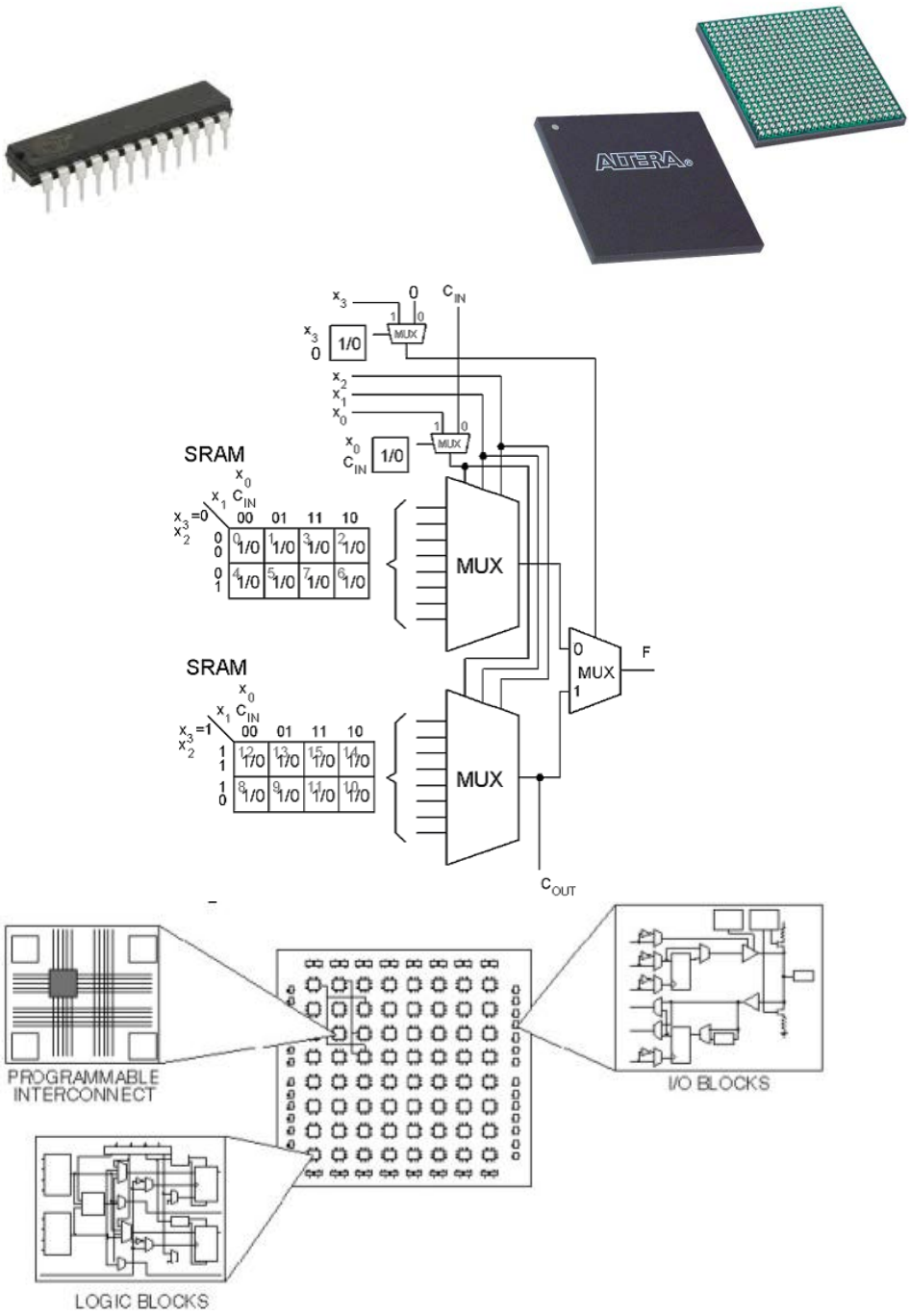
Digital Design *IE1204/5*

Övningshäfte



Sammanställt av *William Sandqvist* `william@kth.se`

ICT/Elektroniksystem

Talsystem och koder

1.1
Nedanstående decimala tal (med basen 10) är givna. Ange motsvarande binära tal.
a) 9 b) 12 c) 71 d) 503

1.2
Omvandla nedanstående binärtal till decimaltal.
a) 101101001_2 b) 110100.010_2

1.3
Omvandla nedanstående binärtal (bas=2) till motsvarande oktala tal (bas=8) och hexadecimala tal (bas=16).
a) $01\ 1101_2$ b) $1000\ 1011_2$ c) $1\ 0011\ 0101_2$ d) $1101\ 1110\ 1001\ 0001_2$ e) $10\ 1001.001_2$

1.4
Omvandla nedanstående hexadecimala tal (bas=16) till motsvarande oktala tal (bas=8).
a) $94D_{16}$ b) $9E.7A_{16}$

1.5
Omvandla det oktala (bas=8) talet 4515_8 till motsvarande hexadecimala tal (bas=16).

1.6
Skriv det hexadecimala (bas=16) talet BAC_{16} på decimal form (bas=10).

1.7
Vad karakteriserar Gray-koder, och hur kan de konstrueras?

1.8
Skriv följande tal ”med tecken” med två-komplementsnotation, $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.
a) -23 b) -1 c) 38 d) -64

1.9
Skriv följande tal ”med tecken” med ett-komplementsnotation, $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.
a) -23 b) -1 c) 38 d) -0

Digital aritmetik

2.1
Addera för hand följande par binära tal.
a) 110 + 010 b) 1110 + 1001 c) 11 0011.01 + 111.1 d) 0.1101 + 0.1110

2.2
Addera eller subtrahera (addition med motsvarande negativa tal) nedanstående tal. Talen skall representeras som binära 4-bitstal (Nibble) på två-komplementform.
a) 1 + 2 b) 4 – 1 c) 7 – 8 d) -3 – 5

2.3
Multiplicera för hand följande par av teckenlösa binära tal.
a) 110·010 b) 1110·1001 c) 11 0011.01·111.1 d) 0.1101·0.1110

2.4
Dividera för hand följande par av teckenlösa binära tal.
a) 110/010 b) 1110/1001

2.5

IEEE-754 standarden för lagring av 32-bitars flyttal.
Antag att ett 32-bitars flyttal lagras i ett register som: $40C80000_{16}$ vilket reellt decimaltal är det?

(2.6)

Flyttalsformatets principer blir mer överskådliga om man av pedagogiska skäl ”skalar ned” det till 4 bitars registerstorlek (Nibble). Däremot skulle ett 4-bitarsformat vara praktiskt oanvändbart.

Antag följande fyrabitars flyttalsformat: $[b_3 b_2 b_1 b_0] = (-1^{b_3}) \cdot (1.b_0) \cdot (2^{b_2 b_1 - 1})$

Tecknet uttrycks med biten b_3, binalerna representeras av en bit b_0, och exponenten har två bitar $b_2 b_1$ uttryckta som exess-1.

a) Räkna upp de tal som kan representeras med full precision. Markera dem på tallinjen.
b) Hur stort är det största kvantiseringsfelet.
c) Kan talet 0 representeras? Om inte, föreslå en förändring av formatet så att 0 kan representeras.

(2.7)

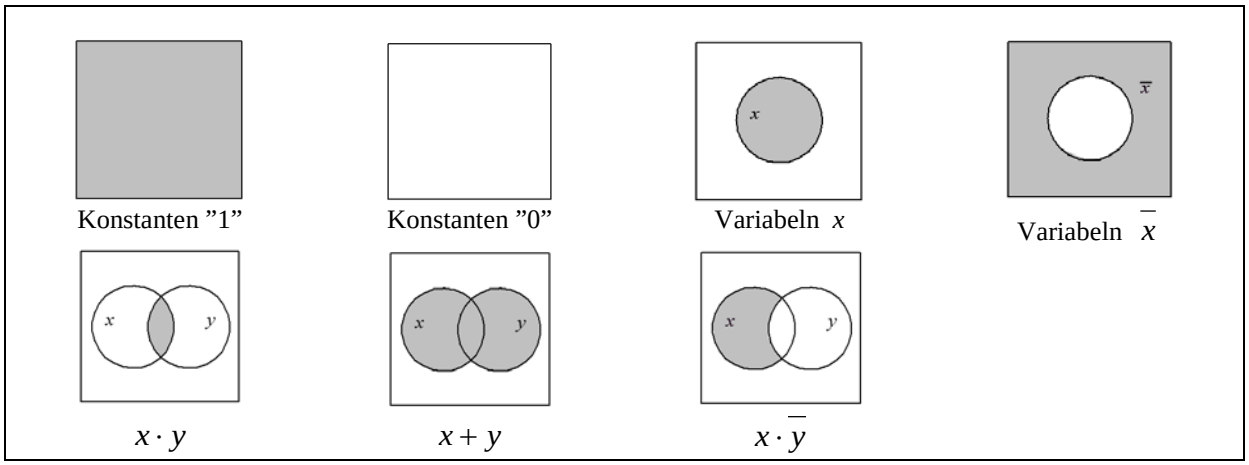
För att undersöka addition och multiplikation av flyttal antar vi nu av pedagogiska skäl ett 6-bitarsformat. (Detta är fortfarande för få bitar för att vara praktiskt användbart).

$[b_5 b_4 b_3 b_2 b_1 b_0] = (-1^{b_5}) \cdot (1.b_2 b_1 b_0) \cdot (2^{b_4 b_3 - 1})$

a) Vilka av följande tal kan representeras i detta format? 0,25 0,8125 -1,375 4,25 7.5
b) Addera talen ($b_5 b_4 b_3 b_2 b_1 b_0$) 001111 och 010010. Vad krävs för att undvika precisionsförlust?
c) Multiplicera de föregående talen med varandra.

Mängdräkning och kubteori

Venn-diagram representation



3.1
Bevisa den distributiva lagen med hjälp av Venn-diagram.
$x + y \cdot z = (x + y) \cdot (x + z)$

3.2
Bevisa De Morgans lag med hjälp av Venn-diagram.
$\overline{x \cdot y} = \overline{x} + \overline{y}$

3.3
a) Rita ett Venn-diagram för tre variabler och markera var sanningstabellens alla mintermer är placerade.
b) Minimera funktionen med hjälp av Venn-diagram.
$f = \overline{x_2}\,\overline{x_1}\,x_0 + \overline{x_2}\,x_1\,x_0 + x_2\,\overline{x_1}\,x_0 + x_2\,x_1\,\overline{x_0} + x_2\,x_1\,x_0$

Kub representation

3.4
a) Representera följande funktion av tre variabler som en 3-dimensionell kub med Gray-kodade hörn.

$$f(x_2, x_1, x_0) = \sum m(0, 2, 3, 4, 6)$$

b) Använd kuben för att förenkla funktionen.

Boolesk algebra och grindar

Booles algebra

$A \cdot A = A \quad A \cdot 0 = 0 \quad A + 0 = A$

$A + A = A \quad A \cdot 1 = A \quad A + 1 = 1$

Distributiva lagarna	$A\cdot(B+C) = A\cdot B + A\cdot C$ $A + (B\cdot C) = (A+B)\cdot(A+C)$
Kommutativa lagarna	$A\cdot B = B\cdot A$ $A + B = B + A$
Associativa lagarna	$(A\cdot B)\cdot C = A\cdot(B\cdot C)$ $(A+B)+C = A+(B+C)$

Absorbtionslagarna	$A + A\cdot B = A$ $A\cdot(A+B) = A$
Koncensuslagen	$A\cdot B + \overline{A}\cdot C =$ $A\cdot B + \overline{A}\cdot C + B\cdot C$
de Morgans lagar	$\overline{(A+B)} = \overline{A}\cdot\overline{B}$ $\overline{(A\cdot B)} = \overline{A} + \overline{B}$

4.1

Använd räknelagarna i den booleska algebran för att förenkla följande logiska uttryck:

a) $f = a\cdot\overline{c}\cdot d + a\cdot d$

b) $f = a\cdot(\overline{b} + \overline{a}\cdot c + a\cdot b)$

c) $f = a + \overline{b} + \overline{a}\cdot b + \overline{c}$

d) $f = (a + b\cdot\overline{c})\cdot(\overline{a}\cdot\overline{b} + c)$

e) $f = (a + \overline{b})\cdot(\overline{a} + b)\cdot(a + b)$

f) $f = \overline{a}\cdot\overline{b}\cdot c + a\cdot b\cdot c + \overline{a}\cdot b\cdot c$

g) $f = \overline{a}\cdot b\cdot\overline{c} + \overline{a}\cdot b\cdot\overline{d} + c\cdot d$

h) $f = a + (\overline{\overline{a}\cdot\overline{b}})$

i) $f = \overline{\overline{a} + \overline{\overline{a}\cdot b + c}}$

4.2

Bevisa algebraiskt att följande samband är giltiga.

a) $(\overline{x_3}x_2 + 1 + x_3\overline{x_2})x_1 + x_2x_1 + x_2 + \overline{x_1} = 1$

b) $\overline{x_3}x_2x_1 + \overline{(x_3 + \overline{x_1})} = \overline{x_3}x_1$

c) $\overline{(x_2 + x_1)}\,\overline{x_2}\,\overline{x_1} + x_3 = \overline{x_2}\,\overline{x_1} + x_3$

d) $\overline{x_2 + x_1} + \overline{x_2}\,\overline{x_1} + x_3 = \overline{x_2}\,\overline{x_1} + x_3$

4.3

Förenkla nedanstående tre uttryck så långt som möjligt.

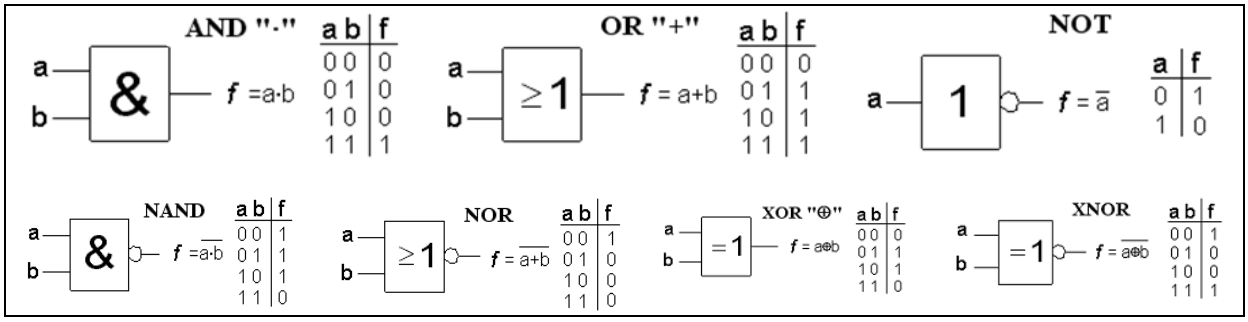
a) $(x+y)(\overline{x}+z)$ b) $(x + \overline{y} + xy)(x + \overline{y})\overline{x}y$ c) $x(1 + \overline{x}y) + \overline{x}$

4.4

Förenkla nedanstående uttryck så långt som möjligt.

$\overline{(a+b+\overline{c})(a+\overline{b}+\overline{c})(\overline{a}+\overline{\overline{b}c}+b\overline{c})}$

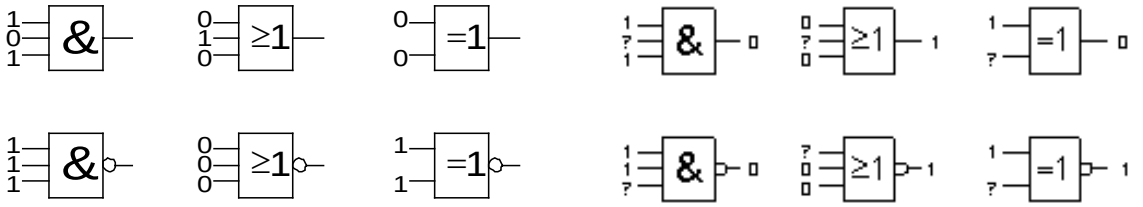
Grindar



4.5

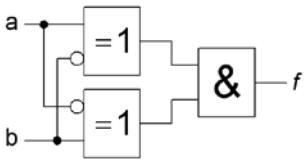
a) Ange **utsignalen** 1/0 för följande sex grindtyper när insignalerna är de som angivits i figuren.

b) Ange **insignalen** 1/0 för följande sex grindtyper när utsignalerna är de som angivits i figuren.



4.6

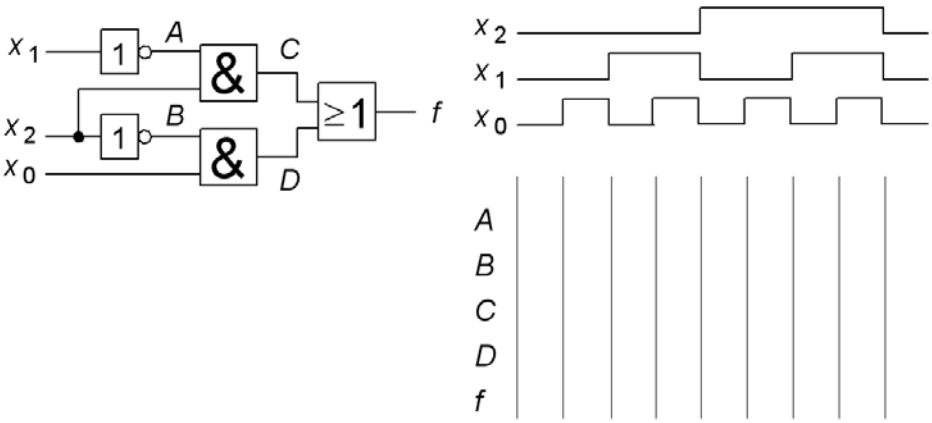
Förenkla $f(a,b)$ som realiseras av figurens grindnät, så långt som möjligt, och ange funktionens namn.



4.7

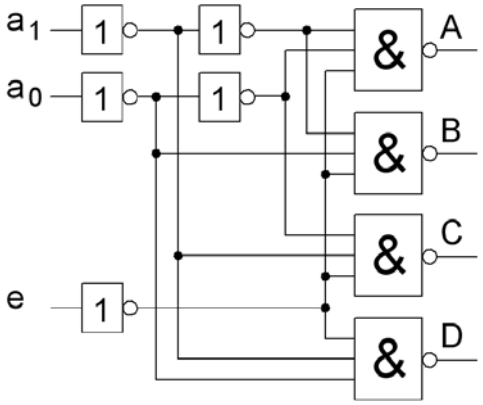
a) Rita tidsdiagram över signalerna *A*, *B*, *C*, *D*, *f*. Insignalerna till x_0, x_1, och x_2 har frekvensförhållandet 4: 2:1 för att ”svepa” igenom sanningstabellens *alla* kombinationer i ”rätt” ordning.

b) Skriv upp sanningstabellen för funktionen *f*.



4.8

Ange de logiska uttrycken för *A*, *B*, *C* och *D*.



4.9

Förenkla det sammansatta uttrycken nedan så långt som möjligt.

a) $x_2 \oplus x_1 \oplus x_1 x_2$ b) $x_2 x_1 \oplus \overline{(x_2 + x_1)}$

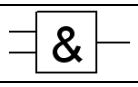
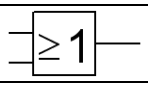
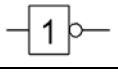
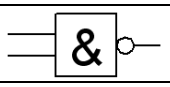
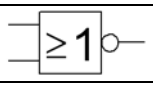
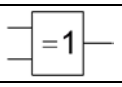
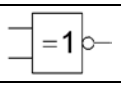
4.10

Visa att

a) $\overline{x_2 \oplus x_1} = \overline{x_2} \oplus x_1 = x_2 \oplus \overline{x_1}$ b) $x_2 \oplus x_1 = \overline{x_2} \oplus \overline{x_1}$

4.11

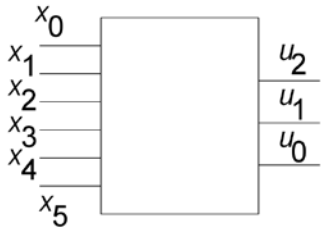
Här visas de Europeiska grindsymbolerna. Amerikas dominans inom halvledarområdet gör att man även måste känna till de amerikanska symbolerna. Namnge grindarna och rita de motsvarande amerikanska grindsymbolerna.

4.12

Ett kombinatoriskt nät med sex insignaler x_5, x_4, x_3, x_2, x_1, x_0 och tre utsignaler u_2, u_1, u_0, beskrivs med text på följande sätt:

- $u_0 = 1$ om och endast om ”antingen både x_0 och x_2 är 0 eller x_4 och x_5 är olika”
- $u_1 = 1$ om och endast om ”x_0 och x_1 är lika och x_5 är inversen av x_2”
- $u_2 = 0$ om och endast om ”x_0 är 1 och någon av x_1 ... x_5 är 0”

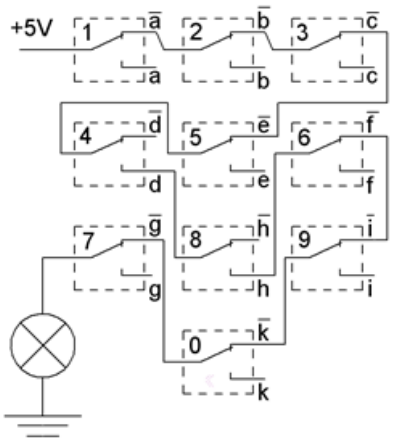


Beskriv nätet med Boolesk algebra och operationerna AND OR NOT XOR i stället.

Sanningstabellen, SP och PS -form, fullständig logik

5.1

I figuren visas ett enkelt ”kodlås” med 10 st växlingskontakter. För en *viss kombination* av samtidigt nedtryckta kontakter lyser lampan.



a) Vilken kombination?

b) Ange den logiska funktionen för lysande lampa. Variablernas beteckningar står i figuren (*a* … *k*).

f =

5.2

En logisk funktion har följande sanningstabell:

a b c	*f*
0 0 0	1
0 0 1	0
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	0
1 1 1	1

Ange funktionen på PS-normalform (produkter av summatermer):
f(*a*, *b*, *c*) =

Ange funktionen på SP-normalform (summa av produkttermer):
f(*a*, *b*, *c*) =

5.3

En minimerad funktion är angiven på SP form (Summa av Produkter).
Ange samma funktion med mintermer som SP, respektive med maxtermer som PS (Produkt av Summor).

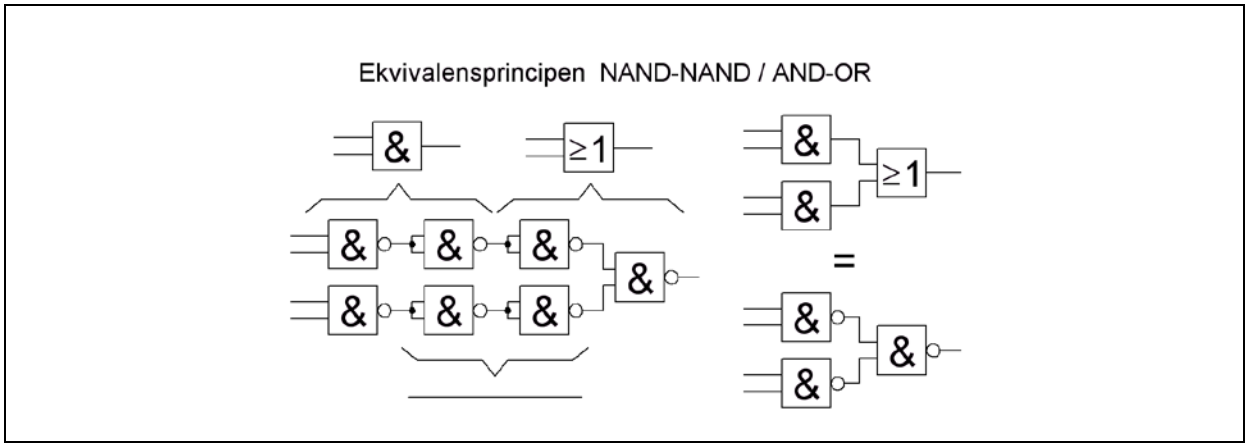
$$f(x,y,z) = x\bar{y} + y\bar{z} + \bar{x}z$$

5.4

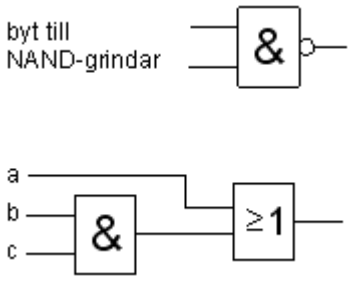
En funktion är angiven med en blandning av produkter och delsummor.
Ange samma funktion som SP (Summa av Produkter) med mintermer, respektive PS (Produkt av Summor) med maxtermer.

$$f(x,y,z) = (x+\bar{y})\left(xyz + \bar{y}(x+z)\right) + xy\bar{z}(x+\bar{x}y)$$

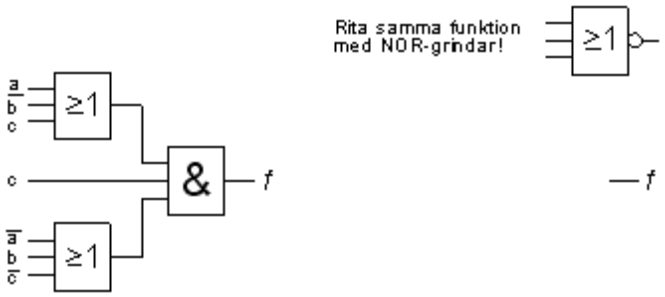
Ekvivalens AND-OR / NAND-NAND och OR-AND / NOR-NOR



5.5
Rita vidstående AND/OR-nät som ett NAND/NAND-nät.



5.6
Rita vidstående OR/AND-nät som ett NOR/NOR nät.



5.7
a) Skriv upp sanningstabellen för en krets med fyra ingångar som definierar jämn paritet; dvs. kretsens utgång är ”1” när ett jämnt antal av ingångarna samtidigt är ”1”.

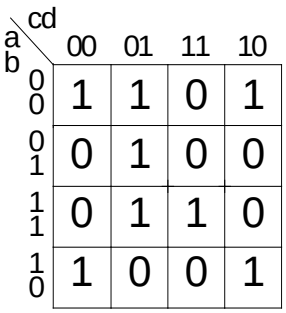
b) Implementera denna funktion med så få NOR-grindar som möjligt.

Karnaughdiagrammet

6.1

Gör bästa möjliga hoptagningar i Karnaughdiagrammet.
Ange den minimerade funktionen på SP-form.

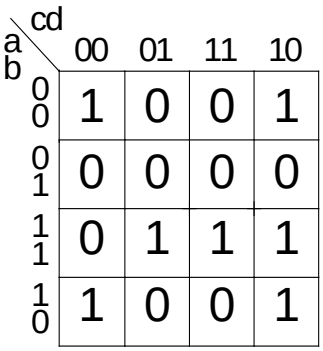
f =



6.2

Gör bästa möjliga hoptagningar i Karnaughdiagrammet.
Ange den minimerade funktionen på SP-form.

f =



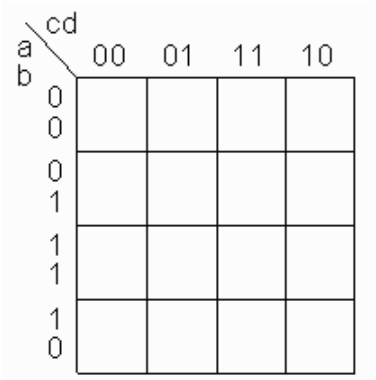
6.3

Placera ut följande funktion i Karnaughdiagrammet.

$f = \overline{a}b\overline{c} + ab\overline{c} + bc\overline{d}$

Försök att göra bättre hoptagningar. Ange därefter den ”minimerade” funktionen:

f =

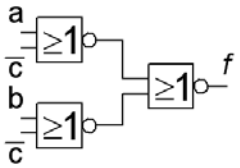


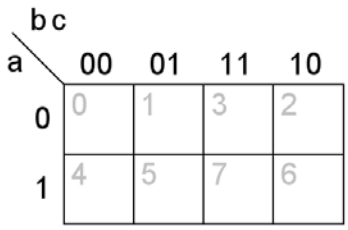
6.4

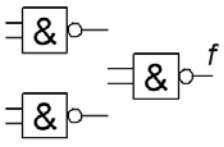
Överst i figuren till höger finns ett NOR-NOR nät.

Analysera detta nät och för in sanningstabellen i Karnaughdiagrammet.

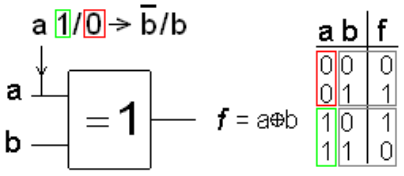
Gör hoptagningar i Karnaughdiagrammet och realisera funktionen med NAND-grindar nederst i figuren i stället.
Variablerna *a b* och *c* finns tillgängliga både normala och inverterade.







PLD-kretsar har ofta en XOR-grind på utgången så att man vid behov skall kunna invertera funktionen. Man kan då välja mellan att ta ihop 0:or eller 1:or efter vad som är fördelaktigast.



6.5
En funktion med fyra variabler definieras med mintermer på SP-form. Använd Karnaughdiagram för att minimera funktionen. Minimera även funktionens invers.

$$f(x_3, x_2, x_1, x_0) = \sum m(0, 2, 4, 8, 10, 12) \quad f = ? \quad \overline{f} = ?$$

6.6
En funktion med fyra variabler definieras med maxtermer på PS-form. Använd Karnaughdiagram för att minimera funktionen. Minimera även funktionens invers.

$$f(x_3, x_2, x_1, x_0) = \prod M(0, 1, 4, 5, 10, 11, 14, 15) \quad f = ? \quad \overline{f} = ?$$

6.7
En funktion med fyra variabler definieras med mintermer på SP-form. Använd Karnaughdiagram för att minimera funktionen. Minimera även funktionens invers.

$$f(x_3, x_2, x_1, x_0) = \sum m(0, 2, 3, 4, 6, 7, 8, 9, 10, 12, 13, 14) \quad f = ? \quad \overline{f} = ?$$

6.8
Ibland kan problemställningen vara sådan att vissa ingångskombinationer är ”omöjliga” och därför *inte* kan inträffa. Sådana mintermer (eller maxtermer) betecknar man med d (”don’t care”) och använder dom som ettor eller nollor allt efter vad som passar bäst för att få så stora hoptagningar som möjligt.

$$f(x_3, x_2, x_1, x_0) = \sum m(3, 5, 7, 11) + d(6, 15) \quad f = ? \quad \overline{f} = ?$$

6.9

$$f(x_3, x_2, x_1, x_0) = \sum m(1, 4, 5) + d(2, 3, 6, 7, 8, 9, 12, 13) \quad f = ? \quad \overline{f} = ?$$

6.10

En funktion med **fem** variabler definieras

$$f(x_4, x_3, x_2, x_1, x_0) = \sum m(9, 11, 12, 13, 14, 15, 16, 18, 24, 25, 26, 27)$$

se ifylld sanningstabell.

