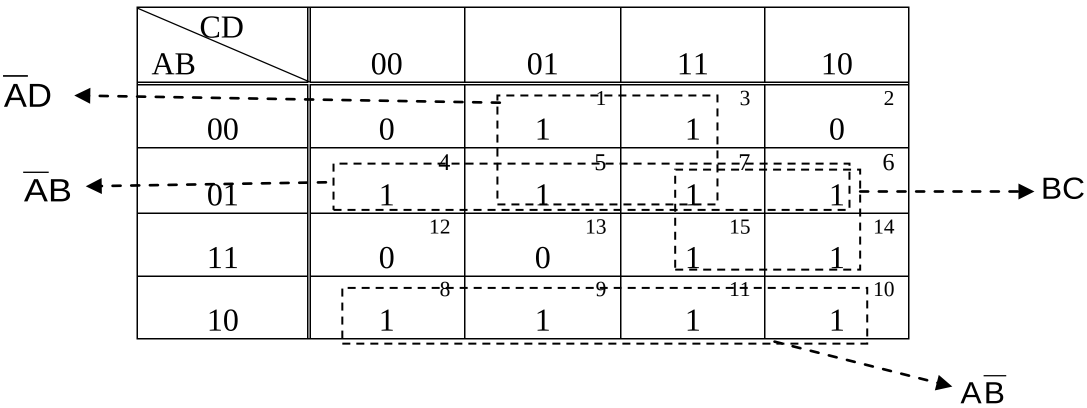
Karnaughdiagram for funksjonen T_1:



$$\underline{\underline{T_1 = AD + BC}}$$

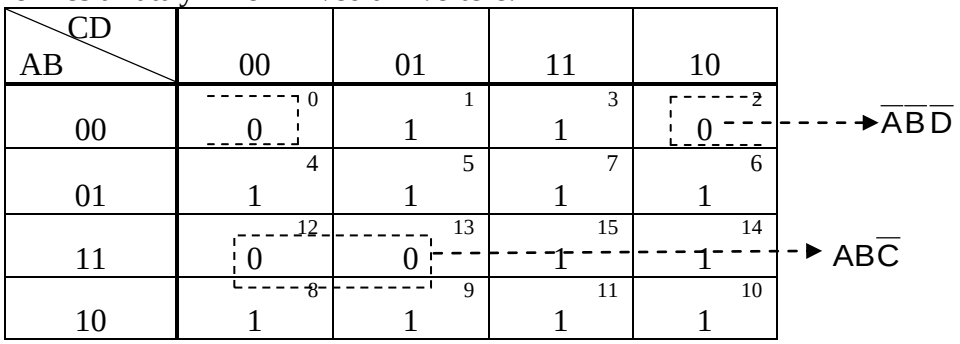
T_2: Når det gjelder funksjonen T_2, kan vi gå frem på to forskjellige måter:

1. Ved å utvikle etter 1-ere får vi følgende Karnaughdiagram (merk at vi kunne valgt andre primimplikanter enn de vi har tatt med her):



$$\underline{\underline{T_2 = \bar{A}D + \bar{A}B + BC + A\bar{B}}}$$

2. Ved å utvikle etter 0-ere i Karnaughdiagrammet, får vi et forenklet uttrykk for $\overline{T}$. Dette omformes til uttrykk for T ved å invertere.



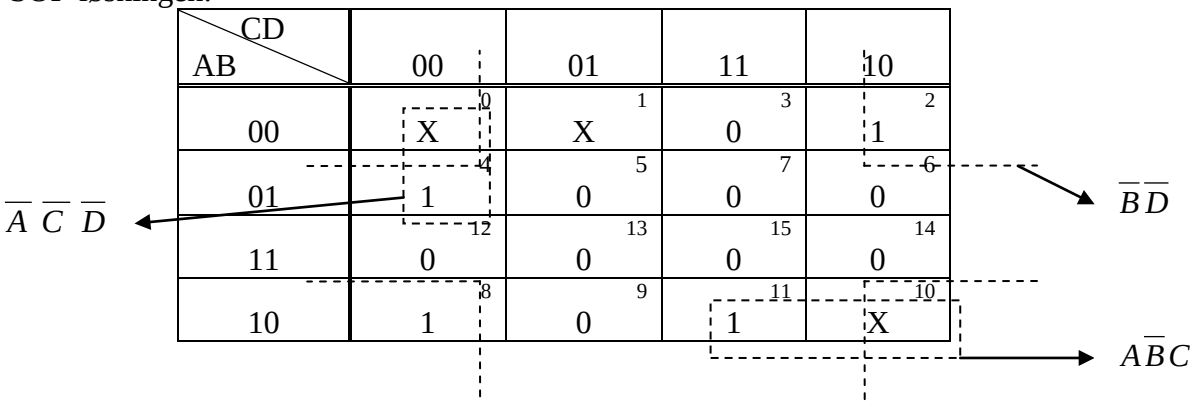
$$\overline{T_2} = \overline{A}\,\overline{B}\,\overline{D} + AB\overline{C} \Rightarrow T_2 = \overline{\overline{A}\,\overline{B}\,\overline{D} + AB\overline{C}} = (A + B + D)(\overline{A} + \overline{B} + C)$$

