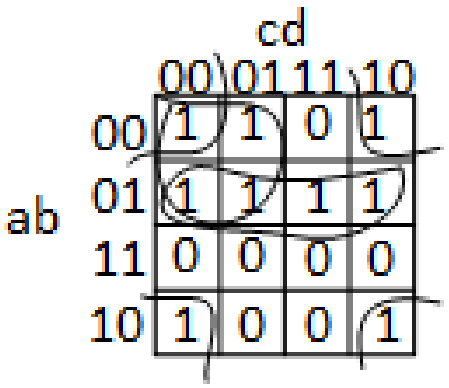
Uppgift 1:

a) u= a’c’ + a’bc+ ab’d’+b’cd’

b)

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

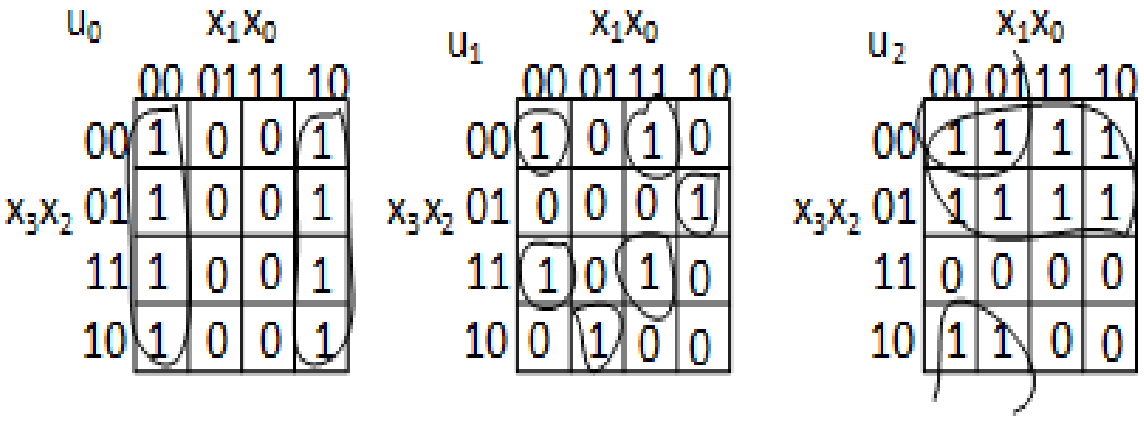
C)



u= b’d’+a’b+a’c’

Uppgift 2:

X_3	X_2	X_1	X_0		U_2	U_1	U_0
0	0	0	0		1	1	1
0	0	0	1		1	0	0
0	0	1	0		1	0	1
0	0	1	1		1	1	0
0	1	0	0		1	0	1
0	1	0	1		1	0	0
0	1	1	0		1	1	1
0	1	1	1		1	0	0
1	0	0	0		1	0	1
1	0	0	1		1	1	0
1	0	1	0		0	0	1
1	0	1	1		0	0	0
1	1	0	0		0	1	1
1	1	0	1		0	0	0
1	1	1	0		0	0	1
1	1	1	1		0	1	0

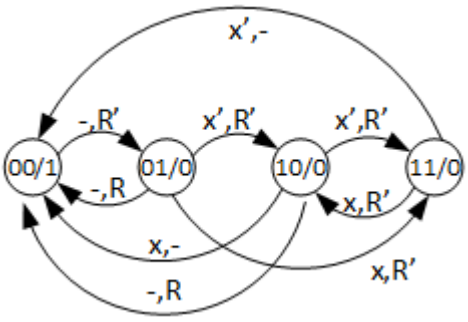


U0 = x0’

U1 = x3’x2’x1’x0’ + x3’x2’x1x0 + x3’x2x1x0’ + x3x2x1’x0’ + x3x2x1x0 + x3x2’x1’x0

U2 = x3’ + x2’x1’

Uppgift 3:



	q0+	q1,q0		
	00	01	11	10
00	1	0	0	1
x,R 01	0	0	0	0
11	0	0	0	0
10	1	1	0	0

	q1+	q1,q0		
	00	01	11	10
00	0	1	0	1
x,R 01	0	0	0	0
11	0	0	0	0
10	0	1	1	0

x	R	q_1	q_0		q_1^+	q_0^+	u
0	0	0	0		0	1	1
0	0	0	1		1	0	0
0	0	1	0		1	1	0
0	0	1	1		0	0	0
0	1	0	0		0	0	1
0	1	0	1		0	0	0
0	1	1	0		0	0	0
0	1	1	1		0	0	0
1	0	0	0		0	1	1
1	0	0	1		1	1	0
1	0	1	0		0	0	0
1	0	1	1		1	0	0
1	1	0	0		0	0	1
1	1	0	1		0	0	0
1	1	1	0		0	0	0
1	1	1	1		0	0	0

u = $q_1'q_0'$ (alltså u='1' i tillstånd 00)

q_0^+ = **x'**R'q0' + xR'q1' = R'(x'q0' + xq1')

q_1^+ = x'R'q1q0' + R'q1'q0 + xR'q0 = R'(q1q0' + q1'q0 + xq0)