Använd Karnaughdiagram-metoden för att minimera funktionen. Minimera även funktionens invers.

$$f(x_4, x_3, x_2, x_1, x_0) \quad f = ? \quad \overline{f} = ?$$

	x_4	x_3	x_2	x_1	x_0	f
0	0	0	0	0	0	0
1	0	0	0	0	1	0
2	0	0	0	1	0	0
3	0	0	0	1	1	0
4	0	0	1	0	0	0
5	0	0	1	0	1	0
6	0	0	1	1	0	0
7	0	0	1	1	1	0
8	0	1	0	0	0	0
9	0	1	0	0	1	1
10	0	1	0	1	0	0
11	0	1	0	1	1	1
12	0	1	1	0	0	1
13	0	1	1	0	1	1
14	0	1	1	1	0	1
15	0	1	1	1	1	1

	x_4	x_3	x_2	x_1	x_0	f
16	1	0	0	0	0	1
17	1	0	0	0	1	0
18	1	0	0	1	0	1
19	1	0	0	1	1	0
20	1	0	1	0	0	0
21	1	0	1	0	1	0
22	1	0	1	1	0	0
23	1	0	1	1	1	0
24	1	1	0	0	0	1
25	1	1	0	0	1	1
26	1	1	0	1	0	1
27	1	1	0	1	1	1
28	1	1	1	0	0	0
29	1	1	1	0	1	0
30	1	1	1	1	0	0
31	1	1	1	1	1	0

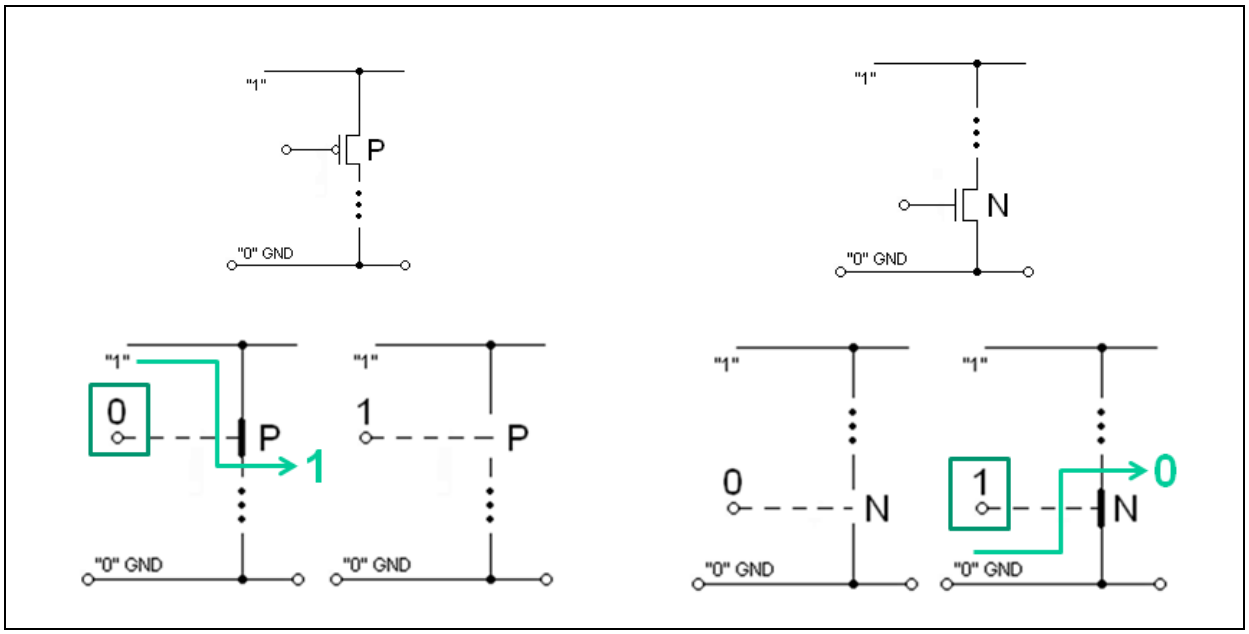
$\overline{x}_4$

x_3x_2 \ x_1x_0	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

x_4

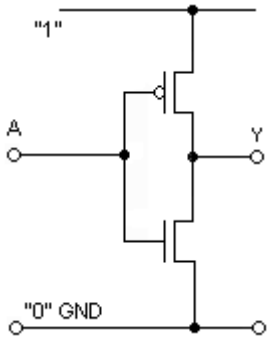
x_3x_2 \ x_1x_0	00	01	11	10
00	16	17	19	18
01	20	21	23	22
11	28	29	31	30
10	24	25	27	26

MOS-transistorn och digitala kretsar



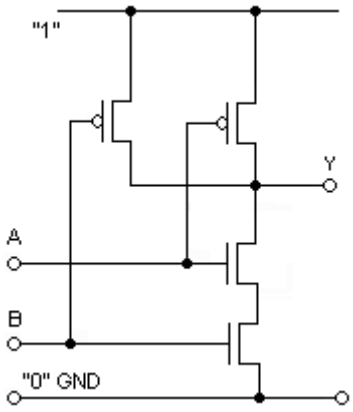
7.1

Identifiera transistorernas beteenden och skriv upp sanningstabellen för *Y*(*A*).
Vilken logisk funktion är det?



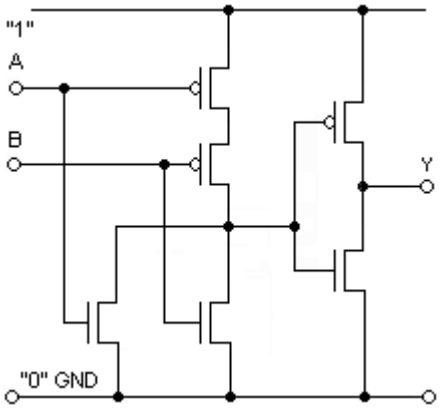
7.2

Identifiera transistorernas beteenden och skriv upp sanningstabellen för *Y*(*A*,*B*).
Vilken logisk funktion är det?



7.3

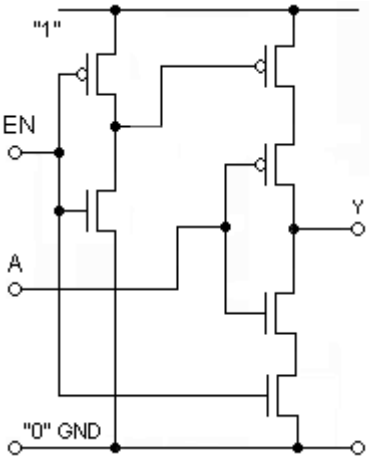
Identifiera transistorernas beteenden och skriv upp sannings-tabellen för *Y*(*A*,*B*).
Vilken logisk funktion är det?



7.4

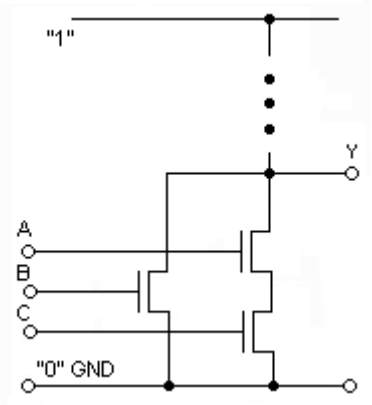
Studera kretsen och beskriv funktionen. Vilken roll spelar signalen *EN*? Vilket samband gäller mellan *Y* och *A*? *Y*(*B*).

Hur många ”tillstånd” kan utgången ha?



7.5

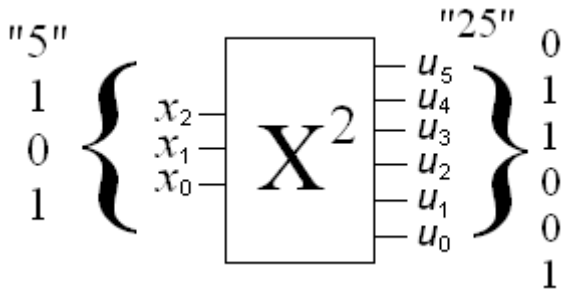
Figuren visar ena halvan av en CMOS krets. Rita dit den andra halvan, som innehåller PMOS transistorerna. Ange den logiska funktionen *Y*(*A*,*B*,*C*).



Kombinatoriska kretsnät

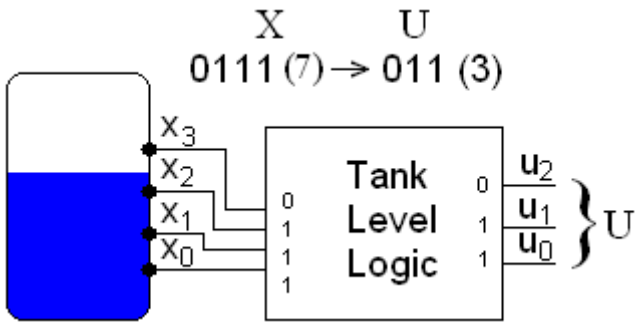
8.1.

Tag fram de Boolska uttrycken på minimerad SP-form för ett kombinatoriskt nät som omvandlar ett trebitars binärkodat tal X (x_2, x_1, x_0) till ett binärkodat sexbitstal U $(u_5, u_4, u_3, u_2, u_1, u_0)$ som är lika med kvadraten på talet, dvs. $U = X^2$. Använd Karnaughdiagram.



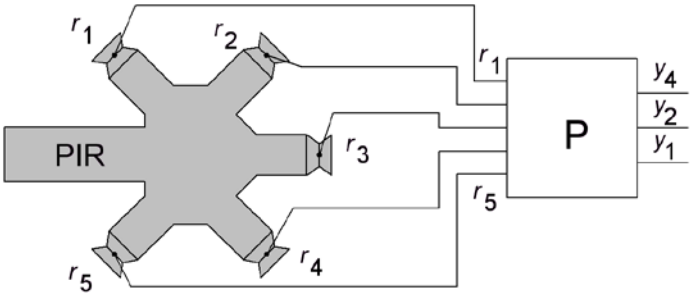
8.2

Ett övervakningssystem till en vattentank består av fyra nivågivare x_3, x_2, x_1, x_0. Signalerna från dessa bildar ett binärt fyr-bitstal X. Ett logiknät ”Tank Level Logic” omkodar X till ett trebitars tal U (u_2, u_1, u_0) som presenterar nivån som ett binärt tal mellan 0 och 4.
Konstruera logiknätet. Ange de Boolska uttrycken på minimerad SP-form. Utnyttja att det är många av insignalskombinationerna som *aldrig* kan uppkomma! Ingångsvariablerna finns tillgängliga i både inverterad och oinverterad form från nivågivarna.
Använd ekvivalensprincipen för att ta fram logiknätet med enbart NAND-grindar.



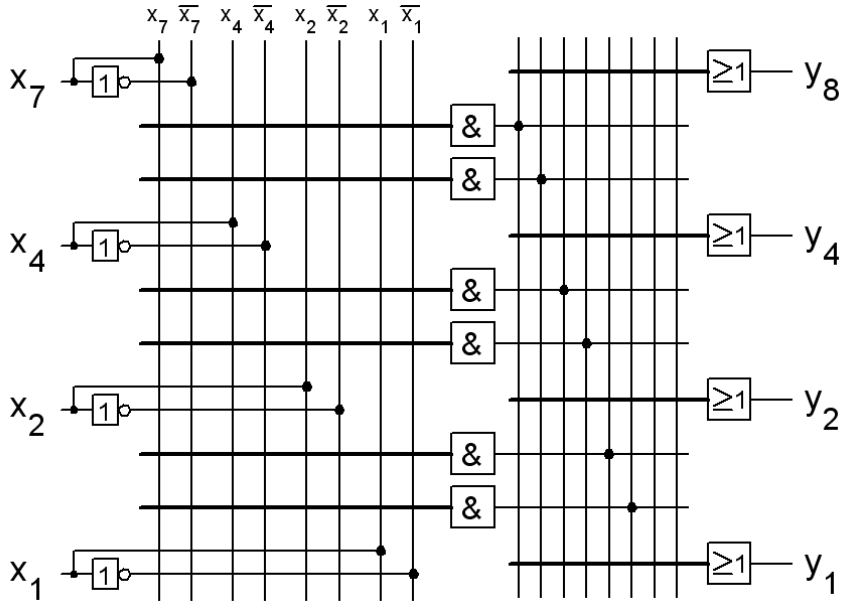
8.3

En PIR på en flygplats har fem anslutningsramper (GATES). Ramperna numreras 1...5. Vid varje ramp finns en givare som ger en utsignal $r_i = 0$ om ett flygplan är anslutet till rampen, annars 1. Ett kombinatoriskt nät, P, hjälper flygtrafikledaren att dirigera anländande flygplan till lediga ramper. Nätet har har insignalerna r_1, r_2, r_3, r_4, r_5 och utsignalerna y_4, y_2, y_1. Kombinationen av utsignalerna y_4, y_2, y_1 skall i binärkod ge numret på den ramp med högst ordningsnummer som är ledig. Om *ingen* ramp är ledig används numret $(y_4, y_2, y_1) = (1, 1, 1)$.
Minimera varje utgång för sig.



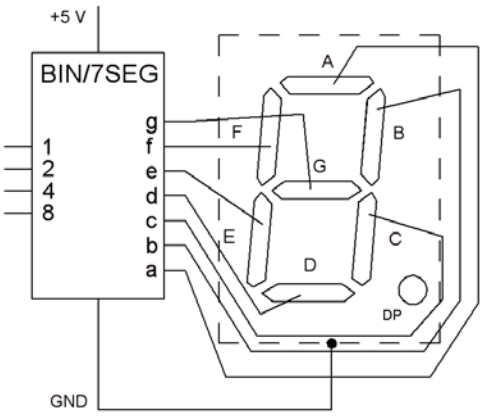
8.4

De decimala siffrorna 0 till 9 kan kodas i så kallad 7421-kod. Det är en viktad binärkod med positionsvikterna 7, 4, 2, och 1. Där två kombinationer av de viktade talen kan ge samma värde väljes det med det minsta antalet ettor. (7421-Koden har egenskapen att den kodar siffrorna 0 till 9 med minimalt antal ettor, totalt 14 st).
Konstruera en krets som översätter från 7421-koden till den mer vanliga BCD-koden (8421-kod).
Använd en PLD-krets av AND-OR typ. Både AND-planet och OR-planet kan programmeras var för sig.
Rita kryss i figuren nedan för att visa vilka programmerade förbindelser som skall göras. Grindarnas ingångar har ritats med ”förenklat” ritsätt”.



8.5

En 7-segmentavkodare avkodar ett binärt 4-bitstal till motsvarande segmentbild för siffrorna 0...9 (eller hexadecimalt 0 ... F).
Ställ upp sanningstabellen, och ange ett minimerat logiskt samband för tex. segmentet ”G”.



8.6

Visa hur en 4-1 multiplexor kan användas som en funktionsgenerator och tex. generera OR-funktionen.

8.7

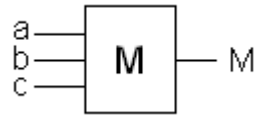
En majoritetsgrind antar på utgången samma värde som en *majoritet* av ingångarna. Grinden kan tex. användas i feltolerant logik, eller till bildbehandlingskretsar.

a) Ställ upp grindens sanningstabell och minimera funktionen med Karnaughdiagram. Realisera funktionen med AND-OR grindar.

b) Realisera majoritetsgrinden med en 8:1 MUX.

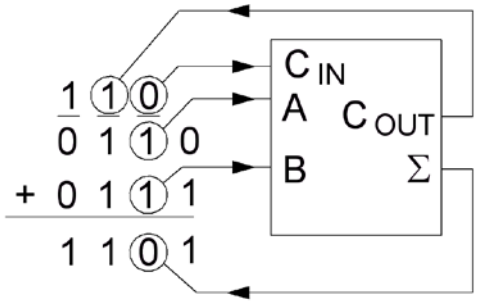
c) Använd Shannon dekomposition och realisera majoritetsgrinden med en 2:1 MUX och grindar.

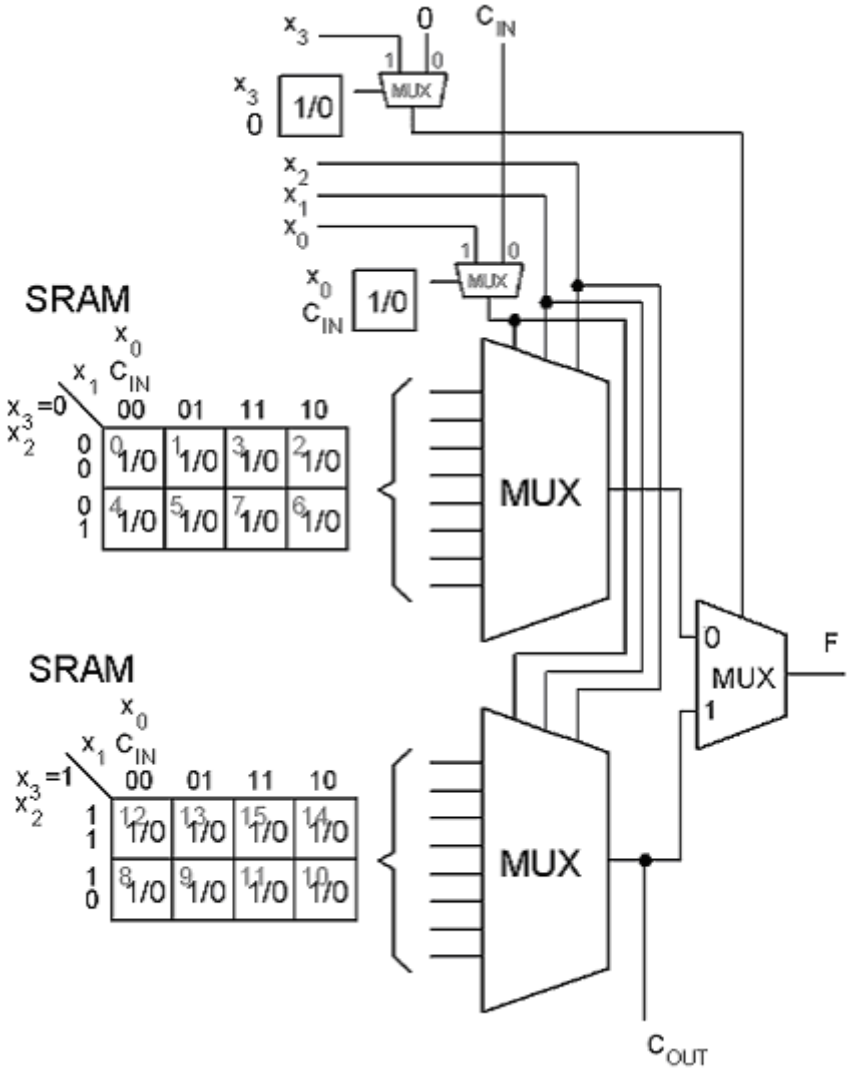
d) Realisera majorotetsgrinden med bara 2:1 MUXar.



8.8

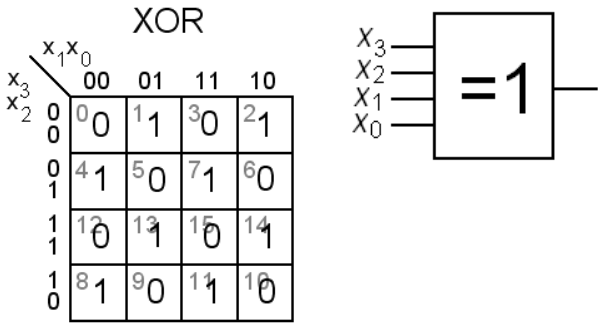
Ställ upp heladderarens sanningstabell. Visa hur en heladderare realiseras i en FPGA-krets. Logikelementen i en FPGA har möjlighet att kaskadkoppla C_{OUT} och C_{IN} mellan ”grannarna”. Visa innehållet i SRAM-cellerna (LUT, LookUp Table).

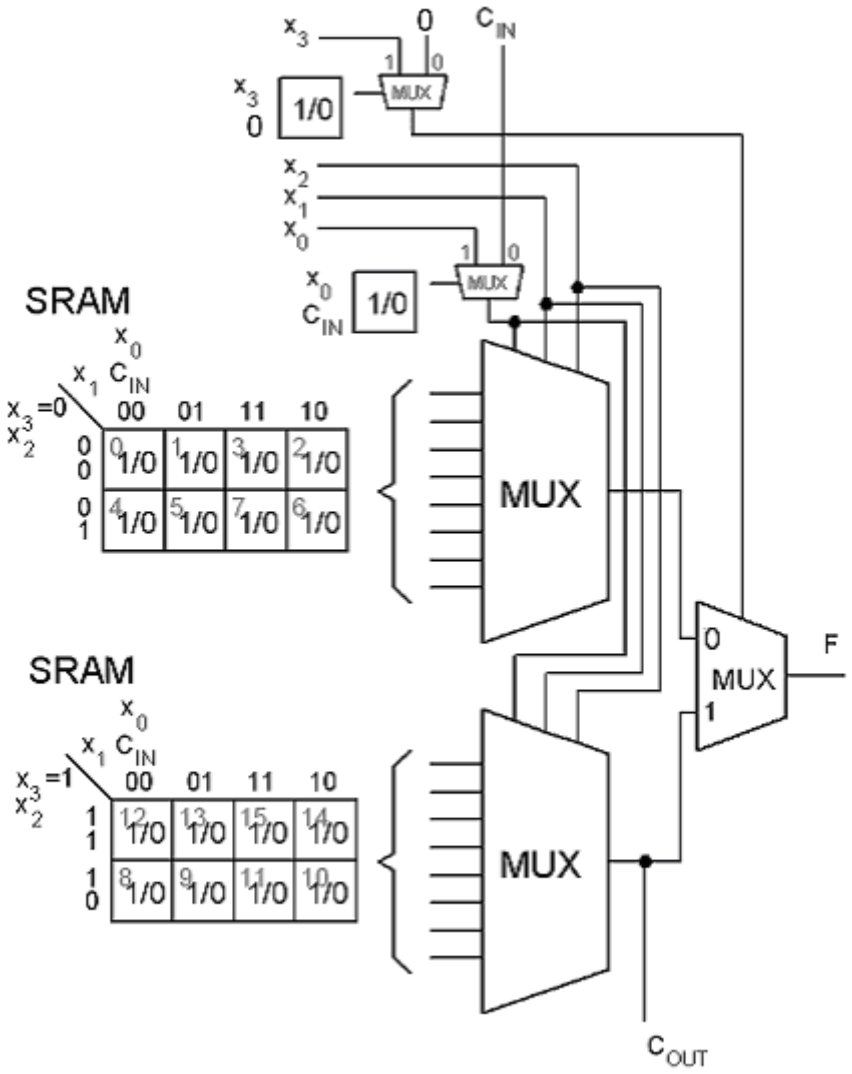




8.9

Visa hur en 4 ingångars exorgrind (XOR) realiseras i en FPGA-krets. Visa innehållet i SRAM-cellerna (LUT, LookUp Table).





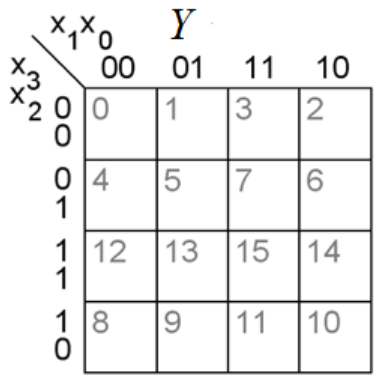
8.10

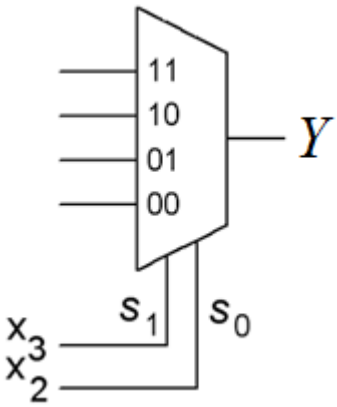
Den Booleska funktionen Y av fyra variabler x_3 x_2 x_1 x_0 definieras av sanningstabellen.

	$x_3x_2x_1x_0$	Y		$x_3x_2x_1x_0$	Y
0	0000	–	8	1000	–
1	0001	–	9	1001	–
2	0010	1	10	1010	1
3	0011	0	11	1011	0
4	0100	0	12	1100	0
5	0101	1	13	1101	1
6	0110	0	14	1110	1
7	0111	1	15	1111	0

a) Använd Karnaughdiagrammet för att konstruera ett minimalt nät för funktionen (utnyttja ”–” som don’t care). Använd valfria grindar.

b) Realisera funktionen Y med en 4:1 multiplexor och (valfri) grindlogik. Använd x_3 och x_2 som multiplexorns dataväljarsignaler.



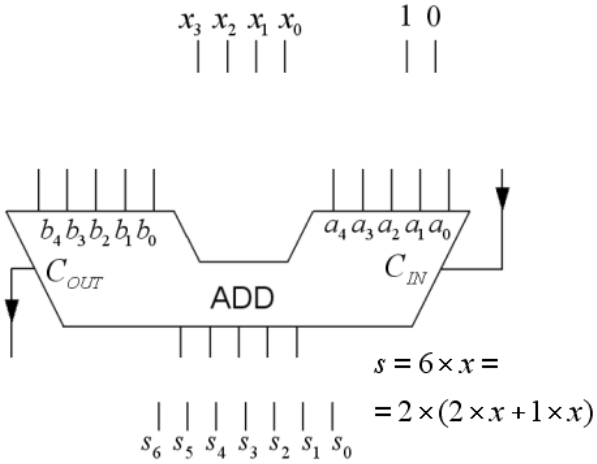


8.11

Ett fyrabitars tal x ($x_3x_2x_1x_0$) ska multipliceras med konstanten 6.
Detta sker genom att talet x ansluts till en fem bitars adderare som konfigurerats för att utföra operationen $6{\cdot}x = 2{\cdot}(2{\cdot}x + 1{\cdot}x)$

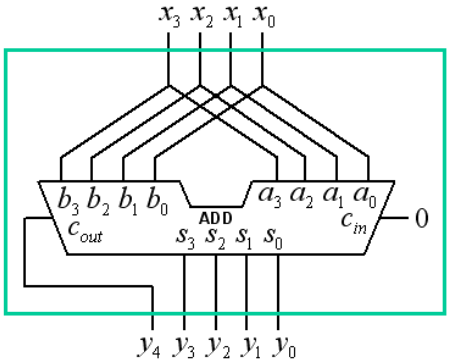
a) Rita hur adderaren ska konfigureras. Förutom de fyra bitarna i talet x så finns även bitar med värdet 0 och 1 tillgängliga vid behov.

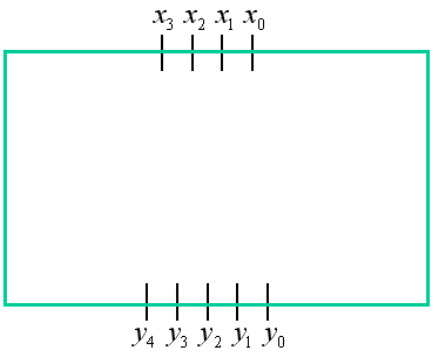
b) Vilket är det största binära tal s ($s_6s_5s_4s_3s_2s_1s_0$) som kan förekomma på utgången efter det att kretsen konfigurerats för operationen? **Svara binärt**.



8.12

Ett fyrabitars teckenlöst tal x ($x_3x_2x_1x_0$) är anslutet till en 4-bits adderare som figuren visar. Resultatet blir ett 5-bitars tal y ($y_4y_3y_2y_1y_0$). Rita in i figuren till höger hur samma resultat kan fås ***utan användning av adderaren***. Konstanter med värdet 0 eller 1 finns tillgängliga.

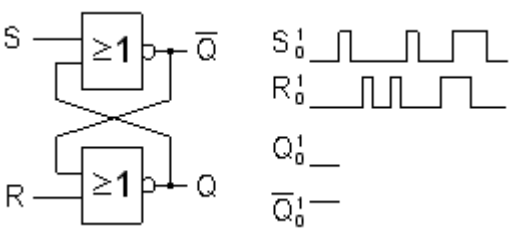




Sekvenskretsar, låskretsar och klockade vippor

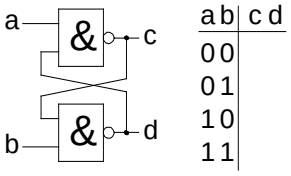
9.1

Komplettera tidsdiagrammet för utsignalerna Q och $\overline{Q}$. Avståndet mellan pulserna är *mycket längre* än grindfördröjningen.
(Ledning, vad är låsande insignal för NOR-grindar)



9.2

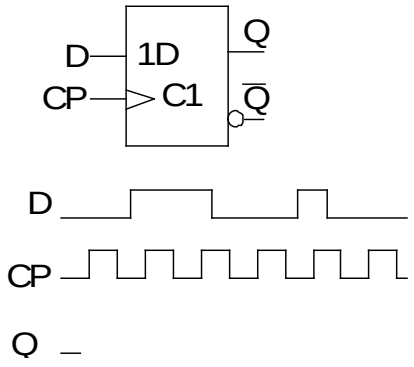
Du känner säkert igen låskretsen till höger. Här har de vanliga beteckningarna bytts ut mot **a b c d**. Fyll i sanningstabellen.



a b	c d
0 0	
0 1	
1 0	
1 1	

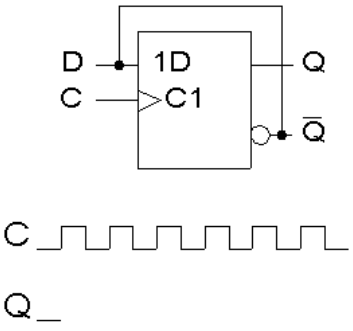
9.3

Rita tidsdiagrammet för utsignalen **Q**, för D-vippan.



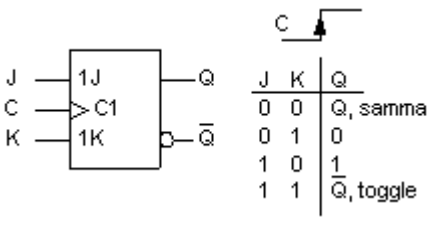
9.4

Rita in Q i detta tidsdiagram.



9.5

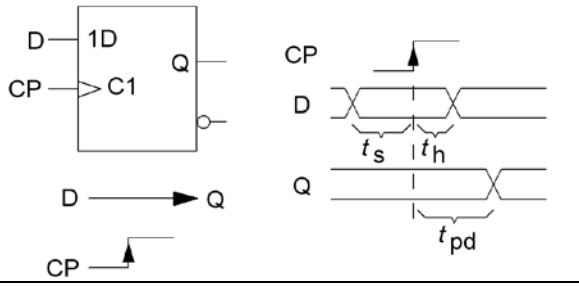
JK-vippan var en äldre typ av ”universalvippa”. Visa hur den kan användas som T-vippa och som D-vippa.



J	K	Q
0	0	Q, samma
0	1	0
1	0	1
1	1	$\overline{Q}$, toggle

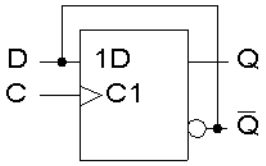
Flip-Flop Timing Parameters.
Vippan laddas med data vid klockpulsens positiva flank, men datat måste vara stabilt tiden t_s före klockpulsens flank och även tiden t_h efter.
Datat återfinns på utgången efter tiden t_{pd}.
(t_{pd} kan vara olika för 0→1 respektive 1→0).

Om dessa tider inte respekteras blir vippans funktion osäker.