I siste overgang brukes DeMorgans teorem. Merk at vi får en løsning med færre literaler og dermed færre porter. Merk at vi også kunne ha funnet T_2 direkte ved å utvikle 0-erne med hensyn på makstermer i steden for mintermer.

T3: $T_3(A,B,C,D) = \sum(2,4,8,11) + \sum_{\phi}(0,1,10)$

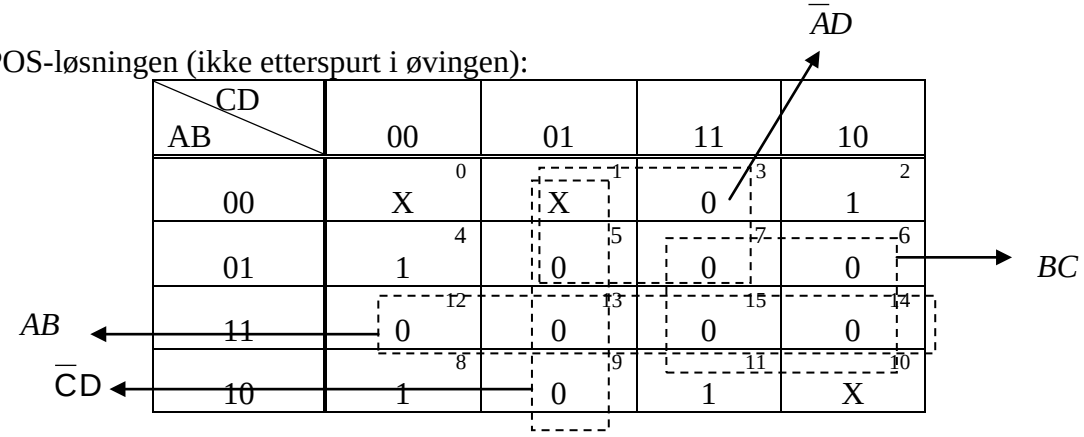
$\sum_{\phi}$ er don't care settet. Denne funksjonen kan altså oppfattes som sammensatt av to deler. Den første delen inneholder en liste over hvilke mintermer som funksjonen ***skal*** inneholde, mens den andre delen er en liste over de mintermer som funksjonen ***kan*** inneholde.

SOP-løsningen:



$$T_3 = \overline{B}\,\overline{D} + \overline{A}\,\overline{C}\,\overline{D} + A\overline{B}C$$

POS-løsningen (ikke etterspurt i øvingen):



Her utvikler vi etter 0-ere for å få løsningen på POS-form.

$$\overline{T_3} = AB + \overline{C}D + BC + \overline{A}D$$

Ved invertering av funksjonen får vi:

$$T_3 = \left(\overline{AB}\right)\left(\overline{\overline{C}D}\right)\left(\overline{BC}\right)\left(\overline{\overline{A}D}\right) = \underline{\underline{(\overline{A}+\overline{B})(C+\overline{D})(\overline{B}+\overline{C})(A+\overline{D})}}$$

Her kunne vi også funnet POS-formen direkte ved å utvikle 0-erne med hensyn på makstermer.

