FÖRELÄSNINGSANTECKNINGAR

DIGITALTEKNIK 3p – ETEA17

Sekvenskretsar

- Innehåll
  - Synkrona sekvenskretsar
  - Tillståndsdiagram / tillståndstabell
  - Definition av Moore- och Mealy-maskiner
  - Tillståndskodning
  - Syntes av sekventiell logik
  - Räknare

Sekvenskretsar – Exempel

Exempel: Bankomat

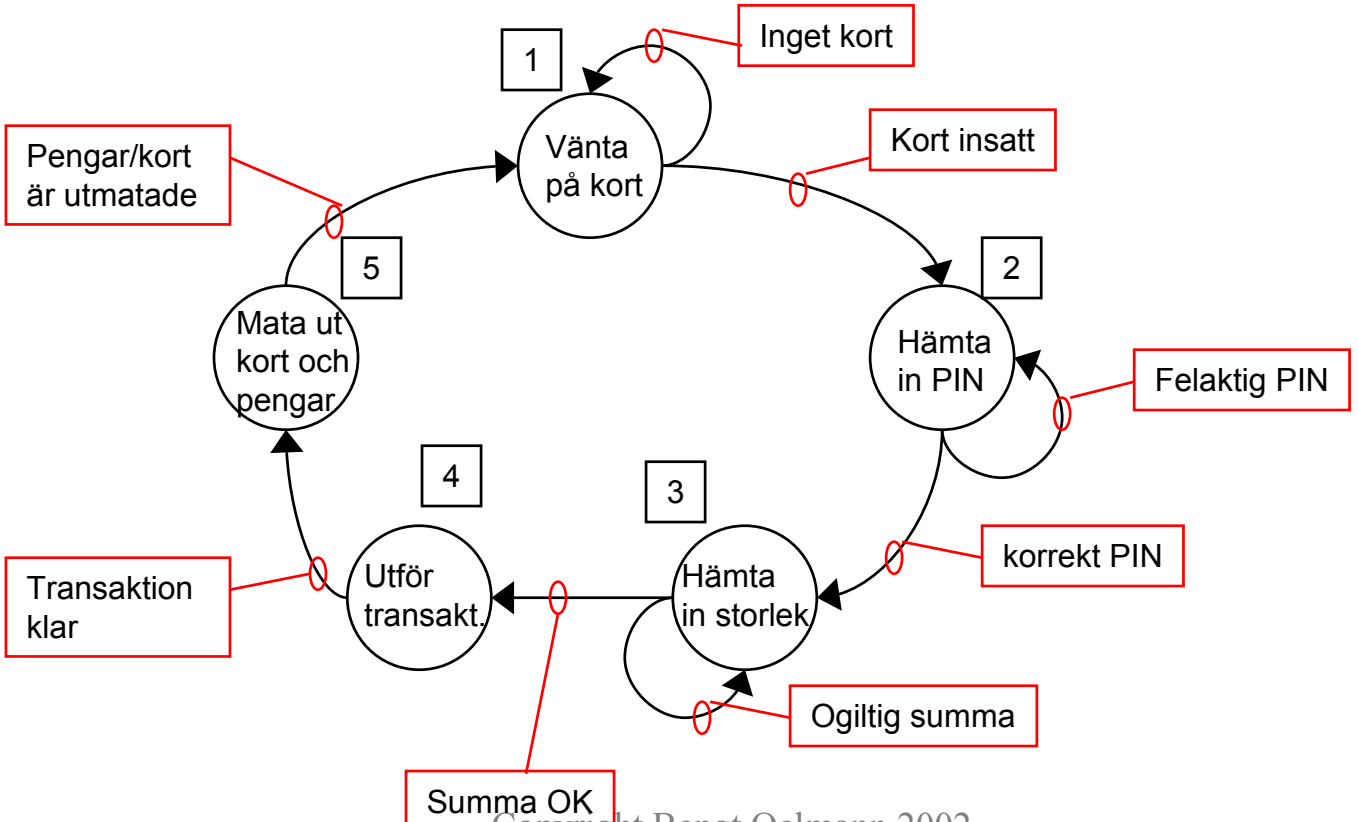
Sekvens av operationer för att göra ett uttag:

- Sätt in kortet
- Mata in PIN-kod
- Mata in storleken på uttaget
- Vänta på pengarna
- Ta ut kort och pengar

Forts. Exempel – Tillståndsdiagram

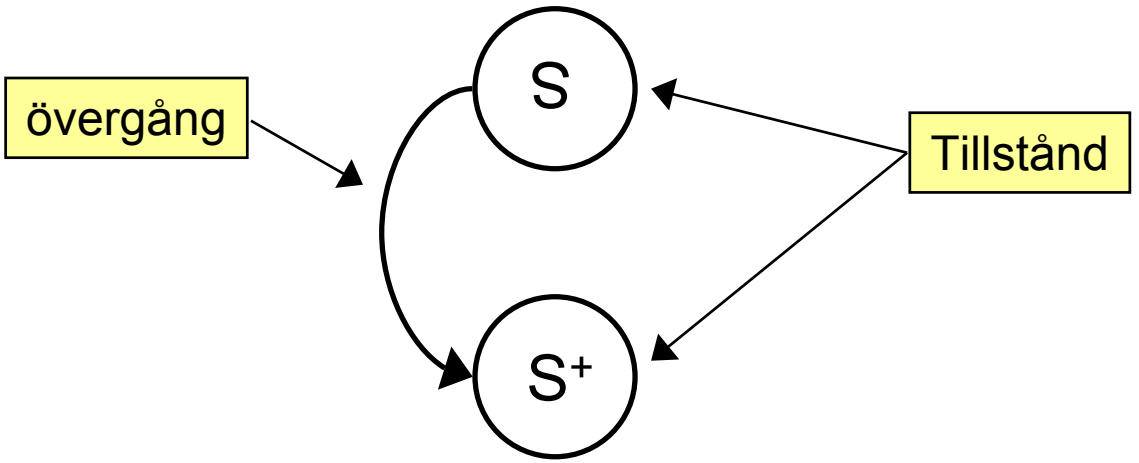
”Operationer” utförda av dig:
1. Sätt in kortet
2. Mata in PIN-kod
3. Mata in storleken på uttaget
4. Vänta på pengarna
5. Ta ut kortet och pengarna

”Operationer” utförda av maskinen
1. Vänta på kortet
2. Hämta in PIN-kod
3. Hämta in storleken på uttaget
4. Utför transaktionen
5. Mata ut kortet och pengarna



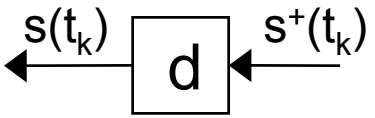
Tillståndsdiagram

- Tillståndsdiagram (*eng. State Transition Graph*)
  - Visar varje individuellt tillstånd
  - Samtliga möjliga sekvenser av tillstånd sekvensnätet kan ha



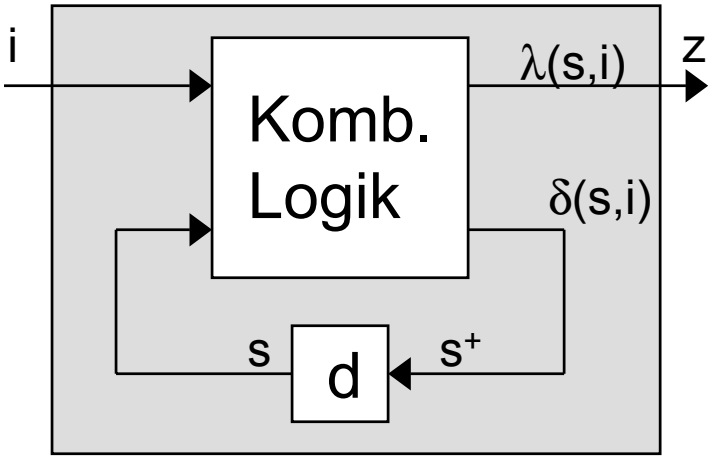
Tillståndsmaskin (*eng. Finite-State Machine*)

- Implementerar ett tillståndsdiagram
- Består av
  - Tillståndsminne – innehåller maskinens tillstånd (s)

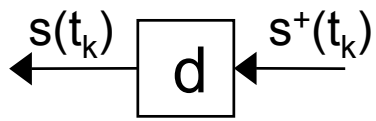


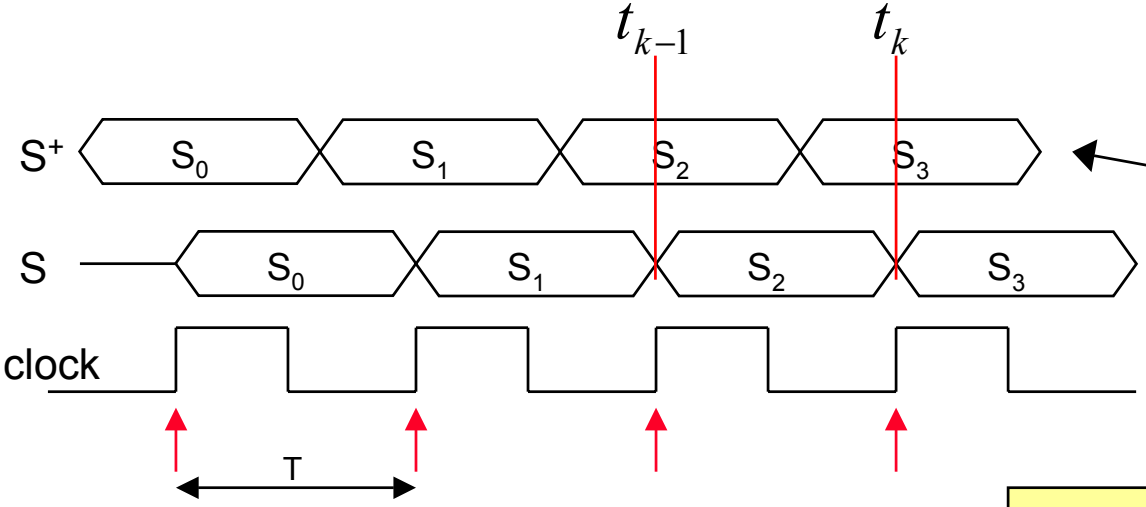
$$s(t_k) = s^+(t_{k-1})$$

  - Funktion för att beräkna nästa tillstånd (δ)
  - Funktion för att beräkna utgångarnas värde (λ)



Tillståndsminne



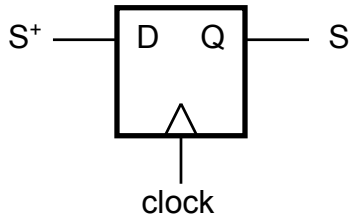


$$s(t_k) = s^+(t_{k-1})$$

Klocksignalen delar upp tiden i steg

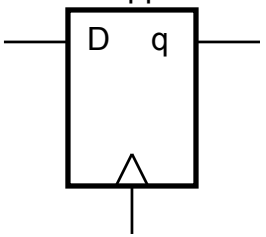
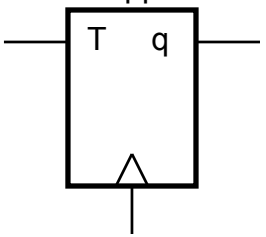
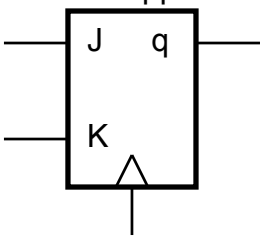
- ett tidsdiskret system
- minnet fördröjer signalen en klockcykel
- alla förändringar i minnet sker samtidigt på aktiv flank
- klockfrekvensen f = 1/T

Synkront D-element som tillståndsminne

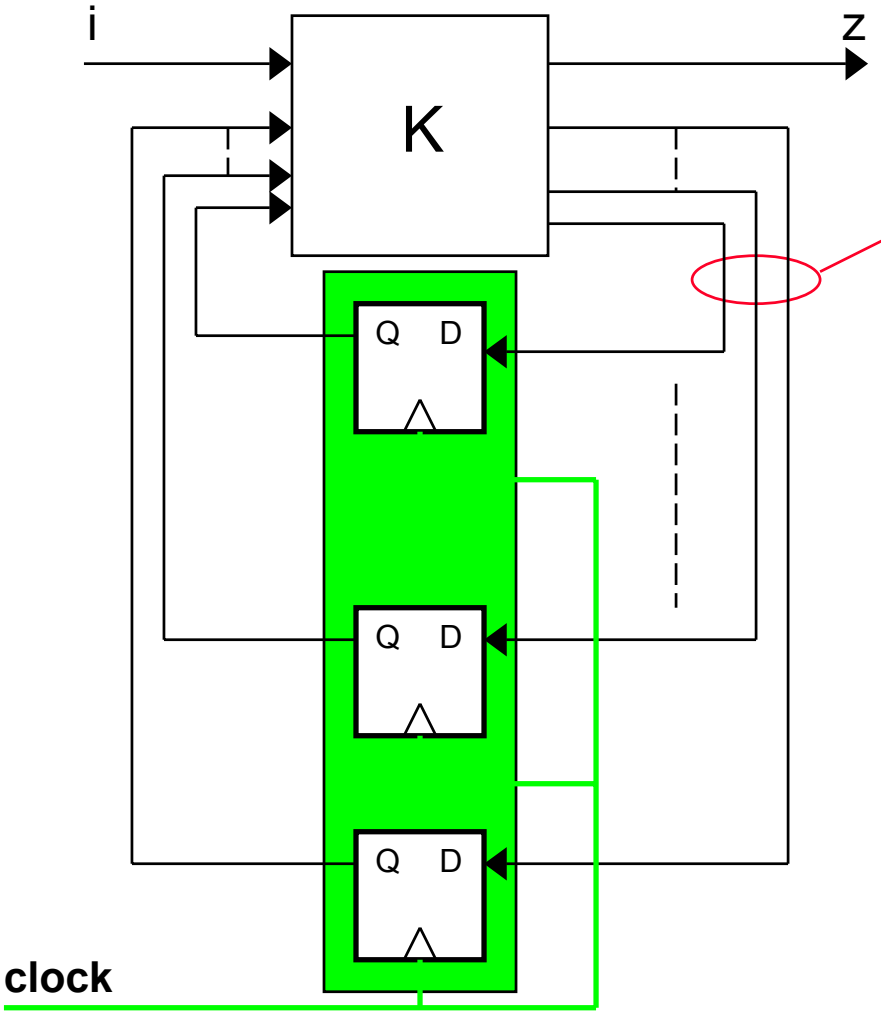


$Q^+ = D$

Typer av minneselement

Symbol	Karakteristisk ekvation	Syntestabell
D-vippa 	$q^+ = D$	q q+ D 0 0 0 0 1 1 1 1 1 1 0 0
T-vippa 	$q^+ = T \cdot \overline{q} + \overline{T} \cdot q$	q q+ T 0 0 0 0 1 1 1 1 0 1 0 1
JK-vippa 	$q^+ = J \cdot \overline{q} + \overline{K} \cdot q$	q q+ J K 0 0 0 - 0 1 1 - 1 1 - 0 1 0 - 1