9.6

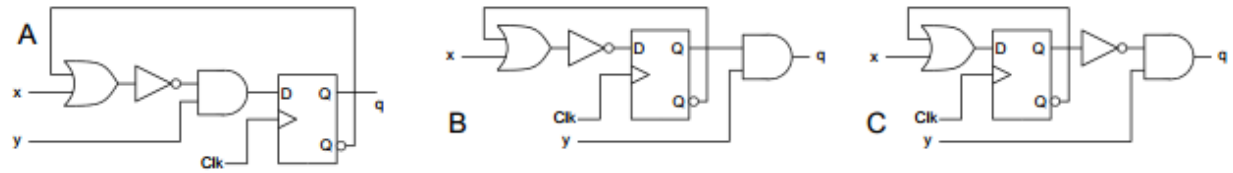
Vilken är den högsta klockfrekvens man kan använda till kretsen i figuren utan att riskera felfunktion?
Antag att
t_s = 20 ns t_h = 5 ns t_{pd} = 30 ns



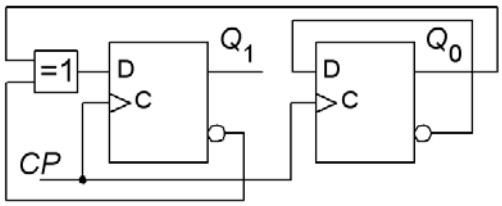
9.7

Figuren visar tre olika tillståndsmaskiner. Ange vilken tillståndsmaskin (A, B eller C) som kan operera vid högst klockfrekvens. Markera den kritiska vägen (den väg som begränsar klockfrekvensen) i denna figur samt beräkna periodtiden för klocksignalen Clk.

$t_{AND} = 0{,}4$ ns, $t_{OR} = 0{,}4$ ns, $t_{NOT} = 0{,}1$ ns, $t_{setup} = 0.3$ ns, $t_{dq} = 0{,}4$ns



9.8

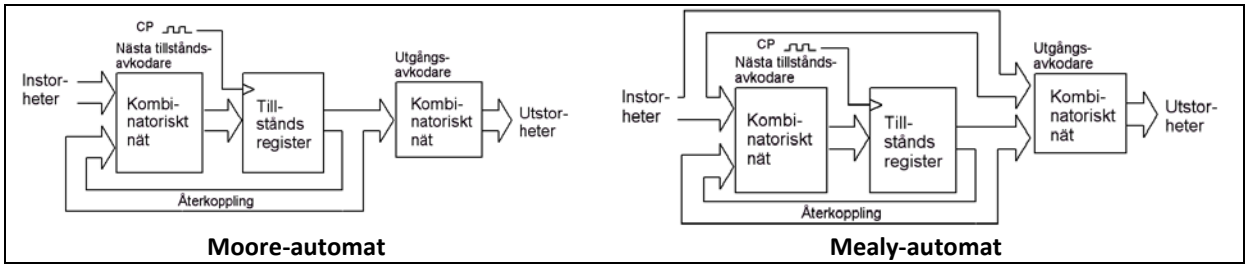


För vipporna i figuren gäller: setuptiden t_{su} = 4 ns, delaytiden på vippans utgångar t_{pdQ} = 3 ns. XOR-grinden har delaytiden t_{pdXOR} = 5 ns.

a) Hur lång tid behöver det vara mellan klockpulserna T_{CP} > ?, för att räknarens funktion skall vara säker?

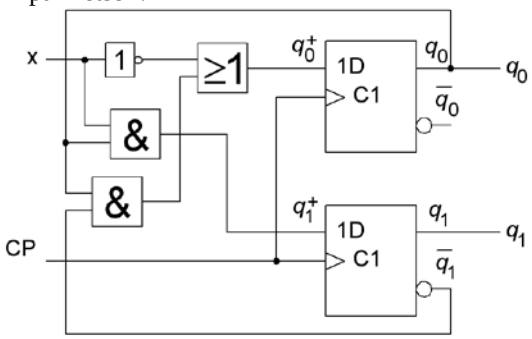
b) Vilket värde på holdtiden t_h måste vipporna ha för att denna krets ska kunna fungera? t_h < ? ns.

Sekvensnät, automater



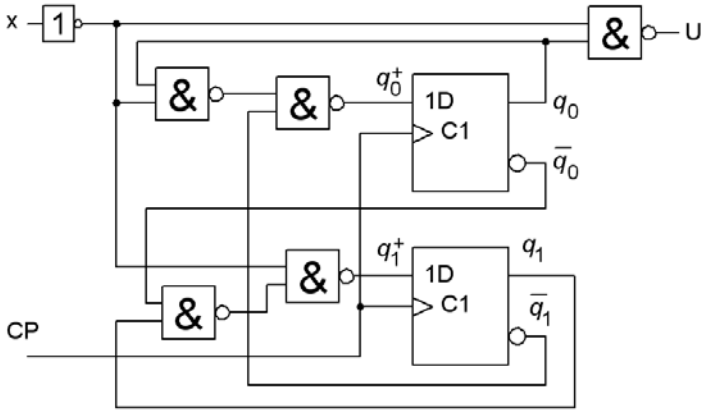
10.1

Bestäm tillståndsdiagram och tillståndstabell för sekvenskretsen. Vilken av modellerna Mealy eller Moore passar in på kretsen?



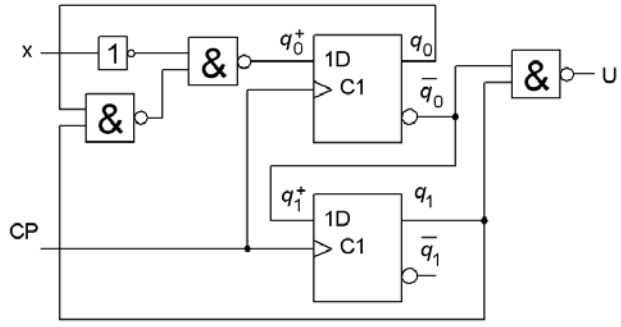
10.2

Bestäm tillståndsdiagram och tillståndstabell för sekvenskretsen. Vilken av modellerna Mealy eller Moore passar på kretsen?



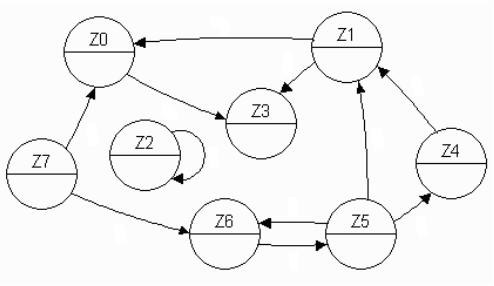
10.3

Bestäm tillståndsdiagram och tillståndstabell för sekvenskretsen. Vilken av modellerna Mealy eller Moore passar in på kretsen?



10.4

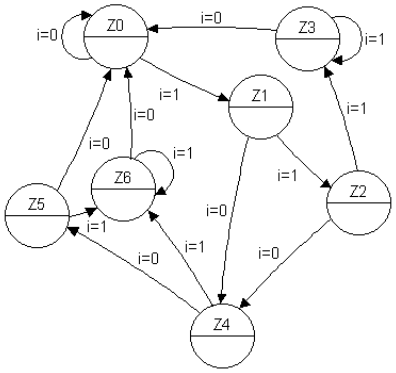
Finns det några stopptillstånd, förlusttillstånd eller isolerade tillstånd i tillståndsdiagrammet till höger?



Stopptillstånd:
Förlusttillstånd:
Isolerade tillstånd:

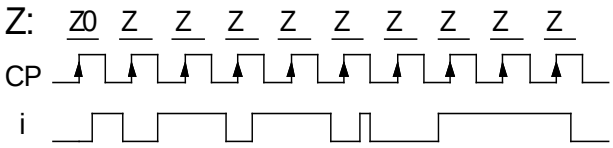
10.5

Till höger finns ett tillståndsdiagram för en Moore-automat. (Den ska upptäcka dubbeltryckning).
En apa råkar få tag i tryckknappen för ingångssignalen **i**, och trycker enligt tidsdiagrammet nedan.



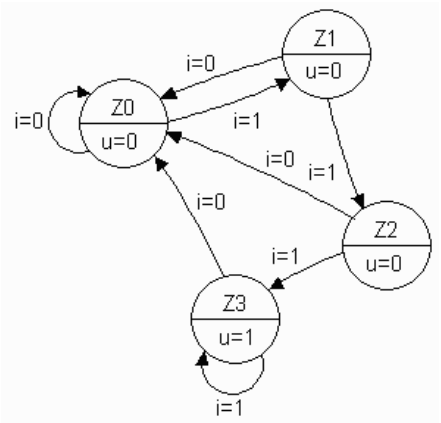
Moore-automaten har vippor som triggas av klockpulsens positiva flank. Antag att den från början står i starttillståndet Z0.

Fyll i vilka tillstånd man hamnar i.

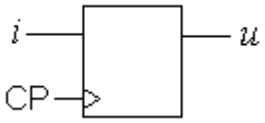


10.6

Konstruera en Moore-automat som kräver att insignalen är lika med ett ($i = 1$) under tre på varandra följande klockpulser för att utsignalen skall bli ett ($u = 1$). Så fort insignalen blir noll under en klockpuls ($i = 0$) skall kretsen återgå till att utsignalen är noll ($u = 0$). Se tillstånds-diagrammet. Välj Graykod för tillståndskodningen. (Z0=00, Z1=01, Z2=11, Z3=10). använd AND-OR grindar.

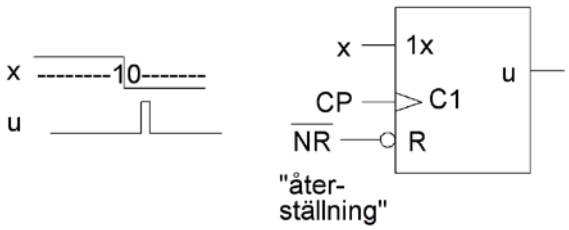


(Kretsen är en säkerhetskrets som skall förhindra ”falsklarm”. Vi kan kalla principen för ”truga kaka” efter den svenska seden att man inte kan tacka nej till en bjuden kaka om den erbjuds tre gånger i rad ...)



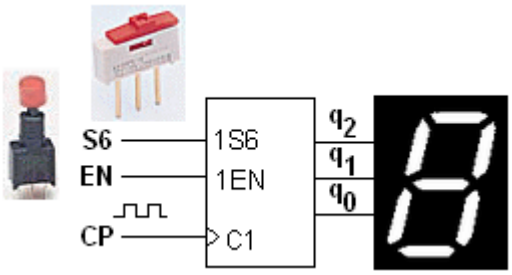
10.7

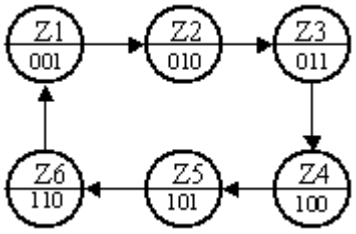
Konstruera ett sekvensnät som upptäcker när insignalen *x* har en övergång mellan 1→0 och då signalerar detta med att *u* = 1 i det nästföljande klock-pulsintervallet för att sedan bli 0 under resten av sekvensen.
Med en asynkron återställningspuls (NR aktiv låg) skall kretsen kunna ”resettas” så att den bevakar insignalen på nytt.



a) Rita tillståndsdiagram för en automat av Moore typ för sekvensnätet.
b) Tag fram de boolska uttrycken för nästa tillståndsavkodaren och utgångsavkodaren för tre olika tillståndskodning:
   1) ”Binärkod”
   2) ”Graykod”
   3) ”One hot” kod
c) Visa hur återställningssignalen NR ansluts till D-vippornas direktverkande PRE och CLR ingångar.

10.8



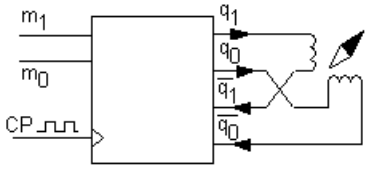


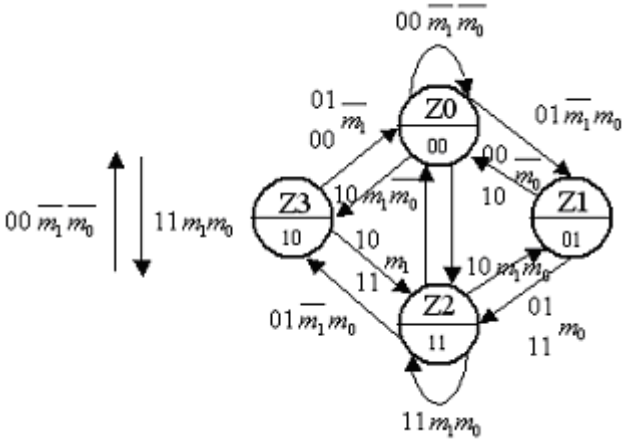
Konstruera en räknare som räknar {… 1, 2, 3, 4, 5, 6, 1 …}. Räknesekvensen, $q_2q_1q_0$, är tänkt att visas på en 7-segmentdisplay, som ett tärningskast.
a) Ange uttrycken för nästatillståndsavkodaren.
b) Komplettera uttrycken med en signal EN som ”fryser” tillståndet för EN = 0 (släppt knapp). Räknaren skall räkna för EN = 1 (nedtryckt knapp).
c) Komplettera uttrycken med en signal S6 som när S6 = 1 tvingar räknaren till tillståndet ”6” (fusk-knappen). S6 är överordnad EN.

10.9

En stegmotor är en digital komponent som drivs med pulser. Stegmotorer brukar anslutas till räknare som räknar Gray-kod. Figurens räknare har dessutom en mode-ingång, m_1m_0.

$m_1m_0 = 00 \rightarrow$ 0-ställning (fix position)
$m_1m_0 = 01 \rightarrow$ upp-räkning (cw)
$m_1m_0 = 10 \rightarrow$ ner-räkning (ccw)
$m_1m_0 = 11 \rightarrow$ 1-ställning (annan fix position)



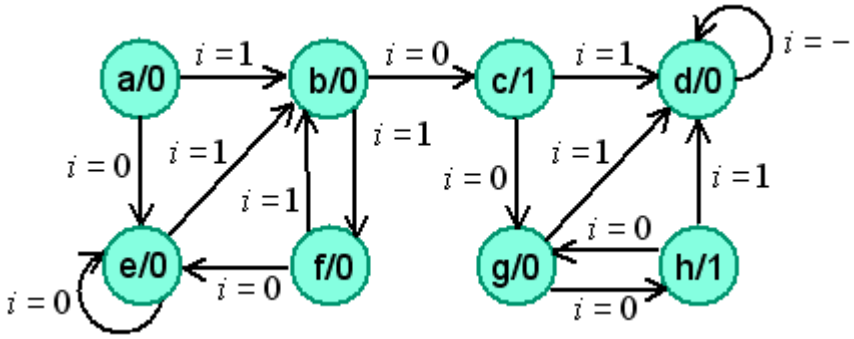


Ibland skriver man boolska vilkor i stället för siffror vid pilarna. I figuren används både vilkor och siffror. Skriv upp de minimerade uttrycken för räknarens nästatillståndsavkodare.

10.10

Detta tillståndsdiagram gäller ett synkront sekvensnät.

- Skriv tillståndstabell.
- Minimera antalet tillstånd.
- Skriv minimerad tillståndstabell
- Rita minimerat tillståndsdiagram.

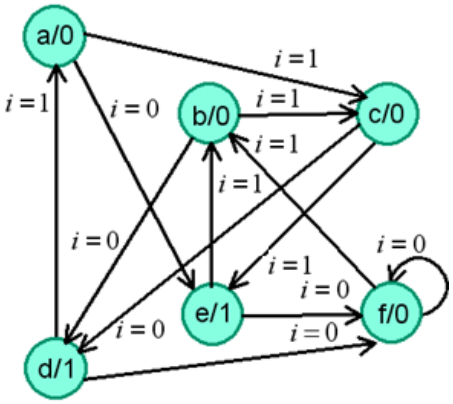


10.11

Detta tillståndsdiagram gäller ett synkront sekvensnät.

- Skriv tillståndstabell. Minimera antalet tillstånd.

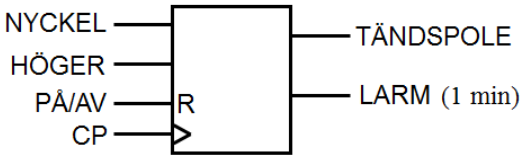
Skriv minimerad tillståndstabell och rita minimerat tillståndsdiagram.



10.12

En teknolog bygger ett biltjuvlarm som en synkron Moore-automat. Larmet får sin ”säkerhet” av att vara hemligt och unikt. För att kunna starta bilen måste man manövrera bilens reglage i följande ordningsföljd:

1) Vrid på startnyckeln (tändning på)
2) Ställ körriktningsvisaren (blinkers) till höger
3) Vrid av startnyckeln (tändning av)
4) Ställ körriktningsvisaren i neutralläge (höger av)
5) Vrid på startnyckeln



Om man vid någon punkt i listan gör ”fel” hamnar man (fastnar) i larmtillståndet. Gör man allt rätt startar bilen (= fastnar man i tändspoletillståndet).

Sekvensnätet har också en ”dold” knapp som går till D-vippornas Reset, och som innebär att larmet kan kopplas PÅ/AV.

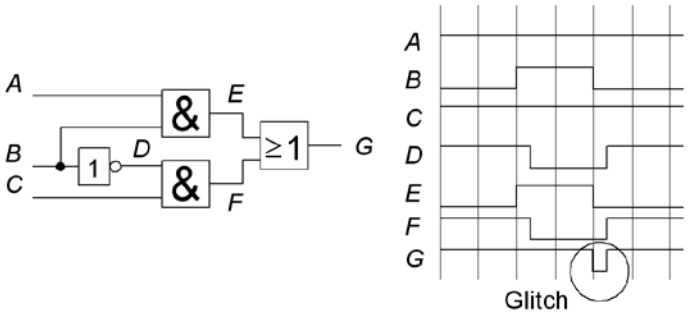
- Rita sekvensnätets tillståndsdiagram.

Asynkrona sekvensnät

11.1

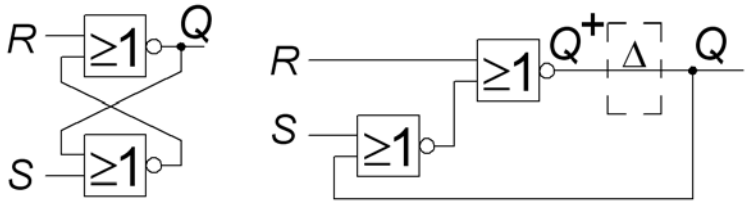
Om signalerna passerar olika många grindsteg på vägen mot utgången kan kortvariga oönskade avvikelser från sanningstabellen uppkomma, så kallade ”glitchar”.

Visa i Karnaughdiagrammet hur man undviker dessa.



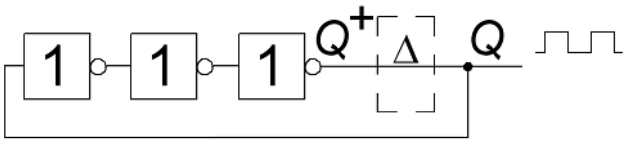
11.2

Till vänster i figuren visas en SR-låskrets som två ”korskopplade” grindnät. Till höger är kretsen omritad som en ”Moore”-automat. Det finns ingen klocksignal, och inget egentligt tillståndsregister. Alla grindfördröjningar som finns i nätet tänks placerade i symbolen Δ mellan Q^+ och Q som får en liknande funktion som D-vippan i ett synkront sekvensnät.
Analysera SR-kretsen på samma sätt som en Moore-automat.



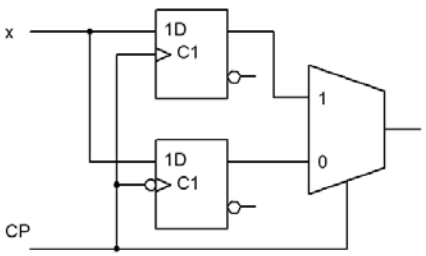
11.3

Visa att det blir ett instabilt nät – en oscillator – om ett udda antal inverterare återkopplas.
Antag att grindfördröjningen t_{pd} är 5 ns och att tre grindar kopplats som i figuren.
Vilket värde får oscillationsfrekvensen?



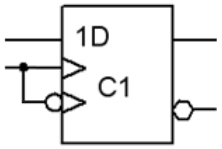
11.4

Analysera följande krets:
Rita ett tillståndsdiagram.
Betrakta kretsen som ett asynkront sekvensnät där klockpulsingången är en av de asynkrona ingångarna.
Vad har kretsen för funktion?



11.5

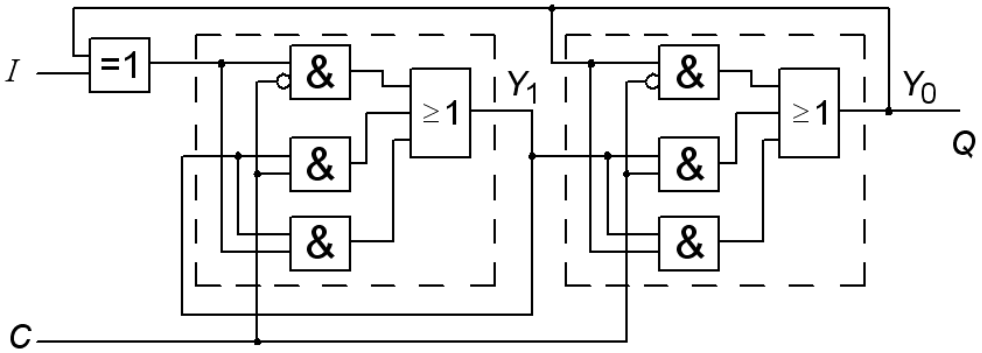
Konstruera en asynkron statemaskin som fungerar som en dubbelflankad D vippa (DETFF), dvs vippan skall ändra värde både på den *positiva* och den *negativa* flanken av klockan.



a) Härled FSMen.
b) Ta fram flödestabellen och minimera den.
c) Tilldela tillstånd (states), överför till Karnaughdiagram och härled de boolska uttrycken.
d) Rita kretsen.

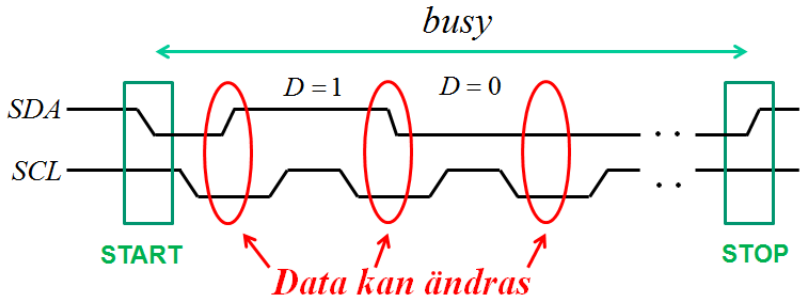
11.6

Analysera följande krets:



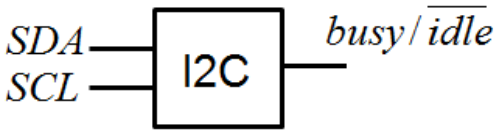
a) Härled de boolska uttrycken för state variablerna Y_1 och Y_0.
b) Härled exitationstabellen. (Ledning: Vilken funktion finns i de två innersta looparna)
c) Härled flödestabell, tilldela symboliska states och rita FSM:en.
d) Identifiera funktionen hos den asynkrona kretsen. Vilken vippa motsvarar den?

11.7



Dataöverföring mellan olika chip inom elektronikutrustningar kan ske med den så kallade I2C-bussen. Den består av två ledningar SDA och SCL. I figuren ovan visas en principbild när ett fåtal bitar överföres, men det vanliga är att mycket större datamängder överförs.
Data *D* får bara ändras när SCL=0.
Positiv och negativ SDA-flank när SCL=1 används som **unika** start och stopp-signaler för dataöverföringen. (Under dataöverföringen förekommer därför *inga* sådana flanker).
(Innan stop-pulsen kan mottagaren ”kvittera” motagningen – i figuren bortses från detta.)

För att kunna studera I2C-dataöverföringen vill man konstruera ett Moore-ekvivalent asynkront sekvensnät som ger utsignalen *busy* = 1 under tiden från start-signalen fram till stopp-signalen. När ingen datakommunikation förekommer är *busy* = 0.



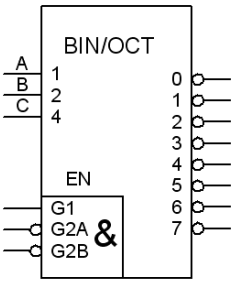
- Ställ upp en primitiv flödestabell
- Minimera flödestabellen
- Välj tillståndskod
- Ställ upp exitationstabellen (motivera att konstruktionen är fri från kritisk kapplöpning)
- Tag fram minimerade booleska uttryck (motivera hazardfrihet)

Avkodning av minnen och I/O-kretsar

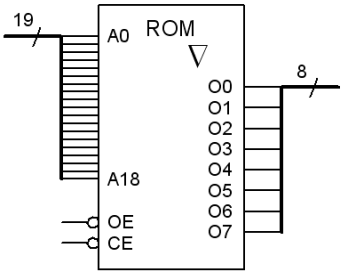
12.1

Ett dynamiskt RAM-minne består av ett antal 256Mbit minneskapslar som är organiserade som 32 M×8.
a) Hur många kapslar behövs för 256M×64?
b) Hur många kapslar behövs för 512M×72? (Vad kan anledningen till den ”underliga” bitbredden ”72” vara?)

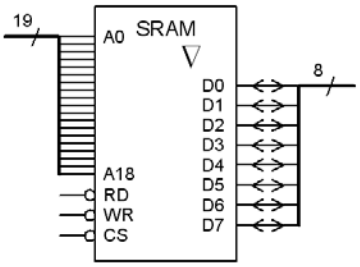
12.2



3:8-avkodare

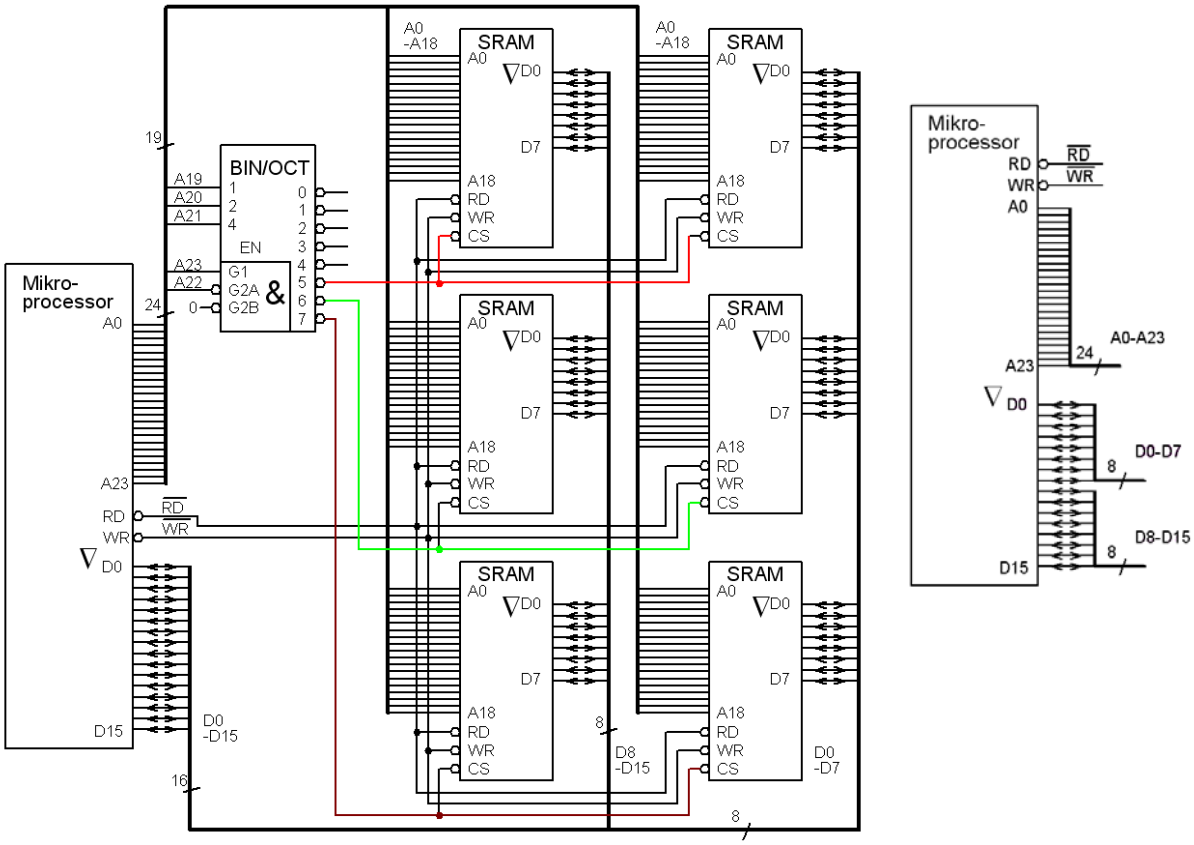


ROM 512k×8



SRAM 512k×8

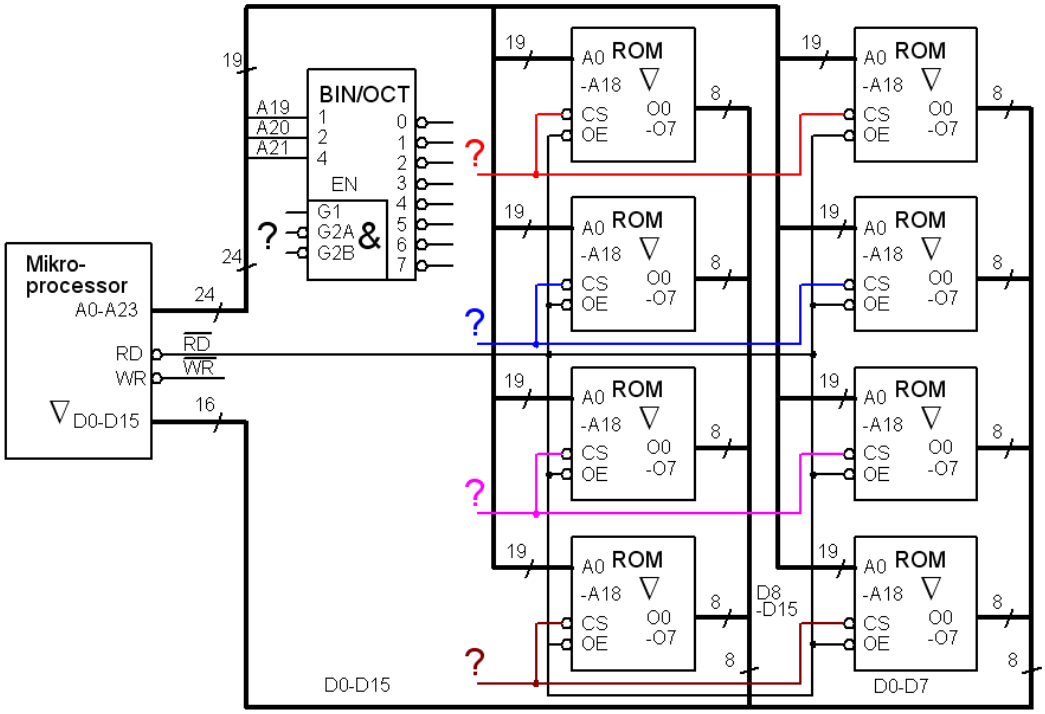
En viss 16 bitars processor kan adressera 24 bitar. Minnesrymden fördelas mellan ROM, SRAM och IO-kretsar.
Adressavkodningen sker med hjälp av en 3:8-avkodare.
a) Hur stort är figurens RAM? Vilket är adressområdet uttryckt i hexadecimala siffror?



b) Hur ändrar man adressområdet till 980000 – AFFFFF ?
c) Ändra till adressområde 480000 – 5FFFFF ?

d) Oftast läser en processor sin första instruktion från adress 0. Då måste det finnas ett läsminne på den adressen. Antag att ROM-minnet är 2M×16 bitar och att adressområdet är 000000 … och framåt. ROM Chip är 512k×8.