Oppgave 2: Tabellmetoden

a)

Gruppe	Minterm	X	Y	Z	W	Dekket
G_0	(0)	0	0	0	0	Ja
G_1	(1)	0	0	0	1	Ja
	(2)	0	0	1	0	Ja
G_2	(9)	1	0	0	1	Ja
	(10)	1	0	1	0	Ja
G_3	(7)	0	1	1	1	Ja
	(11)	1	0	1	1	Ja
	(14)	1	1	1	0	Ja
G_4	(15)	1	1	1	1	Ja
G_0	(0,1)	0	0	0	-	
	(0,2)	0	0	-	0	
G_1	(1,9)	-	0	0	1	
	(2,10)	-	0	1	0	
G_2	(9,11)	1	0	-	1	
	(10,11)	1	0	1	-	Ja
	(10,14)	1	-	1	0	Ja
G_3	(7,15)	-	1	1	1	
	(11,15)	1	-	1	1	Ja
	(14,15)	1	1	1	-	Ja
G_2	(10,11,14,15)	1	-	1	-	Nei

Tabell 2.1: Søk etter primledd for funksjonen F

I tabell 2.1 har vi funnet alle primleddene til funksjonen F (produktleddene som ikke har Ja i kolonnen for Dekket). Disse kalles $P_1 \ldots P_7$ i tabell 2.2, og vi vil her finne en irredundant dekning for funksjonen F basert på et subsett av primleddene slik det er vist i tabell 2.2.

Essensielle primledd er primledd som må være med for å dekke de distingverte mintermene $\mathbf{m_7}$ og $\mathbf{m_{14}}$, dvs. de mintermer som ikke er dekket av andre primledd. De essensielle primleddene dekker også mintermene $\mathbf{m_{10}}$, $\mathbf{m_{11}}$ og $\mathbf{m_{15}}$. For å få komplett dekning må vi også ta med ikke-essensielle primpledd. $\mathbf{P_1}$, $\mathbf{P_2}$, og $\mathbf{P_3}$ har alle to udekkede mintermer, mens $\mathbf{P_4}$ og $\mathbf{P_5}$ bare dekker en udekket minterm. Vi starter derfor med en av de første. Hvis vi velger $\mathbf{P_3}$ får vi dekket $\mathbf{m_1}$ og $\mathbf{m_9}$. $\mathbf{P_1}$ har nå bare en udekket minterm og vi velger derfor $\mathbf{P_2}$ som dekker de resterende $\mathbf{m_0}$ og $\mathbf{m_2}$.

			0	1	2	7	9	10	11	14	15	E.P.
P_1	0,1	X' Y' Z'	x	x								
P_2	0,2	X' Y' W'	x		x							
P_3	1,9	Y' Z' W		x			x					
P_4	2,10	Y' Z W'			x			x				
P_5	9,11	X Y' W					x		x			
P_6	7,15	Y Z W				⊗					x	√
P_7	10,11,14,15	X Z						x	x	⊗	x	√
	Distingverte mintermer					7				14		
	Dekket av essensielle primledd					7		10	11	14	15	
	Dekket av P_3			1			9					
	Dekket av P_2		0		2							

Tabell 2.2: Valg av irredundant dekning for funksjonen F

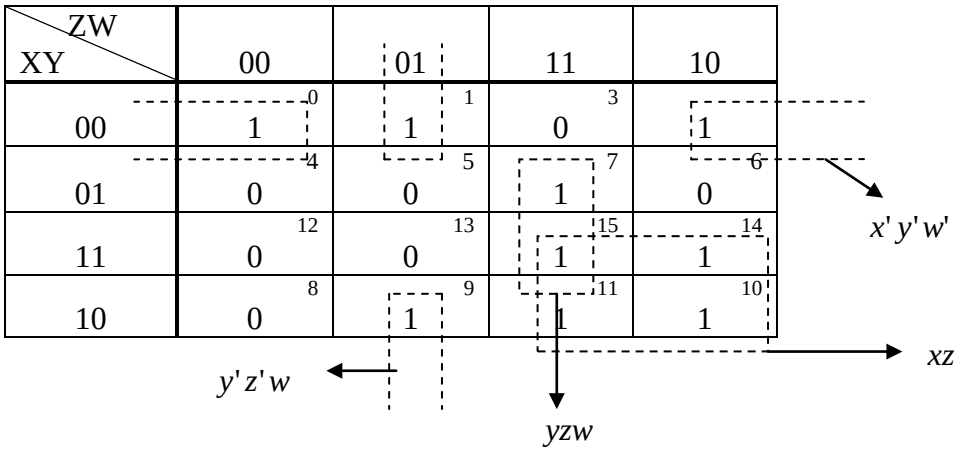
b) Primleddene er gitt av **P_1…P_7** i tabell 2.2, nemlig:

X' Y' Z' , X' Y' W' , Y' Z' W , Y' Z W' , X Y' W , Y Z W og **X Z**

c) De essensielle primleddene er: **Y Z W** og **X Z**

d) F = Y Z W + X Z + Y' Z' W + X' Y' W'

NB: andre irredundante løsninger er mulig her, hvis man istedenfor **P_3**, velger **P_1** først. Denne løsningen ville imidlertid krevd fem primledd. Under er Karnaughdiagrammet tegnet opp, som viser at vi har funnet en irredundant dekning av primledd:



Oppgave 3: Teknologi-mapping

a) Siden funksjonen T er på SOP-form, kan vi lage kretsen basert på NOR-porter hvis vi tar utgangspunkt i den inverterte til T. Siste nivå i funksjonen $\overline{T}$ kan da realiseres som en 3-inngangs NOR-port (som kan splittes opp i to 2-inngangs porter):

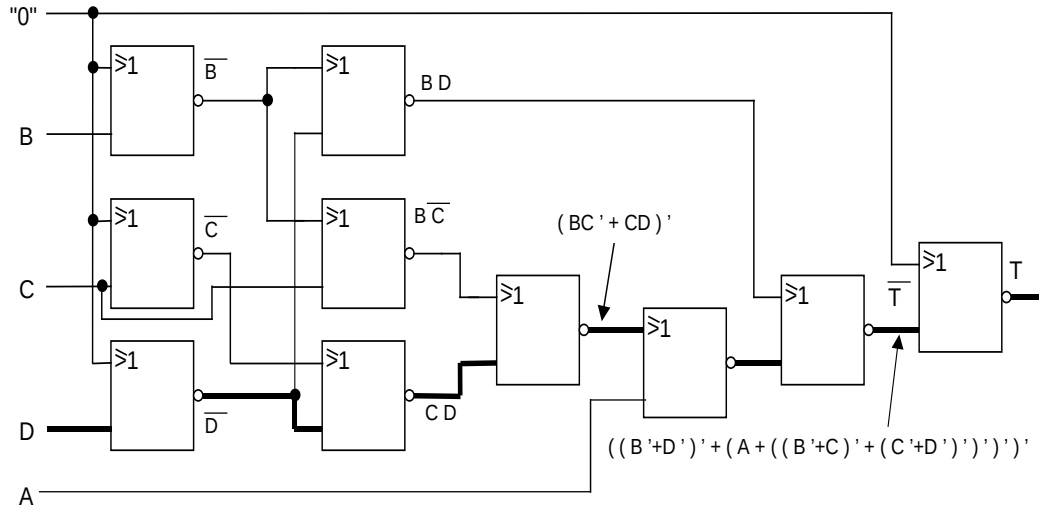
$$\overline{T} = \overline{BD + \overline{A}B\overline{C} + \overline{A}CD}$$

Deretter fortsetter vi å jobbe med underliggende nivåer som er produkter av inngangsvariabler. Ved å bruke DeMorgans teorem kan disse realiseres som NOR-porter. Legg merke til at to siste leddene er faktorisert i uttrykket nedenfor for lettere å kunne realisere funksjonen med 2-inngangs porter:

$$\overline{T} = \overline{\overline{BD} + \overline{\overline{A}(B\overline{C} + CD)}} = \overline{\overline{B} + \overline{D} + (A + (\overline{B\overline{C} + CD}))}$$

Dette utføres rekursivt med alle nivåer inntil vi får en krets realisert fullstendig med 2-inngangs NOR-porter (legg merke til at dobbeltinvertering ikke er nødvendig, men er tatt med her for å vise overgangen mellom f.eks. BD og $\overline{\overline{B}+\overline{D}}$, som er identiske uttrykk):

$$\overline{T} = \overline{\overline{B}+\overline{D}+(A+(B\overline{C}+\overline{C}D))} = \overline{\overline{B}+\overline{D}+A+\overline{B}+C+\overline{C}+D}$$



Figur 2.2: Realisering av funksjonen T med NOR porter

Kommentar: I denne oppgaven ble europeisk IEC-symboler brukt.

Vi ser også at inverterende funksjoner er realisert med en NOR-port, der den ene inngangen settes konstant lik logisk 0. En annen måte å gjøre dette på, er å koble sammen inngangene til en NOR-port.

b) Vi har benyttet ti NOR-porter. Hver av disse krever fire transistorer. Løsningens kostnad er følgelig 40 transistorer.

c) Kritisk sti gjennom kretsen er markert med tykk strek i figur 2.2. Denne går gjennom seks NOR-kretser. Hver av disse har en portforsinkelse på 1,4ns. Total forsinkelsen er følgelig 8,4ns.