Synkron tillståndsmaskin



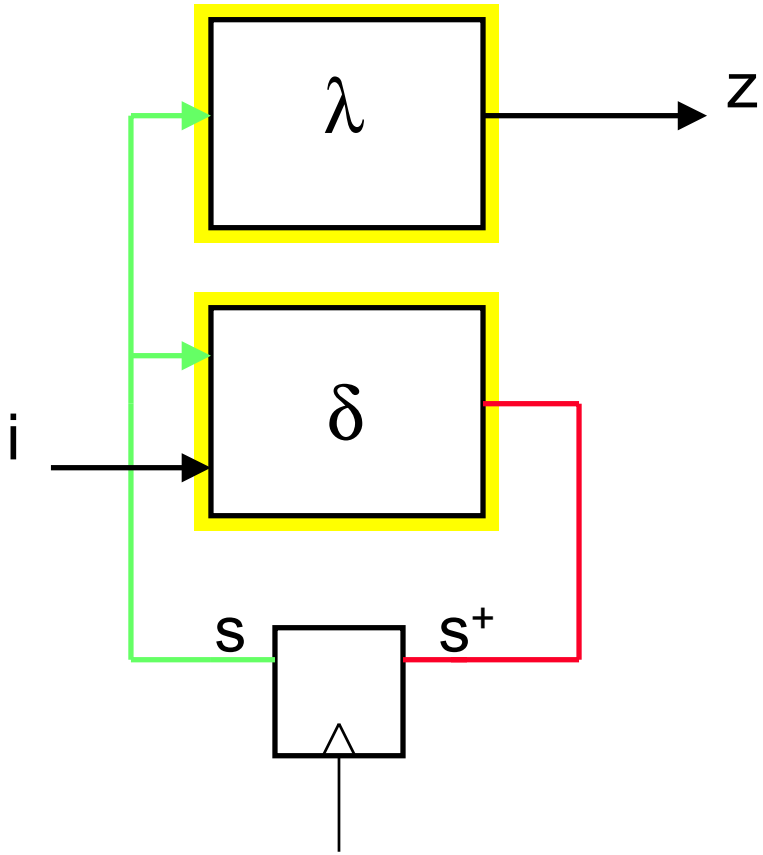
Tillståndet kodat i digitalt tal

- Binär kodning med n bitar kan representera 2^n tillstånd

Exempel på kodning:

Tillstånd	Kod
S0	00
S1	01
S2	10
S3	11

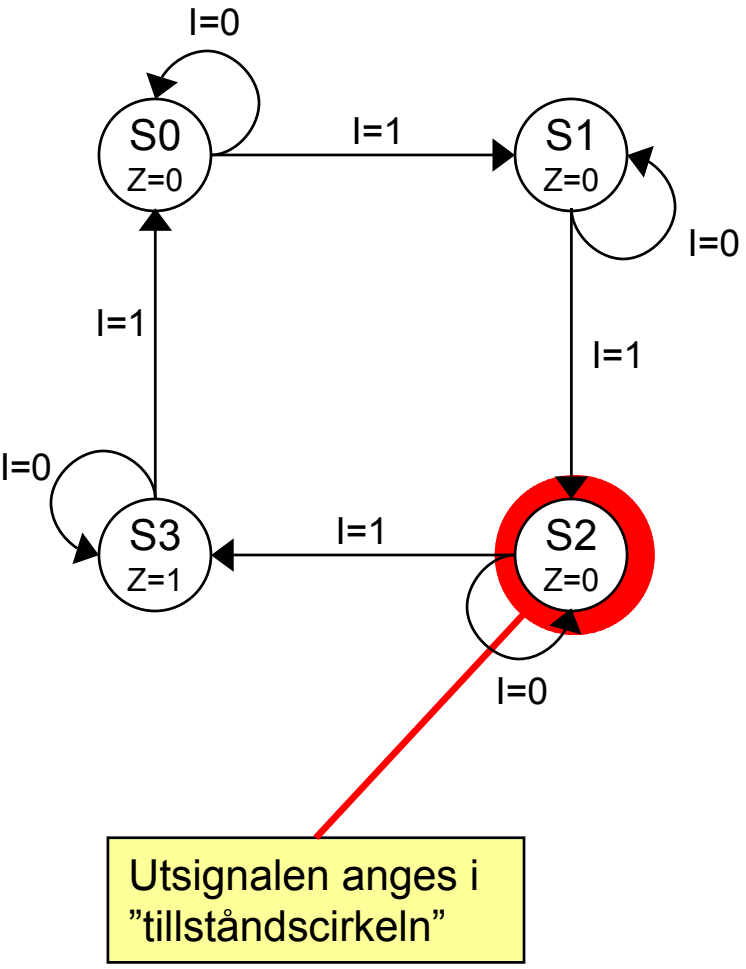
Moore-maskin – struktur



Utsignalfunktionen:
$z=\lambda(s)$
d.v.s: utgången beror endast av tillståndet s.

Nästatillstånds-funktionen:
$s^+=\delta(i,s)$
d.v.s: nästa tillstånd beror av värdet på ingångarna och nuvarande tillstånd s

Moore-maskin – tillståndsdiagram och tabell



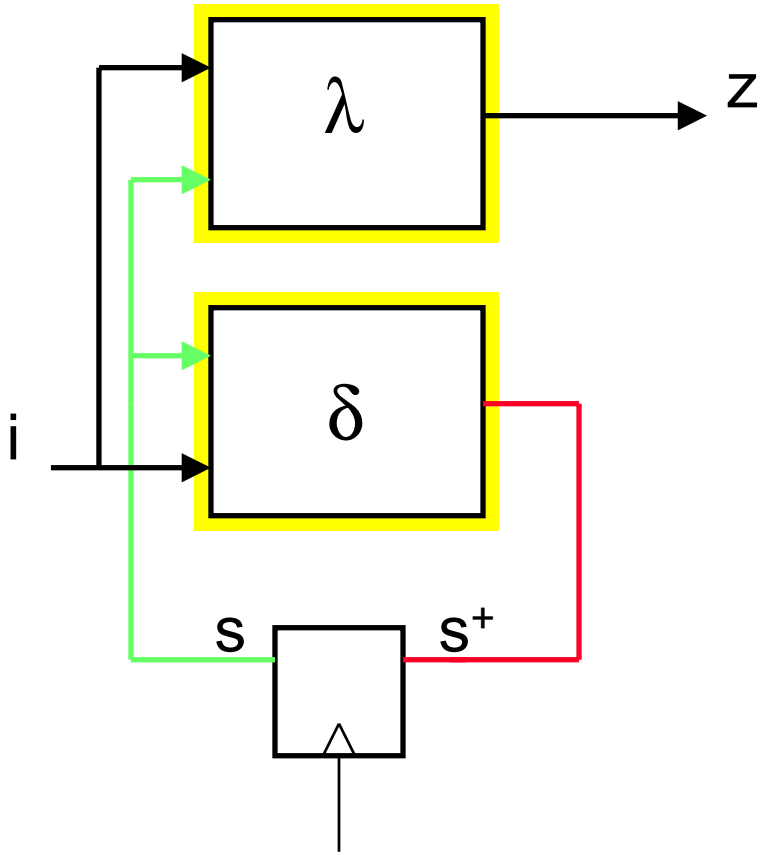
S	I = 0	I = 1	Z
S0	S0	S1	0
S1	S1	S2	0
S2	S2	S3	0
S3	S3	S0	1
		S^+	

Nuvarande tillstånd

Nästa tillstånd

Utsignalvärde
För ett givet tillstånd är det oberoende av I.

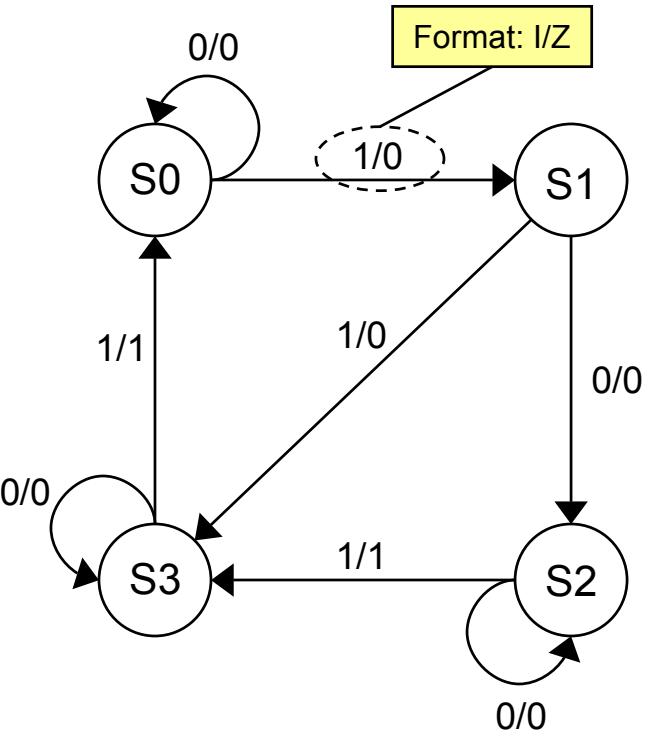
Mealy-maskin – struktur



Utsignalfunktionen:
$z=\lambda(i,s)$
d.v.s: utgången beror både av s och i.

Nästatillstånds-funktionen:
$s^+=\delta(i,s)$
d.v.s: nästa tillstånd beror av värdet på ingångarna och nuvarande tillstånd s

Mealy-maskin – tillståndsdiagram och tabell



≡

S	I = 0	I = 1
S0	S0,0	S1,0
S1	S3,0	S2,0
S2	S2,0	S3,1
S3	S3,0	S0,1

S^+, Z

Nuvarande tillstånd

För I=1 fås ett annat nästa tillstånd och utsignal

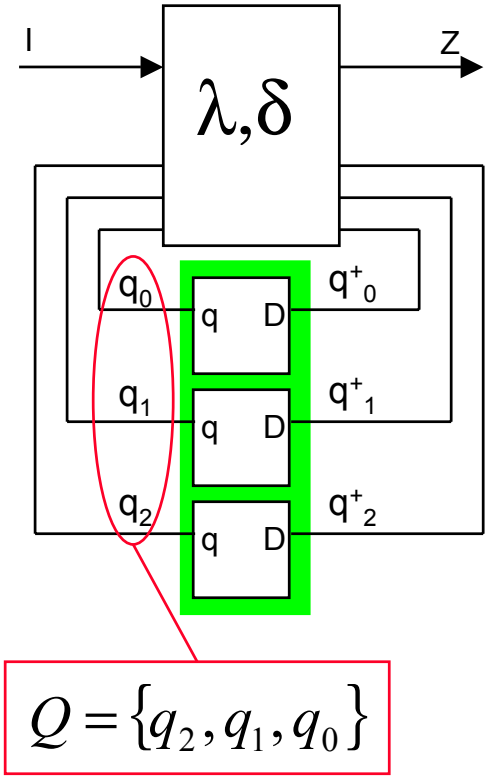
För I=0 fås ett nästa tillstånd och utsignal

Utgången Z är beroende av Nuvarande tillstånd OCH ingångsvärdet

Tillståndskodning

Tillstånd	Binär	Gray	One-hot	Almost One-hot
INIT	000	000	00001	0000
A0	001	001	00010	0001
A1	010	011	00100	0010
OK0	011	010	01000	0100
OK1	100	110	10000	1000
	$q_2q_1q_0$	$q_2q_1q_0$	$q_4q_3q_2q_1q_0$	$q_3q_2q_1q_0$

Symboliskt namn ges en binär kod



Q är det kodade tillståndet

Syntes av tillståndsmaskin – översikt

Specifikation

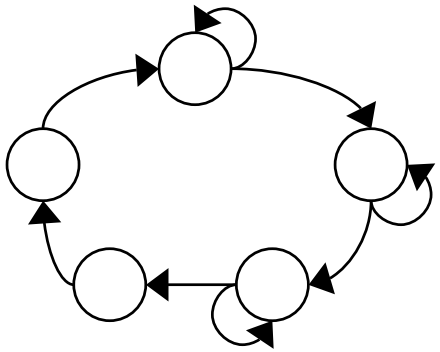
Bankomaten ska först …

.....

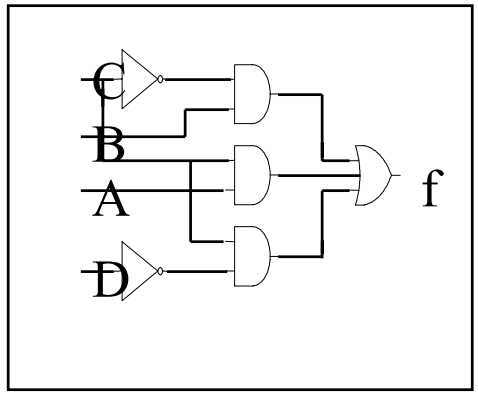
.....

.....

Otvetydig funktionell specifikation av tillståndsmaskinen



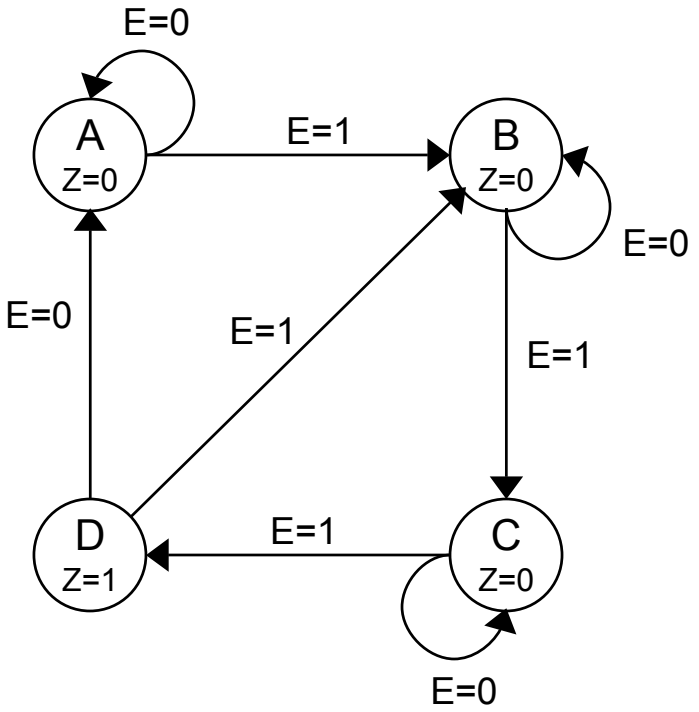
Kombinatoriska nät med logiska grindar



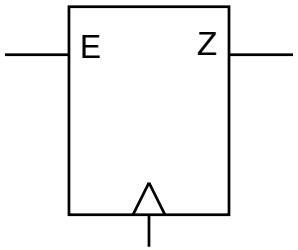
Procedur för syntes:

- Konstruera en tillståndstabell
- Tilldela varje tillstånd en kod (tillståndskodning)
- Konstruera en transitionstabell
- Bestäm minnestyp (vilken vippa ska användas)
- Konstruera en excitationstabell
- Ta fram logiska uttryck för λ och δ.
- Konstruera ett schema som visar grindar och vippor

Tillståndstabell



	E		
S	0	1	Z
A	A	B	0
B	B	C	0
C	C	D	0
D	A	B	1
	S^+		



En Moore-maskin

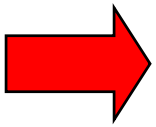
Transitionstabell

Tillstånd (S)	Binär (Q)
A	00
B	01
C	10
D	11
	q_1q_0

Tillstånden kodas med binär kod

S byts ut mot Q

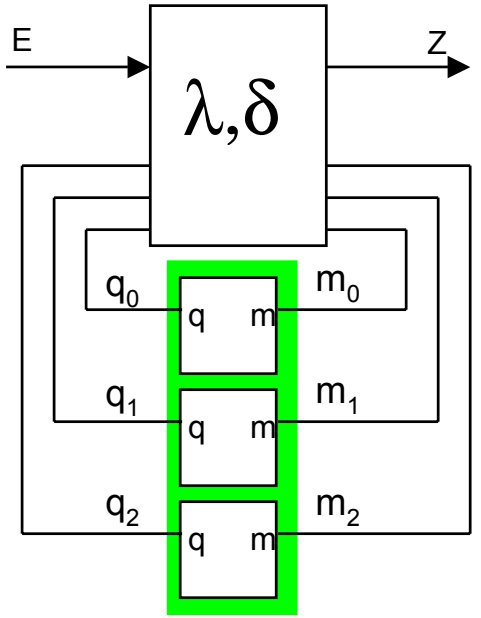
	E		
S	0	1	Z
A	A	B	0
B	B	C	0
C	C	D	0
D	A	B	1
	S^+		



Transitionstabell

	E		
Q	0	1	Z
00	00	01	0
01	01	10	0
10	10	11	0
11	00	01	1
	Q^+		

Excitationstabell



Transitionstabellen ger relationen mellan Q och Q^+

$$Q^+ = f(Q, E)$$

Egentligen vill vi veta relationen mellan Q och M, där M är insignalerna till minneselementen.

$$M = f(Q, E)$$

Specialfall: Då D-vippor används så är $M=Q^+$. Karakteristiska ekvationen är $Q^+=D$.