- Hur många kapslar behövs?
- Hur skall avkodaren anslutas?
- Hur skall minneskretsarna anslutas?
- Ange adressområdena för avkodarens utgångar med hexadecimala siffror.

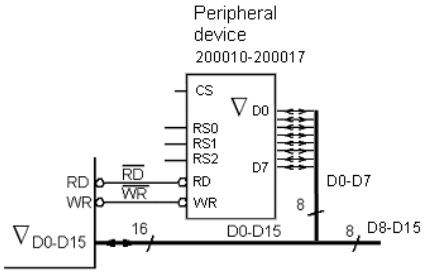


e) Vilket adressområde blir ledigt för SRAM och IO-kretsar?

12.3

Periferienheter, I/O, ansluts ofta till en CPU som om dom vore minneskretsar (fast med bara ett fåtal ”minnesceller”). Ex. en realtidsklock-krets – håller reda på tid och datum. Den styrs/avläses från 8 inbyggda 8-bitars register.

a) Anslut en 8 registers minnesmappad periferienhet (I/O) till en CPU. CPU:n har 16 bitars databuss (vi använd er bara 8), och en 24 bitars adressbuss. Använd en 3:8-avkodare och vid behov grindar. Periferienheten skall kopplas in så att den får adresserna `0x200010` … `0x200017`. (Jämför med föregående uppgift).



b) Vad är ofullständig avkodning?

Lösningar

Talsystem och koder

1.1

Nedanstående decimala tal (med basen 10) är givna. Ange motsvarande binära tal.

a) $9_{10} = (1\cdot2^3+0\cdot2^2+0\cdot2^1+1\cdot2^0) = (8+1) = 1001_2$

b) $12_{10} = (8+4) = 1100_2$ c) $71_{10} = (64+7 = 64+4+2+1) = 1000111_2$

d) $503_{10} = (512 - 9 = 511 - 8) = (111111111_2 - 1000_2) = 111110111_2$

1.2

Omvandla nedanstående binärtal till decimaltal.

a) $101101001_2 = (2^8+2^6+2^5+2^3+2^0 = 256+64+32+8+1) = 361_{10}$

b) $110100.010_2 = (2^5+2^4+2^2 + 2^{-2} = 32+16+4 + 0.25) = 52{,}25_{10}$

1.3

Omvandla nedanstående binärtal (bas=2) till motsvarande oktala tal (bas=8) och hexadecimala tal (bas=16).

a) $01\ 1101_2 = 1D_{16} = (011\ 101_2) = 35_8$

b) $1000\ 1011_2 = 8B_{16} = (010\ 001\ 011_2) = 213_8$

c) $1\ 0011\ 0101_2 = 135_{16} = (100\ 110\ 101_2) = 465_8$

d) $1101\ 1110\ 1001\ 0001_2 = DE91_{16} = (001\ 101\ 111\ 010\ 010\ 001_2) = 157221_8$

e) $10\ 1001.001_2 = 29.2_{16} = (101\ 001\ .\ 001_2) = 51.1_8$

1.4

Omvandla nedanstående hexadecimala tal (bas=16) till motsvarande oktala tal (bas=8).

a) $94D_{16} = (1001\ 0100\ 1101_2 = 100\ 101\ 001\ 101_2) = 4515_8$

b) $9E.7A_{16} = (1001\ 1110\ .\ 0111\ 1010_2 = 010\ 011\ 110\ .\ 011\ 110\ 100_2) = 236.364_8$

1.5

Omvandla det oktala (bas=8) talet 4515_8 till motsvarande hexadecimala tal (bas=16).

$4515_8 = (100\ 101\ 001\ 101_2 = 1001\ 01\ 00\ 1101_2) = 94D_{16}$

1.6

Skriv det hexadecimala (bas=16) talet BAC_{16} på decimal form (bas=10).

$BAC_{16} = (11\cdot16^2+10\cdot16^1+12\cdot16^0 = 11\cdot256+10\cdot16+12\cdot1) = 2988_{10}$

1.7

Vad karakteriserar Gray-koder, och hur kan de konstrueras?

Graykoder har avståndet ”ett” mellan kodorden. Det är aldrig mer än en bit i taget som ändras vid övergångarna från ett kodord till nästa. Om man vill konstruera en N-bitars Gray-kod kan man göra detta ur koden för N-1 bitar. Först följer N-1 bitskoden med "0" som bit N, därefter fortsätter man nästa halva med N-1 koden en gång till men med kodorden i omvänd ordning och med "1" som bit N. Detta är en "speglad binärkod".

Så här gör man 3-bits Graykod från 2-bits Graykod:
00 01 11 10 är en 2-bits Graykod. 000 001 011 010 är koden kompletterad "0" som bit 3. 10 11 01 00 är 2-bits koden i omvänd ordning. 110 111 101 100 är den omvända koden kompletterad med "1" som bit 3.
Sammantaget blir 3-bits Graykoden:
000 001 011 010 110 111 101 100
Detta är inte den enda tänkbara 3-bits Graykoden, en annan möjlig kod för tre bitar är tex.
000 010 011 001 101 111 110 100.
(I allmänhet är det den "speglade binärkoden" man menar när man talar om Graykod).

1.8

Skriv följande tal ”med tecken” med två-komplementsnotation, $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.

a) $-23 = (+23_{10} = 0010111_2 \rightarrow -23_{10} = 1101000_2 + 1_2) = 1101001_2 = 105_{10}$

b) $-1 = (+1_{10} = 0000001_2 \rightarrow -1_{10} = 1111110_2 + 1_2) = 1111111_2 = 127_{10}$

c) +38 = (32_{10}+4_{10}+2_{10}) = 0100110_2 = 38_{10}
d) -64 = (+64_{10} = 1000000_2 är för stort positivt tal! -64_{10} = 0111111_2 + 1_2) = 1000000_2 = 64_{10}

1.9

Skriv följande tal ”med tecken” med ett-komplementsnotation, $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.
a) -23 = (+23_{10} = 0010111_2 → -23_{10} = 1101000_2) = 1101000_2 = 104_{10}
b) -1 → 1111110_2 = 126_{10} c) 38 = 0100110_2 = 38_{10} d) -0 → 1111111_2 = 127_{10}

Digital aritmetik

2.1

Addera för hand följande par binära tal.
a) 110 + 010 b) 1110 + 1001 c) 11 0011.01 + 111.1 d) 0.1101 + 0.1110

a)
```
  1 1
    1 1 0
+   0 1 0
-------
  1 0 0 0
```

b)
```
  1
    1 1 1 0
+   1 0 0 1
-------
  1 0 1 1 1
```

c)
```
      1 1 1
  1 1 0 0 1 1.0 1
+         1 1 1.1
---------------
  1 1 1 0 1 0.1 1
```

d)
```
  1 1
  0.1 1 0 1
+ 0.1 1 1 0
---------
  1.1 0 1 1
```

2.2

Addera eller subtrahera (addition med motsvarande negativa tal) nedanstående tal. Talen skall representeras som binära 4-bitstal (Nibble) på två-komplementform.
a) 1 + 2 b) 4 – 1 c) 7 – 8 d) -3 – 5
-1_{10} = (+1_{10} = 0001_2 → -1_{10} = 1110_2 +1_2) = 1111_2
-8_{10} = (+8_{10} = 1000_2 → -8_{10} = 0111_2 +1_2) = 1000_2
-3_{10} = (+3_{10} = 0011_2 → -3_{10} = 1100_2 +1_2) = 1101_2
-5_{10} = (+5_{10} = 0101_2 → -5_{10} = 1010_2 +1_2) = 1011_2

1+2=3

a)
```
    0 0 0 1  =1
+   0 0 1 0  =2
-----------
    0 0 1 1  =3
```

4-1=3

b)
```
  1 1
    0 1 0 0  =4
+   1 1 1 1  =-1
-----------
  1 0 0 1 1  =3
```

7-8=-1

c)
```
    0 1 1 1  =7
+   1 0 0 0  =-8
-----------
    1 1 1 1  =-1
```

-3-5=-8

d)
```
  1 1 1 1
    1 1 0 1  =-3
+   1 0 1 1  =-5
-----------
  1 1 0 0 0  =-8
```

2.3

Multiplicera för hand följande par av teckenlösa binära tal.
a) 110·010 b) 1110·1001 c) 11 0011.01·111.1 d) 0.1101·0.1110

110·010=(6·2=12)=1100

a)
```
      1 1 0  =6
  ×   0 1 0  =2
-----------
      0 0 0
    1 1 0
+ 0 0 0
-----------
  0 1 1 0 0  =12
```

1110·1001=1111110

b)
```
      1 1 1 0  =14
  ×   1 0 0 1  =9
-----------
      1 1 1 0
    0 0 0 0
  0 0 0 0
+ 1 1 1 0
-----------
  1 1 1 1 1 1 0  =126
```

110011.01·111.1=
=110000000.011

c)
```
  11001101
×     1111
--------
  11001101
 11001101
11001101
+11001101
---------
110000000011
```

0.1101·0.1110=
=0.10110110

d)
```
    1101
×   1110
--------
    0000
   1101
  1101
+ 1101
--------
  10110110
```

2.4

Dividera för hand följande par av teckenlösa binära tal.
a) 110/010 b) 1110/1001
Om divisionen är en **heltalsdivision** blir svaret påuppgift b) heltalet 1.

110/010=(6/2=3)=011

a)
```
      1 1
  ---------
1 0 | 1 1 0
    - 1 0
    -------
        1 0
      - 1 0
      -----
          0
```

1110/1001=(14/9)=1.10…

b)
```
          1. 1 0 0 0 1…
  -------------------
1 0 0 1 | 1 1 1 0
        - 1 0 0 1
        ---------
          1 0 1 0
        - 1 0 0 1
        ---------
            1 0 0 0 0
          - 1 0 0 1
          ---------
              1 1 1…
```

2.5

Antag att ett 32-bitars flyttal lagras i ett register som: $40C80000_{16}$ vilket reellt decimaltal är det?
32-bitars flyttal lagras normerade som ”1.” En ”teckenbit”, 8 bitar för 2-exponenten (utryckt som ett exess-127 tal), 23 bitar för binalerna. Eftersom alla tal börjar med ”1.” så behöver denna ”1.” inte lagras, utan underförstås.

Exess-127 innebär att talet 127 läggs till alla exponenter, som därmed alltid lagras som positiva tal. Detta har fördelen att flyttalen kan storlekssorteras som om de vore heltal!

IEEE 32 bit flyttal

```
s  eeeeeeee fffffffffffffffffffffff
31 30    23 22                    0
```

Vad blir:

```
   4   0   C   8   0   0   0   0
01000000110010000000000000000000

0 10000001 10010000000000000000000

+  129-127    1 + 0.5+0.0625
```

$+1{,}5625 \cdot 2^2 = +6{,}25$

2.6

a)

Floating point format

Sign b_3

Significand $1.b_0$

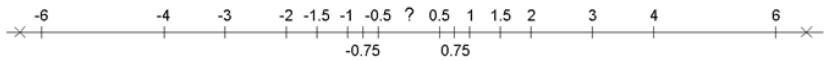
$1.0_2 = 1.0_{10}$
$1.1_2 = 1.5_{10}$

Exponent b_2b_1-1 (exess1)

$2^{00-1} = 0.5$
$2^{01-1} = 1$
$2^{10-1} = 2$
$2^{11-1} = 4$

+ 1.0 · 0.5 = +0.5	- 1.0 · 0.5 = -0.5
+ 1.0 · 1 = +1	- 1.0 · 1 = -1
+ 1.0 · 2 = +2	- 1.0 · 2 = -2
+ 1.0 · 4 = +4	- 1.0 · 4 = -4
+ 1.5 · 0.5 = +0.75	- 1.5 · 0.5 = -0.75
+ 1.5 · 1 = +1.5	- 1.5 · 1 = -1.5
+ 1.5 · 2 = +3	- 1.5 · 2 = -3
+ 1.5 · 4 = +6	- 1.5 · 4 = -6

Floating point number line!



b) Det största kvantiseringsfelet inträffar mellan 4 och 6 (eller mellan -6 och -4). Felet blir (6-4)/2 = 1.
c) Någon representation för talet 0 finns inte, man kan välja att använda det minsta positiva och minsta negativa talen som +0 och -0. Så görs i IEEE-standarden.

2.7

$$[b_5 b_4 b_3 b_2 b_1 b_0] = (-1^{b_5}) \cdot (1.b_2 b_1 b_0) \cdot (2^{b_4 b_3 - 1})$$

Significand $1.b_2b_1b_0$	**Exponent b_4b_3-1 (exess1)**
$1.000_2 = 1.000_{10}$	00-1 = -1 $2^{00-1} = 0.5$
$1.001_2 = 1.125_{10}$	01-1 = 0 $2^{01-1} = 1$
$1.010_2 = 1.25_{10}$	10-1 = 1 $2^{10-1} = 2$
$1.011_2 = 1.375_{10}$	11-1 = 2 $2^{11-1} = 4$
$1.100_2 = 1.5_{10}$	
$1.101_2 = 1.625_{10}$	
$1.110_2 = 1.75_{10}$	
$1.111_2 = 1.875_{10}$	

- $0.25_{10} = 0.01_2 = 1.0_2 \cdot 2^{-2}$ exponent -2 *not* representable
- $0.8125_{10} = 0.1101_2 = 1.101_2 \cdot 2^{-1}$ representable
- $-1{,}375_{10} = -1.011_2 = -1.011_2 \cdot 2^0$ representable

- $4.25_{10} = 100.01_2 = 1.0001_2 \cdot 2^2$ significand 1.0001 *not* representable
- $7.5_{10} = 111.1_2 = 1.111_2 \cdot 2^2$ representable

Flyttalsaddition

Algorithm:
1. Check for zeroes
2. Align significands
3. Add/Sub significands
4. Normalize result

$a = 001111_2 \rightarrow 0\ 01\ 111 \rightarrow +\ \mathbf{01}\ 1.111$
$b = 010010_2 \rightarrow 0\ 10\ 010 \rightarrow +\ \mathbf{10}\ 1.010$

Align significands. **10** > **01** shift smaller number *a* to right to get *same* exponent:
Significand 0.1111 exponent **10**

Add significands:

```
  0.1111
+ 1.010
 10.0011
```

Normalize result:

10.0011 exp **10** → 1.00011 exp **11**
Rounding: 1.0001**1** ~ 1.001

Result: 0 11 001

$a = 1.875$ $b = 2.5$ $a+b = 4.5$ (4.375)

Flyttals multiplikation (enklare än additionen!)

Simpler than addition!

Algorithm:
1. Check for zeroes
2. Add exponents and subtract *Bias*
3. Multiply significands
4. Normalize

Exponents: 01 10 Bias = 1
exp = 01+10 -1 = 10

Multiply significands

```
    1.010
  * 1.111
  -------
    1 010
   10 10
  101 0
+ 1 010
  -------
  10.010 110
```

Normalize result:

10.010110 exp **10** → 1.0010110 exp **11**

Rounding: 1.001**0**110 ~ 1.001

Result: 0 11 001

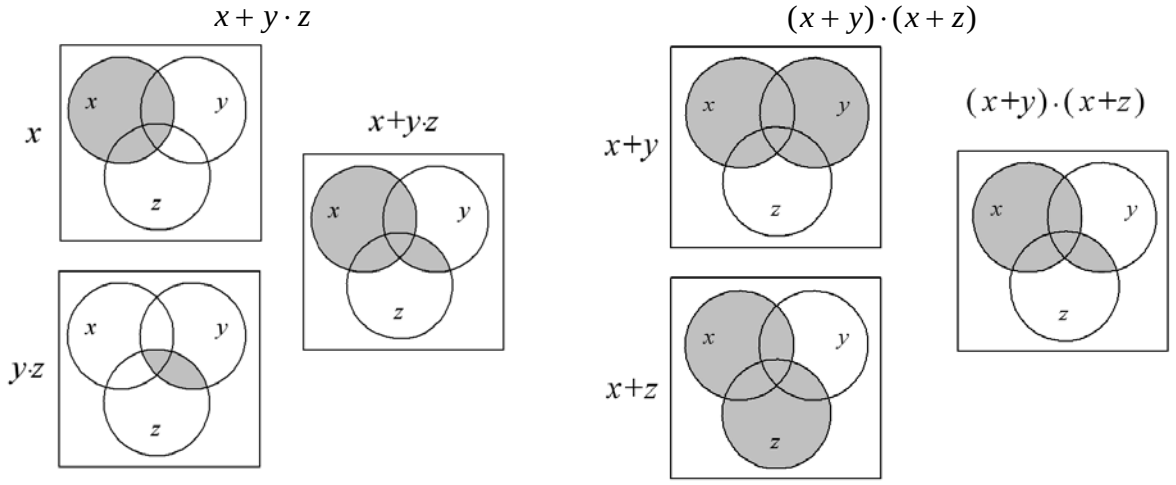
$a = 1.875$ $b = 2.5$ $a*b = 4.5$ (4.6875)

Mängdräkning och kubteori

3.1

Bevis av den distributiva lagen med hjälp av Venn-diagram.

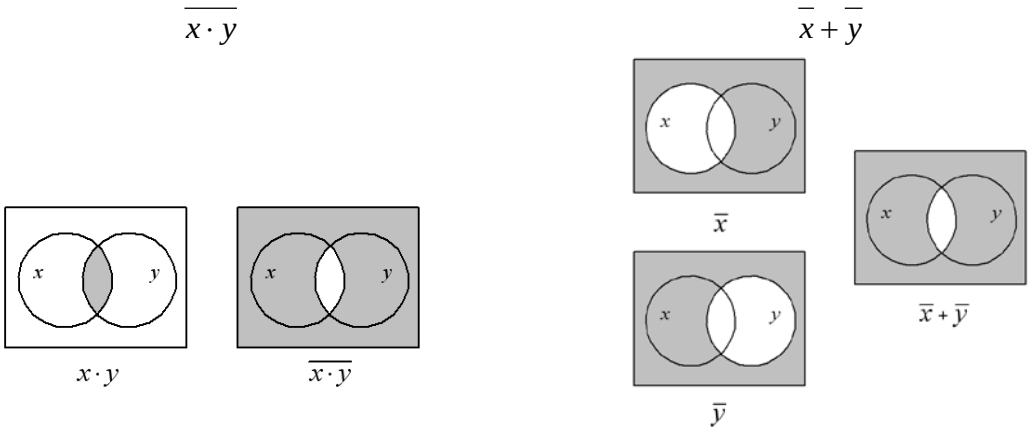
$x + y \cdot z = (x + y) \cdot (x + z)$



3.2

Bevis av De Morgans lag med hjälp av Venn-diagram.

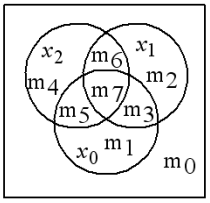
$\overline{x \cdot y} = \overline{x} + \overline{y}$



3.3

a) Mintermernas placering i ett tre variablers Venn-diagram.

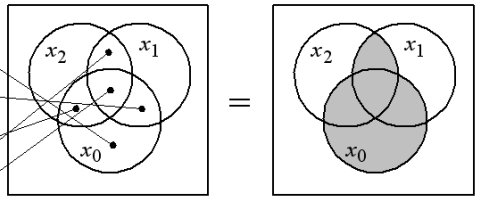
x_2	x_1	x_0	
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7



b)

$f = \bar{x}_2\bar{x}_1x_0 + \bar{x}_2x_1x_0 + x_2\bar{x}_1x_0 + x_2x_1\bar{x}_0 + x_2x_1x_0$

x_2	x_1	x_0	
0	0	0	0
0	0	1	1 $\bar{x}_2\bar{x}_1x_0$
0	1	0	0
0	1	1	1 $\bar{x}_2x_1x_0$
1	0	0	0
1	0	1	1 $x_2\bar{x}_1x_0$
1	1	0	1 $x_2x_1\bar{x}_0$
1	1	1	1 $x_2x_1x_0$



$f = x_2x_1 + x_0$

Venndiagram-metoden visar tydligt de boolska sambanden, men är svår att använda vid fler än tre variabler. Den är opraktisk att göra om till en datoralgoritm.

3.4

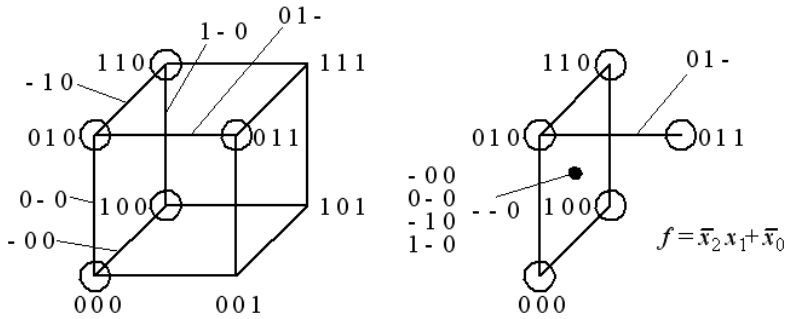
a) Representera följande funktion av tre variabler, som en 3-dimensionell kub med Gray-kodade hörn.

$$f(x_2, x_1, x_0) = \sum m(0,2,3,4,6) = \bar{x}_2\bar{x}_1\bar{x}_0 + \bar{x}_2x_1\bar{x}_0 + \bar{x}_2x_1x_0 + x_2\bar{x}_1\bar{x}_0 + x_2x_1\bar{x}_0$$

b) Använd kuben för att förenkla funktionen.

$f = \bar{x}_2\bar{x}_1\bar{x}_0 + \bar{x}_2x_1\bar{x}_0 + \bar{x}_2x_1x_0 + x_2\bar{x}_1\bar{x}_0 + x_2x_1\bar{x}_0$

x_2	x_1	x_0	
0	0	0	1 $\bar{x}_2\bar{x}_1\bar{x}_0$
0	0	1	0
0	1	0	1 $\bar{x}_2x_1\bar{x}_0$
0	1	1	1 $\bar{x}_2x_1x_0$
1	0	0	1 $x_2\bar{x}_1\bar{x}_0$
1	0	1	0
1	1	0	1 $x_2x_1\bar{x}_0$
1	1	1	0



Den kubiska representationen är svår att åskådliggöra för fler än 3 dimensioner, men minimerings-metoden kan enkelt definieras för valfritt antal variabler och dimensioner och sedan ligga till grund för dator-algoritmer.

Boolsk algebra och grindar

4.1

a) $f = a \cdot \bar{c} \cdot d + a \cdot d \quad = a \cdot d \cdot (\bar{c} + 1) = a \cdot d$

b) $f = a \cdot (\bar{b} + \bar{a} \cdot c + a \cdot b) \quad = a \cdot \bar{b} + a \cdot \bar{a} \cdot c + a \cdot a \cdot b = a \cdot \bar{b} + 0 + a \cdot b = a \cdot (\bar{b} + b) = a$

c) $f = a + \bar{b} + \bar{a} \cdot b + \bar{c} \quad = (1 \cdot \bar{b} + \bar{a} \cdot b) \quad + a + \bar{c} = \{konsensus\} =$

$= (1 \cdot \bar{b} + \bar{a} \cdot b + 1 \cdot \bar{a}) \quad + a + \bar{c} = ... \bar{a} + a \, ... = 1$

d) $f = (a + b \cdot \bar{c}) \cdot (\bar{a} \cdot \bar{b} + c) \quad = a \cdot \bar{a} \cdot \bar{b} + a \cdot c + \bar{a} \cdot b \cdot \bar{b} \cdot \bar{c} + b \cdot c \cdot \bar{c} = 0 + a \cdot c + 0 + 0 = a \cdot c$

e) $f = (a + \bar{b}) \cdot (\bar{a} + b) \cdot (a + b) \quad = (a \cdot \bar{a} + a \cdot b + \bar{a} \cdot \bar{b} + b \cdot \bar{b}) \cdot (a + b) =$

$(0 + a \cdot b + \bar{a} \cdot \bar{b} + 0) \cdot (a + b) = a \cdot a \cdot b + a \cdot \bar{a} \cdot \bar{b} + a \cdot b \cdot b + \bar{a} \cdot \bar{b} \cdot b = a \cdot b + a \cdot b = a \cdot b$

f) $f = \bar{a} \cdot \bar{b} \cdot c + a \cdot b \cdot c + \bar{a} \cdot b \cdot c \quad = c \cdot (\bar{a} \cdot \bar{b} + a \cdot b + \bar{a} \cdot b) = c \cdot (\bar{a} \cdot \bar{b} + b \cdot (a + \bar{a})) =$

$= c \cdot (\bar{a} \cdot \bar{b} + 1 \cdot b) = \{konsensus\} = c \cdot (\bar{a} \cdot \bar{b} + 1 \cdot b + \bar{a} \cdot 1) = c \cdot (\bar{a} \cdot \bar{b} + b + \bar{a}) = c \cdot (\bar{a} \cdot (\bar{b} + 1) + b) =$

$= c \cdot (\bar{a} + b) = \bar{a} \cdot c + b \cdot c$

g) $f = \bar{a} \cdot b \cdot \bar{c} + \bar{a} \cdot b \cdot \bar{d} + c \cdot d \quad = \bar{a} \cdot b \cdot (\bar{c} + \bar{d}) + c \cdot d = \bar{a} \cdot b \cdot \overline{\overline{(\bar{c} + \bar{d})}} + c \cdot d =$

$\bar{a} \cdot b \cdot \overline{c \cdot d} + c \cdot d = \{x + \bar{x}y = x + y\} = \bar{a} \cdot b + c \cdot d$

h) $f = a + \overline{\overline{(\bar{a} \cdot \bar{b})}} \quad = \{deMorgan\} = a + \bar{\bar{a}} + \bar{\bar{b}} = a + b$

i) $f = \overline{\bar{a} + \overline{\bar{a} \cdot b + c}} \quad = \{deMorgan\} = a \cdot (\bar{a} \cdot b + c) = a \cdot \bar{a} \cdot b + a \cdot c = ac$

4.2

Bevisa algebraiskt att följande samband är giltiga.

a) $(\overline{x_3} x_2 + 1 + x_3 \overline{x_2}) x_1 + x_2 x_1 + x_2 + \overline{x_1} = 1$ **VL:** $(... + 1) x_1 = x_1 \quad x_1 + \overline{x_1} + ... = 1$

b) $\overline{x_3} x_2 x_1 + \overline{(x_3 + \overline{x_1})} = \overline{x_3} x_1$ **VL:** $\overline{x_3} x_2 x_1 + \overline{(x_3 + \overline{x_1})} = \overline{x_3} x_2 x_1 + \overline{x_3}\, x_1 = \overline{x_3}\, x_1 (x_2 + 1) = \overline{x_3}\, x_1$

c) $\overline{(x_2 + x_1)}\, \overline{x_2}\, \overline{x_1} + x_3 = \overline{x_2}\, \overline{x_1} + x_3$ **VL:** $\overline{(x_2 + x_1)}\, \overline{x_2}\, \overline{x_1} + x_3 = \overline{x_2}\, \overline{x_1}\, \overline{x_2}\, \overline{x_1} + x_3 = \overline{x_2}\, \overline{x_1} + x_3$

d) $\overline{x_2 + x_1} + \overline{x_2}\, \overline{x_1} + x_3 = \overline{x_2}\, \overline{x_1} + x_3$ **VL:** $\overline{x_2 + x_1} + \overline{x_2}\, \overline{x_1} + x_3 = \overline{x_2}\, \overline{x_1} + \overline{x_2}\, \overline{x_1} + x_3 = \overline{x_2}\, \overline{x_1} + x_3$

4.3

Förenkla nedanstående tre uttryck så långt som möjligt.

a) $(x + y)(\bar{x} + z) = x\bar{x} + y\bar{x} + xz + yz = \bar{x}y + xz + yz = \{konsensus\ ta\ bort\ \ yz\} = \bar{x}y + xz$

b) $(x + \bar{y} + xy)(x + \bar{y})\bar{x}y = (x + xy + x\bar{y} + \bar{y})\bar{x}y = 0 + 0 + 0 + 0$

c) $x(1 + \bar{x}y) + \bar{x} = x(1) + \bar{x} = 1$

4.4

Förenkla nedanstående uttryck så långt som möjligt.

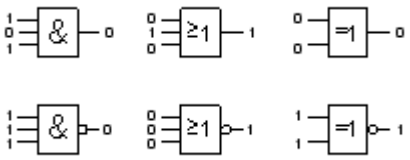
$\overline{(a + b + \bar{c})(a + \bar{b} + \bar{c})(\bar{a} + \overline{\bar{b}\bar{c}} + b\bar{c})} = \overline{(a + b + \bar{c})(a + \bar{b} + \bar{c})(\bar{a} + b + \bar{c} + b\bar{c})} =$

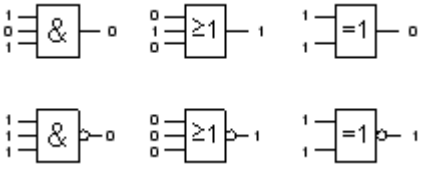
$= \overline{(a + b + \bar{c})(a + \bar{b} + \bar{c})(\bar{a} + b + \bar{c})} =$

$\overline{(a + b + \bar{c})} + \overline{(a + \bar{b} + \bar{c})} + \overline{(\bar{a} + b + \bar{c})} = \bar{a}\bar{b}c + \bar{a}bc + a\bar{b}c = \bar{a}\bar{b}c + \bar{a}\bar{\bar{b}}c + \bar{a}bc + a\bar{b}c =$

$= \bar{a}c(\bar{b} + b) + \bar{b}c(a + \bar{a}) = \bar{a}c + \bar{b}c$

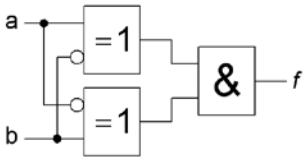
4.5

a) 

b) 

4.6

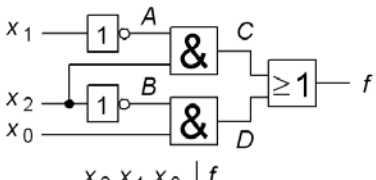
Förenkla *f*(*a*,*b*) som realiseras av figurens grindnät, så långt som möjligt, och ange funktionens namn.