Oppgave 4: Design 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 B' C + A B + B 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 B' C + A B C' + A B C + A' B C' + A B C'

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

Lys = A B' C + A B C + A' B C' + A B C'

Nå kan vi trekke ut A C fra de to første leddene og B C' fra de to siste

Lys = A C (B' + B) + (A' + A) B C'

Parentesene kan nå fjernes og vi står igjen med:

$$\text{Lys} = A\,C + B\,C'$$

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

b) Adderer

Løsningen på denne oppgaven baseres på metoden presentert i kapittel 5.1 og 5.2 i Gajski. Vi starter med å bruke Karnaughdiagram for å finne uttrykket for C_{i+1} og S_i. Bitkombinasjoner som ikke står i sannhetstabellen betrakter vi som "Don't care"-tilstander.

Karnaughdiagram for C_{i+1}:

X_i \ Y_iC_i	00	01	11	10
0	0 (0)	0 (1)	1 (3)	0 (2)
1	0 (4)	1 (5)	X (7)	X (6)

$$C_{i+1} = C_i(X_i \oplus Y_i)$$

Karnaughdiagram for S_i:

X_i \ Y_iC_i	00	01	11	10
0	0 (0)	1 (1)	0 (3)	1 (2)
1	1 (4)	0 (5)	X (7)	X (6)

$$S_i = C_i \oplus X_i \oplus Y_i$$

Skriver: $P_i = X_i \oplus Y_i$. Nå kan vi sette opp uttrykkene for C_{i+n}, $n \in \{1,2,3,4\}$.

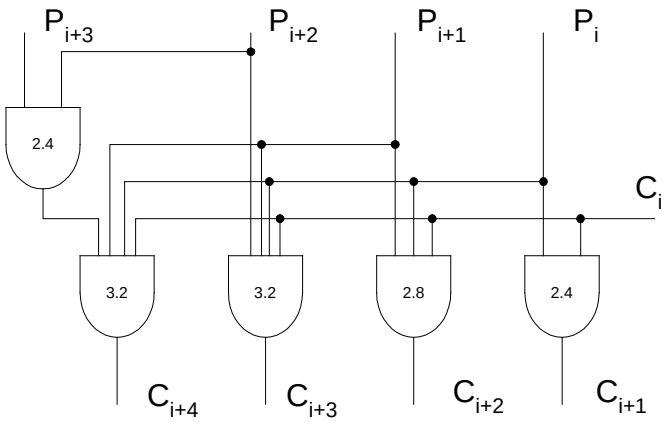
$$C_{i+1} = C_i P_i$$

$$C_{i+2} = C_i P_i P_{i+1}$$

$$C_{i+3} = C_i P_i P_{i+1} P_{i+2}$$

$$C_{i+4} = C_i P_i P_{i+1} P_{i+2} P_{i+3}$$

Siden alle carry-signalene kun er avhengige av inngangs-carry-signalet C_i, kan vi lage en CLA-generator (se fig. 2.3).