Transitionstabell

Q	E = 0	E = 1	Z
00	00	01	0
01	01	10	0
10	10	11	0
11	00	01	1
	Q^+		

Syntestabell för D-vippa

q	q+	D
0	0	0
0	1	1
1	1	1
1	0	0

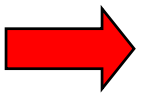
Excitationstabell

Q	E = 0	E = 1	Z
00	00	01	0
01	01	10	0
10	10	11	0
11	00	01	1
	D		

Excitationstabell för T-vippa

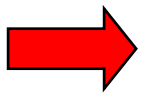
Transitionstabell

	E		
Q	0	1	Z
00	00	01	0
01	01	10	0
10	10	11	0
11	00	01	1
	Q^+		



Syntestabell för T-vippa

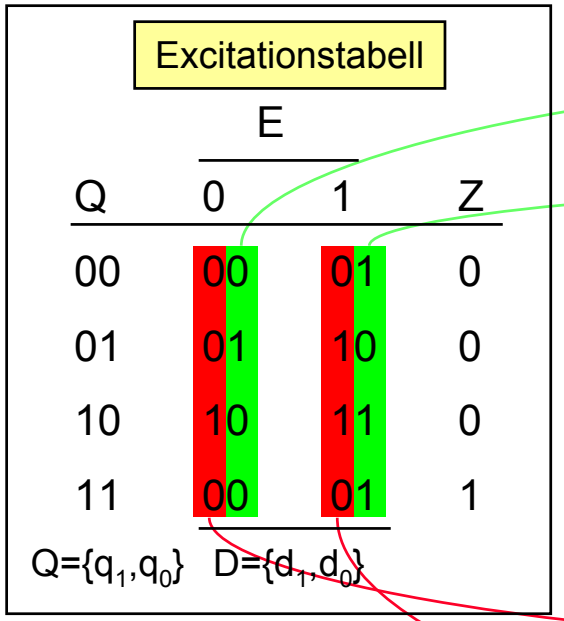
q	q+	T
0	0	0
0	1	1
1	1	0
1	0	1

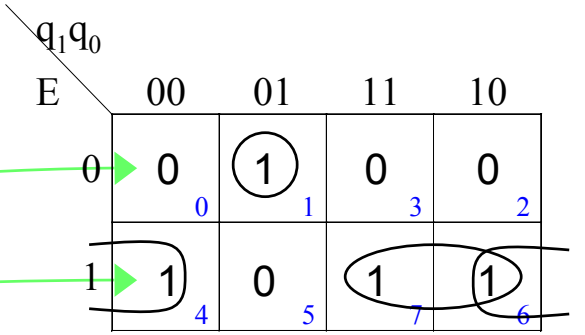


Excitationstabell

	E		
Q	0	1	Z
00	00	01	0
01	00	11	0
10	00	01	0
11	11	10	1
	T		

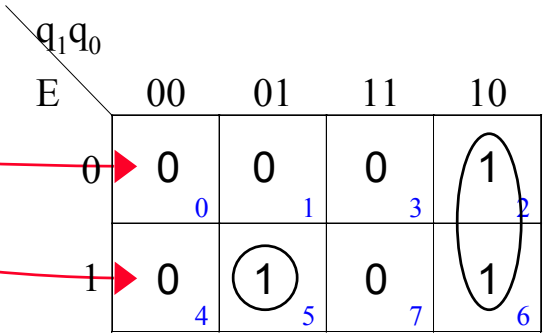
Logiskt uttryck för δ-funktionen då D-vippor används





$d_0=f(E,q_1,q_0)$

$$d_0 = \overline{E}\,\overline{q_1}q_0 + Eq_1 + E\overline{q_0}$$



$d_1=f(E,q_1,q_0)$

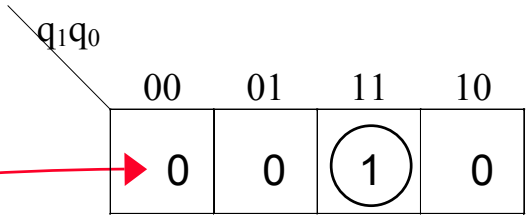
$$d_1 = E\overline{q_1}q_0 + q_1\overline{q_0}$$

Logiskt uttryck för λ-funktionen

Excitationstabell

	E		
Q	0	1	Z
00	00	01	0
01	01	10	0
10	10	11	0
11	00	01	1

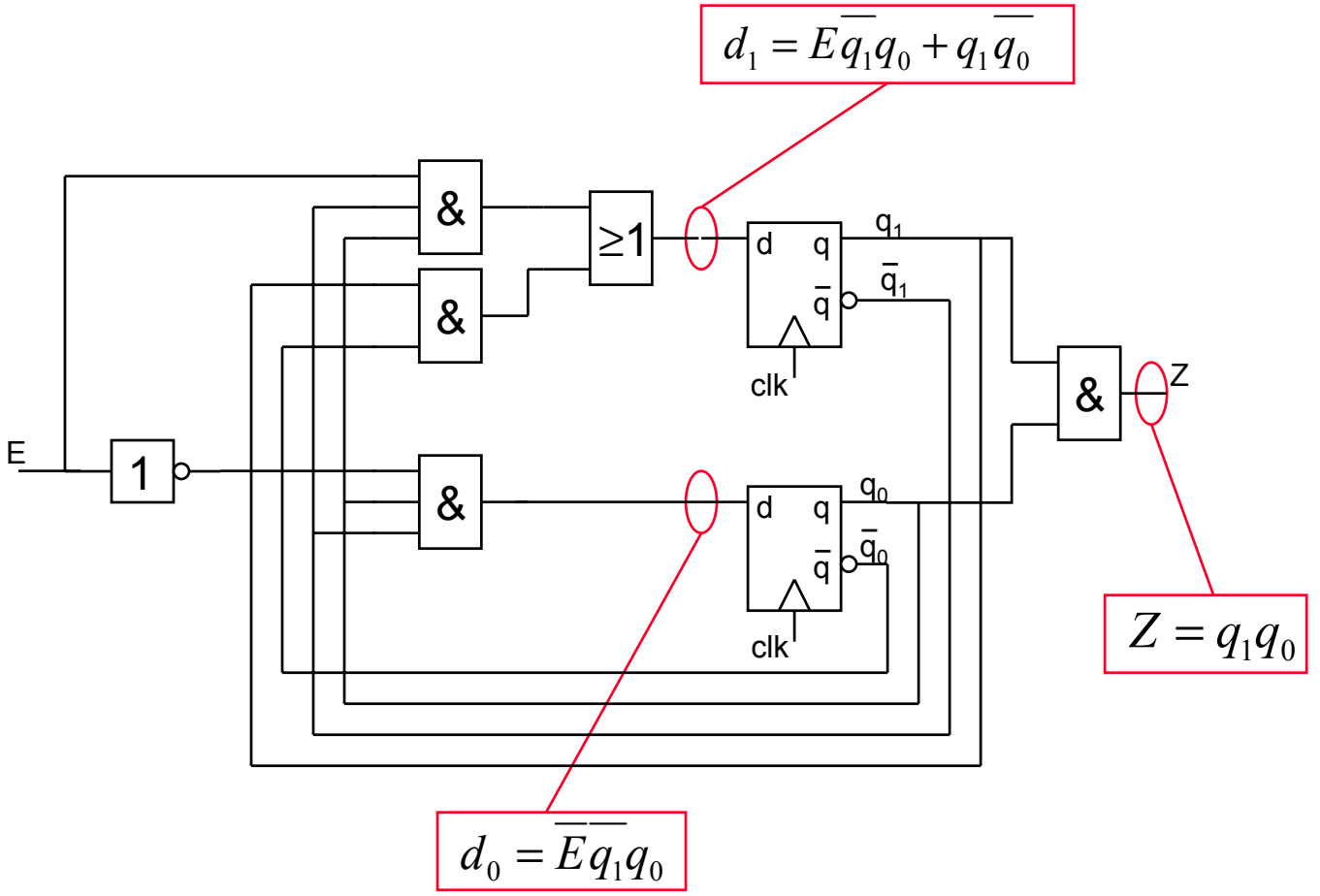
$Q=\{q_1,q_0\}$ $D=\{d_1,d_0\}$



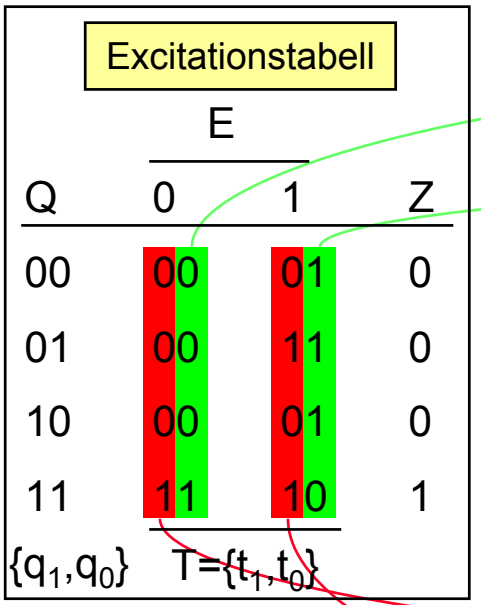
$Z=f(q_1,q_0)$

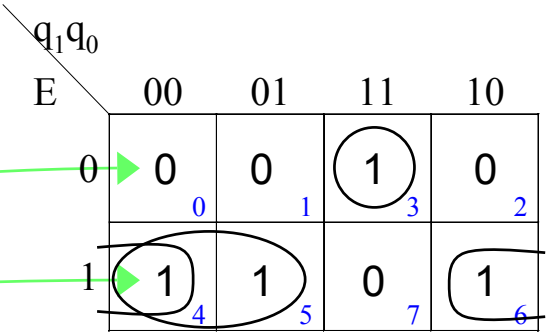
$$Z = q_1 q_0$$

Schema för tillståndsmaskin med D-vippor



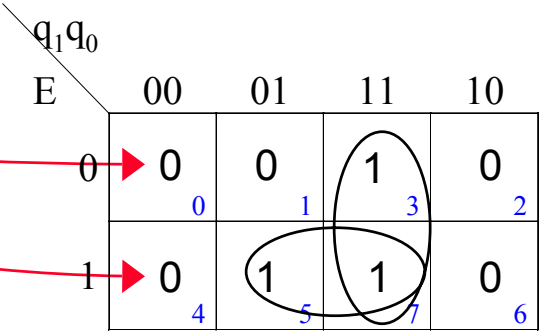
Logiskt uttryck för δ-funktionen då T-vippor används





$t_0=f(E,q_1,q_0)$

$$t_0 = \overline{E}q_1q_0 + E\overline{q_1} + E\overline{q_0}$$



$t_1=f(E,q_1,q_0)$

$$t_1 = Eq_0 + q_1q_0$$

Räknare

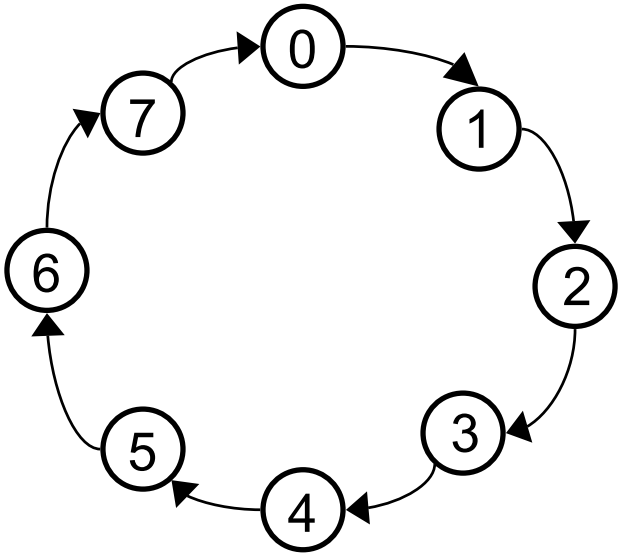
- Räknar antalet inkommande klockpulser
- De är sekvenskretsar
- Olika typer av räknare
  - Modulo-2^n räknare
  - Räknare med *enable*
  - Upp- och nedräknare

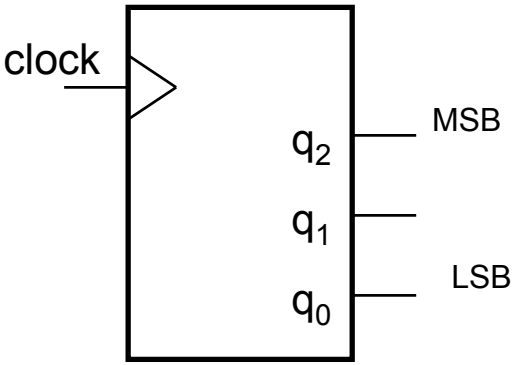
Modulo-2^n räknare

- Generellt
  - Räknar sekvensen …0, 1, … 2^n-1, 0, …

Max. värde för ett n-bitars tal

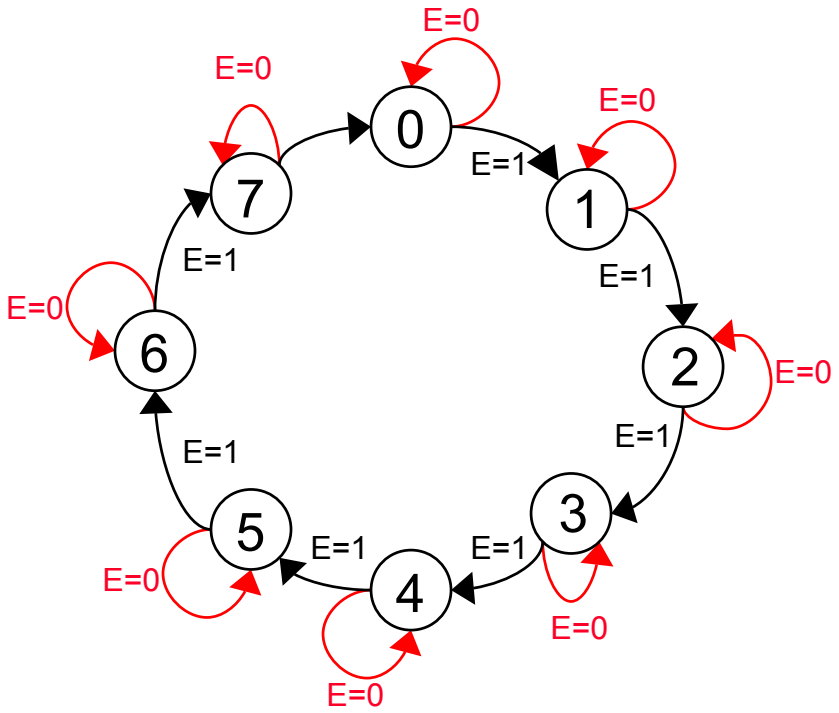
- Exempel: Modulo-8 (2^3) räknare
  - Räknar sekvensen …0, 1, … 7, 0, …