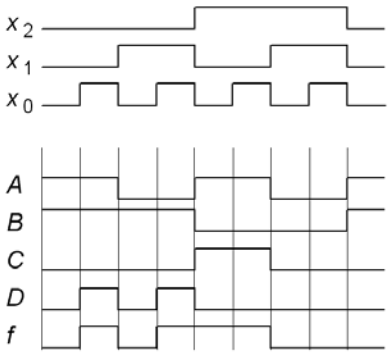
$$(a \oplus \bar{b})(b \oplus \bar{a}) = (a\bar{\bar{b}} + \bar{a}\bar{b})(b\bar{\bar{a}} + \bar{b}\bar{a}) = (ab + \bar{a}\bar{b})(ab + \bar{a}\bar{b}) = ab + \bar{a}\,\bar{b}$$

Det blir en XNOR –funktion.

4.7

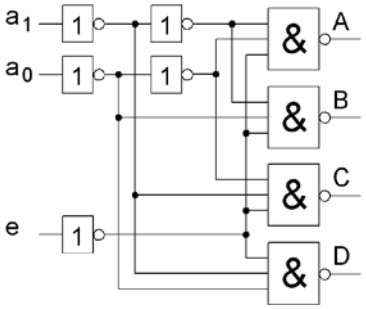


x_2	x_1	x_0	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



4.8

Ange de logiska uttrycken för *A*, *B*, *C* och *D*.



$$A = \bar{\bar{a}_1}\,\bar{\bar{a}_0}\,\bar{e} = \bar{a}_1 + \bar{a}_0 + e$$

$$B = \bar{\bar{a}_1}\,\bar{a}_0 e = \bar{a}_1 + a_0 + e$$

$$C = \bar{a}_1\,\bar{\bar{a}_0}\,\bar{e} = a_1 + \bar{a}_0 + e$$

$$D = \bar{\bar{a}_1}\,\bar{\bar{a}_0}\,\bar{\bar{e}} = a_1 + a_0 + e$$

4.9

Förenkla det sammansatta uttrycken nedan så långt som möjligt.

a)

$$XOR - \text{funktionen} \quad a \oplus b = a\bar{b} + \bar{a}b$$

$$x_2 \oplus x_1 \oplus x_1x_2 = (x_2\bar{x_1} + \bar{x_2}x_1)\overline{\bar{x_1}x_2} + \overline{(x_2\bar{x_1} + \bar{x_2}x_1)}\,x_1x_2 =$$

$$= (x_2\bar{x_1} + \bar{x_2}x_1)(\bar{x_1} + \bar{x_2}) + \overline{x_2\bar{x_1}}\;\overline{\bar{x_2}x_1}\;x_1x_2 = (x_2\bar{x_1} + 0 + 0 + \bar{x_2}x_1) + (\bar{x_2} + x_1)(x_2 + \bar{x_1})x_1x_2 =$$

$$= +(0 + x_1x_2 + \bar{x_2}\,\bar{x_2} + 0)x_1x_2 = \bar{x_1} + \bar{x_2}x_1 + x_1x_2 = x_2\bar{x_1} + \bar{x_2}x_1 + x_1x_2 + x_1x_2 =$$

$$= x_2(\bar{x_1} + x_1) + x_1(\bar{x_2} + x_2) = x_2 + x_1$$

b)

$$x_2x_1 \oplus \overline{(x_2 + x_1)} = x_2x_1(x_2 + x_1) + \overline{x_2x_1}\,\overline{(x_2 + x_1)} = x_2x_1 + (\bar{x_2} + \bar{x_1})\bar{x_2}\,\bar{x_1} =$$

$$= x_2x_1 + \bar{x_2}\,\bar{x_1}$$

4.10

Visa att

a)

$$\overline{x_2 \oplus x_1} = \overline{x_2} \oplus x_1 = x_2 \oplus \overline{x_1}$$

Denna gång prövar vi att bevisa sambanden med så kallad ”perfekt induktion”. Det innebär att man direkt sätter in alla fyra kombinationerna av de två variablerna i de olika uttrycken. Om utrycken har samma sanningstabell så är dom ekvivalenta. När variablerna är få, kan denna icke algebraiska metod användas.

x_2	x_1	$\overline{x_2}$	$\overline{x_1}$	$\overline{x_2 \oplus x_1}$	$\overline{x_2} \oplus x_1$	$x_2 \oplus \overline{x_1}$
0	0	1	1	1	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	1	1

b)

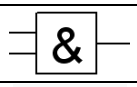
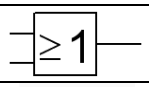
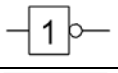
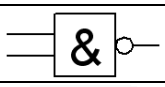
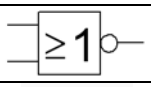
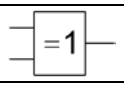
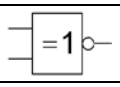
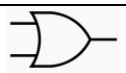
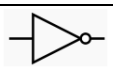
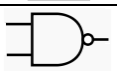
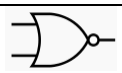
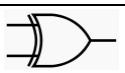
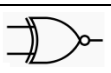
$$x_2 \oplus x_1 = \overline{x_2} \oplus \overline{x_1}$$

$$VL: \quad x_2 \oplus x_1 = x_2\overline{x_1} + \overline{x_2}x_1$$

$$HL: \quad \overline{x_2} \oplus \overline{x_1} = \overline{x_2}\,\overline{\overline{x_1}} + \overline{\overline{x_2}}\,\overline{x_1} = x_2\overline{x_1} + \overline{x_2}x_1 \quad VL = HL$$

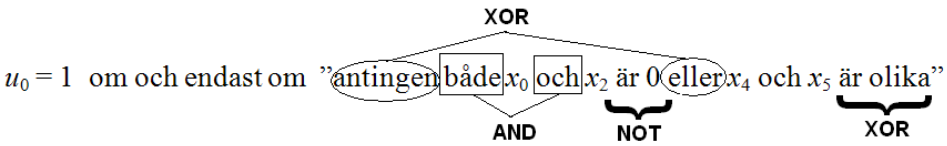
4.11

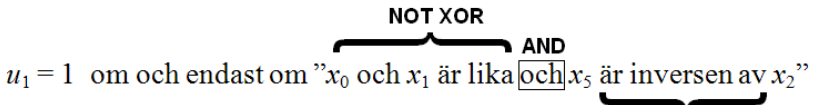
Här visas de Europeiska grindsymbolerna. Rita de motsvarande amerikanska grindsymbolerna.

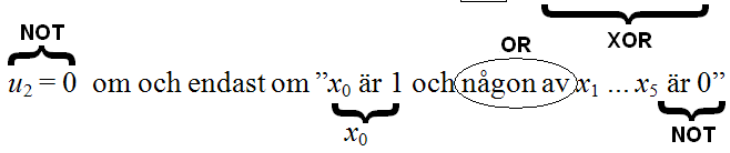
AND	OR	NOT	NAND	NOR	XOR	XNOR
						
						

4.12

Från text till Booleskt uttryck.







$$u_0 = \overline{x_0} \cdot \overline{x_2} \oplus (x_4 \oplus x_5)$$

$$u_1 = \overline{x_0 \oplus x_1} \cdot (x_5 \oplus x_2)$$

$$\overline{u_2} = x_0 \cdot (\overline{x_1} + \overline{x_2} + \overline{x_3} + \overline{x_4} + \overline{x_5}) \quad \Rightarrow \quad u_2 = \overline{x_0 \cdot (\overline{x_1} + \overline{x_2} + \overline{x_3} + \overline{x_4} + \overline{x_5})} = \overline{x_0} + x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5$$

Sanningstabellen, SP och PS –form, fullständig logik

5.1

Kontakter avbildas alltid i opåverkat läge. För att få lampan att lysa ska man samtidigt trycka på siffrorna ”4” och ”8” det vill säga kontakt *d* och *h*. Observera att man *inte* får trycka ned någon annan kontakt! Den logiska funktionen (lampan lyser) blir:

$$f = \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot d \cdot \bar{e} \cdot \bar{\bar{f}} \cdot \bar{g} \cdot h \cdot \bar{i} \cdot \bar{k}$$

Kodlåset är en avkodare, det avkodar en enda minterm i saningstabellen.

5.2

a b c	*f*	*a b c*	*f*
0 0 0	1	100	1
0 0 1	0	1 0 1	1
0 1 0	0	1 1 0	0
0 1 1	0	1 1 1	1

Funktionen på SP-normalform:

$$f = \bar{a}\bar{b}\bar{c} + a\bar{b}\bar{c} + a\bar{b}c + abc$$

Funktionen på PS-normalform:

$$f = (a + b + \bar{c}) \cdot (a + \bar{b} + c) \cdot (a + \bar{b} + \bar{c}) \cdot (\bar{a} + \bar{b} + c)$$

5.3

$$f(x, y, z) = x\bar{y} + y\bar{z} + \bar{x}z \quad = x\bar{y}(z + \bar{z}) + (x + \bar{x})y\bar{z} + \bar{x}(y + \bar{y})z =$$

$$= x\bar{y}z + x\bar{y}\bar{z} + xy\bar{z} + \bar{x}y\bar{z} + \bar{x}yz + \bar{x}\bar{y}z$$

$$\Rightarrow \quad f(x, y, z) = \sum m(001, 010, 011, 100, 101, 110) = \sum m(1, 2, 3, 4, 5, 6)$$

$$\Rightarrow \quad f(x, y, z) = \prod M(0,7) = (\bar{x} + \bar{y} + \bar{z})(x + y + z)$$

5.4

$$f(x, y, z) = (x + \bar{y})\left(xyz + \bar{y}(x + z)\right) + xy\bar{z}(x + \bar{x}y) \quad = (x + \bar{y})\left(xyz + \bar{y}(x + z)\right) + xy\bar{z}(x + y) =$$

$$= (x + \bar{y})\left(xyz + \bar{y}(x + z)\right) + xy\bar{z} =$$

$$= xyz + x\bar{y} + \bar{x}yz + xy\bar{y}z + \bar{x}\bar{y} + \bar{y}z + xy\bar{z} =$$

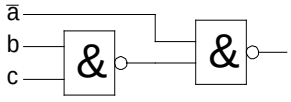
$$= xyz + x\bar{y}(z + \bar{z}) + \bar{x}yz + \bar{x}\bar{y}(z + \bar{z}) + (x + \bar{x})\bar{y}z + xy\bar{z} =$$

$$= xyz + x\bar{y}z + x\bar{y}\bar{z} + \bar{x}yz + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}z + xy\bar{z} \quad = \bar{x}\bar{y}z + \bar{x}y\bar{z} + \bar{x}\bar{y}z + xy\bar{z} + xyz$$

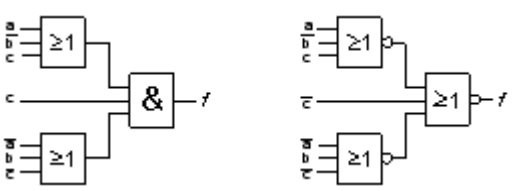
$$\Rightarrow \quad f(x, y, z) = \sum m(001, 100, 101, 110, 111) = \sum m(1, 4, 5, 6, 7)$$

$$\Rightarrow \quad f(x, y, z) = \prod M(0, 2, 3) \quad = (x + y + z)(x + \bar{y} + z)(x + \bar{y} + \bar{z})$$

5.5



5.6



5.7

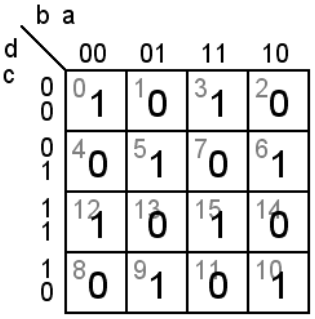
a) Paritetskrets för jämn paritet, antalet ettor skall vara ett jämnt tal (0, 2, eller 4) för ”1” på utgången.

	d	*c*	*b*	*a*	*J*
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1

Hälften av sanningstabellens rader skall vara “1”. Denna funktion går *inte* att minimera, utan alla 8 mintermerna behöver vara med i SP-formen!

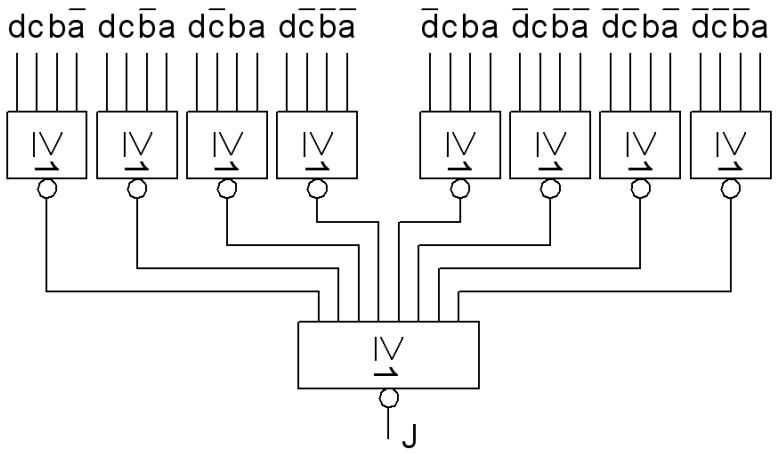
$$J = \bar{d}\bar{c}\bar{b}\bar{a} + \bar{d}\bar{c}bc + \bar{d}c\bar{b}a + \bar{d}cb\bar{a} + d\bar{c}\bar{b}a + d\bar{c}b\bar{a} + dc\bar{b}\bar{a} + dcba$$

(Den som redan nu känner till Karnaugh-diagrammet kan i ett sådant direkt se att inga ”hoptagningar” låter sig göras.)



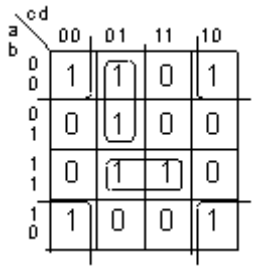
b) Med NOR-grindar lämpar sig PS-formen bättre.

$$J = (d + c + b + \bar{a})(d + c + \bar{b} + a)(d + \bar{c} + b + a)(d + \bar{c} + \bar{b} + \bar{a}) \cdot$$
$$\cdot (\bar{d} + c + b + a)(\bar{d} + c + \bar{b} + \bar{a})(\bar{d} + \bar{c} + b + \bar{a})(\bar{d} + \bar{c} + \bar{b} + a)$$



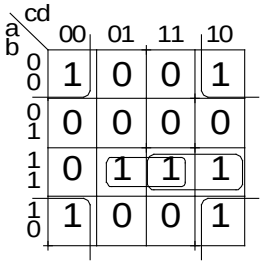
Karnaughdiagrammet

6.1



$$f = \bar{a}\,\bar{c}\,d + abd + \bar{b}\,\bar{d}$$

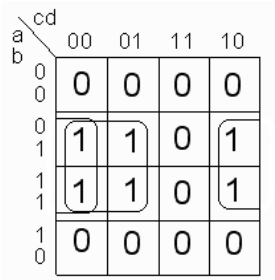
6.2



$$f = abd + abc + \bar{b}\bar{d}$$

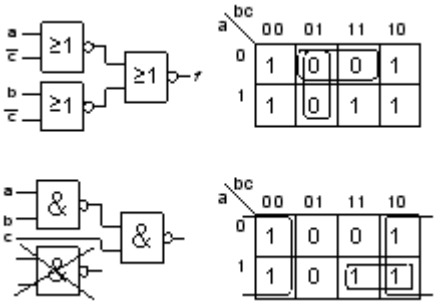
6.3

$$f = \bar{a}b\bar{c} + ab\bar{c} + bc\bar{d}$$



$$f = b\bar{c} + b\bar{d}$$

6.4

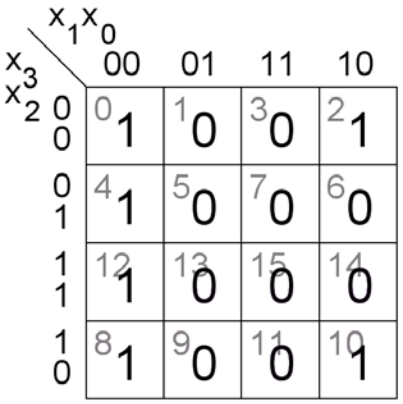


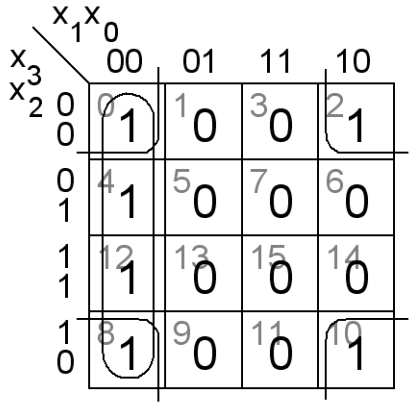
6.5

Sanningstabell och Karnaughdiagram. Den minimerade funktionen erhålls genom hoptagning av 1:or i Karnaughdiagrammet. Funktionens invers fås om 0:orna ”felaktigt” tas ihop som om dom vore 1:or.

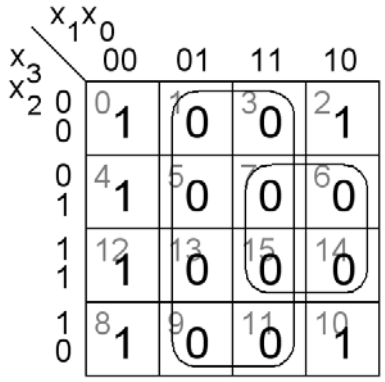
$f(x_3, x_2, x_1, x_0) = \sum m(0, 2, 4, 8, 10, 12) \quad f = ? \quad \bar{f} = ?$

	x_3	x_2	x_1	x_0	f
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0





$f = \overline{x}_1\overline{x}_0 + \overline{x}_2\overline{x}_0$



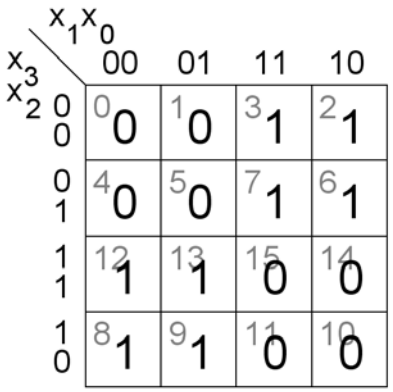
$\overline{f} = \{\,0 : \text{or som } 1 : \text{or}\,\} = x_0 + x_2x_1$

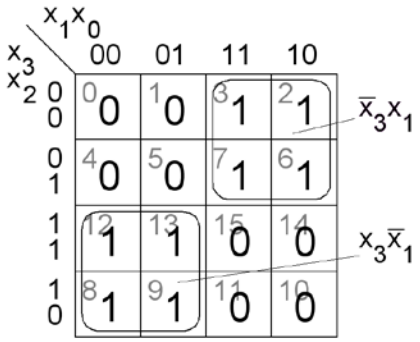
6.6

Sanningstabell och Karnaughdiagram. Den minimerade funktionen erhålls genom hoptagning av 1:or i Karnaughdiagrammet. Funktionens invers fås om 0:orna ”felaktigt” tas ihop som om dom vore 1:or.

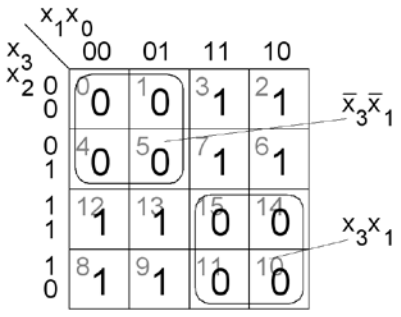
$f(x_3, x_2, x_1, x_0) = \prod M(0, 1, 4, 5, 10, 11, 14, 15) \quad f = ? \quad \overline{f} = ?$

	x_3	x_2	x_1	x_0	f
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0





$f = x_3\overline{x_1} + \overline{x_3}\,x_1 = x_3 \oplus x_1$



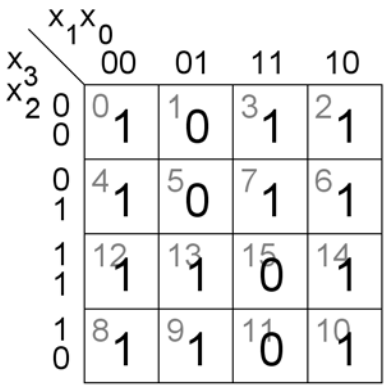
$\overline{f} = \{\,0 : \text{or som } 1 : \text{or}\,\} = \overline{x_3}\,\overline{x_1} + x_3x_1 = \overline{x_3 \oplus x_1}$

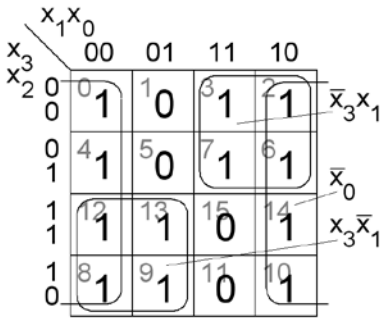
6.7

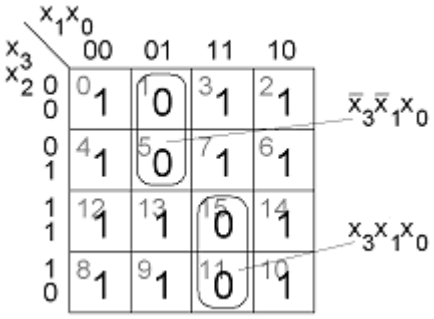
Sanningstabell och Karnaughdiagram. Den minimerade funktionen erhålls genom hoptagning av 1:or i Karnaughdiagrammet. Funktionens invers fås om 0:orna ”felaktigt” tas ihop som om dom vore 1:or.

$f(x_3, x_2, x_1, x_0) = \sum m(0, 2, 3, 4, 6, 7, 8, 9, 10, 12, 13, 14) \quad f = ? \quad \overline{f} = ?$

	x_3	x_2	x_1	x_0	f
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0





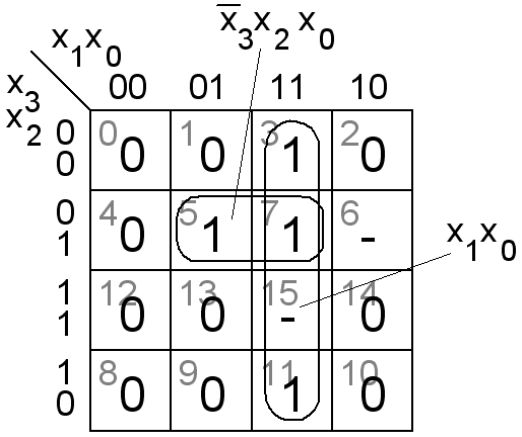


$f = \overline{x_0} + x_3\overline{x_1} + \overline{x_3}\,x_1$

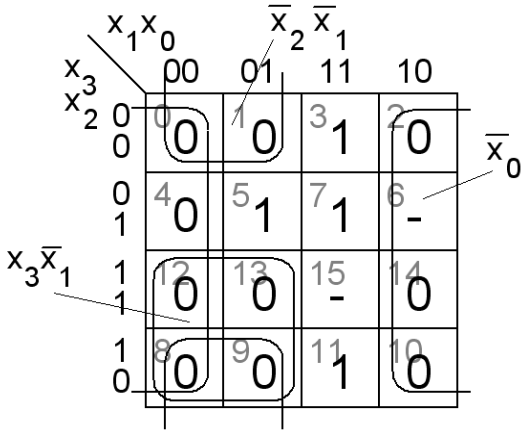
$\overline{f} = \{\,0:\text{or som}\,1:\text{or}\,\} = \overline{x_3}\,\overline{x_1}\,x_0 + x_3x_1x_0$

6.8

$f(x_3, x_2, x_1, x_0) = \sum m(3, 5, 7, 11) + d(6, 15) \quad f = ? \quad \overline{f} = ?$



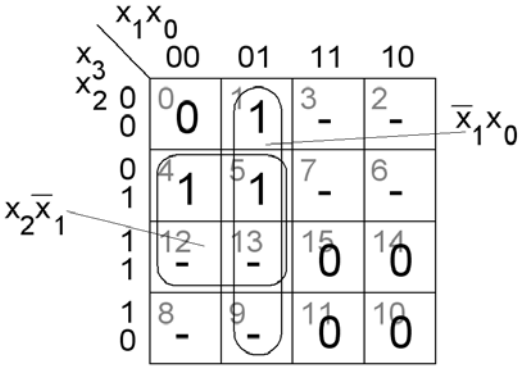
$f = x_1 x_0 + \overline{x_3} x_2 x_0$



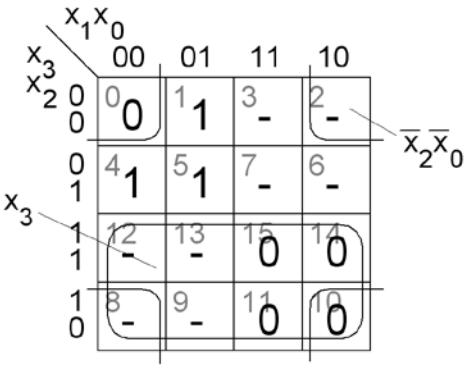
$\overline{f} = \overline{x_0} + \overline{x_2}\,\overline{x_1} + x_3\,\overline{x_1}$

6.9

$f(x_3, x_2, x_1, x_0) = \sum m(1, 4, 5) + d(2, 3, 6, 7, 8, 9, 12, 13) \quad f = ? \quad \overline{f} = ?$



$f = x_2\,\overline{x_1} + \overline{x_1}\,x_0$

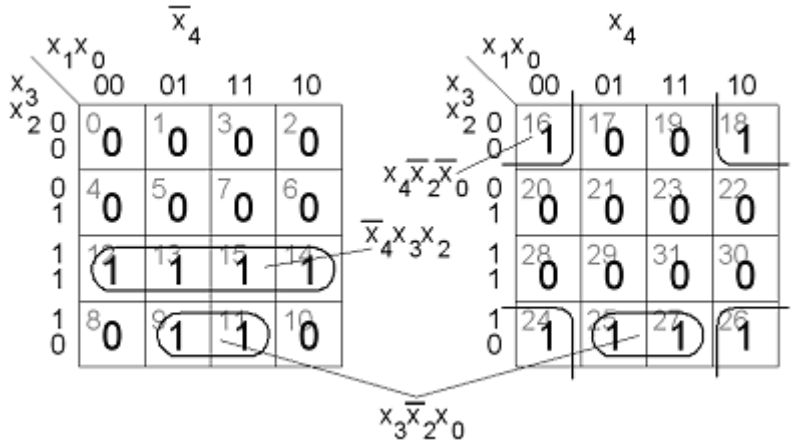


$\overline{f} = x_3 + \overline{x_2}\,\overline{x_0}$ eller $\overline{f} = x_1 + \overline{x_2}\,\overline{x_0}$

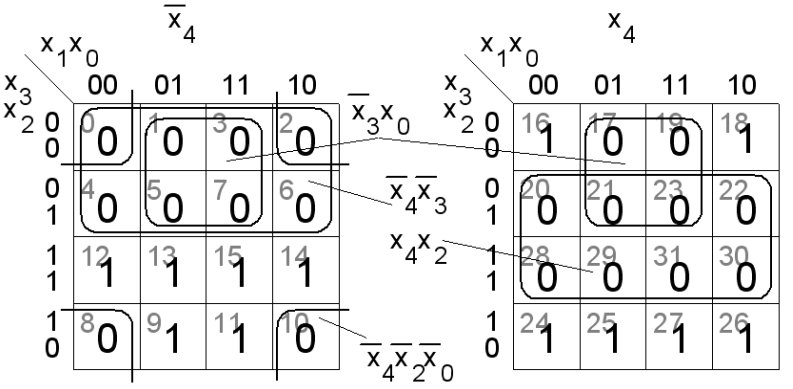
6.10

Karnaughdiagram för **fem** variabler. Det vänstra diagrammet är för $\overline{x}_4$ och det högra för x_4. Om samma hoptagning kan göras i *båda* diagrammen utgår variabeln $\overline{x}_4$ eller x_4 annars tas respektive variabel med.

$f(x_4, x_3, x_2, x_1, x_0) \quad f = ? \quad \overline{f} = ?$



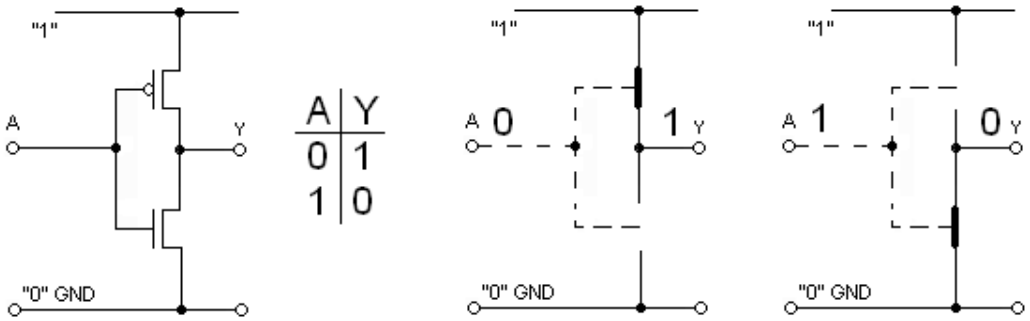
$$f = \overline{x_4}\, x_3\, x_2 + x_3\, \overline{x_2}\, x_0 + x_4\, \overline{x_2}\, \overline{x_0}$$



$$\overline{f} = \overline{x_4}\,\overline{x_3} + \overline{x_3}\, x_0 + x_4\, x_2 + \overline{x_4}\,\overline{x_2}\,\overline{x_0}$$

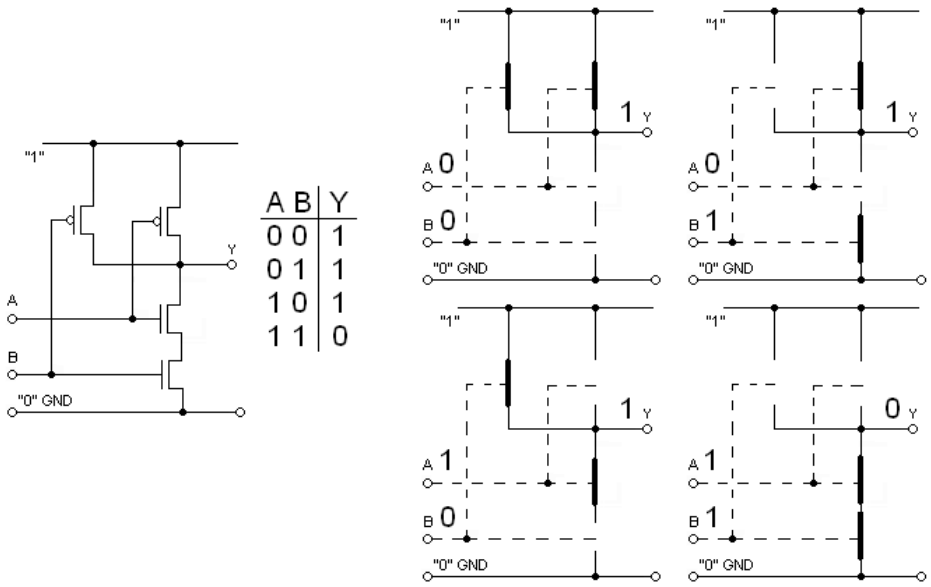
MOS-transistorn och digitala kretsar

7.1



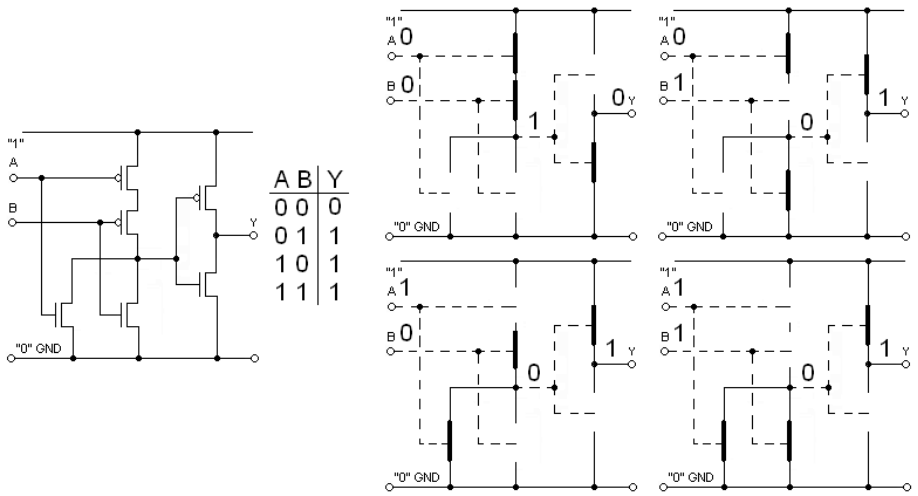
Det är en CMOS-inverterare. $Y = \overline{A}$.