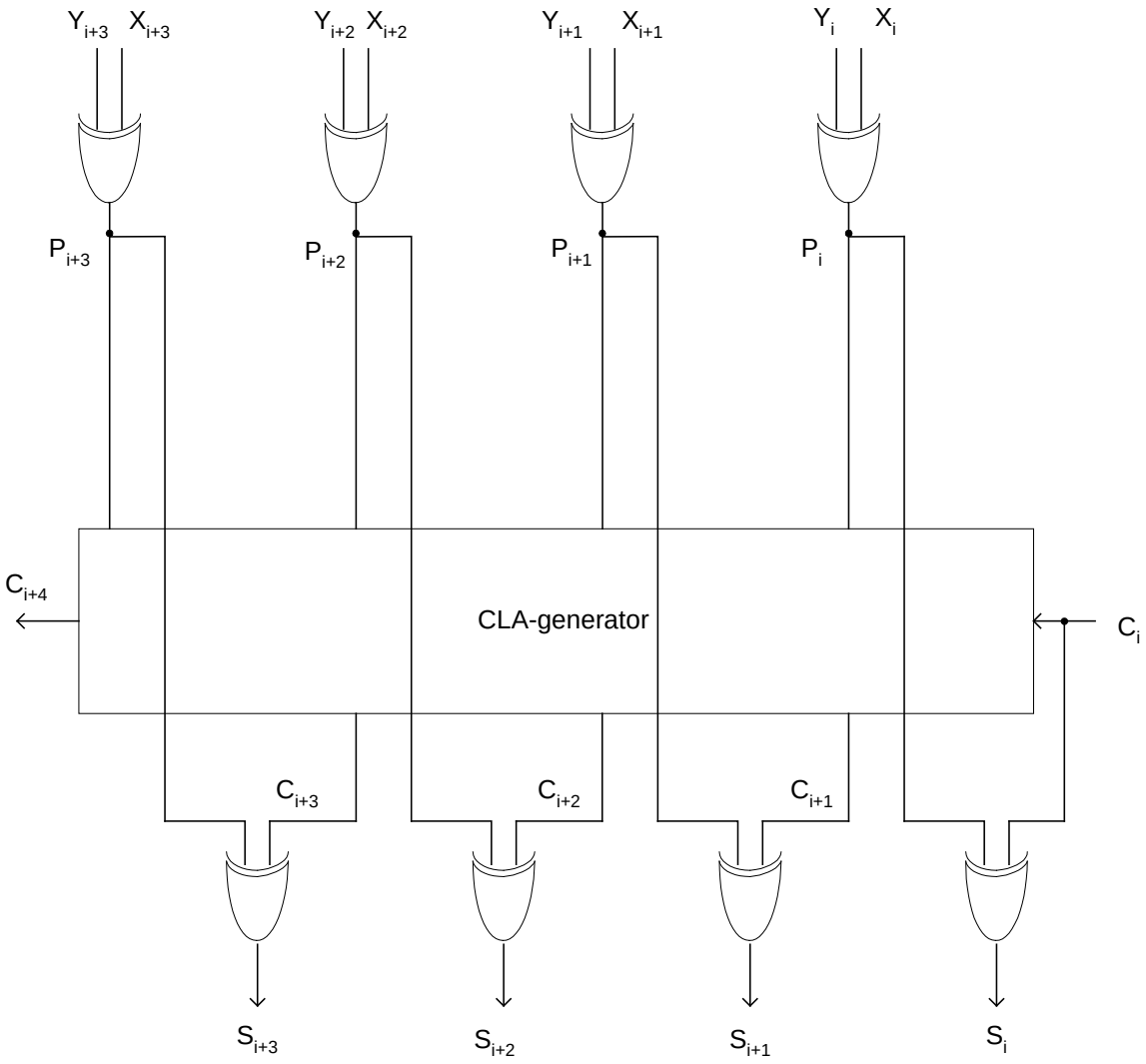
Figur 2.3: CLA-generator

Forplantningstiden er enkel å finne, fordi det finnes kun ett nivå med logiske porter mellom inngangs-carry-signalet og de andre carry-signalene. Dermed blir forplantningstiden identisk med tidsforsinkelsen gjennom de enkelte portene, se tabell 2.3:

	Forplantningstid (ns)
$C_i \rightarrow C_{i+1}$	2.4
$C_i \rightarrow C_{i+2}$	2.8
$C_i \rightarrow C_{i+3}$	3.2

Tabell 2.3: Forplantningstiden gjennom CLA-generatoren

Addereren for oppgave 4b er tegnet på fig 2.4.



Figur 2.4: CLA-adderer for oppgave 4b

En CLA-adderer krever vesentlig mer kombinatorikk (porter) enn en Riple-Carry adderer.

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:

$U_1 = Q_1'Q_0' + Q_0I_1 + Q_1Q_0'I_0 + Q_1Q_0'I_1$

$U_0 = Q_1'Q_0'I_1' + Q_1'Q_0I_1 + Q_1'Q_0'I_0 + Q_1Q_0'I_1'I_0'$

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:

$Q_1\,Q_0$ \ $I_1\,I_0$	00	01	11	10
00	0: 1	1: 1	3: 1	2: 1
01	4: 0	5: 0	7: 1	6: 1
11	12: 0	13: 0	15: 1	14: 1
10	8: 0	9: 1	11: 1	10: 1