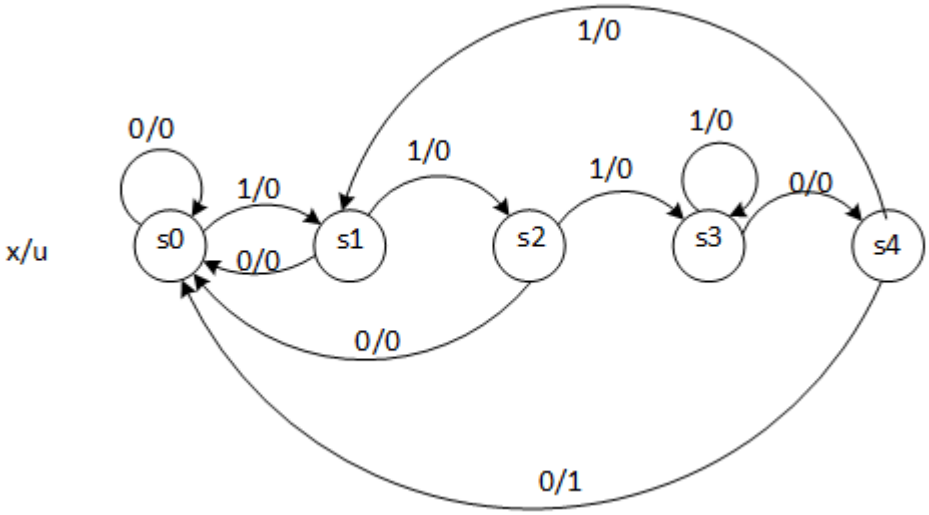
Med två insignaler blir det fyra vägar ut ur ett tillstånd. Om den ena insignalen är reset kan den antingen ingå i tillståndstabellen (som här) eller läggas till senare som q+ = R'(q_uttryck) + R(q=00).

Exempel 5.11 i Hemert, Digitala kretsar är ett exempel med två insignaler.

I tabellen är de första fyra raderna binärräknaren, nästa fyra är reset, rad 9-12 är Grayräknaren och de sista fyra är reset.

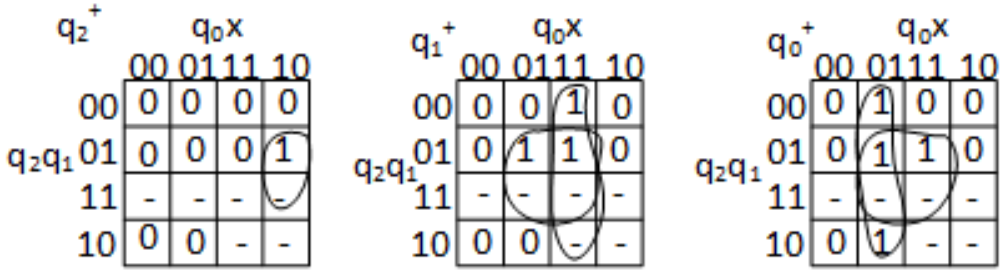
Synkron reset betyder att reset händer vid klockpulsen som de andra tillståndsändringarna. Vid asynkron reset sker reset omedelbart. I hårdvaran får det ske direkt på vippornas resetingång och detta hanteras då i ett separat nät för den ingången.

Uppgift 4:



Kodning: s0= 000, s1= 001, s2= 010, s3= 011, s4 =100.

Nuv.	Nästa	
q2q1q0	**x = 0**	**x = 1**
000	000/0	001/0
001	000/0	010/0
010	000/0	011/0
011	100/0	011/0
100	000/1	001/0



$q_2^+ = q_1 q_0 x'$

$q_1^+ = q_0 x + q_1 x$

$q_0 + = q_0' x + q_1 x$

$u = q_2 x'$

Uppgift 5:

1. Hur många bitar behövs för att lagra en instruktion?
   Svar: Instruktioner ligger i minnet på adress 00000, 00010, 00100, d v s på avstånd om 2 byte, vilket gör att varje instruktion har längden 2 byte.

2. Hur många bitar innehåller instruktionsregistret?
   Svar: I och med att varje instruktion är på 2 bytes, måste instruktionsregistret bestå av 16 bitar

3. Vilka instruktioner kommer exekvera?
   Svar: Dessa instruktioner kommer att exekvera:

ADRESS	INSTRUKTION	FÖRKLARING
00000	XOR R1, R1, R1	R1<-R1 XOR R1
00010	ADD R2, R1, 5	R2<-R1 + 5
00100	MUL R2, R1, R2	R2<-R1*R2
00110	BEZ 10000	IF ZERO BRANCH TO 10000
10000	AND R1, R1, R2	R1<- R1 AND R2