7.2



Det är en CMOS-NAND-grind. $Y = \overline{A \cdot B}$.

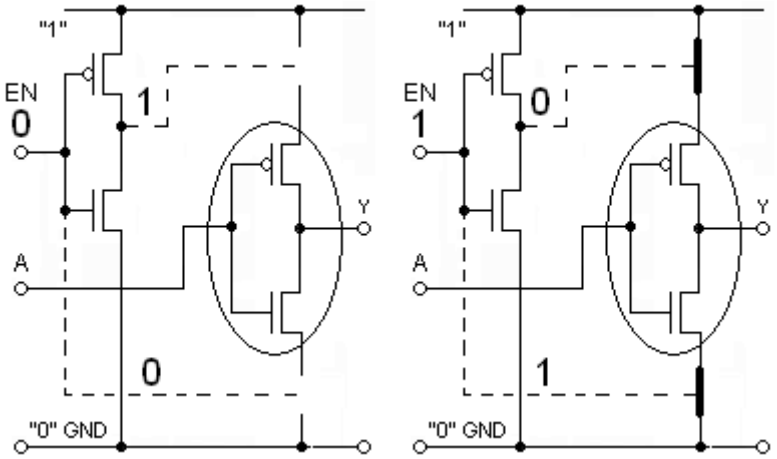
7.3



Det är en CMOS-OR-grind. $Y = A + B$.

7.4

Kretsen är en inverterare med THREE-state utgång. När *EN* = 1 blir den inringade delen ”inkopplad” och sambandet mellan *Y* och *A* blir då invertering, $Y = \overline{A}$. När EN = 0 blir den inringade delen ”urkopplad”. Utgången är då i ett tredje tillstånd, förutom ”1” och ”0” finns det således ett tillstånd ”urkopplad”. Eftersom utgången *Y* nu inte är inkopplad kan den då inte heller påverkas av *A*.

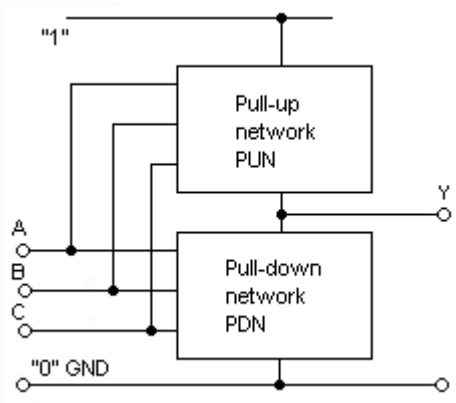


THREE-state utgångar används för att göra det möjligt att koppla samman många utgångar till en och samma ledning (**bussledning**). Flera kretsars utgångar kan utnyttja en gemensam ledning, förutsatt att bara en av kretsarna är aktiv (*EN* = 1) åt gången – de övriga har *EN* = 0 och är ”urkopplade”.

7.5

CMOS-kretsar består av två delnät som är varandras inverser. Pull-up-nätet, PUN, styr ”1” till utgången medans Pull-down-nätet styr ”0”. Analyserar man Pull-down-nätet så får man därför funktionen *Y*:s invers**.**

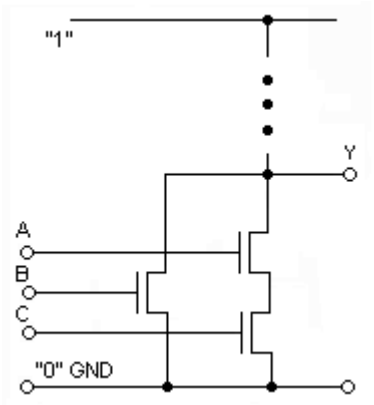
$$\overline{Y} = A \cdot C + B \quad \Rightarrow \quad Y = \overline{A \cdot C + B} = \overline{A \cdot C} \cdot \overline{B} = (\overline{A} + \overline{C})\overline{B}$$

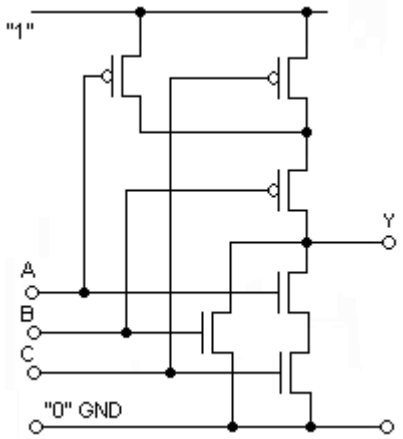


$$(\overline{A} + \overline{C})\overline{B}$$

Pull-up-nätet skall således bestå av *A* och *C* i parallell-koppling (+) sedan seriekopplade (·) med B. Användandet av PMOS-transistorer inverterar variablerna A, B och C.

$(\overline{A} + \overline{C})\overline{B}$



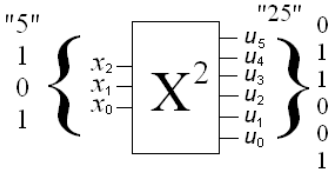


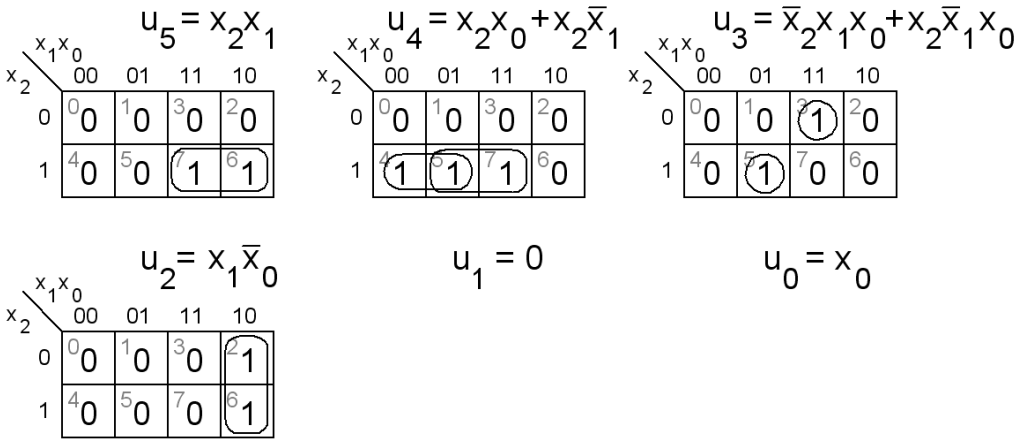
Kombinatoriska kretsnät

8.1

X	x_2	x_1	x_0	$U = X^2$	u_5	u_4	u_3	u_2	u_1	u_0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	1
2	0	1	0	4	0	0	0	1	0	0
3	0	1	1	9	0	0	1	0	0	1
4	1	0	0	16	0	1	0	0	0	0
5	1	0	1	25	0	1	1	0	0	1
6	1	1	0	36	1	0	0	1	0	0
7	1	1	1	49	1	1	0	0	0	1

Av sanningstabellen framgår att u_1 alltid är lika med 0. u_1 utgången kan därför anslutas 0V (jord) så att den får konstanten 0. Man kan vidare se att u_0 alltid är samma som x_0. u_0 utgången kan därför förbindas direkt med x_0 ingången. De övriga funktionerna tas fram med Karnaughdiagrammen.

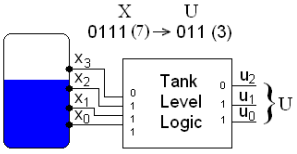


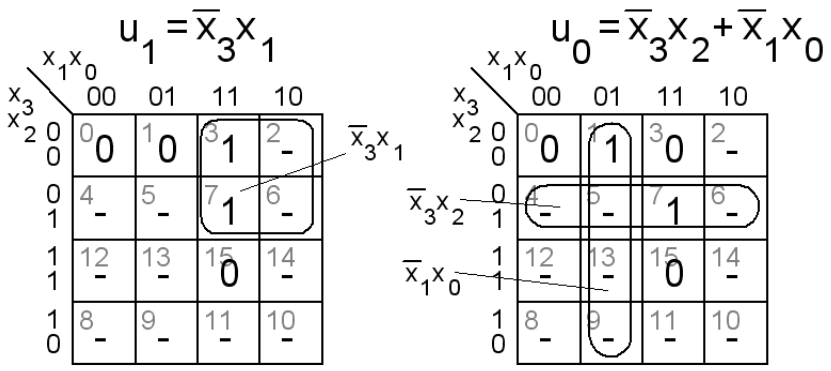


8.2

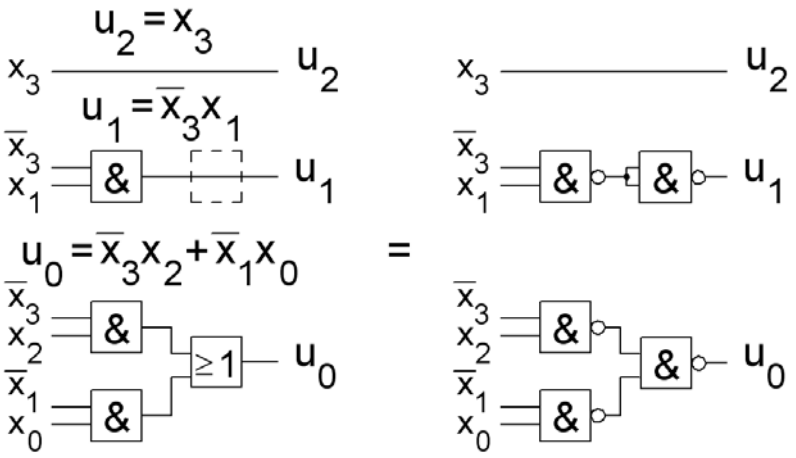
X	x_3	x_2	x_1	x_0	U	u_2	u_1	u_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	0	0	1
3	0	0	1	1	2	0	1	0
7	0	1	1	1	3	0	1	1
15	1	1	1	1	4	1	0	0

Vi kan direkt se i tabellen att u_2 och x_3 är lika varför u_2 kan anslutas direkt till x_3. $u_2 = x_3$.
De övriga uttrycken fås med hjälp av deras Karnaughdiagram.





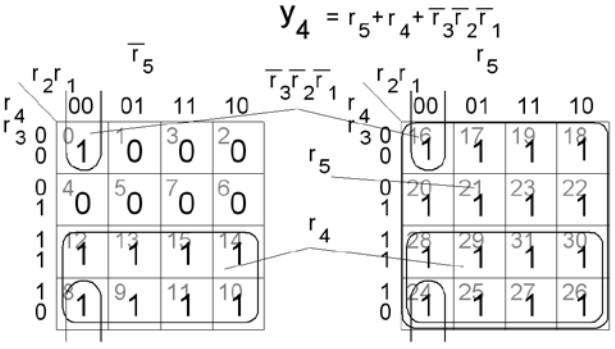
Grindnät:

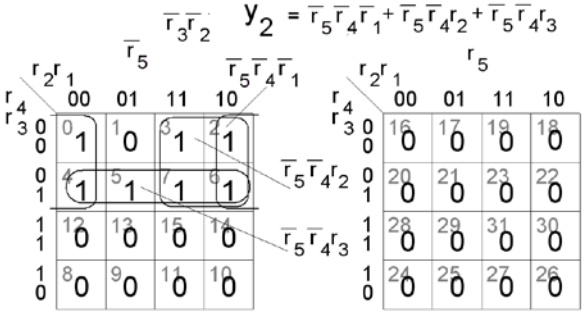


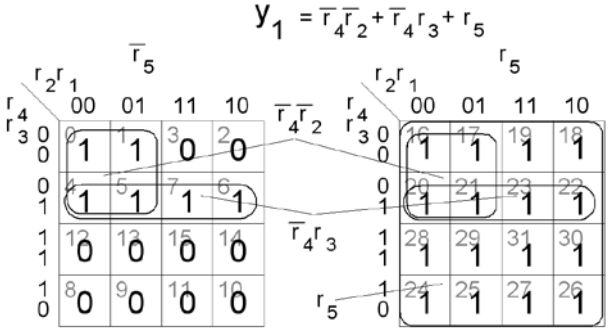
8.3

Sanningstabell

r_5	r_4	r_3	r_2	r_1	y_4	y_2	y_1
1	-	-	-	-	1	0	1
0	1	-	-	-	1	0	0
0	0	1	-	-	0	1	1
0	0	0	1	-	0	1	0
0	0	0	0	1	0	0	1
0	0	0	0	0	1	1	1







8.4

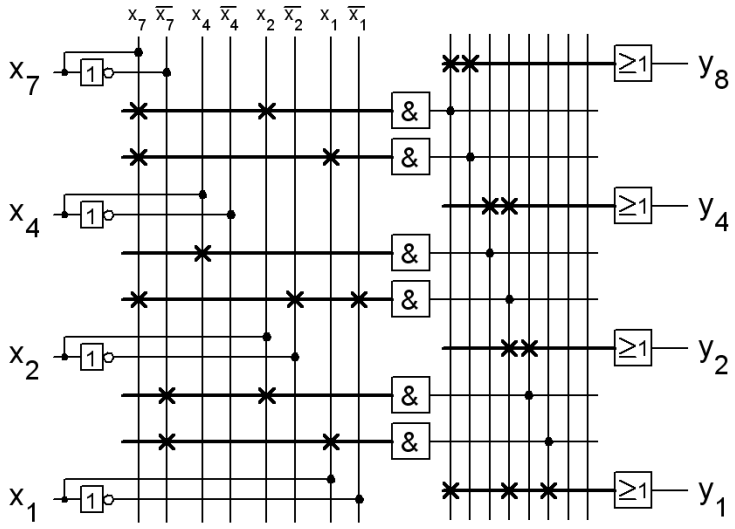
	7	4	2	1		8	4	2	1
	x_7	x_4	x_2	x_1		y_8	y_4	y_2	y_1
(0)	0	0	0	0	0	0	0	0	0
(1)	0	0	0	1	1	0	0	0	1
(2)	0	0	1	0	2	0	0	1	0
(3)	0	0	1	1	3	0	0	1	1
(4)	0	1	0	0	4	0	1	0	0
(5)	0	1	0	1	5	0	1	0	1
(6)	0	1	1	0	6	0	1	1	0
(8)	1	0	0	0	7	0	1	1	1
(9)	1	0	0	1	8	1	0	0	0
(10)	1	0	1	0	9	1	0	0	1

$$y_8 = x_7 x_2 + x_7 x_1$$

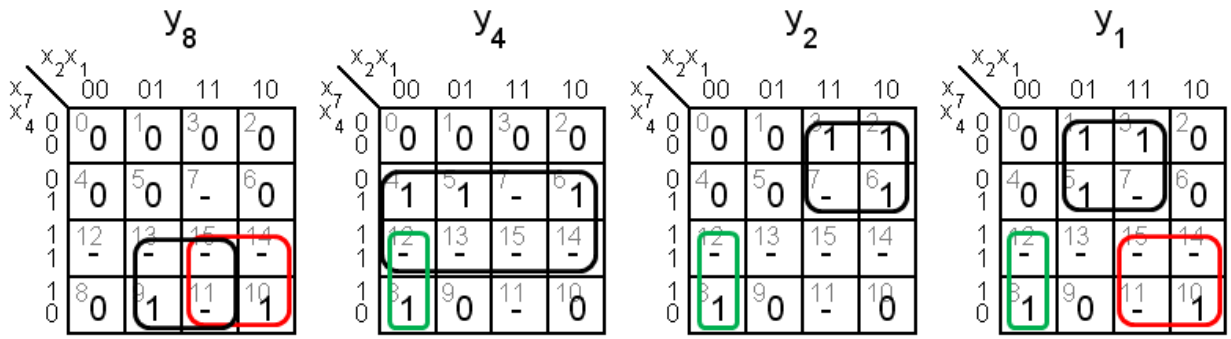
$$y_4 = x_4 + x_7 \overline{x_2}\,\overline{x_1}$$

$$y_2 = \overline{x_7}\, x_2 + x_7 \overline{x_2}\,\overline{x_1}$$

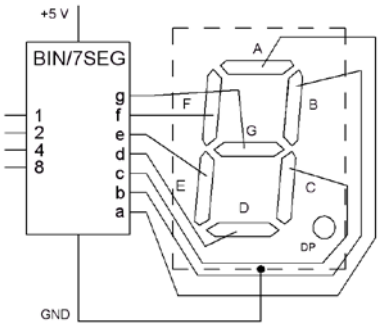
$$y_1 = \overline{x_7}\, x_1 + x_7 x_2 + x_7 \overline{x_2}\,\overline{x_1}$$



Två av AND-grindarna kan *delas* av y_8,y_1 och y_8,y_2,y_1 -näten.



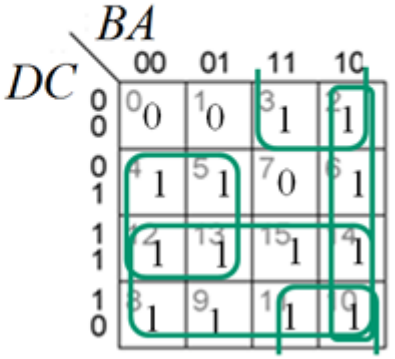
8.5



D C B A	a b c d e f g
0 0 0 0	1 1 1 1 1 1 0
0 0 0 1	0 1 1 0 0 0 0
0 0 1 0	1 1 0 1 1 0 1
0 0 1 1	1 1 1 1 0 0 1
0 1 0 0	0 1 1 0 0 1 1
0 1 0 1	1 0 1 1 0 1 1
0 1 1 0	1 0 1 1 1 1 1
0 1 1 1	1 1 1 0 0 0 0
1 0 0 0	1 1 1 1 1 1 1
1 0 0 1	1 1 1 0 0 1 1
1 0 1 0	1 1 1 0 1 1 1
1 0 1 1	0 0 1 1 1 1 1
1 1 0 0	0 0 0 1 1 0 1
1 1 0 1	0 1 1 1 1 0 1
1 1 1 0	1 0 0 1 1 1 1
1 1 1 1	1 0 0 0 1 1 1

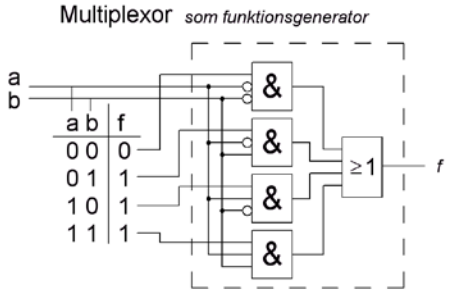
segment

$$g = D + \overline{B}C + B\overline{C} + B\overline{A}$$



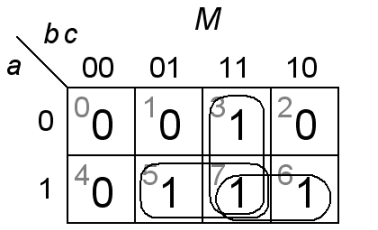
8.6

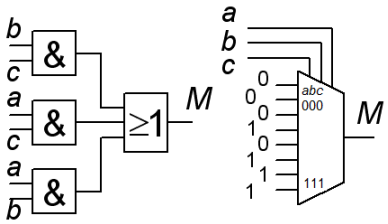
En 4-1 multiplexor som funktionsgenerator för OR-funktionen.



8.7

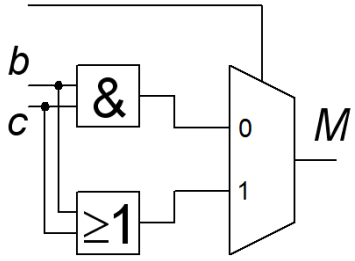
	a	b	c	M	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	$\bar{a}bc$
4	1	0	0	0	
5	1	0	1	1	$a\bar{b}c$
6	1	1	0	1	$ab\bar{c}$
7	1	1	1	1	abc





$$M = bc + ac + ab$$

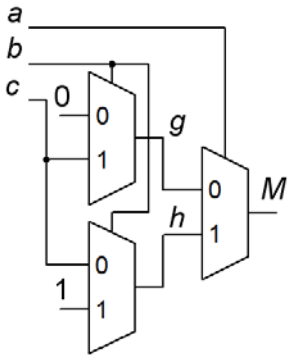
$$M = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc = \bar{a}(bc) + a(\bar{b}c + b\bar{c} + bc) = \bar{a}(bc) + a(b + c)$$



$M = \bar{a}(bc) + a(b + c) \quad g = bc \quad h = b + c$

$g = \bar{b}(0) + b(c) \quad = bc$

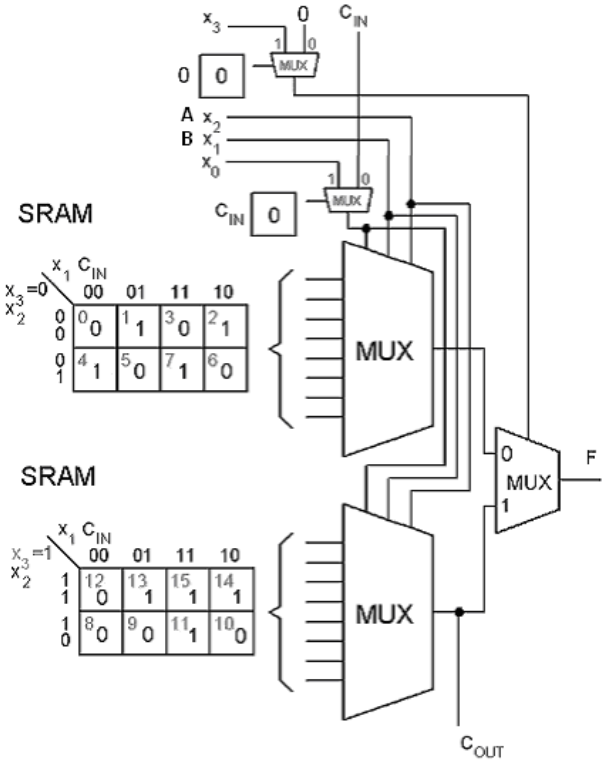
$h = \bar{b}(c) + b(1) \quad = \bar{b}c + b(\bar{c} + c) = \bar{b}c + b\bar{c} + bc = c(b + \bar{b}) + b(c + \bar{c}) = b + c$



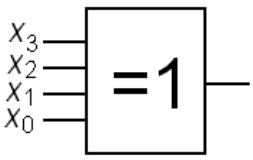
8.8

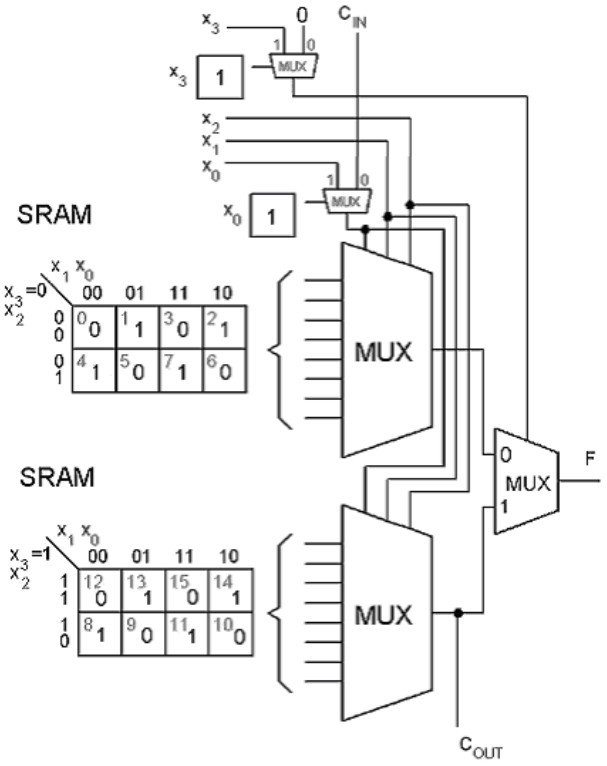
För att göra en heladderare behöver vi använda den övre MUXen till summafunktionen, och den nedre MUXen, som alltid är ansluten till C_{OUT} , används till Carry-funktionen. I stället för x_0 väljer vi C_{IN}. För att den övre MUXen skall anslutas till logikelementets utgång ska utgångsmuxen styras med 0 i stället för med x_3 .

x_2	x_1	x_0		
A	B	C_{IN}	$\sum$	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

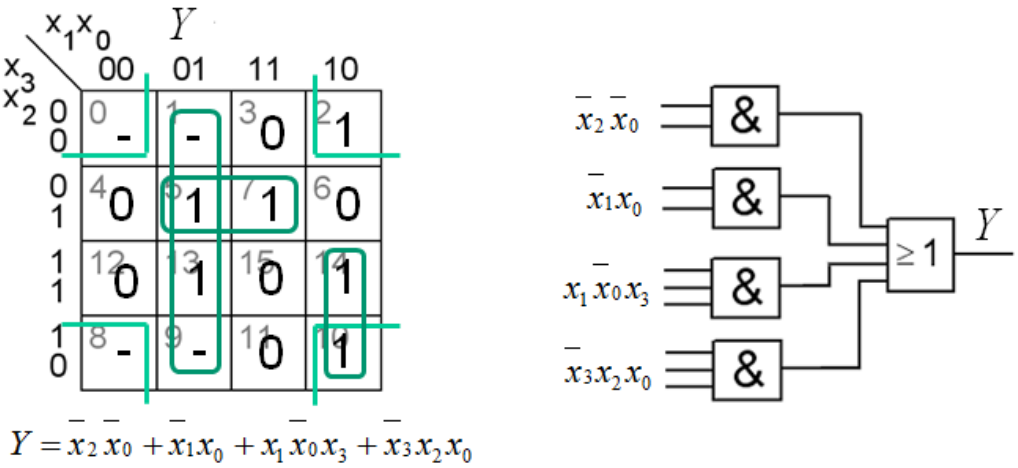


8.9

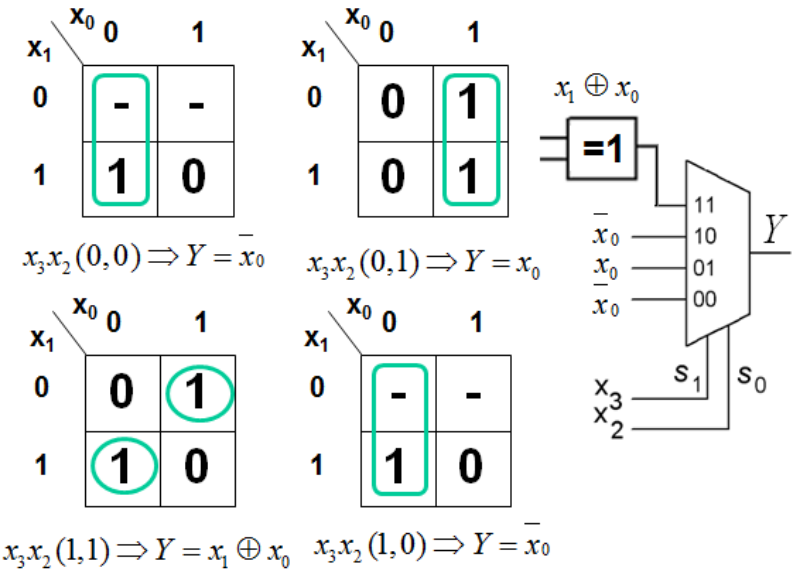




8.10

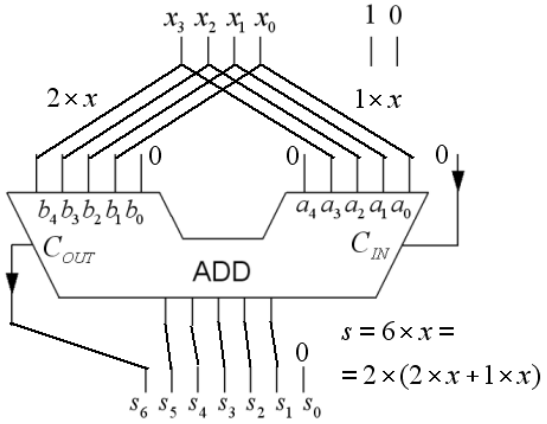


$$Y = \bar{x}_2\bar{x}_0 + \bar{x}_1x_0 + x_1\bar{x}_0x_3 + \bar{x}_3x_2x_0$$



Alternativt kan XOR-grinden även användas till MUX-ingångarna 00 och 10.

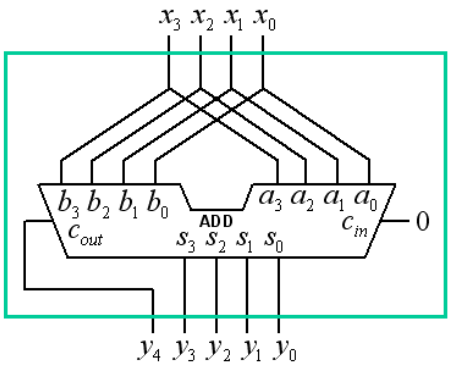
8.11 a)

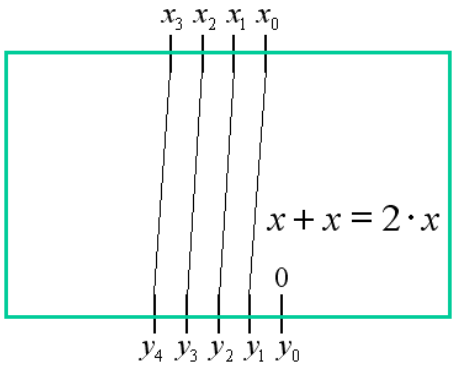


b) Största talet blir $s_{max} = 6 \cdot 15 = 90$. Eftersom miniräknare inte är tillåten vid tentamen kan vi denna gång välja att omvandla 90 till ett **binärt tal** på samma sätt som i uppgiften (ok även med andra omvandlingssätt):

$$\begin{array}{rcccccc} & \underline{1} & \underline{1} & \underline{1} & \underline{1} & \underline{0} & \underline{0} & \\ & 0 & 1 & 1 & 1 & 1 & 0 & 15\times 2 \\ + & 0 & 0 & 1 & 1 & 1 & 1 & 15\times 1 \\ \hline & 1 & 0 & 1 & 1 & 0 & 1 & \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & \times 2 \end{array}$$

8.12


x_3 x_2 x_1 x_0
b_3 b_2 b_1 b_0
ADD
a_3 a_2 a_1 a_0
c_{out}
s_3 s_2 s_1 s_0
c_{in}
0
y_4 y_3 y_2 y_1 y_0

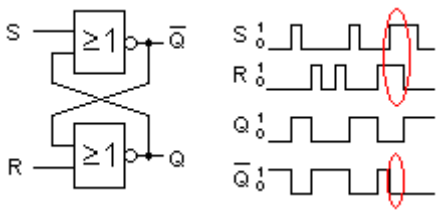

x_3 x_2 x_1 x_0
$x + x = 2 \cdot x$
0
y_4 y_3 y_2 y_1 y_0

Sekvenskretsar, låskretsar och klockade vippor

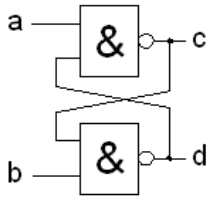
9.1

Figuren visar en *SR*-latch, men mot slutet av insignals-sekvensen förekommer den ”förbjudna” insignalskombinationen $S = 1$, $R = 1$. Utgångarna blir inte varandras inverser för denna kombination.