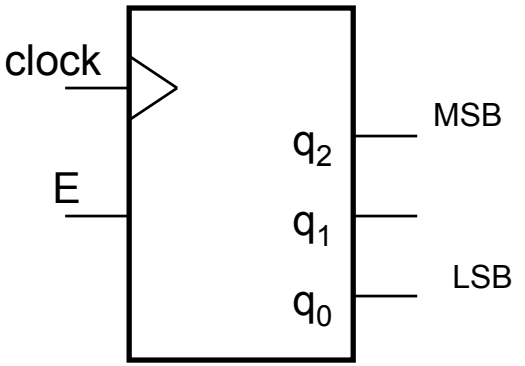




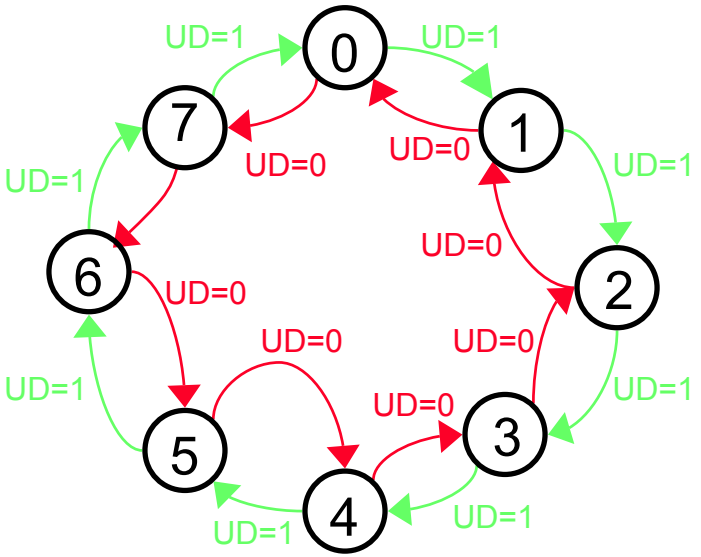
Räknare med *enable*

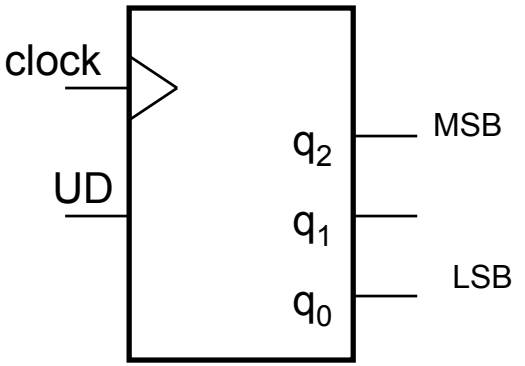
- Funktion
  - Med en *enable* signal kan man styra om räknaren ska räkna eller inte





- Funktion
  - Med en styrsignal UD (upp eller ned) kan man välja om räknaren ska räkna upp eller ned





SLUT på Föreläsning 4

- Innehåll
  - Minneselement
  - Tillståndsdiagram / tillståndstabell
  - Definition av Moore- och Mealy-maskiner
  - Tillståndskodning
  - Syntes av sekvenskretsar
  - Räknare

Digitala tal och Boolesk algebra

Innehåll

- Talsystem och koder
- Aritmetik för binära tal
- Grundläggande logiska operationer
- Logiska grindar
- Definitioner i Boolesk algebra
- Räknelagar

Binära talsystemet

- Binärt
  - Positionssystem
  - Två symboler används, B = { 0, 1 }
  - Binära tal gör det lätt att bygga elektronik baserade på elektroniska omkopplare
  - En algebra utvecklad av Boole gör det lätt att hantera logiska uttryck baserade på binära tal

Positionsbaserade talsystem

- Ett generellt positionsbaserat talsystem med basen b

Låt

$S = \{s_0, s_1, \ldots, s_{b-2}, s_{b-1}\}$ = mängd symboler

b = antal symboler $\equiv$ basen

N = positivt heltal, $N \geq 0$

q = antal positioner som krävs för att representera N i basen b

Representera positiva heltal

- För positiva heltal:

$$N = \sum_{i=0}^{q-1} s_i b^i$$

- *ex.,* Decimalt tal

$$(234)_{10} = s_2 b^2 + s_1 b^1 + s_0 b^0$$

låt $b = 10, \quad S = \{0,1,\ldots,8,9\}$

$$(234)_{10} = 2 \bullet 10^2 + 3 \bullet 10^1 + 4 \bullet 10^0$$

$$(234)_{10} = 2 \bullet 100 + 3 \bullet 10 + 4 \bullet 1$$

$$(234)_{10} = 200 + 30 + 4$$

Representera positiva heltal

- *exempel*, Binärt tal (basen 2)

$$(1011)_2 = s_3 b^3 + s_2 b^2 + s_1 b^1 + s_0 b^0$$

låt $b = 2$, $S = \{0,1\}$

$$(1011)_2 = 1 \bullet 2^3 + 0 \bullet 2^2 + 1 \bullet 2^1 + 1 \bullet 2^0$$

$$(1011)_2 = 1 \bullet (1000)_2 + 0 \bullet (100)_2 + 1 \bullet (10)_2 + 1 \bullet (1)_2$$

$$(1011)_2 = (1)_{10} \bullet (8)_{10} + (0)_{10} \bullet (4)_{10} + (1)_{10} \bullet (2)_{10} + (1)_{10} \bullet (1)_{10}$$

$$(1011)_2 = (8+2+1)_{10} = (11)_{10}$$

Representera delar av heltal (*eng. Fraction*)

Låt

$S = \{s_{-1}, \ldots, s_{r-2}, s_{r-1}\} =$ mängd symboler

$b =$ antal symboler i S $\equiv$ basen

$1 > F \geq 0$

$p =$ Antal positioner som krävs för att representera F i basen b, eller antal positioner som tillåts

då blir

$$F = \sum_{i=-p}^{-1} s_i b^i$$

Exempel på decimal tal

- Låt p = 3 (*tre positioner till höger om decimalpunkten*)

$$b = 10$$

$$F = s_{-1} b^{-1} + s_{-2} b^{-2} + s_{-3} b^{-3}$$

$$(0.625)_{10} = 6 \bullet 10^{-1} + 2 \bullet 10^{-2} + 5 \bullet 10^{-3}$$

$$(0.625)_{10} = 6 \bullet 0.1 + 2 \bullet 0.01 + 5 \bullet 0.001$$

$$(0.625)_{10} = 0.600 + 0.0200 + 0.005$$

Exempel på Binära ”decimaltal”?

- Låt p = 3

$$b = 2$$

$$(0.101)_2 = 1 \bullet 2^{-1} + 0 \bullet 2^{-2} + 1 \bullet 2^{-3}$$

$$(0.101)_2 = 1 \bullet \frac{1}{2} + 0 \bullet \frac{1}{4} + 1 \bullet \frac{1}{8}$$

$$(0.101)_2 = (0.100)_2 + (0.000)_2 + (0.001)_2$$

Bas-konvertering

- Ett tal med basen b skrivs om som:

$$N_0 = s_p \bullet b^p + s_{p-1} \bullet b^{p-1} + \ldots + s_1 \bullet b^1 + s_0 \bullet b^0$$
$$= (s_p \bullet b^{p-1} + s_{p-1} \bullet b^{p-2} + \ldots + s_1) \bullet b + s_0$$
$$= N_1 \bullet b + s_0$$

- En division av N_0 med b ger:

$$N_1 + \frac{s_0}{b}$$

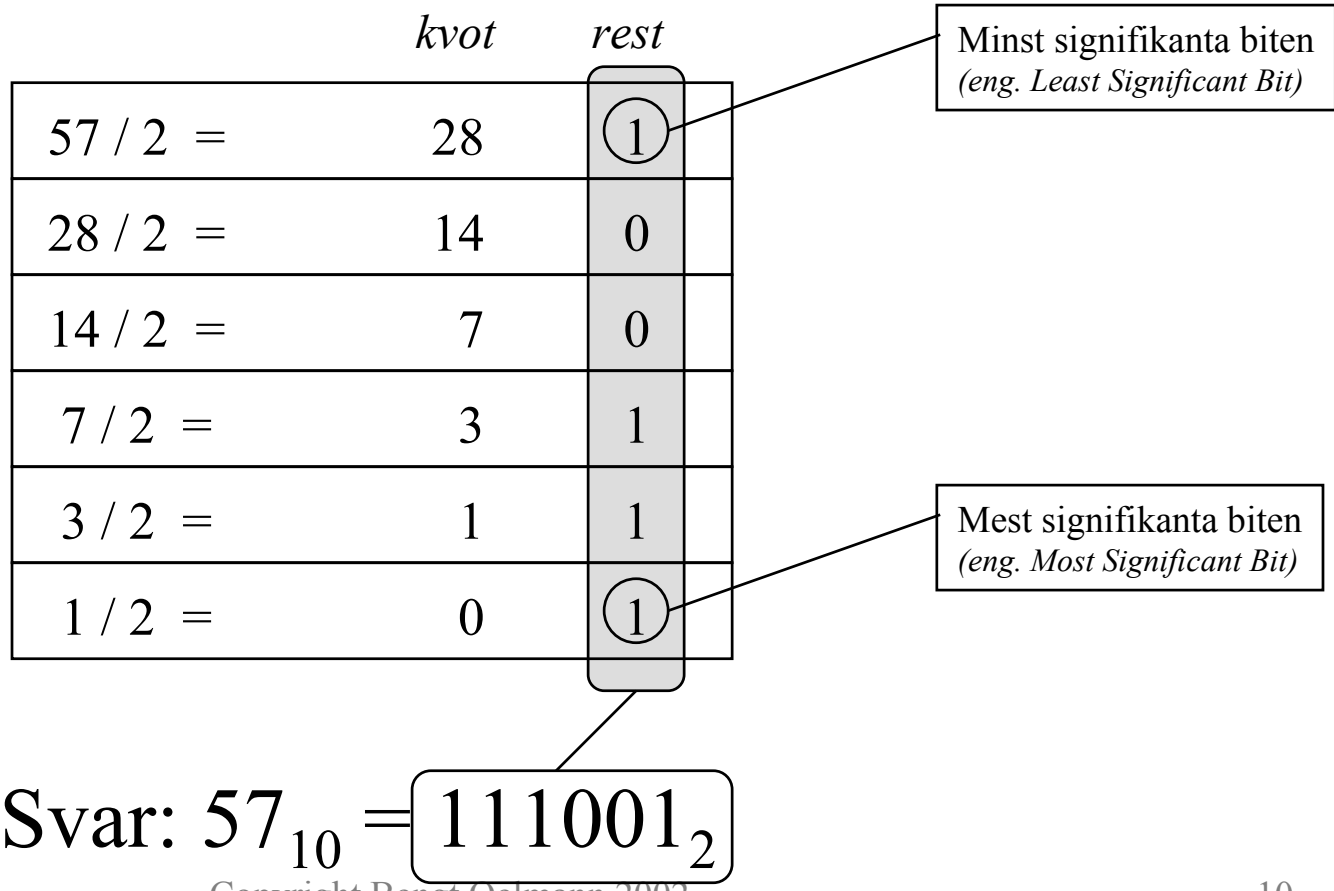
Resten = s_0 = minst signifikanta siffran

$$N_2 + \frac{s_1}{b}$$

Resten = s_1 = näst minst signifikanta siffran

Procedur för bas-konvertering

Exempel: Omvandla 57_{10} till binärt tal



Svar: $57_{10} = 111001_2$

Att tänka på vid omvandling

- Syftet med binär representation är att erhålla tal i ett format som passar digital logik
- Ju större noggrannhet ett binärt tal har desto fler bitar krävs → mer digitala kretsar
- Alla tal i en bas kan INTE representeras exakt i en annan bas (avrundningsfel)

Binära, Oktala och Hexadecimala tal

- Det är lätt att konvertera binära tal till andra, mer lättarbetade format genom att gruppera bitar tillsammans och sedan konvertera till lämplig bas
  - Oktala tal S={ 0, 1, ... , 7 }, basen = 8
    - 3-bits grupper
  - Hexadecimal S={ 0, ... , 9, A, B, C, D, E, F }, basen = 16
    - 4-bits grupper

Exempel: Binär till Oktal omvandling

$N_2 = 10101101 00.1$

gruppera i 3 - bitars grupper från "decimal"-punkten

$N_2 = 1\underline{010}\,\underline{110}\,\underline{100}.1$

fyll ut med två nollor för att få ett oktalt tal (en full grupp)

$N_2 = \underline{001}\,010\,110\,100.\underline{100}$

Konvertera varje 3 - bitars grupp

$N_8 = 1 \quad 2 \quad 6 \quad 4 \;.\; 4$

$N_8 = (1264.4)_8$

Exempel: Binär till hexadecimal

N_2=1010110100.1

gruppera i 4 - bitars grupper från "decimal"-punkten

N_2=10 1011 0100 . 1

fyll ut med nollor för att få en komplett grupp

N_2=0010 1011 0100 . 1000

Konvertera varje 4 - bitars grupp

N_{16}= 2 B 4 . 8

N_{16}= $(2B4.8)_{16}$= $(2B4.8)_H$

Viktade koder

- Godtycklig vikt kan tilldelas varje position
  - Binary Coded Decimal (BCD)
    - 8, 4, 2, 1
    - exempel, 1001 = 8 + _ + _ + 1 = 9
    - Alla kodord används inte
      - Enbart 0_{10}-9_{10} används
      - Ej 10_{10}-15_{10}

Icke-viktade koder

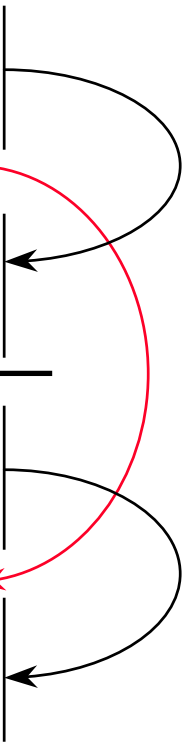
- Cykliska
  - På varandra följande kodord skiljer sig åt med endast en bit och det gäller också då de ”slår runt”
  - Gray Code är den vanligaste

Gray kod

- Fördelar
  - Enkelt att konstruera för vilket antal bitar som helst
  - Cyklisk
  - Unik

3-bitars Gray kod

Ordning	*Kodord*
0	0 0 0
1	0 0 1
2	0 1 1
3	0 1 0
4	1 1 0
5	1 1 1
6	1 0 1
7	1 0 0



Alfanumeriska koder

- Alfanumeriska koder representerar både
  - Decimala siffersymboler
    - 0 - 9
  - Alfabetets tecken
    - A – Z, a – z
  - Övriga skrivbara tecken
    - T.ex: %, &, ?, *, @
  - Styrsymboler
    - Blanksteg, ny rad, etc.
  - Standardkoder: **ASCII**, EBCDIC, etc.