   Instruktionerna

01000	SUB R2, R2, 1	R2<-R2-1
01010	BR 10000	BRANCH TO 01110
01100	ADD R1, R1, 1	R1<- R1 ADD 1
01110	MUL R1, 5, 2	R1 <- 5 MUL 2

   kommer inte exekvera eftersom instruktionen:

00100	MUL R2, R1, R2	R2<-R1*R2

   kommer resultera i 0, vilket gör att ZERO flaggan sätts, och det gör att denna instruktion leder till ett hopp till adress 10000:

00110	BEZ 01100	IF ZERO BRANCH TO 10000

4. Vad som händer vid varje instruktion?

ADRESS	INSTRUKTION	FÖRKLARING
00000	XOR R1, R1, R1	R1<-R1 XOR R1

   Detta händer: Bitvis XOR med sig själv ger resultatet till 0. Alltså blir R1=0

00010	ADD R2, R1, 5	R2<-R1 + 5

   Detta händer: R2 får värdet R1 (som är 0) plus 5, d v s R2=5

00100	MUL R2, R1, R2	R2<-R1*R2

   Detta händer: R2 får värdet av R1 (som har värdet 0) multiplicerat med värdet av R2 (som har värdet 5), d v s R2 får värdet 0. Detta gör att ZERO flaggan sätts

00110 BEZ 01100 IF ZERO BRANCH TO 10000
Detta händer: Om ZERO flaggan är satt, och den sattes av tidigare instruktion, så kommer hopp ske till adress 10000. Nästa instruktion är alltså den på adress 10000.

10000 AND R1, R1, R2 R1 <- R1 AND R2
Detta händer: R1 får värdet av bitvis AND mellan innehållet i R1 och R2. R1 har värdet 0, d v s alla bitar är 0, och R2 har värdet 0, d v s alla bitar är 0. En bitvis AND operation resulterar i 0, d v s R1 får värdet 0.

5. Vad innehåll i programräknare och instruktionsregister vid varje instruktion som exekverats?

För instruktion:
00000 XOR R1, R1, R1 R1<-R1 XOR R1
gäller att:
Programräknaren innehåller: 00000
Instruktionsregistret innehåller maskinkoden som svarar mot: XOR R1, R1, R1

För instruktion:
00010 ADD R2, R1, 5 R2<-R1 + 5
gäller att:
Programräknaren innehåller: 00010
Instruktionsregistret innehåller maskinkoden som svarar mot: ADD R2, R1, 5

För instruktion:
00100 MUL R2, R1, R2 R2<-R1*R2
gäller att:
Programräknaren innehåller: 00100
Instruktionsregistret innehåller maskinkoden som svarar mot: MUL R2, R1, R2

För instruktion:
00110 BEZ 10000 IF ZERO BRANCH TO 10000
gäller att:
Programräknaren innehåller: 00110
Instruktionsregistret innehåller maskinkoden som svarar mot: BEZ 10000

För instruktion:
10000 AND R1, R1, R2 R1 <- R1 AND R2
gäller att:
Programräknaren innehåller: 10000
Instruktionsregistret innehåller maskinkoden som svarar mot: AND R1, R1, R2

Uppgift 6:

Antag ett datorsystem med ett cacheminne som består av 16 bytes och ett primärminne som består av 64 bytes. Cacheminnet är direktmappat med cacherader om 4 bytes.

1. Hur många bitar behövs för att specificera datorsystemets adressrymd?
   Svar: Primärminnet är på 64 bytes. Det behövs 6 bitar att beskriva tal mellan 0 och 63 eftersom 2^6=64
2. Vid exekvering av ett program har processorn läst adress 1 och vill läsa på adress 5, är det möjligt att avgöra om den läsningen kommer ge en hit eller en miss?
   Svar: Om processorn läst på adress 1 innebär det att det som finns i motsvarande block finns i cacheminnet. Varje block är på 4 bytes och detta block hamnar på adress 0, 1, 2 och 3. Hade vi läst adress 0, 1, 2, eller 3 skulle det bli en hit. Men, för adress 5 vet vi ingenting så vi kan inte avgöra om det blir en hit eller en miss.
3. I cacheminnet, när processorn läser i minnet, vilka adresser kan hamna i den första cacheraden?
   Svar: Eftersom blockstorleken är på 4 bytes så hamnar dessa 4 bytes på adress 0, 1, 2, och 3.
4. Lokalitet av referenser är en viktig princip för cacheminnen, vad säger principen?
   Svar: Lokalitet av referenser finns i rum och tid. Detta gäller instruktioner och data. Lokalitet i tid innebär att något som nyligen använts (tid) kommer med stor sannolikhet användas igen. Lokalitet i rum innebär att något som nyligen använts kommer annat som finns i närheten med stor sannolikhet användas igen.
5. Enligt principen om lokalitet av referenser, kan man säga något om storleken av block?
   Svar: När det gäller tid kan man inte säga så mycket men när det gäller rum, d v s när man använt något som nyligen använts kommer annat som finns i närheten med stor sannolikhet användas igen. Det leder till att man skulle kunna argumentera för stora cacherader/block.