För NOR-grindar är 1 låsande insignal, därför blir $\overline{Q} = 0$ så länge *S* är kvar på 1.

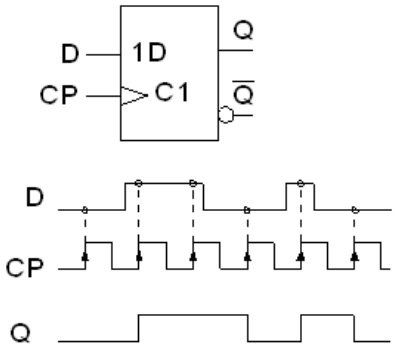


9.2

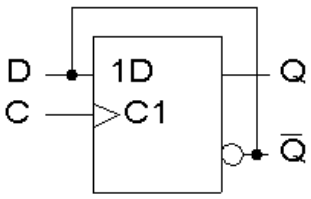


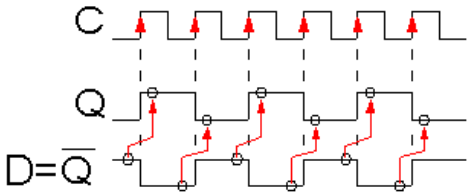
a b	c d
0 0	1 1
0 1	1 0
1 0	0 1
1 1	c d

9.3

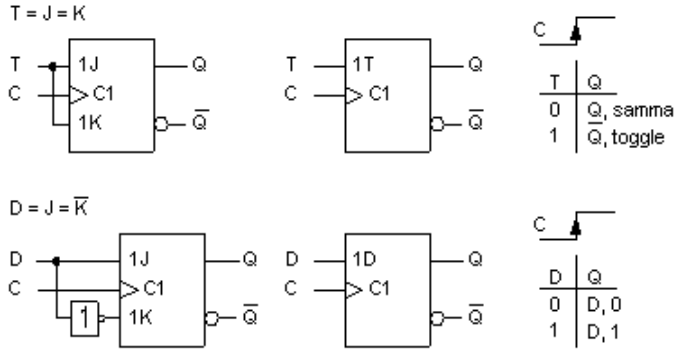


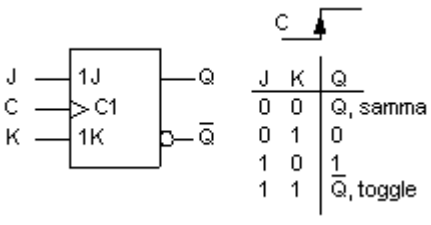
9.4





9.5



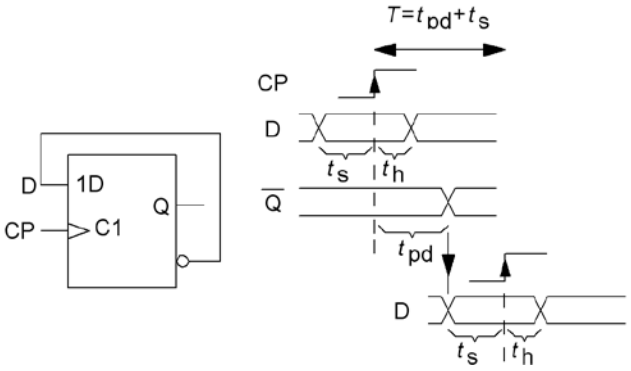


JK-vippan kan användas som T-vippa eller som D-vippa. (När vippor kopplas ihop med varandra finns det inverterade utgångar att tillgå, då behövs inte inverteraren för att göra JK-vippan till D-vippa.)

9.6

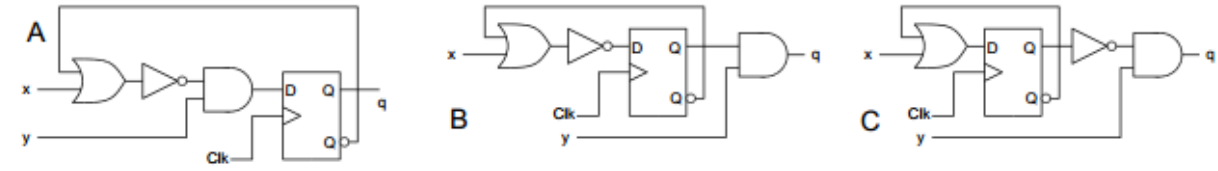
$$T = t_{pd} + t_s$$

$$f = \frac{1}{T} = \frac{1}{t_{pd} + t_s} = \frac{1}{(20+30)[\text{ns}]} = 20 \text{ MHz}$$

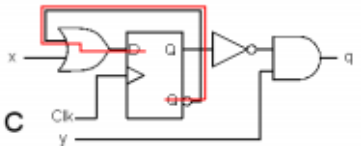


9.7

$t_{AND} = 0{,}4$ ns, $t_{OR} = 0{,}4$ ns, $t_{NOT} = 0{,}1$ ns, $t_{setup} = 0.3$ ns, $t_{dq} = 0{,}4$ns



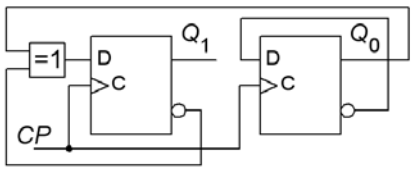
$$T = T_{OR} + T_{setup} + T_{dq} = 0{,}4 + 0{,}3 + 0{,}4 = 1{,}1 \text{ ns}$$



9.8

a) Klockperioden bestäms av den längsta datavägen, den som innehåller XOR-grinden.
$T_{CP} > t_{pdQ} + t_{su} + t_{pdXOR} = 3+4+5 =$ **12 ns**.

b) Holdtiden är den tid som dataingången måste hållas stabil efter klockpulsen. Den vippa som har *D*-ingången kopplad direkt till $\overline{Q}_0$ får sin D-ingång ändrad snabbast, direkt efter $t_{pdQ} = 3$ ns. $t_h <$ **3 ns**.



Sekvensnät, automater

10.1

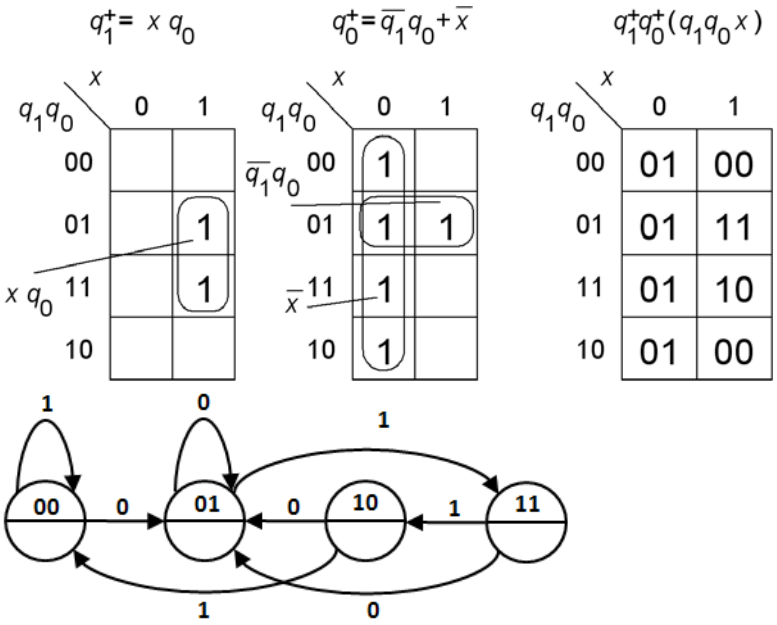
Ur kretsschemat kan följande samband ställas upp:

$q_1 \quad q_0$

$q_1^+ = x \cdot q_0$

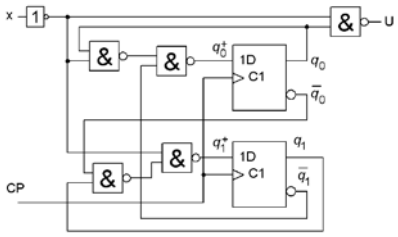
$q_0^+ = \overline{x} + \overline{q_1} \cdot q_0$

Ingen utgångsavkodare finns utan vippornas tillstånd används direkt. Moore-modellen skall användas.



10.2

Ur kretsschemat kan följande samband ställas upp:

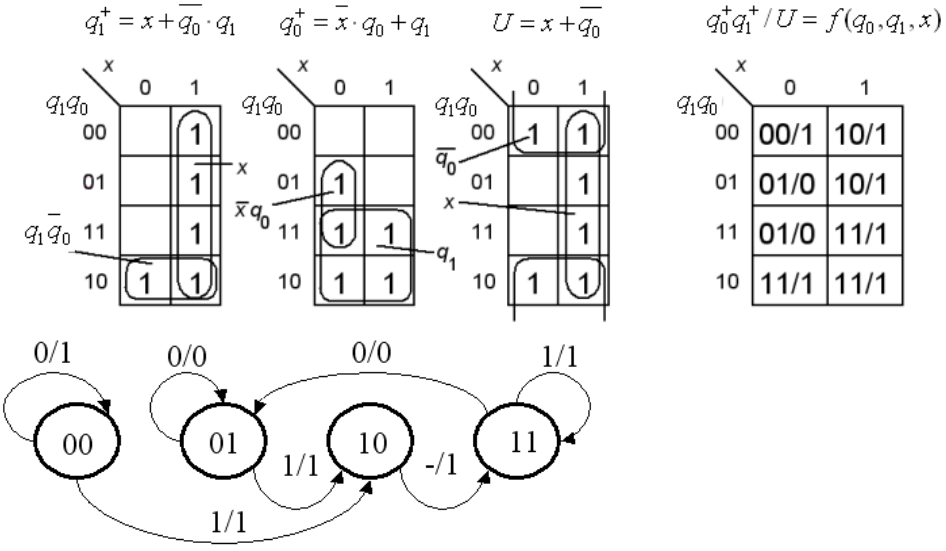


$U = \overline{\overline{x} \cdot q_0} = x + \overline{q_0}$

$q_1^+ = \overline{\overline{q_1} \cdot (q_0 \cdot x)} = x + \overline{q_0} \cdot q_1$

$q_0^+ = \overline{(q_1 \cdot \overline{q_0}) \cdot \overline{x}} = \overline{x} \cdot q_0 + q_1$

Eftersom *U* beror direkt av *x* så måste Mealy-modellen användas.



10.3

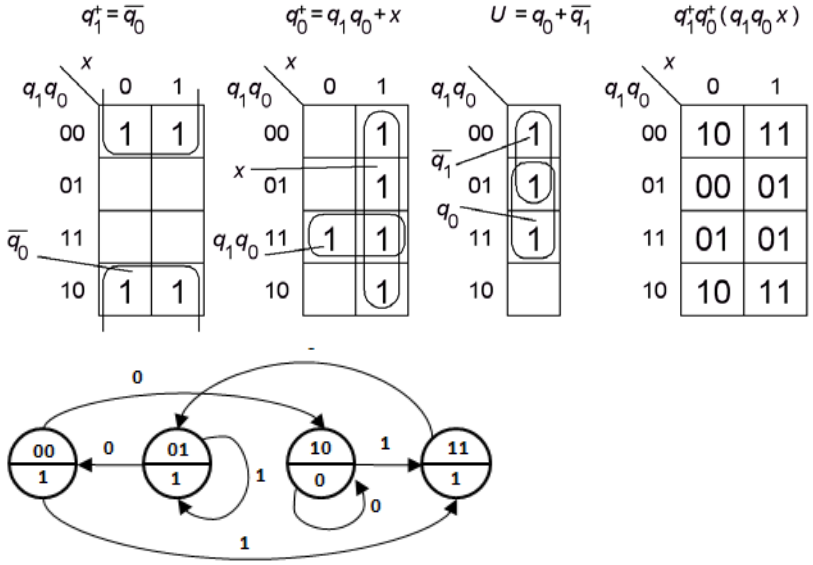
Ur kretsschemat kan följande samband ställas upp:

$$U = \overline{\overline{q_0} \cdot q_1} = q_0 + \overline{q_1}$$

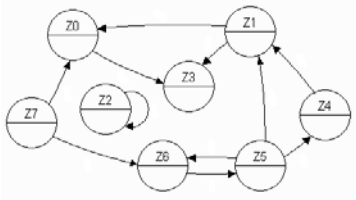
$$q_1^+ = \overline{q_0}$$

$$q_0^+ = \overline{x} \cdot \overline{\overline{q_0} \cdot q_1} = q_1 q_0 + x$$

Eftersom *U* bara beror på tillståndet och är oberoende av *x* så måste Moore-modellen användas.

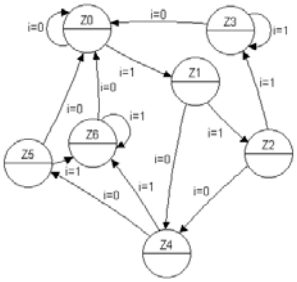


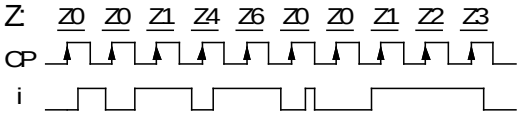
10.4.



Stopptillstånd: Z3
Förlusttillstånd: Z7
Isolerade tillstånd: Z2

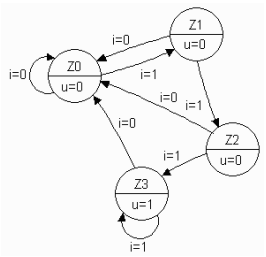
10.5

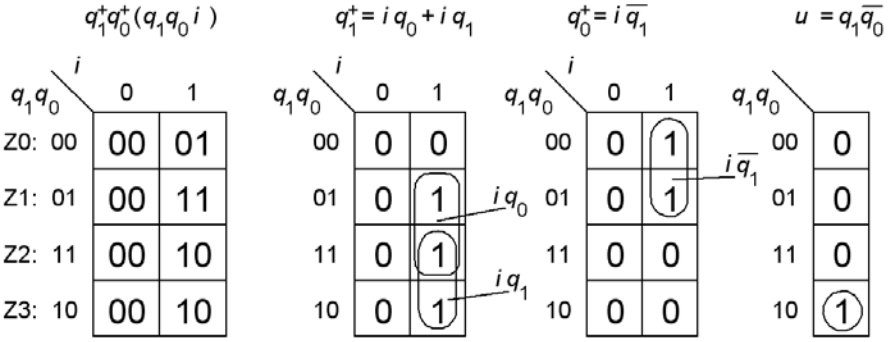


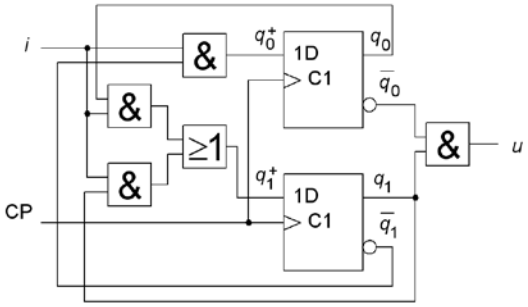


10.6

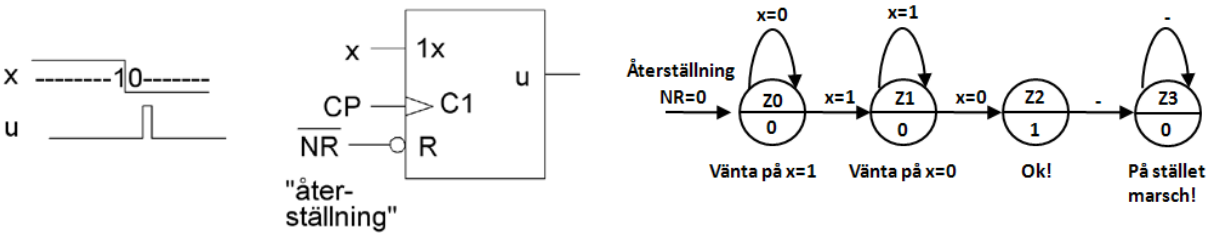
Från tillståndsdiagram till kodad tillståndstabell.



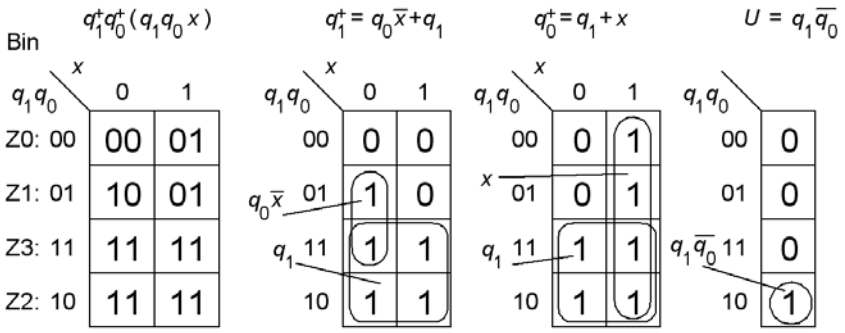




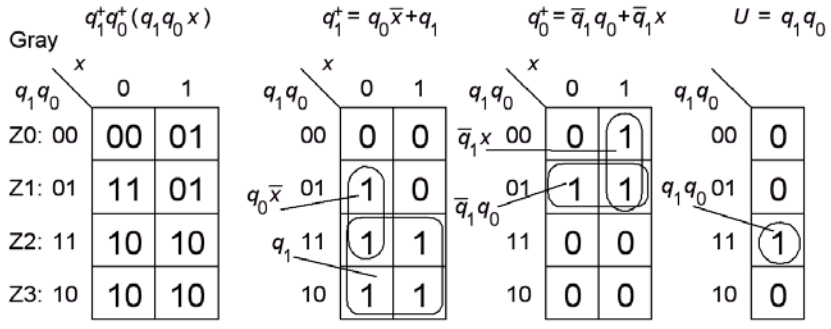
10.7



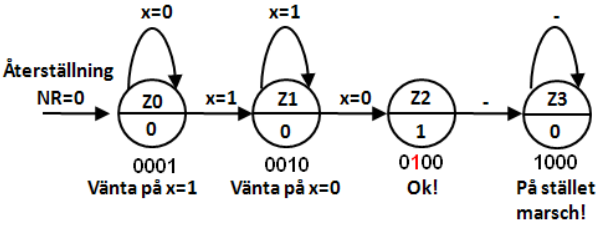
Tillståndskodning Binär:



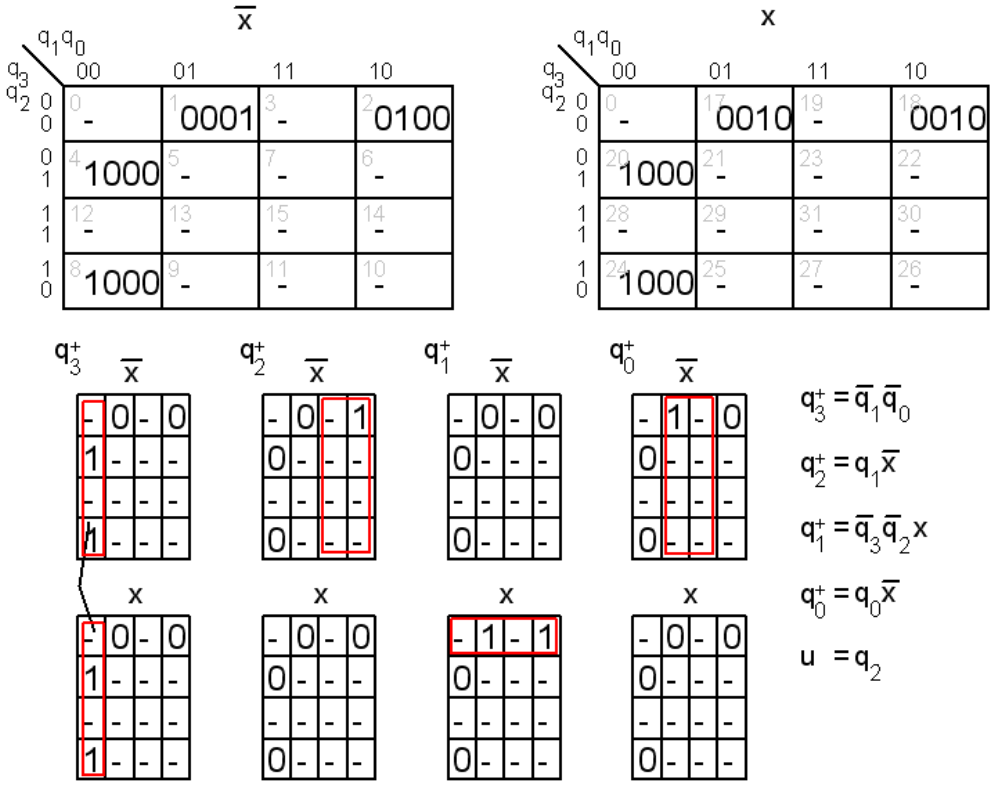
Tillståndskodning Gray:



Tillståndskodning One hot:



$$q_3^+ q_2^+ q_1^+ q_0^+ (x\, q_3\, q_2\, q_1\, q_0)$$



$$q_3^+ = \bar{q}_1 \bar{q}_0$$

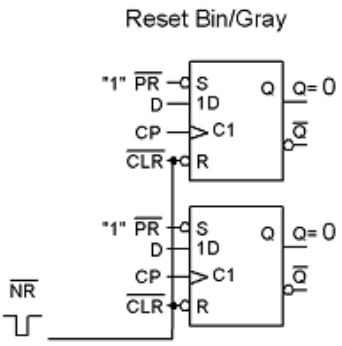
$$q_2^+ = q_1 \bar{x}$$

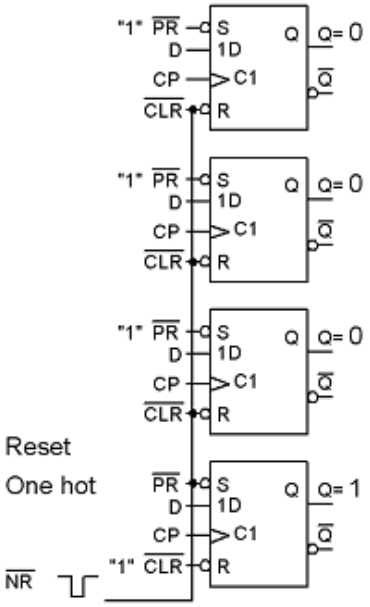
$$q_1^+ = \bar{q}_3 \bar{q}_2 x$$

$$q_0^+ = q_0 \bar{x}$$

$$u = q_2$$

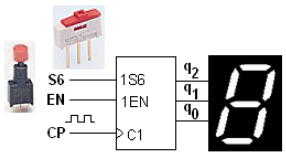
Så här gör man återställning till ”00” med CLR ingångarna för Bin/Gray tillståndskodningen, respektive till ”0001” med PRE/CLR ingångarna för ”one hot” tillståndskodningen.





10.8

För sex tillstånd krävs det tre vippor. Då finns det 8 tillstånd totalt, två tillstånd som inte ingår i sekvensen. Vi specificerar för säkerhets skull vad som skall ske med dom tillstånden, så att räknaren inte skall riskera att ”fastna”.



$q_2^+ q_1^+ q_0^+ (q_2 q_1 q_0)$

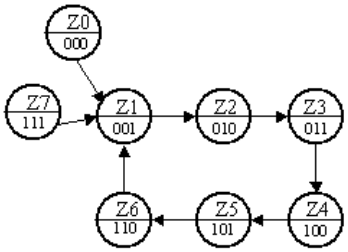
$q_2 q_1$ \ q_0	0	1
00	001	010
01	011	100
11	001	001
10	101	110

q_2^+				q_1^+				q_0^+			
0	0			0	1			1	0		
0	1			1	0			1	0		
0	0			0	0			1	1		
1	1			0	1			1	0		

$q_2^+ = q_2\bar{q}_1 + \bar{q}_2 q_1 q_0$

$q_1^+ = \bar{q}_1 q_0 + \bar{q}_2 q_1 \bar{q}_0$

$q_0^+ = \bar{q}_0 + q_2 q_1$



Omskrivning med EN (EN=0 → på stället marsch) :

$(q_2^+)' = EN \cdot (q_2^+) + \overline{EN} \cdot (q_2)$

$(q_1^+)' = EN \cdot (q_1^+) + \overline{EN} \cdot (q_1)$

$(q_0^+)' = EN \cdot (q_0^+) + \overline{EN} \cdot (q_0)$

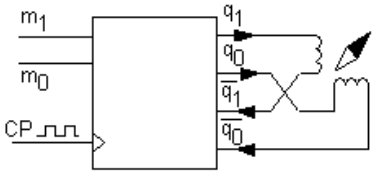
Omskrivning med S6 (S6 = 1 → nästa tillstånd 110) :

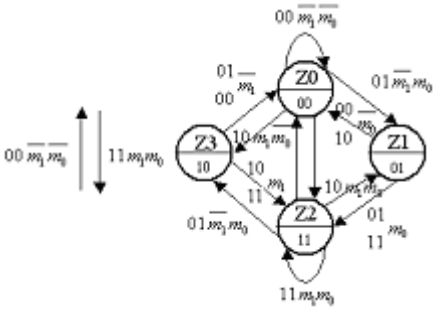
$(q_2^+)'' = (q_2^+)' + S6$

$(q_1^+)'' = (q_1^+)' + S6$

$(q_0^+)'' = (q_0^+)' \cdot \overline{S6}$

10.9





$q_1^+ q_0^+ (q_1 q_0 m_1 m_0)$

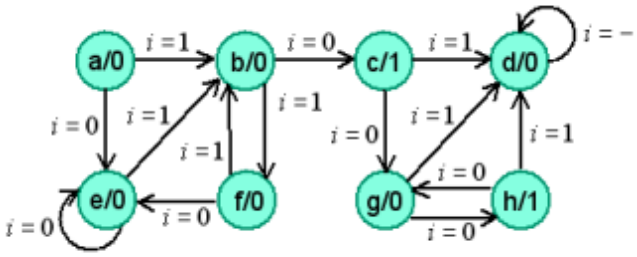
$q_1 q_0$ \ $m_1 m_0$	00	01	11	10
00	00	01	11	10
01	00	11	11	00
11	00	10	11	01
10	00	00	11	11

q_1^+				q_0^+			
0	0	1	1	0	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	0	1	1
0	0	1	1	0	0	1	1

$q_1^+ = q_0 m_0 + \bar{q}_0 m_1$

$q_0^+ = q_1 m_1 + \bar{q}_1 m_0$

10.10



Skriv tillståndstabell.

now	next		out
$i =$	0	1	z
a	e	b	0
b	c	f	0
c	g	d	1
d	d	d	0
e	e	b	0
f	e	b	0
g	h	d	0
h	g	d	1

Två tillstånd kan *inte* vara ekvivalenta om utsignalerna är olika eller om efterföljande tillstånds utsignaler är olika.