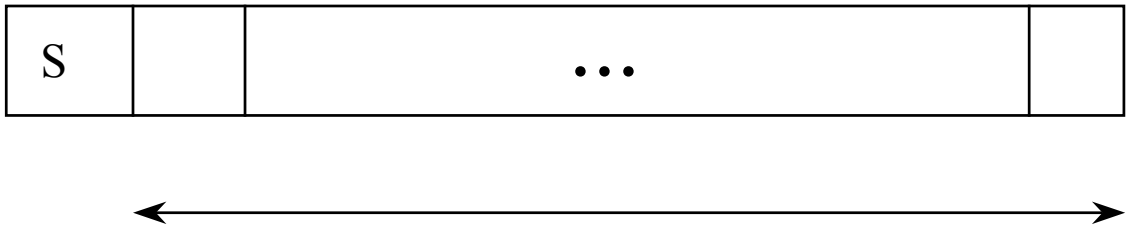
ASCII

- ASCII kodning
  - *American Standard Code for Information Interchange*
  - ASCII-tabell
    - Ger hexadecimal kod
    - Rad: minst signifikanta positionen
    - Kolumn: mest signifikanta positionen
    - Exempel: ASCII('C') = 43_{16}

	000	001	002	003	004	005	006	007
0	NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	` 0060	p 0070
1	STX 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
2	SOT 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
3	ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
4	EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
5	ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
6	ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
7	BEL 0007	ETB 0017	' 0027	7 0037	G 0047	W 0057	g 0067	w 0077
8	BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
9	HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
A	LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
B	VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
C	FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
D	CR 000D	GS 001D	- 002D	= 003D	M 004D	] 005D	m 006D	} 007D
E	SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
F	SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

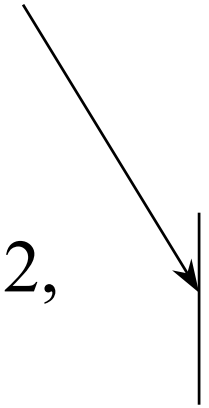
Negativa tal

- teckenbit
  - Biten längst till vänster representerar talets tecken
    - 1 negativt
    - 0 positivt
  - Bitarna till höger om teckenbiten är storleken på talet



Exempel: tal med teckenbit

- 2 koder för noll
- Skiftning
  - Ger ej mult/div med 2,
  - Tar ej hänsyn till tecknet



Decimal	s	$(\text{storlek})_2$
+3	0	11
+2	0	10
+1	0	01
+0	0	00
-0	1	00
-1	1	01
-2	1	10
-3	1	11

Två-komplement

- Två-komplement representation
  - För ett n-bitars tal
    - är värdet för MSB -2^{p-1} (istället för $+2^{p-1}$)
    - övriga bitars värde är samma som för positiva tal
  - Procedur för att utföra två-komplement
    - invertera samtliga bitar i talet
    - addera 1 till talet
  - Exempel:

Bilda två-komplement till 8-bitars talet 00010001_2 ($=17_{10}$)

00010001

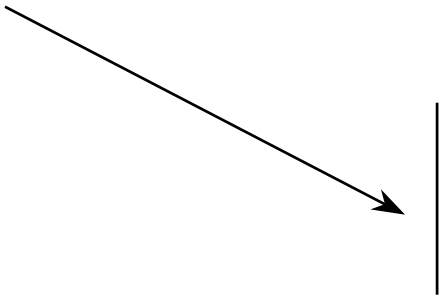
11101110

$+ \quad 1$

11101111 = -17_{10}

Två-komplement – 3-bitars tal

- En kod för noll



Decimal	2-komp
+3	011
+2	010
+1	001
+0	000
-0	000
-1	111
-2	110
-3	101
-4	100

Addition och subtraktion

- Exempel:

$3 + 2 = ?$

$$\begin{array}{r} 0011 \\ \underline{0010} \\ 0101 \end{array} = 5_{10}$$

$7 - 1 = ?$

$$\begin{array}{r} 0111 \\ \underline{1111} \\ 1\ 0110 \end{array} = 6_{10}$$

$-1 - 3 = ?$

$$\begin{array}{r} 1111 \\ \underline{1101} \\ 1\ 1100 \end{array} = -4_{10}$$

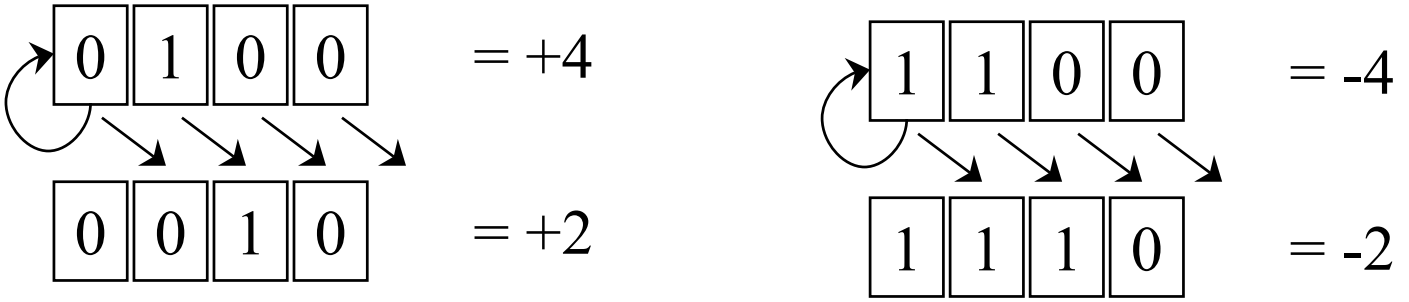
$2 - 3 = ?$

$$\begin{array}{r} 0010 \\ \underline{1101} \\ 1111 \end{array} = -1_{10}$$

Multiplikation/Division med 2

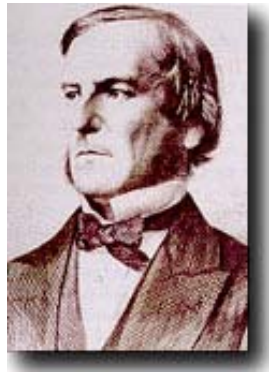
- Skifta det binära talet ett steg vänster

- Skifta det binära talet ett steg höger



Boolesk Algebra

- Historik



George Boole (1815-1864), en engelsk matematiker visade att logik kan uttryckas som algebraiska ekvationer. Han gav upphov till vad vi kallar *Boolesk algebra.* (*1854: An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities*)

Används idag inom matematik, informationsteori, switching algebra, grafteori, datorvetenskap och artificiell intelligens.

Edward Huntington: (1874-1952), en amerikansk matematiker som gav Boolesk algebra sina axiom.

Claude Shannon (1916-2001), en amerikansk matematiker som beskrev informationens minsta beståndsdel som 0 eller 1. Han myntade begreppet ’bit’. Han lade grunden till informationsteorin som har haft avgörande betydelse för utvecklingen av kommunikationssystem.

Boolesk Algebra – Definitioner

Konstanter

0	(Falsk)
1	(Sann)

Operationer

+	(ELLER)
·	(OCH)
’	(ICKE)

Axiom

$0 + 0 = 0$
$1 \cdot 1 = 1$

$1 + 1 = 1$
$0 \cdot 0 = 0$

$0 + 1 = 1 + 0 = 1$
$1 \cdot 0 = 0 \cdot 1 = 0$

$0' = 1$
$1' = 0$

Räknelagar för en variabel

$x + x = x$
$x \cdot x = x$

$x + x' = 1$
$x \cdot x' = 0$

$x + 1 = 1$
$x \cdot 0 = 0$

$x + 0 = x$
$x \cdot 1 = x$

$(x')' = x$

Dessa räknelagar kan enkelt visas utifrån axiomen.

Visa att $x + x = x$

Låt $x = 1$: $1 + 1 = 1$

Låt $x = 0$: $0 + 0 = 0$

Räknelagar för flera variabler

- Associativa lagar

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

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

- Kommutativa lagar

  $x + y = y + x$

  $x \cdot y = y \cdot x$

- Distributiva lagar

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

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

Räknelagar för flera variabler

- absorptionslagar

  $x + x \cdot y = x$

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

$x + x \cdot y = x \cdot (1 + y) = x \cdot 1 = x$

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

$= x \cdot (1 + y) = x \cdot 1 = x$

- Concensuslagen

  $x \cdot y + x' \cdot z = x \cdot y + x' \cdot z + y \cdot z$

Augustus de Morgan

- De Morgans lag

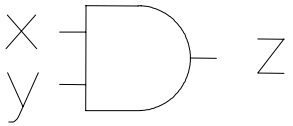
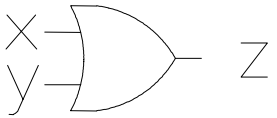
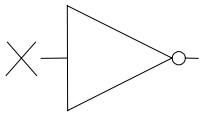
  $(x + y)' = x' \cdot y'$

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

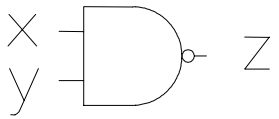
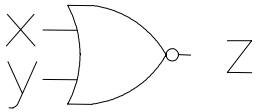
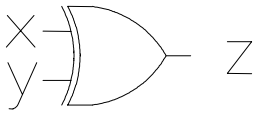
$\overline{x + y} = \overline{x} \cdot \overline{y}$

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

Grundläggande logiska grindar

Namn/operator	Symbol	Funktion	Logisk operation
OCH, *eng. AND* •		X Y Z 0 0 0 0 1 0 1 0 0 1 1 1	$Z = X \bullet Y$
ELLER, *eng. OR* +		X Y Z 0 0 0 0 1 1 1 0 1 1 1 1	$Z = X + Y$
ICKE, *eng. NOT* ’		X Z 0 1 1 0	$Z = X'$

Grundläggande logiska grindar

Namn/operator	Symbol	Funktion	Logisk operation
NAND		X Y Z 0 0 1 0 1 1 1 0 1 1 1 0	$Z = \overline{X \cdot Y}$ $Z = (X \cdot Y)'$
NOR		X Y Z 0 0 1 0 1 0 1 0 0 1 1 0	$Z = \overline{X + Y}$ $Z = (X + Y)'$
XOR $\oplus$		X Y Z 0 0 0 0 1 1 1 0 1 1 1 0	$Z = X \oplus Y$

SLUT på Föreläsning 1

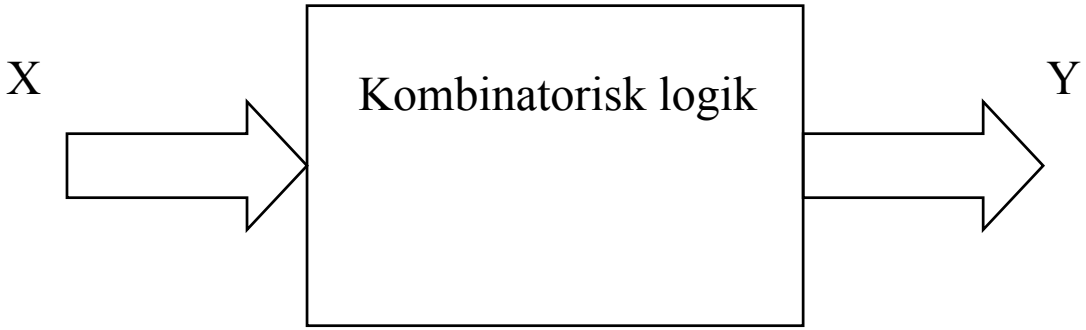
- Innehåll
  - Talsystem och koder
  - Aritmetik för binära tal
  - Grundläggande logiska operationer
  - Logiska grindar
  - Definitioner i Boolesk algebra
  - Räknelagar

Kombinatorisk logik

Innehåll

- Definition av kombinatorisk logik
- Olika sätt att representera kombinatorisk logik
- Minimering av logiska uttryck
  - Boolesk algebra
  - Karnaugh-diagram
- Realisering i av logiska funktioner i grindnät
- Ofullständigt specificerade funktioner

Definition av kombinatorisk logik



$$X = (x_{n-1}, \ldots, x_0) \qquad Y = f(X) \qquad Y = (y_{m-1}, \ldots, y_0)$$

$$x_i \in \{0,1\}, \forall i \in \{0, n-1\} \qquad y_j \in \{0,1\}, \forall j \in \{0, m-1\}$$

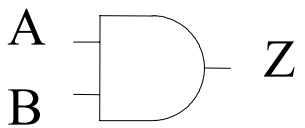
Utgångarnas värde, för en given tidpunkt, beror endast på värdet på ingångarna vid samma tidpunkt.

Kombinatorisk logik har inget minne.

Olika sätt att representera logiska funktioner

- Sanningstabell
- Grindnät
- Boolesk algebra
- Normalform

Sanningstabell



Logisk grind

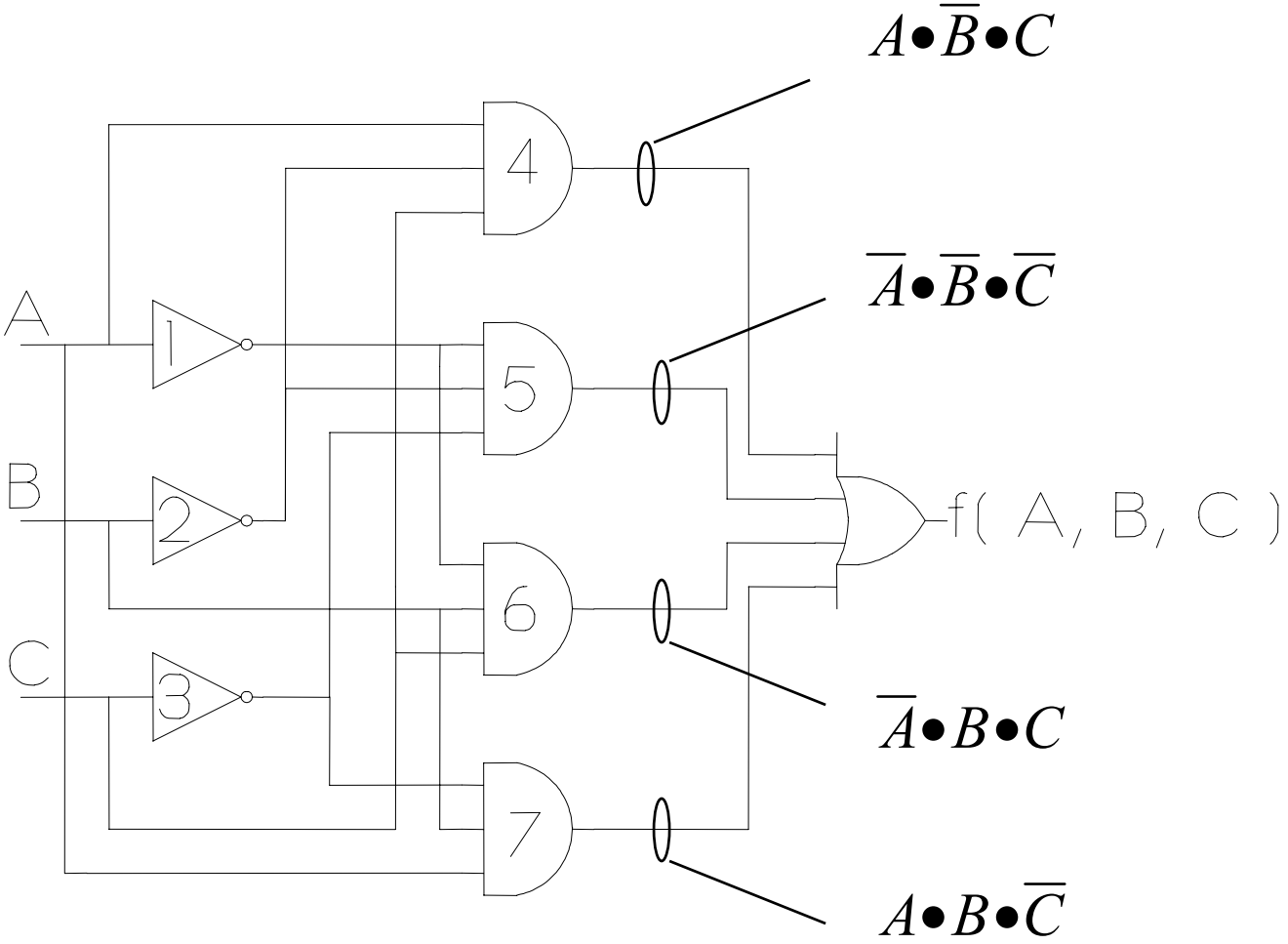
$Z = A \bullet B$

Logisk funktion

Ingångar		Utgång
A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

Sanningstabell

Grindnät



$$f(A,B,C) = A\overline{B}C + \overline{A}\overline{B}\overline{C} + \overline{A}BC + AB\overline{C}$$

Boolesk algebra

- Algebraisk manipulering
  - Visa att: $x + yz = (x + y)(x + z)$

$$
\begin{aligned}
(x + y)(x + z) &= xx + yx + xz + yz \\
&= x + yx + xz + yz \\
&= x + xy + xz + yz \\
&= x + x(y + z) + yz \\
&= x(1 + y + z) + yz \\
&= x + yz
\end{aligned}
$$