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

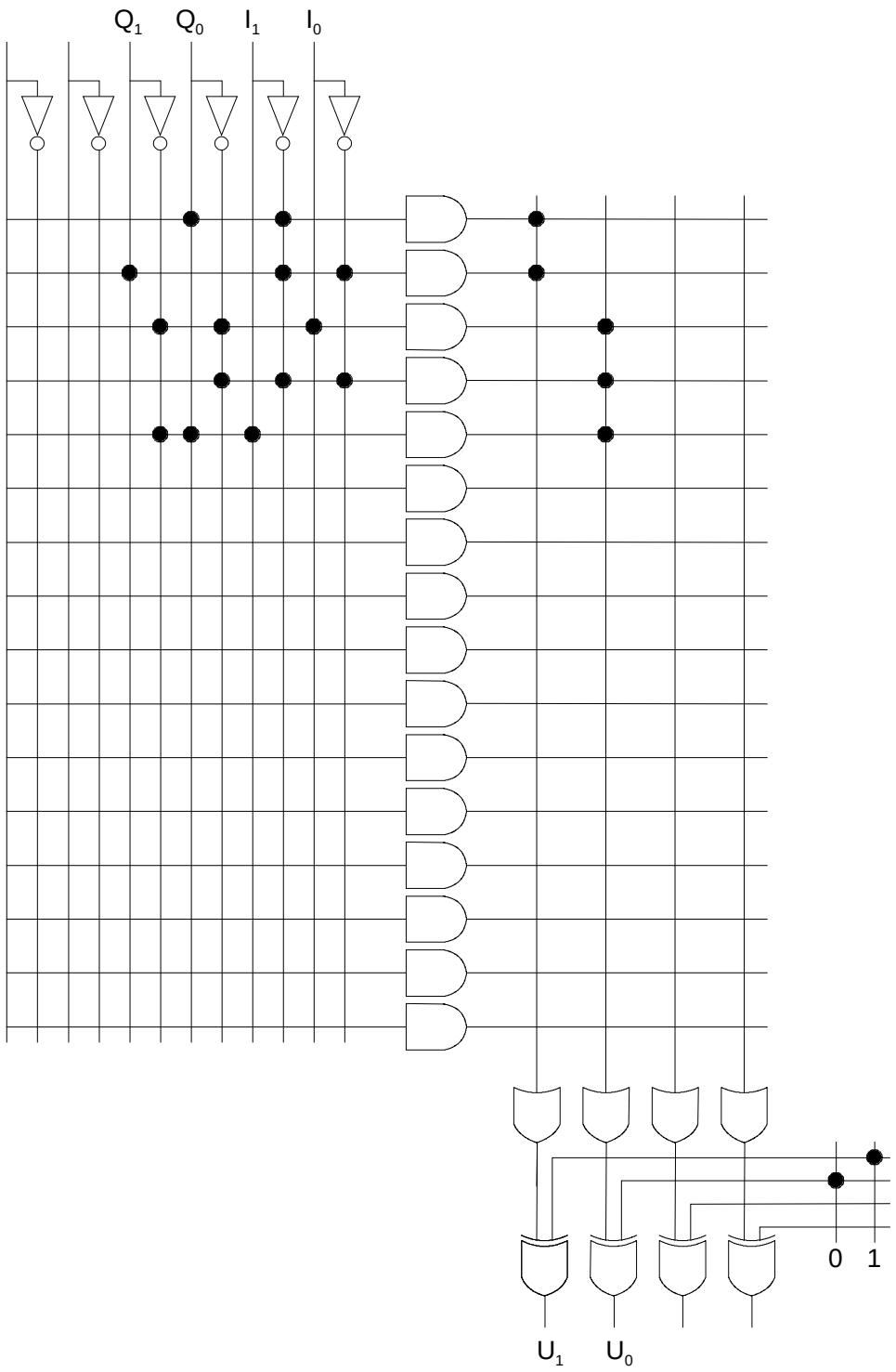
$$U_1 = (Q_0\,I_1' \;+\; Q_1\,I_1'\,I_0' \;)'$$

U_0:

$Q_1\,Q_0$ \ $I_1\,I_0$	00	01	11	10
00	0: 1	1: 1	3: 1	2: 0
01	4: 0	5: 0	7: 1	6: 1
11	12: 0	13: 0	15: 0	14: 0
10	8: 1	9: 0	11: 0	10: 0

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

$$U_0 = Q_0'\,I_1'\,I_0' \;+\; Q_1'\,Q_0'\,I_0 \;+\; Q_1'\,Q_0\,I_1$$



Figur 2-5: PLA med realisering av kombinatorikk

I figur 2-5 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.