Grupper med samma utsignal:

$$P_1 = (a,b,d,e,f,g)(c,h)$$

Undersök efterföljande tillstånd:

$a_{i=0} \to (a,b,d,\mathrm{e},f,g)$ $a_{i=1} \to (a,\mathrm{b},d,e,f,g)$

$b_{i=0} \to (\mathrm{c},h)$ $b_{i=1} \to (a,b,d,e,\mathrm{f},g)$

$d_{i=0} \to (a,b,\mathrm{d},e,f,g)$ $d_{i=1} \to (a,b,\mathrm{d},e,f,g)$

$e_{i=0} \to (a,b,d,\mathbf{e},f,g)$ $e_{i=1} \to (a,\mathbf{b},d,e,f,g)$

$f_{i=0} \to (a,b,d,\mathbf{e},f,g)$ $f_{i=1} \to (a,\mathbf{b},d,e,f,g)$

$g_{i=0} \to (c,\mathbf{h})$ $g_{i=1} \to (a,b,\mathbf{d},e,f,g)$

(b, g) bildar egen grupp.

$$P_2 = (a,d,e,f)(b,g)(c,h)$$

$$P_2 = (a,d,e,f)(b,g)(c,h)$$

Undersök efterföljande tillstånd:

$a_{i=0} \to (a,d,\mathbf{e},f)$ $a_{i=1} \to (\mathbf{b},g)$

$d_{i=0} \to (a,\mathbf{d},e,f)$ $d_{i=1} \to (a,\mathbf{d},e,f)$

$e_{i=0} \to (a,d,\mathbf{e},f)$ $e_{i=1} \to (\mathbf{b},g)$

$f_{i=0} \to (a,d,\mathbf{e},f)$ $f_{i=1} \to (\mathbf{b},g)$

(d) bildar en egen grupp.

$$P_3 = (a,e,f)(b,g)(d)(c,h)$$

$$P_3 = (a,e,f)(b,g)(d)(c,h)$$

Undersök efterföljande tillstånd:

$b_{i=0} \to (\mathbf{c},h) \quad b_{i=1} \to (a,e,\mathbf{f})$

$g_{i=0} \to (c,\mathbf{h}) \quad g_{i=1} \to (\mathbf{d})$

(b) (g) bildar egna grupper.

$$P_4 = (a,e,f)(b)(d)(g)(c,h)$$

Byte av beteckningar

$(a,e,f) \Rightarrow a$

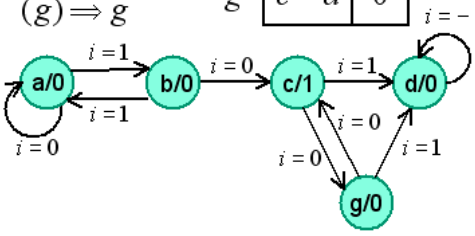
$(b) \Rightarrow b$

$(c,h) \Rightarrow c$

$(d) \Rightarrow d$

$(g) \Rightarrow g$

	now	next	out
$i=$	0	1	z
a	a	b	0
b	c	a	0
c	g	d	1
d	d	d	0
g	c	d	0



10.11

Grupper med samma utsignal:

$P_1 = (a,b,c,f)(d,e)$

Undersök efterföljande tillstånd:

$a_{i=0} \to (d,e) \quad a_{i=1} \to (a,b,c,f)$

$b_{i=0} \to (d,e) \quad b_{i=1} \to (a,b,c,f)$

$c_{i=0} \to (d,e) \quad c_{i=1} \to (d,e)$

$f_{i=0} \to (a,b,c,f) \quad f_{i=1} \to (a,b,c,f)$

$d_{i=0} \to (a,b,c,f) \quad d_{i=1} \to (a,b,c,f)$

$e_{i=0} \to (a,b,c,f) \quad e_{i=1} \to (a,b,c,f)$

	next		out
$i:$	0	1	z
a	e	c	0
b	d	c	0
c	d	e	0
d	f	a	1
e	f	b	1
f	f	b	0

Grupper med också samma efterföljare:

$P_2 = (a,b)(c)(f)(d,e)$

$P_3 = P_2$

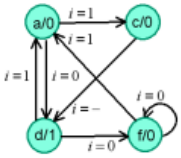
$(a,b) \to a$

$(c) \to c$

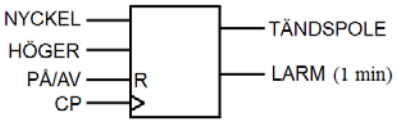
$(f) \to f$

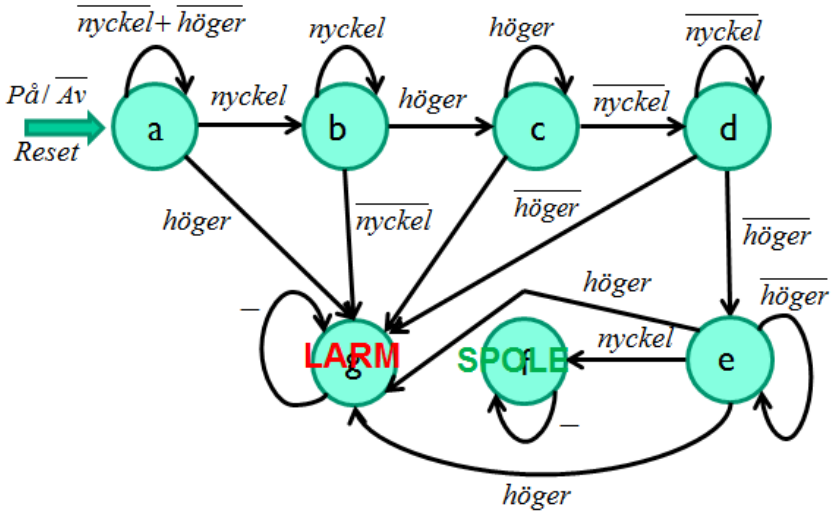
$(d,e) \to d$

	next		out
	0	1	z
a	d	c	0
c	d	d	0
d	f	a	1
f	f	a	0



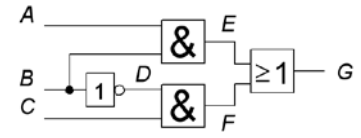
10.12

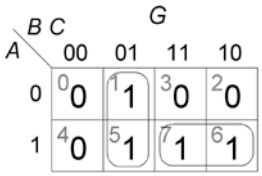


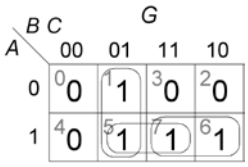


Asynkrona sekvensnät

11.1



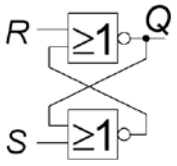


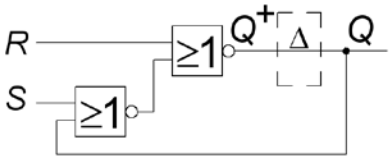


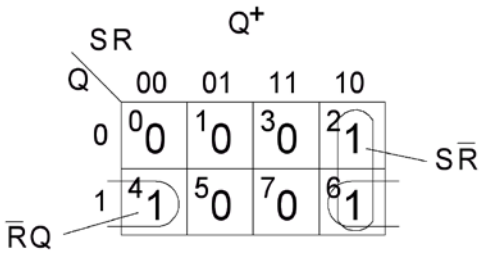
$G = \overline{B}C + AB \quad \{Hazardfritt\} \quad G = \overline{B}C + AB + AC$

11.2

Till vänster i figuren visas en SR-låskrets som två ”korskopplade” grindnät. Till höger är kretsen omritad som en Moore-automat.



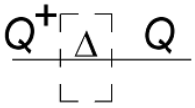




$Q^+ = \overline{R + \overline{S + Q}} = \overline{R} \cdot \overline{\overline{S + Q}} = \overline{R} \cdot (S + Q) = S\overline{R} + \overline{R}Q$

Den kodade tillståndstabellen brukar kallas för **excitationstabell** när man arbetar med asynkrona tillstånds-maskiner.

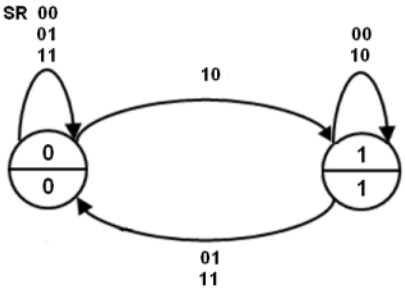
Nuvarande tillstånd Q	Nästa tillstånd Q⁺			
	Insignaler SR			
	00	01	11	10
0	(0)	(0)	(0)	1
1	(1)	0	0	(1)



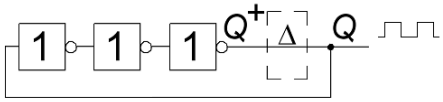
För varje insignal (kolumn) måste det finnas åtminstone något tillstånd där $Q = Q^+$. Sådana tillstånd är *stabila* och de brukar markeras genom att ringas in.
Tillståndsdiagrammet följer ur exitationstabellen.

Den okodade tillståndstabellen brukar kallas för **flödestabell** när det gäller asynkrona sekvensnät.

Nuvarande tillstånd Q	Nästa tillstånd Q⁺			
	Insignaler SR			
	00	01	11	10
A	(A)	(A)	(A)	B
B	(B)	A	A	(B)



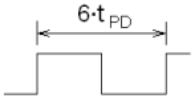
11.3



$Q^+ = \overline{Q}$

Q	Q^+
0	1
1	0

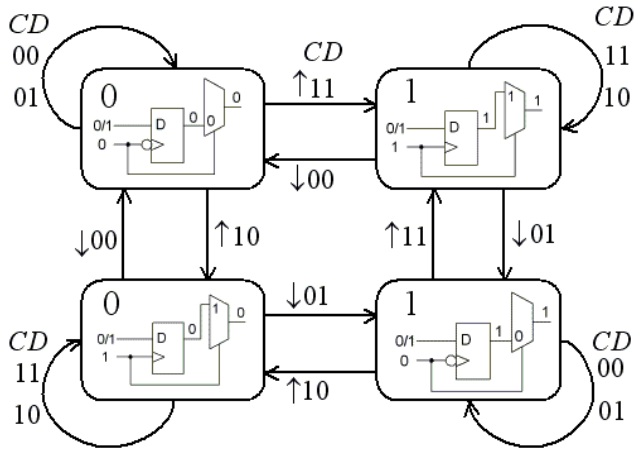
Inga stabila tillstånd!

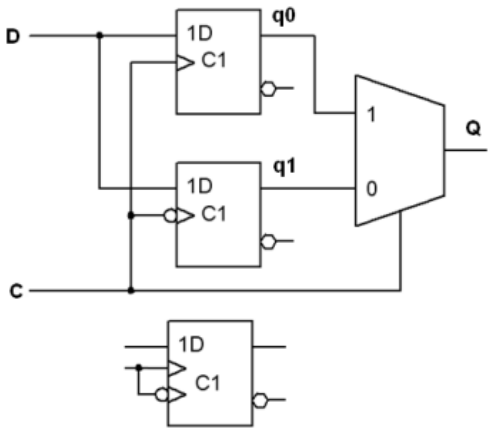


$T = 6 \cdot t_{PD} \quad \Rightarrow \quad f = \dfrac{1}{6 \cdot t_{PD}}$

Sifferexempel: $t_{pd} = 5 \cdot 10^{-9} \quad f = \dfrac{1}{6 \cdot 5 \cdot 10^{-9}} = 33 \text{ MHz}$

11.4





- Vid positiv flank ↑ går **C** från 0 till 1 och **C**=1 kopplar den övre **q0** vippan till utgången.
- Vid negativ flank ↓ går **C** från 1 till 0 och **C**=0 kopplar den undre **q1** vippan till utgången.

Resultatet blir en **D**-vippa som som reagerar på klockans *båda* flanker.

11.5

Det finns åtta kombinationer som kan inträffa för de fyra ingångsvärdeskombinationerna (CD) och de två utgångsvärdeskombinationerna O:

Möjliga input/output kombinationer

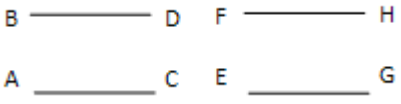
Present state		Next state		Kommentar
State tag	CDO	(CDO)+	(CDO)+	
A	000	010	100	Utgången får D's värde när C ändrar värde
B	001	011	100	
C	010	000	111	Ingen ändring av O när D ändrar värde
D	011	001	111	
E	100	000	110	
F	101	000	111	
G	110	011	100	
H	111	011	101	

Flödestabell (stabile states markerade som bold font)

Present state	Next State (CD)				Output
	00	01	11	10	
A	**A**	C	-	E	0
B	**B**	D	-	E	1
C	A	**C**	H	-	0
D	B	**D**	H	-	1
E	A	-	G	**E**	0
F	A	-	H	**F**	1
G	-	D	**G**	E	0
H	-	D	**H**	F	1

Vi ser direkt att ingen minimering kan göras genom att formera ekvivalensklasser eftersom all åtta tillstånden har olika utgångar där de har stabila tillstånd och där de har don't cares I tabellen.

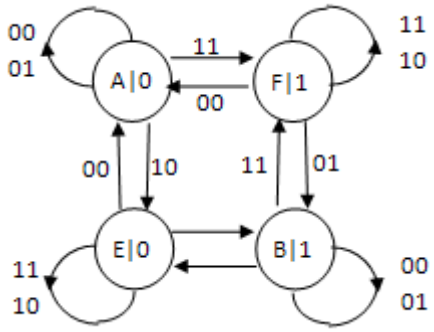
Merger-diagrammet blir som följer:



Den minimerade flödestabellen blir sålunda

Flow Table (stable states marked as bold)

Present state	Next State (CD)				Output
	00	01	11	10	
A	**A**	**A**	F	E	0
B	**B**	**B**	F	E	1
E	A	B	**E**	**E**	0
F	A	B	**F**	**F**	1



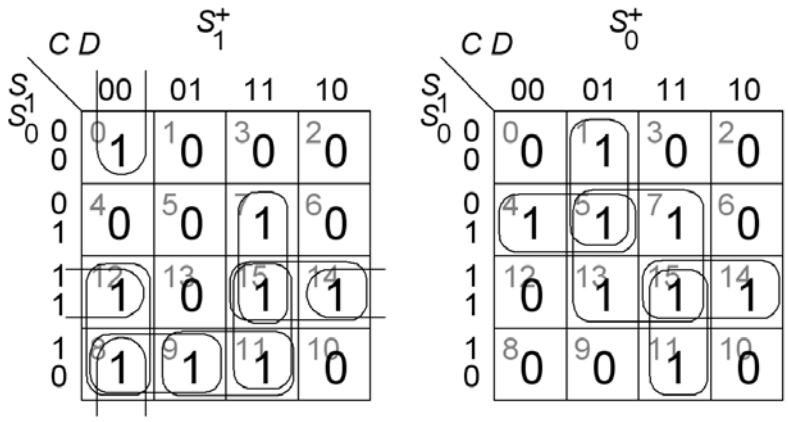
c) Tilldela states, gör Karnaugh-minimering och ta fram booleska tilldelningar. De möjliga state tilldelningarna är (E=00, B=01, A=10, F=11) plus deras rotationer och spegellösningar:

Possible state assignments			
A	F	B	E
00	01	11	10
01	11	10	00
11	10	00	01
10	00	01	11
11	01	00	10
01	00	10	11
00	10	11	01
10	11	01	00

Den resulterande flödestabellen blir då

Flow Table (stable states marked as bold)					
Present state	Next State (CD)				Output
	00	01	11	10	
10	10	10	11	00	0
01	01	01	11	00	1
00	10	01	00	00	0
11	10	01	11	11	1

Och de motsvarande Karnaugh-diagrammen och booleska uttrycken blir:



$$S_1^+ = CD(S_1 + S_0) + \overline{C}\,\overline{D}(S_1 + \overline{S_0}) + S_1 S_0 (C + \overline{D})$$

$$S_0^+ = S_0 D + \overline{S_1}\, S_0 \overline{C} + \overline{S_1}\,\overline{C}\, D + S_1 CD + S_1 S_0 C$$

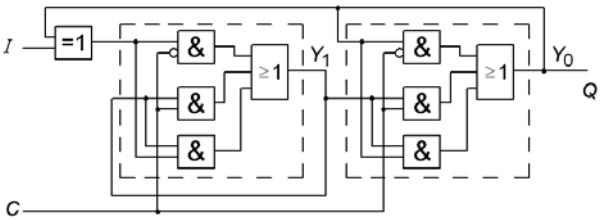
$$O = S_0$$

11.6

a) Härled de boolska uttrycken för tillståndsvariablerna. Svar:

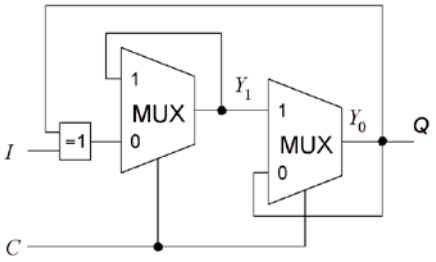
$$Y_0^+ = Y_0 Y_1 + Y_0 \overline{C} + Y_1 C$$

$$Y_1^+ = Y_1 (Y_0 \oplus I) + (Y_0 \oplus I)\overline{C} + \overline{Y_1} C$$



b) Härled exitationstabellen. (Ledning! Vilken funktion finns i de två innersta looparna)

De två inre looparna är hazard-fria MUX:ar!



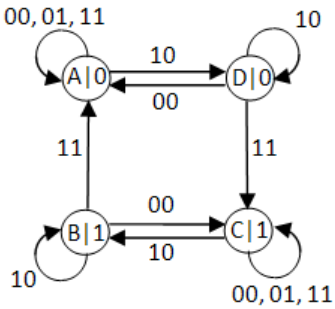
Pres. State	Next State IC=				Q
Y_1Y_0	00	01	11	10	
00	**00**	**00**	**00**	10	0
01	11	~~00~~	00	**01**	1
11	**11**	**11**	**11**	01	1
10	00	~~11~~	11	**10**	0

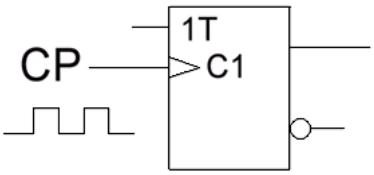
*Stable states marked in **Bold***
Impossible transitions marked as ~~strikethrough~~

Härled flödestabell, tilldela symboliska states och rita FSM:en.

Pres. State	Next State IC=				Q
Y_1Y_0	00	01	11	10	
A	**A**	**A**	**A**	D	0
B	C	~~A~~	A	**B**	1
C	**C**	**C**	**C**	B	1
D	A	~~C~~	C	**D**	0

*Stable states marked in **Bold***
Impossible transitions marked as ~~strikethrough~~

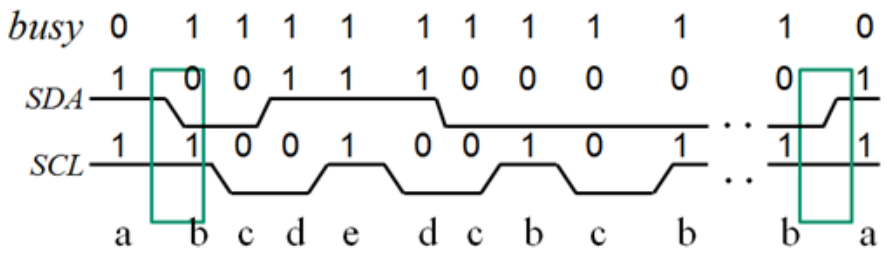




Identifiera funktionen hos den asynkrona kretsen. Vilken vippa motsvarar den?

- Positiv flank-triggad T-vippa.

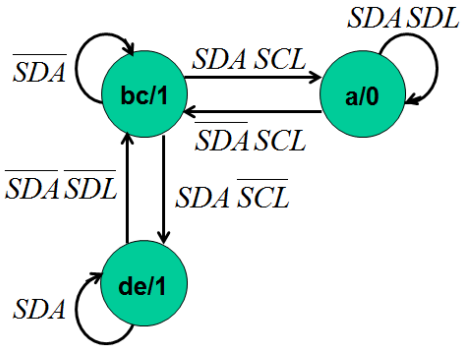
11.7



	SDA SCL 00	01	11	10	q
a	–	b	(a)	*	0
b	c	(b)	a	–	1
c	(c)	b	–	d	1
d	c	–	e	(d)	1
e	–	*	(e)	d	1

Följ tidsdiagrammet och inför nytt tillstånd vid varje händelse som inte varit med tidigare. I tillstånd *a* ”väntar” vi på startflanken (*b*), då är insignalen 10 omöjlig (markerad med *). Protokollet förbjuder ändring av data SDA när SCL är **hög**. Därför är insignal 01 omöjlig i tillstånd *e* (markerad med *). Detta ger två extra don’t care positioner i tabellen. Man ser sedan direkt vilka tillstånd som kan slås ihop.

	SDA SCL 00	01	11	10	q
a	–	bc	(a)	–	0
bc	(bc)	(bc)	a	de	1
de	bc	–	(de)	(de)	1



Som tillståndskod kan Gray-kod användas. *a* 00, *bc* 01, *de* 11, och *x* 10. *x* kan användas som don’t care bortsett från 10.

q_1q_0	SDA SCL 00	01	11	10	busy
00	–	01	00	–	0
01	01	01	00	11	1
11	01	–	11	11	1
10	≠10	≠10	≠10	≠10	–

$busy = q_0$

$q_1^+ q_0^+$

q_1q_0	SDA SCL 00	01	11	10
00	–	0	0	–
01	0	0	0	1
11	0	–	1	1
10	–	–	–	–

$q_1^+ = SDA(\overline{SCL} + q_1)$

q_1q_0	SDA SCL 00	01	11	10
00	–	1	0	–
01	1	1	0	1
11	1	–	1	1
10	–	–	≠10	–

$q_0^+ = (\overline{SDA} + \overline{SCL} + q_1)$

Näten bildar sammanhängande områden i Karnaughdiagrammen och är därför hazardfria (om näten har två nivåer). Att realisera med valfria grindar.

Avkodning av minnen och I/O-kretsar

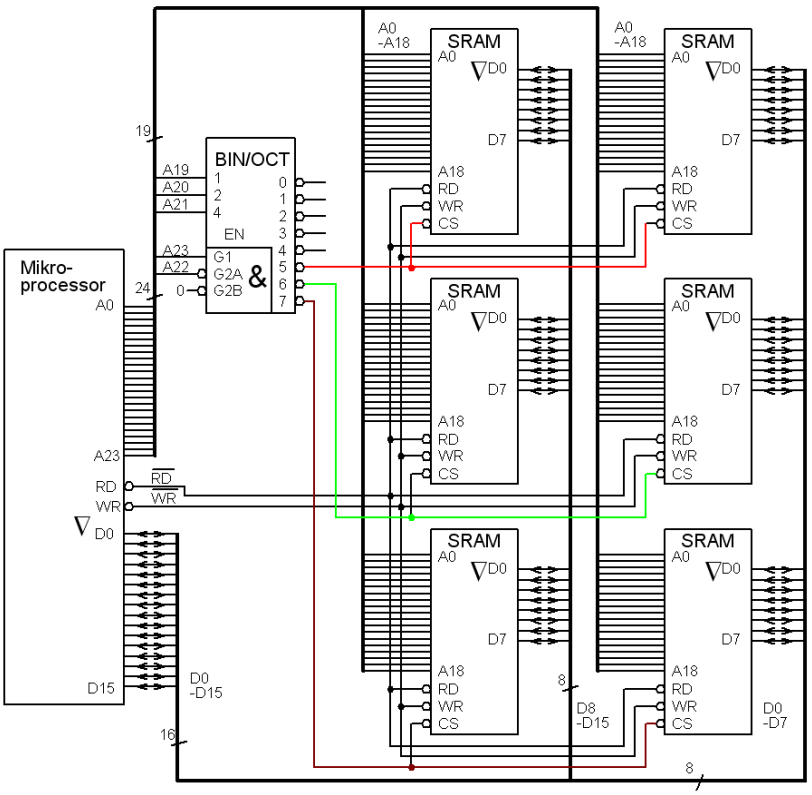
12.1

Ett dynamiskt RAM-minne består av ett antal 256Mbit minneskapslar som är organiserade som 32 M×8.
a) Hur många kapslar behövs för 256M×64?
Minne N = 256M M = 64 bitar. **Kapsel** p = 32M q = 8 bitar.
Antal kolumner $k = M/q$ = 64/8 = 8.
Antalet rader $r = N/p$ = 256M/32M = 8.
Antal kapslar $K = r \times k$ = 8×8 = 64.

b) Hur många kapslar behövs för 512M×72? (Vad kan anledningen till den ”underliga” bitbredden ”72” vara?)
Minne N = 512M M = 72 bitar. **Kapsel** p = 32M q = 8 bitar.
Antal kolumner $k = M/q$ = 72/8 = 9.
Antalet rader $r = N/p$ = 512M/32M = 16.
Antal kapslar $K = r \times k$ = 9 ×1 6 = 144.
Den ”ovanliga” bitbredden 72 (= 64 + 8). De 8 *extra* bitarna används för att korrigera enkelfel, och för att kunna upptäcka dubbelfel.

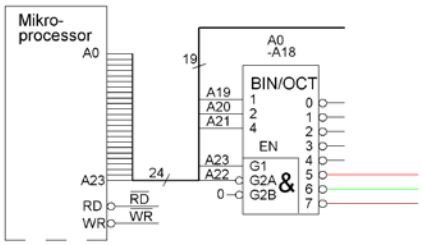
12.2

En viss 16 bitars processor kan adressera 24 bitar. Minnesrymden fördelas mellan ROM, SRAM och IO-kretsar.
Adressavkodningen sker med hjälp av en 3:8-avkodare.
a) Hur stort är figurens RAM? Vilket är adressområdet uttryckt i hexadecimala siffror?



- **Minneskapsel**:
p = 512k q = 8 bitar
- **Minne**:
r = 3 k = 2 K = 2 × 3 = 6
$M = k \times q$ = 2 × 8 = 16 bitar
N = p × r = 512k × 3 = 1,5M

Adressområde:



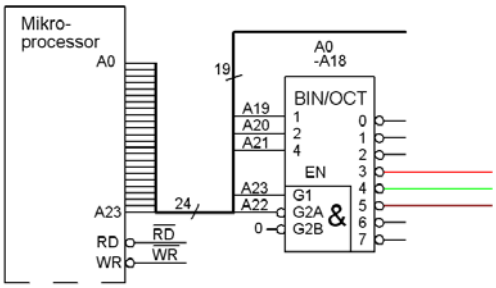
Computer:	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decoder:	1	0	0 … 7																					
Mem start:	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
• Begin hex	A					8			0				0				0				0			
Mem end:	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
• End hex	B					F			F				F				F				F			

b) Hur ändrar man adressområdet till 980000 – AFFFFF ?

980000
1001|1000|0000|0000|0000|0000|

AFFFFF
1010|1111|1111|1111|1111|1111|

”10|011” ⟶ ”3”
”10|101” ⟶ ”5”

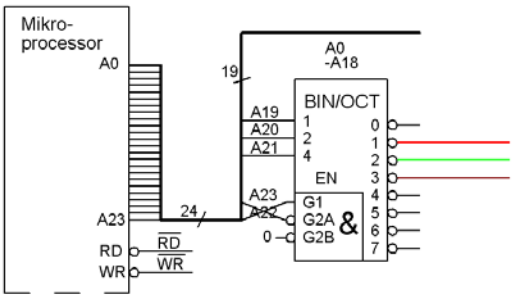


c) Ändra till adressområde 480000 – 5FFFFF ?

480000
0100|1000|0000|0000|0000|0000|

5FFFFF
0101|1111|1111|1111|1111|1111|

”01|001” ⟶ ”1”
”01|011” ⟶ ”3”

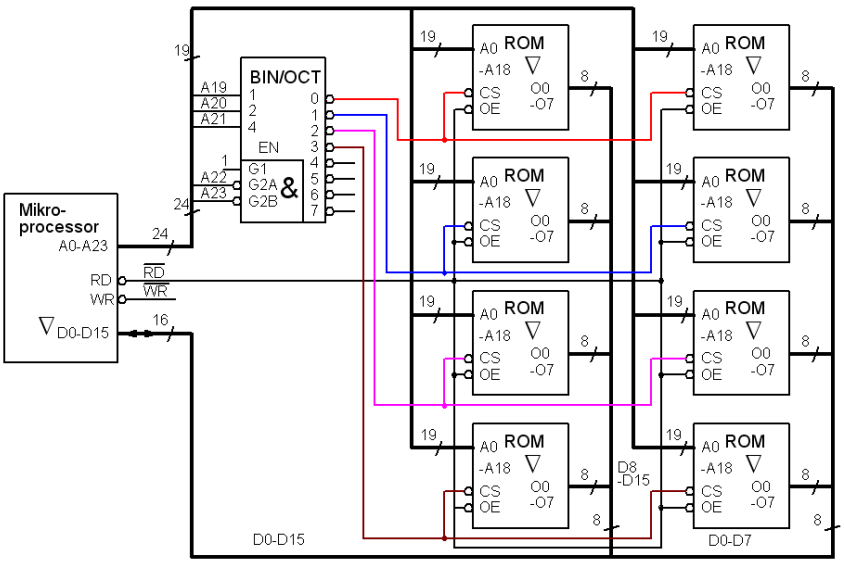


Vi byter plats på A23 och A22 !

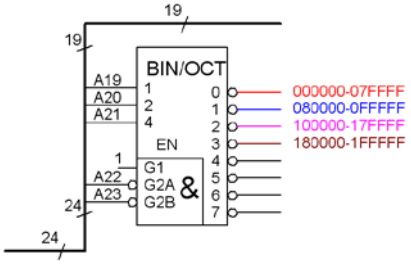
d) ROM-minnet är 2M×16 bitar och att adressområdet är 000000 … och framåt. ROM Chip är 512k×8.

- Hur många kapslar behövs?
- Hur skall avkodaren anslutas?
- Hur skall minneskretsarna anslutas?
- Ange adressområdena för avkodarens utgångar med hexadecimala siffror.

Minne:
N = 2 M (4·512k) ordlängden M = 16 bitar
Minneskapsel:
p = 512 k ordlängden q = 8 bitar
- Antalet kapselrader $r \le N/p$ = 4·512k/512k = 4
- Antalet kapselkolumner $k \ge M/q$ = 16/8 = 2
- Antalet kapslar $K = r \times k = 4 \times 2 = 8$



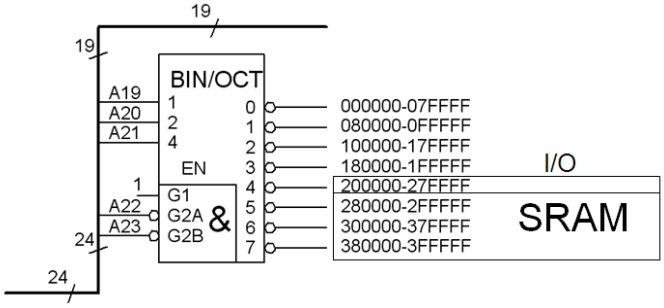
```
00ab|cmmm|mmmm|mmmm|mmmm|mmmm
0000|0000|0|0|0|0 - 0000|0111|F|F|F|F    000000-07FFFF
0000|1000|0|0|0|0 - 0000|1111|F|F|F|F    080000-0FFFFF
0001|0000|0|0|0|0 - 0001|0111|F|F|F|F    100000-17FFFF
0001|1000|0|0|0|0 - 0001|1111|F|F|F|F    180000-1FFFFF
```



Totalt ROM 000000 – 1FFFFF

e) Vilket adressområde blir ledigt för SRAM och IO-kretsar?

```
00ab|cmmm|mmmm|mmmm|mmmm|mmmm
0010|0000|0|0|0|0 - 0010|0111|F|F|F|F    200000-27FFFF
0010|1000|0|0|0|0 - 0010|1111|F|F|F|F    280000-2FFFFF
0011|0000|0|0|0|0 - 0011|0111|F|F|F|F    300000-37FFFF
0011|1000|0|0|0|0 - 0011|1111|F|F|F|F    380000-3FFFFF
```



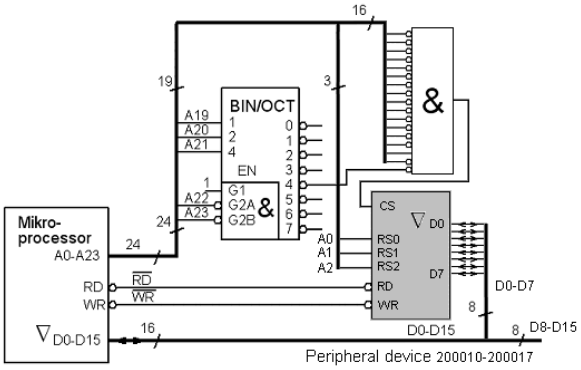
Möjliga SRAM+I/O adresser 200000 – 3FFFFF

12.3

a) Anslut en 8 registers minnesmappad periferienhet (I/O) till en CPU. CPU:n har 16 bitars databuss (vi använder bara 8), och en 24 bitars adressbuss. Använd en 3:8-avkodare och vid behov grindar. Periferienheten skall kopplas in så att den får adresserna `0x200010` … `0x200017`. (Jämför med föregående uppgift).

```
0x200010 = 0010|0.000|0000|0000|0001|0.000
0x200011 = 0010|0.000|0000|0000|0001|0.001
0x200012 = 0010|0.000|0000|0000|0001|0.010
0x200013 = 0010|0.000|0000|0000|0001|0.011
0x200014 = 0010|0.000|0000|0000|0001|0.100
0x200015 = 0010|0.000|0000|0000|0001|0.101
0x200016 = 0010|0.000|0000|0000|0001|0.110
0x200017 = 0010|0.000|0000|0000|0001|0.111
```

I/O adresser, 200000 – 27FFFF finns enligt tidigare uppgift på 3:8-avkodarens utgång ”4”. Den avkodar A23... A19, periferienheten avkodar själv A2 ... A0, resten får avkodas med en och-grind.



b) Vad är ofullständig avkodning?

För fullständig avkodning använde vi en &-grind med 17 ingångar! I bland gör man en **ofullständig avkodning**. Man struntar då i att ta med alla adress-signalerna och kan därmed använda en grind med färre ingångar. I/O-enhetens adressering blir mångtydig, den kan adresseras med många olika adresser, men den som skriver programkoden bestämmer ju själv vilka adresser det är som används. Huvudsaken är att man ser till att I/O-enhetens adresser *inte kolliderar* med någon annan enhets adresser.