Normalformer

- Icke-minimalt standardsätt att skriva algebraiska uttryck
- En boolesk funktion kan skrivas på två normalformer
  - Summa av produkter, SP-normalform
    - Mintermer
  - Produkt av summa, PS-normalform
    - Maxtermer

Minterm

- Definitioner
  - Produktterm
    - Är en variabel eller en logisk produkt av två eller flera variabler
    - Exempel: A, A’, AC, ABD
  - Minterm
    - En n-variabel minterm är en produktterm med n variabler. För en funktion av n variabler så är dess mintermer av n variabler.
    - Exempel: A’BCD, A’B’C’D’, ABCD för f(A,B,C,D)

Minterm

- Samband mellan mintermer och sanningstabell
  - En minterm är en produktterm som är 1 i exakt en rad i sanningstabellen

rad	Ingångar		f(A,B)	minterm
	A	B	Z	
0	0	0	0	A'B'
1	0	1	1	A'B
2	1	0	0	AB'
3	1	1	1	AB

Mintermerna 1 och 3 leder till att funktionen f(A,B) blir sann

$$f(A,B) = \overline{A}B + AB$$

1 3

SP-normalform

- Summa av produkt
  - engelska: SOP (Sum-of-products)
  - Ett algebraiskt uttryck som är en logisk summa (ELLER) av logiska produkter (produkttermer)
  - Exempel: AB + AC, A + ABC
- SP-normalform
  - Summan av mintermerna som motsvaras av sanningstabellens rader där utgången är 1
    Notation: $f(A,B,C) = \sum(1,3)$

Exempel: SP-normalform

rad	Ingångar			f(A,B,C)	minterm
	A	B	C	Z	
0	0	0	0	1	A’B’C’
1	0	0	1	0	A’B’C
2	0	1	0	0	A’BC’
3	0	1	1	1	A’BC
4	1	0	0	1	AB’C’
5	1	0	1	0	AB’C
6	1	1	0	1	ABC’
7	1	1	1	1	ABC

$$f(A,B,C) = \sum(0,3,4,6,7)$$

$$f(A,B,C) = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}BC + A\overline{B}\,\overline{C} + AB\overline{C} + ABC$$

Maxterm

- Definitioner
  - Summaterm
    - Är en variabel eller en logisk summa av två eller flera variabler
    - Exempel: A, A’, A+C, A+B+D
  - Maxterm
    - En n-variabel maxterm är en summaterm med n variabler. För en funktion av n variabler så är dess maxtermer av n variabler.
    - Exempel: A’+B+C+D, A’+B’+C’+D’, A+B+C+D för f(A,B,C,D)

Maxterm

- Samband mellan maxtermer och sanningstabell
  - En maxterm är en summaterm som är 0 i exakt en rad i sanningstabellen

rad	Ingångar		f(A,B)	minterm
	A	B	Z	
0	0	0	0	A+B
1	0	1	1	A+B’
2	1	0	0	A’+B
3	1	1	1	A’+B’

Maxtermerna 0 och 2 leder till att funktionen f(A,B) blir falsk

$$f(A,B) = (A+B)(\overline{A}+B)$$

PS-normalform

- Produkt av summa
  - engelska: POS (Product-of-sums)
  - Ett algebraiskt uttryck som är en logisk produkt (OCH) av logiska summor (summatermer)
  - Exempel: (A+B)(A+C), A(B+C)
- PS-normalform
  - Produkten av maxtermerna som motsvaras av sanningstabellens rader där utgången är 0
    Notation: $f(A,B) = \prod(0,2)$

Exempel: PS-normalform

rad	Ingångar			f(A,B,C)	maxterm
	A	B	C	Z	
0	0	0	0	1	A+B+C
1	0	0	1	0	A+B+C’
2	0	1	0	0	A+B’+C
3	0	1	1	1	A+B’+C’
4	1	0	0	1	A’+B+C
5	1	0	1	0	A’+B+C’
6	1	1	0	1	A’+B’+C
7	1	1	1	1	**A’+B’+C’**

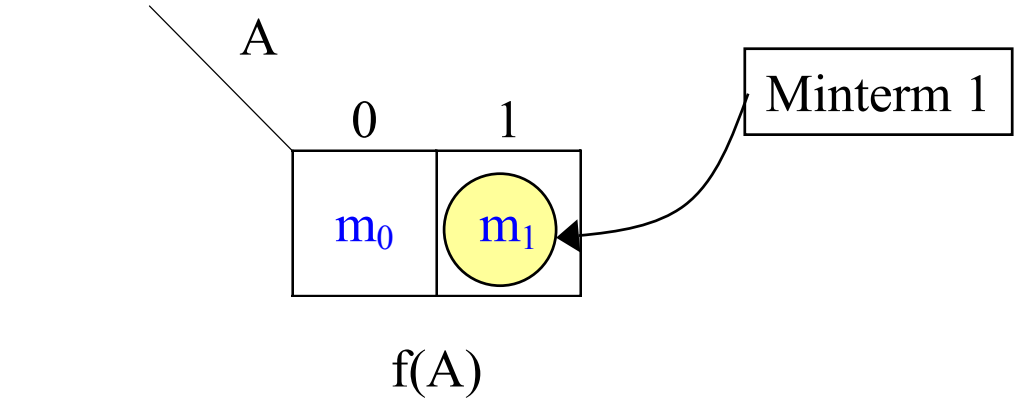
$$f(A,B,C) = \prod(1,2,5)$$

$$f(A,B,C) = (A + B + \overline{C}) \bullet (A + \overline{B} + C) \bullet (\overline{A} + B + \overline{C})$$

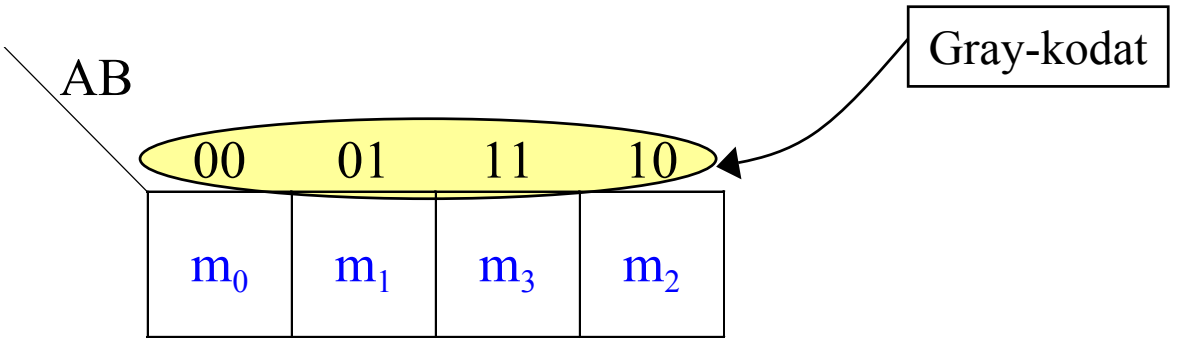
Karnaugh diagram

- Representation av en funktion i en två-dimensionell sanningstabell (matris)
- Horisontella/Vertikala celler i matrisen skiljer sig bara i en variabel
  - Hamming avstånd = 1
- Om närliggande celler (mintermer) är 1, så täcks dem av en enda term
- Algebraisk princip för minimering i K-diagram $x + \bar{x} = 1$

1 & 2 Variabel K-diagram

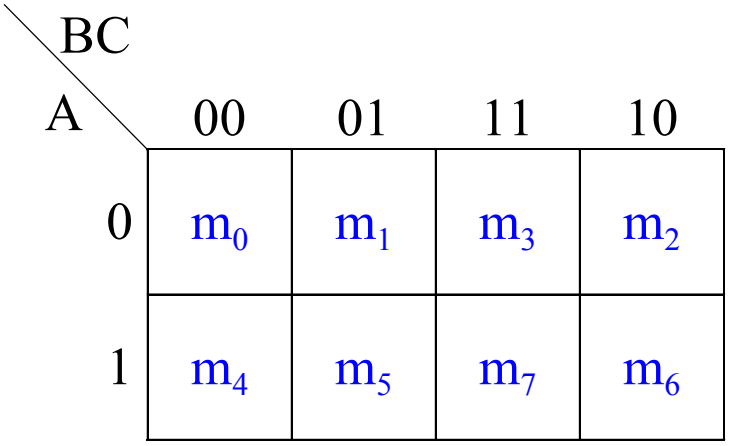


f(A)

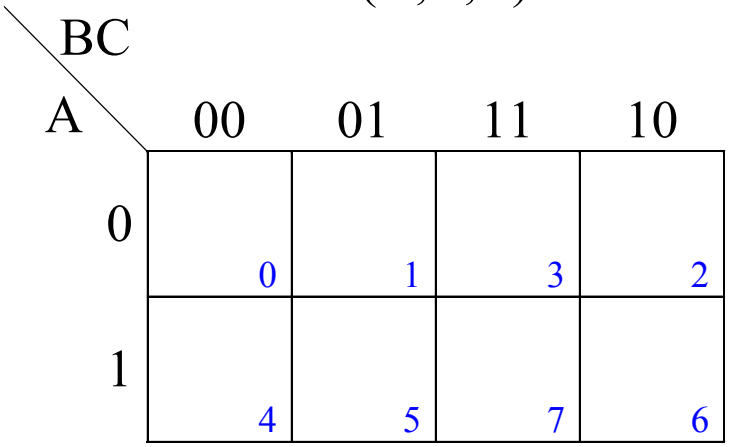


f(A,B)

3 Variabel K-diagram

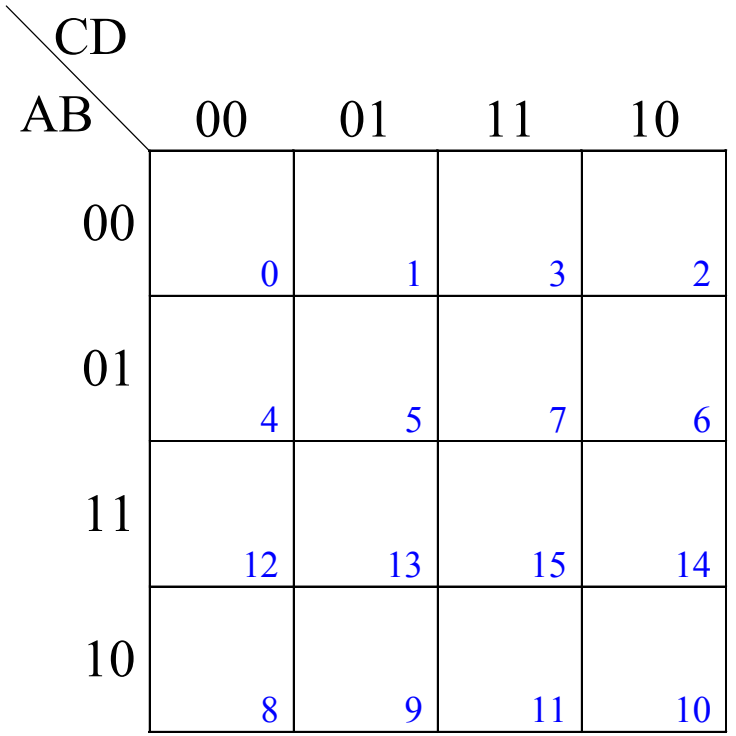


f(A,B,C)



f(A,B,C)

4 Variabel K-diagram

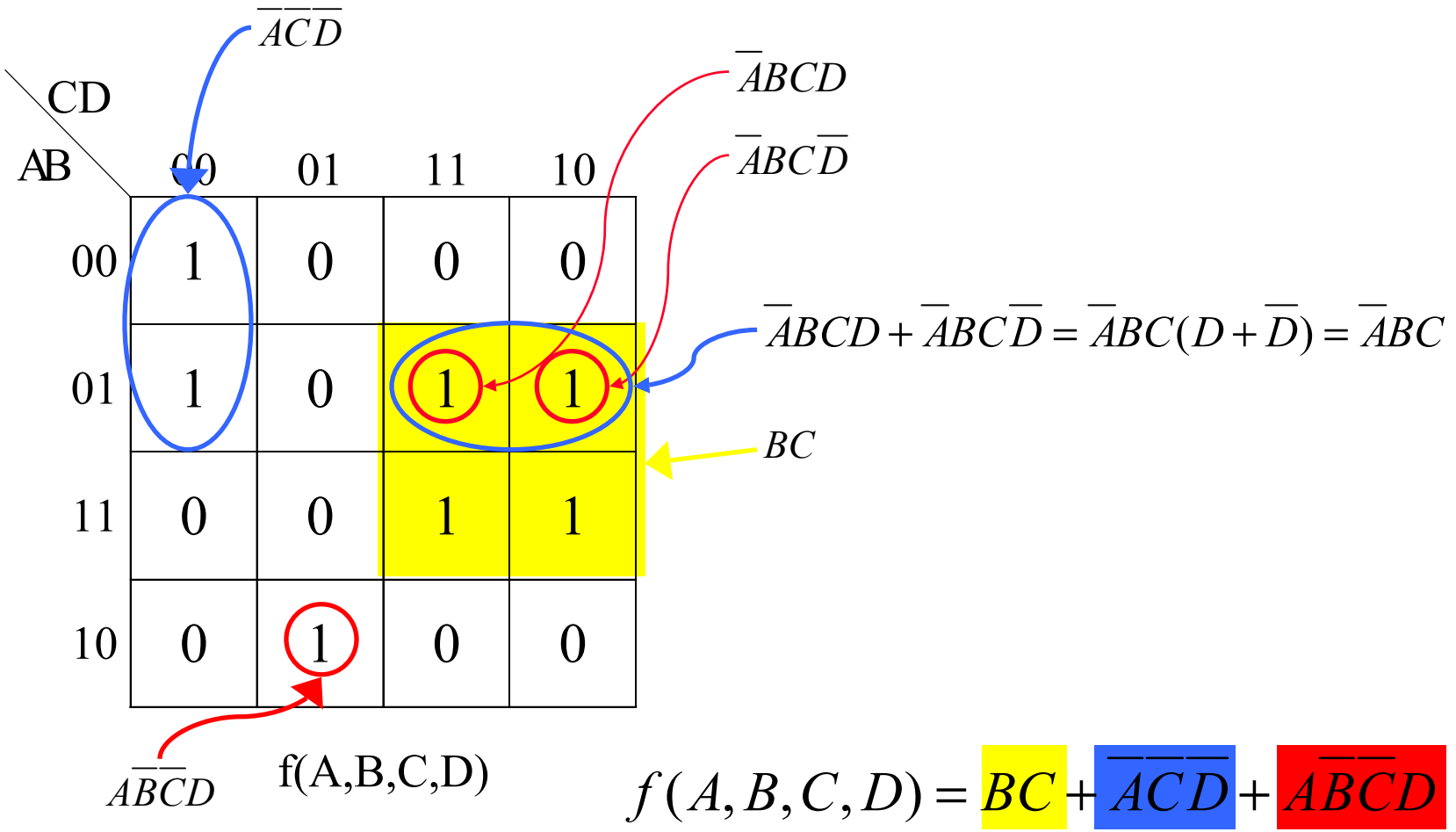


f(A,B,C,D)

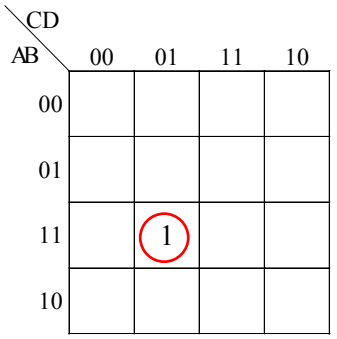
Användning av K-diagram

- Grafisk metod för minimering av uttryck
- Inringningar av mintermer för *f*
  - Ger uttryck på summa-av-produktform
- Maxtermer för *f*
  - Ger uttryck på produkt-av-summaform

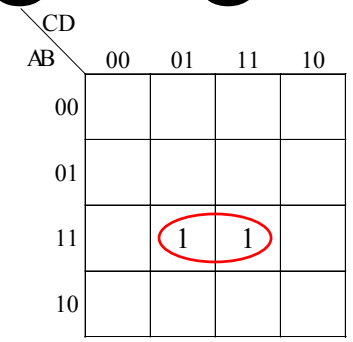
Inringningar i K-diagram



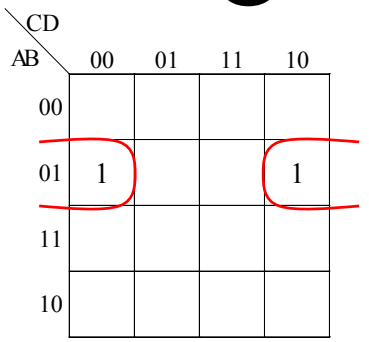
Inringningar i K-diagram



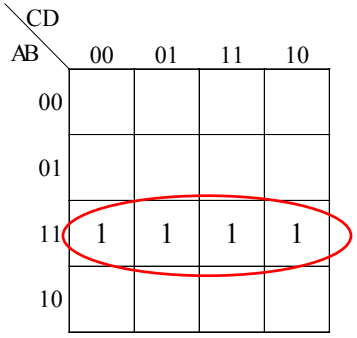
ABC'D



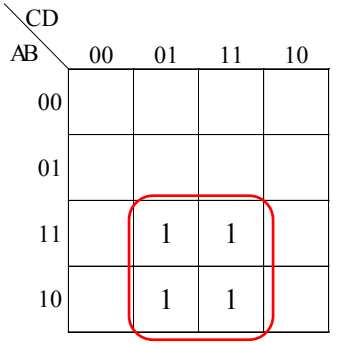
ABD



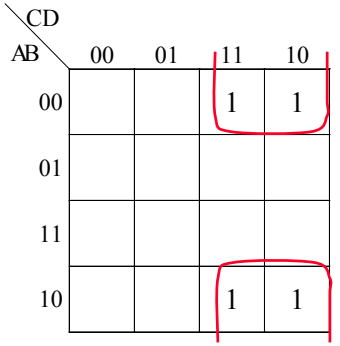
A'BD'



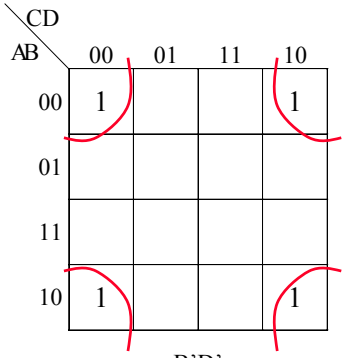
AB



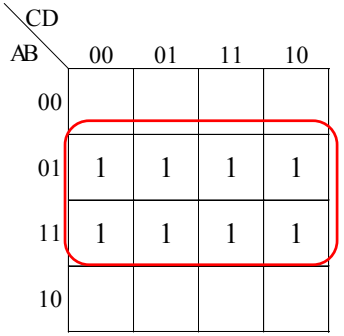
AD



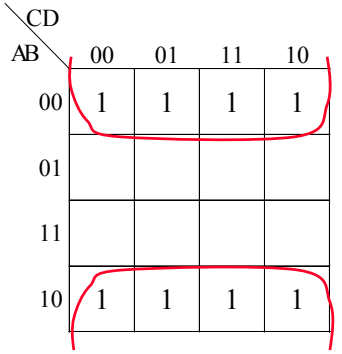
B'C



B'D'



B



B'

Ofullständigt specificerade funktioner

- Vissa ingångskombinationer förekommer aldrig
  - Vid minimering kan utgångsvärdet för dessa väljas fritt mellan 0 eller 1
  - Indikeras som ”don’t care” (d) eller –
  - Exempel: kombinationerna 00, 01 förekommer aldrig

rad	Ingångar		f(A,B)
	A	B	Z
0	0	0	-
1	0	1	-
2	1	0	0
3	1	1	1

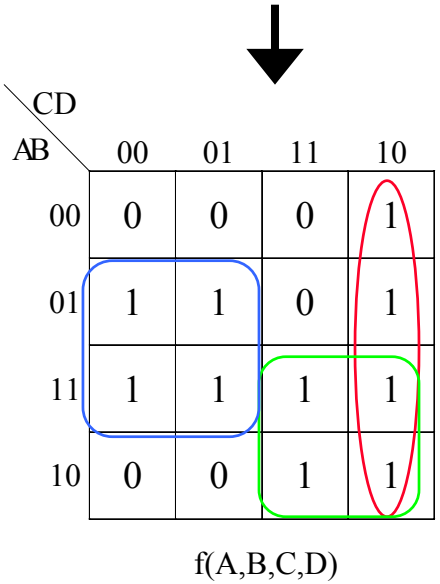
$$f(A,B) = \sum(3) + d(0,1)$$

Vid inringning i K-diagram
kan – väljas som 0 eller 1
så att största inringningarna erhålls

Realisering i grindnät

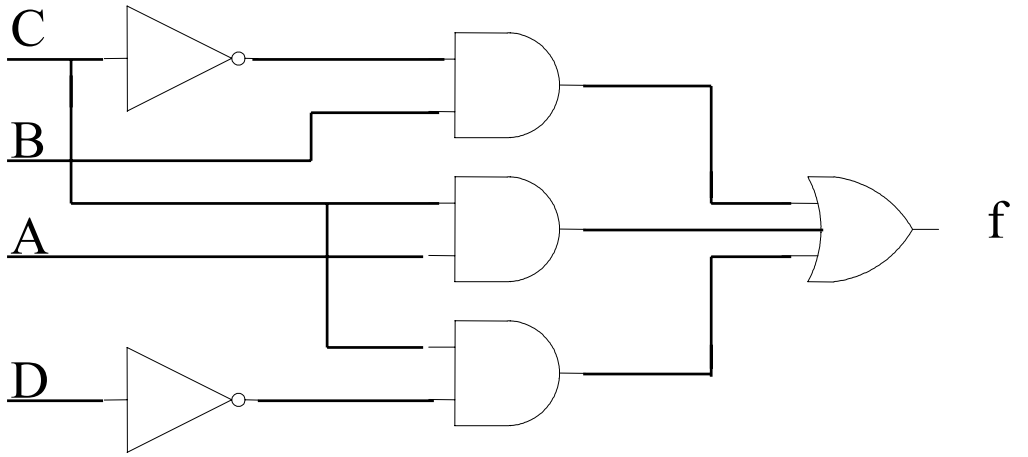
- Ta fram grindnät för funktionen:

$$f(A,B,C,D) = \sum(2,4,5,6,10,11,12,13,14,15)$$



f(A,B,C,D)

$$f(A,B,C,D) = B\overline{C} + AC + C\overline{D}$$



SLUT på Föreläsning 2

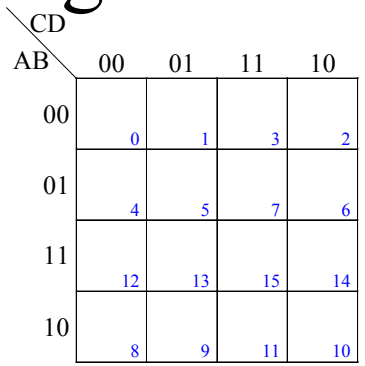
- Innehåll
  - Definition av kombinatorisk logik
  - Olika sätt att representera kombinatorisk logik
  - Minimering av logiska uttryck
    - Boolesk algebra
    - Karnaugh-diagram
  - Realisering i av logiska funktioner i grindnät
  - Ofullständigt specificerade funktioner

Minimeringsmetoder

- Innehåll
  - Karnaugh-diagram för 5 och 6 variabler
  - Quine-McCluskey metoden

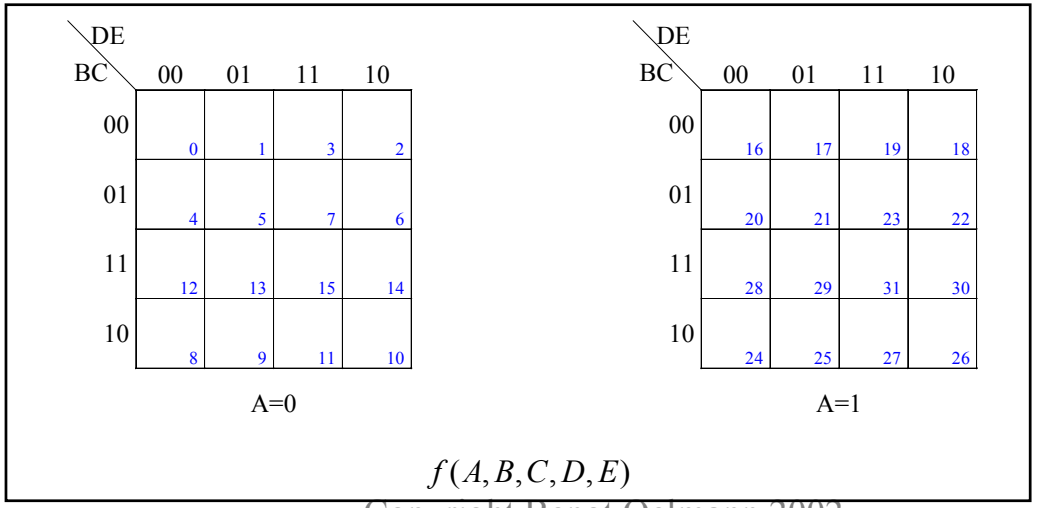
K-diagram för 5 variabler

- K-diagram för funktioner av 4 variabler



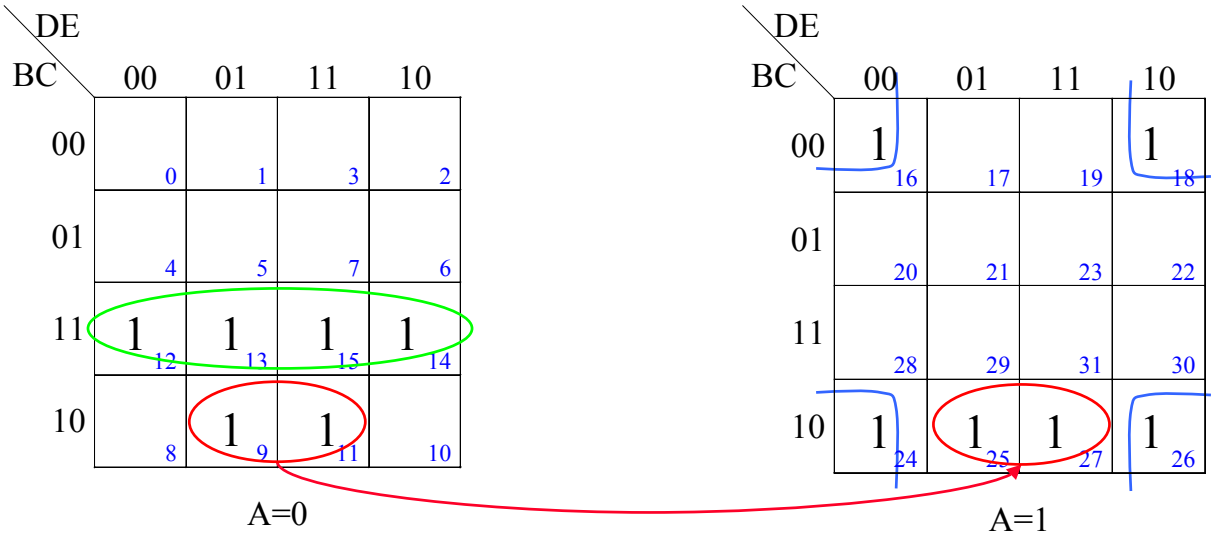
f(A,B,C,D)

- K-diagram för funktioner av 5 variabler



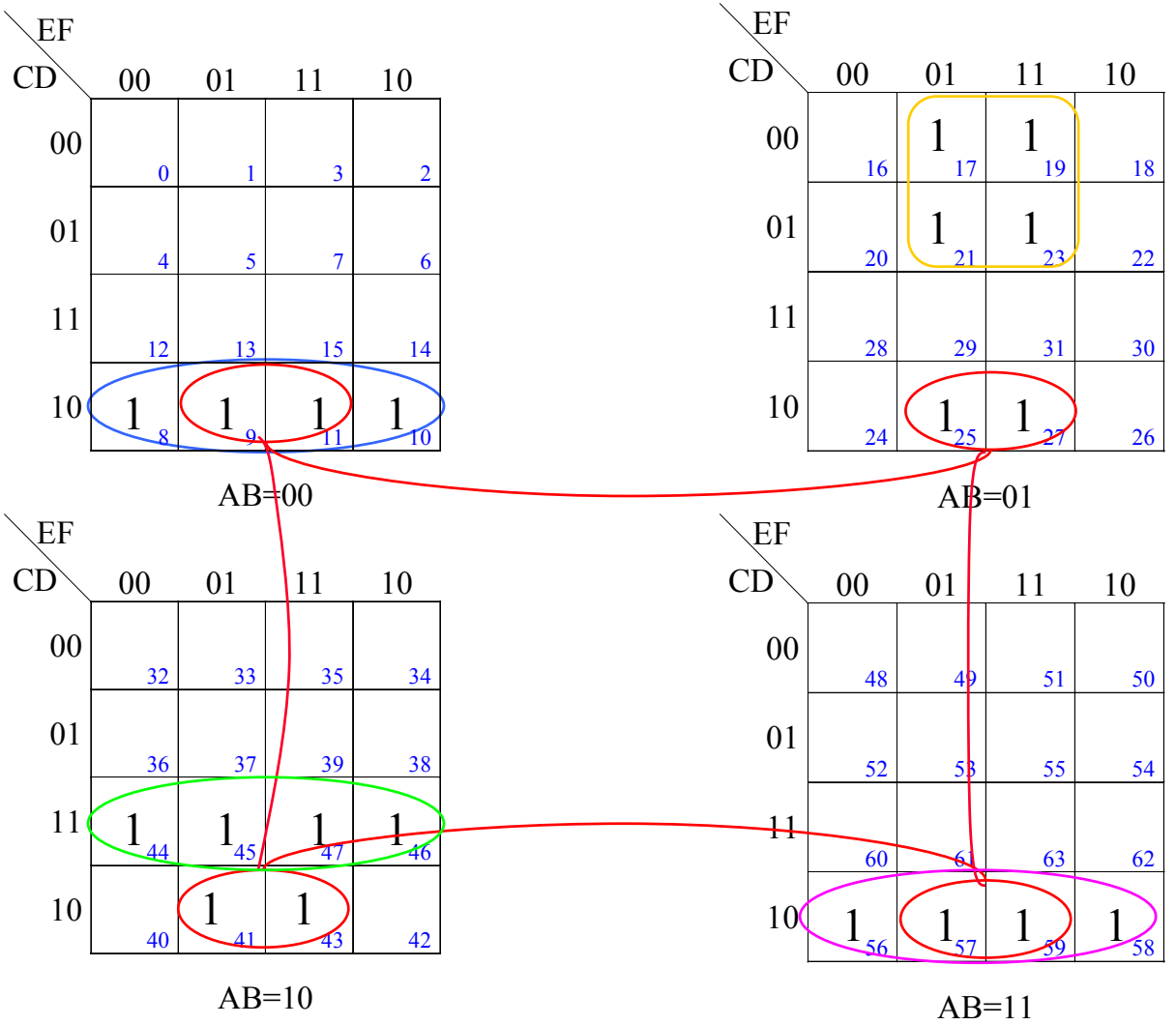
Exempel: Använd K-diagram för 5 variabler

$$f(A,B,C,D,E) = \sum(9,11,12,13,14,15,16,18,24,25,26,27)$$



$$f(A,B,C,D,E) = \overline{A}BC + B\overline{C}E + A\overline{C}\,\overline{E}$$

K-diagram för 6 variabler



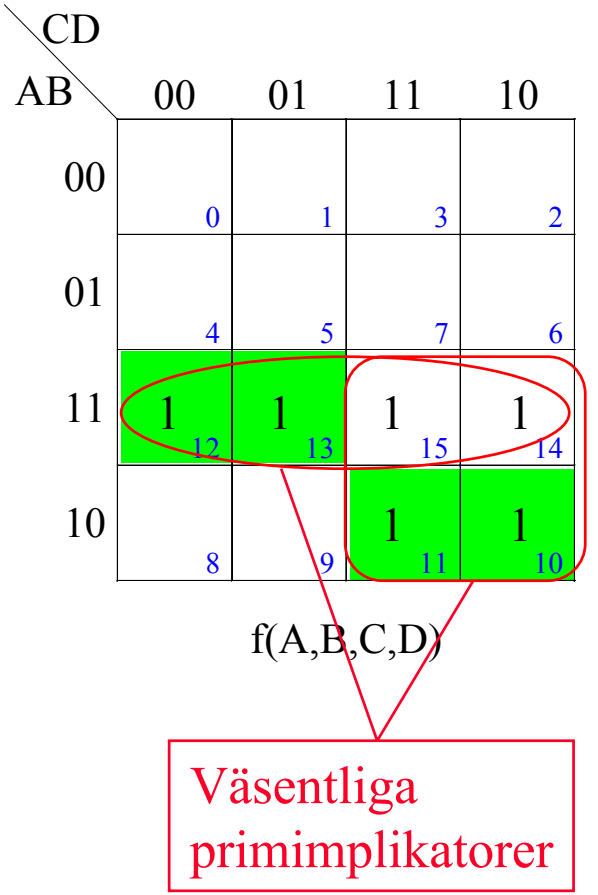
$$f(A,B,C,D,E,F) = C\bar{D}F + A\bar{B}CD + \bar{A}B\bar{C}F + \bar{A}\bar{B}C\bar{D} + AB\bar{C}\bar{D}$$

Quine-McCluskey minimering

- K-diagram är bra på
  - Minimering av små funktioner (< 5variabler)
  - Funktioner med endast en utgång
- Kan inte implementeras i datorprogram
- Subjektiv tolkning kan ge upphov till olika inringningar
- Quine-McCluskey löser dessa problem
  - Tabell-baserad minimeringsmetod

Grundläggande definitioner

- Primimplikator
  - Produktterm som hör till en maximal inringning
- Väsentlig primimplikator
  - Primimplikator som täcker en minterm som inte täcks av annan primimplikator
- Hammingvikt
  - Antal ettor $hv(B) = \sum_{i=0}^{N-1} b_i$
  - Exempel: $hv(10011) = 3$



Procedur för Quine-McCluskey

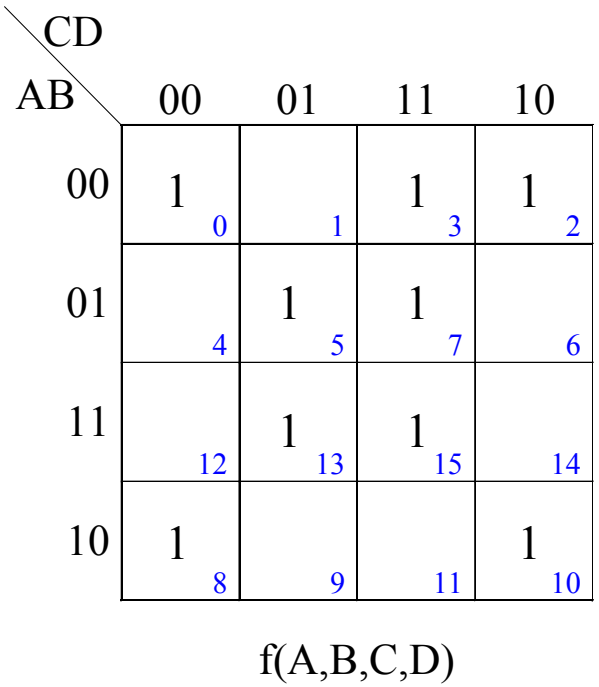
- Generering av samtliga primimplikatorer
- Bestämning av minsta antalet primimplikatorer för funktionen

Q-M Exempel

$$f(A,B,C,D) = \sum(0,2,3,5,7,8,10,13,15)$$

- Skapa en tabell med alla mintermer sorterade efter Hammingvikt

Hammingvikt	minterm	binärkod
0	0	0000
1	2	0010
	8	1000
2	3	0011
	5	0101
	8	1010
3	7	0111
	13	1101
4	15	1111

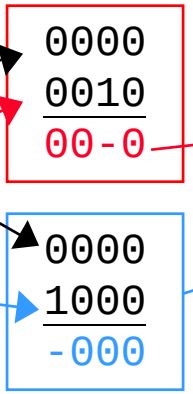


f(A,B,C,D)

1:a Reduktionen

- Utnyttja att AB+A'B = B(A+A') = B
  - Termerna i varje grupp jämförs med termerna i gruppen med närmast högre Hammingvikt
  - Varje term som ingått i en reduktion markeras

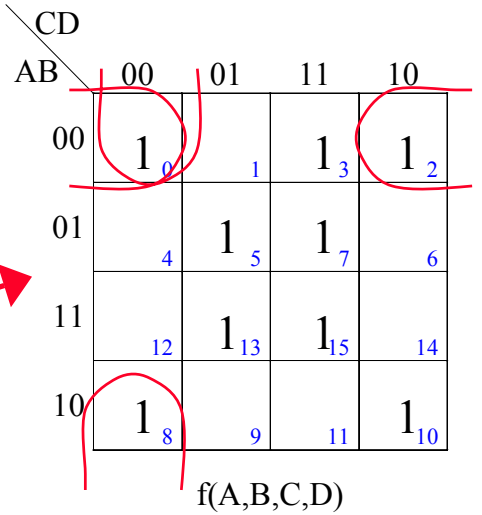
minterm	binärkod	
0	0000	x
2	0010	x
8	1000	x
3	0011	x
5	0101	x
8	1010	x
7	0111	x
13	1101	x
15	1111	x

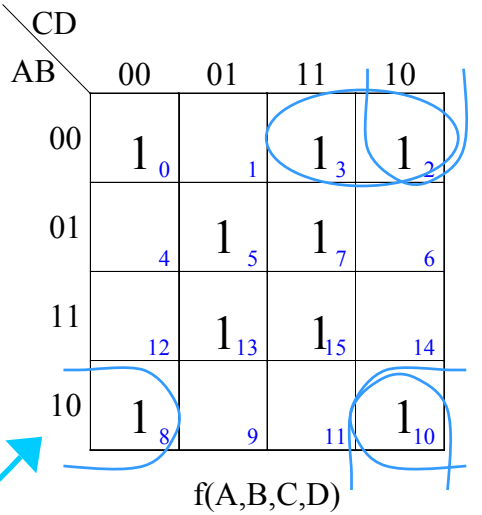


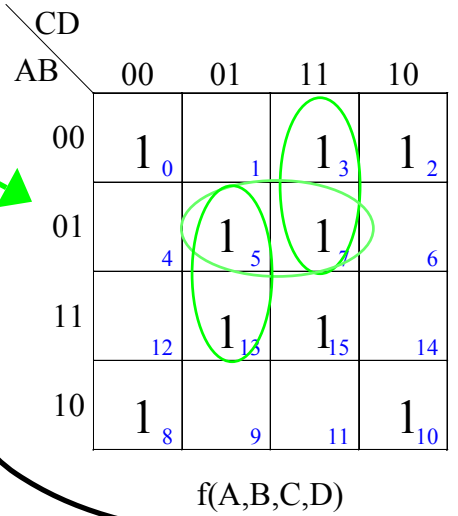
1:a reduktion
00-0 -000
001- -010 10-0
0-11 01-1 -101
-111 11-1

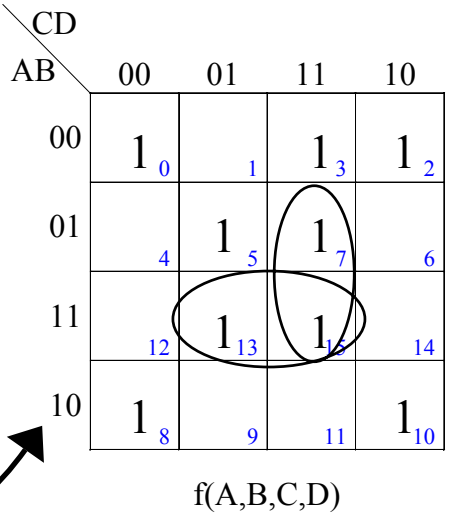
K-diagram ekvivalent

1:a reduktion
ABCD
00-0 -000
001- -010 10-0
0-11 01-1 -101
-111 11-1





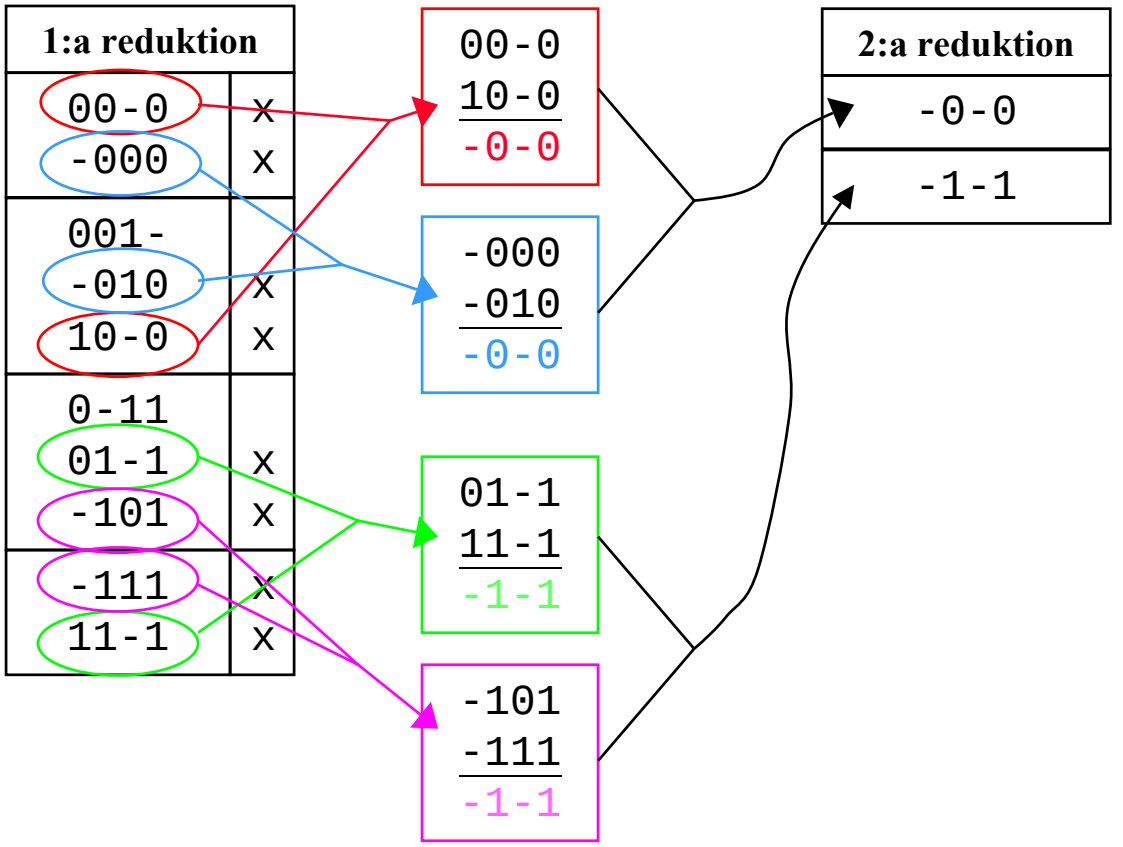




2:a Reduktionen

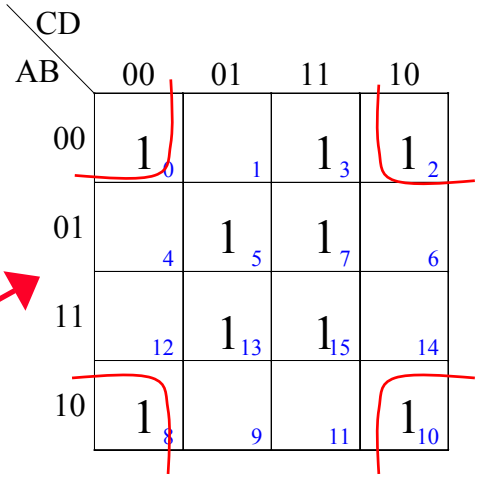
- Kombinera ihop termerna från 1:a reduktionen
  - Termer som kan kombineras har ’-’ på samma position
  - Markera alla termer som har använts för att bilda nya kombinationer

minterm	Binärkod	
0	0000	x
2	0010	x
8	1000	x
3	0011	x
5	0101	x
8	1010	x
7	0111	x
13	1101	x
15	1111	x

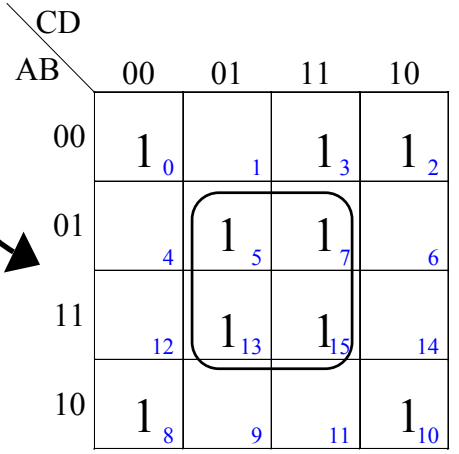


K-diagram ekvivalent

2:a reduktion
-0-0
-1-1



f(A,B,C,D)



f(A,B,C,D)

Samla ihop primimplikatorerna

- Alla termer som inte är markerade (x) är primimplikatorer

minterm	Binärkod	
0	0000	x
2	0010	x
8	1000	x
3	0011	x
5	0101	x
8	1010	x
7	0111	x
13	1101	x
15	1111	x

1:a reduktion	
00-0	x
-000	x
001-	
-010	x
10-0	x
0-11	
01-1	x
-101	x
-111	x
11-1	x

2:a reduktion
-0-0
-1-1

Primimplikatorerna är genererade !!

$$p_1 = \overline{A}\,\overline{B}C$$

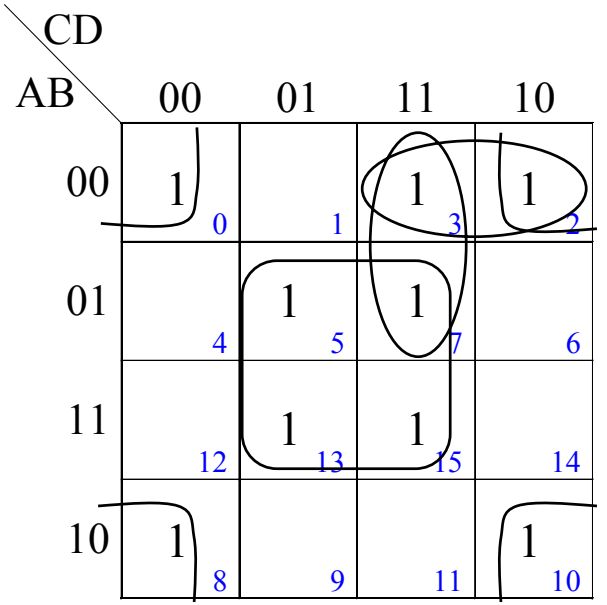
$$p_2 = \overline{A}CD$$

$$p_3 = \overline{B}\,\overline{D}$$

$$p_4 = BD$$

Primimplikatorer i ett K-diagram

Primimplikator	minterm
$p_1 = \overline{A}\overline{B}C$	2,3
$p_2 = \overline{A}CD$	3,7
$p_3 = \overline{B}\overline{D}$	0,2,8,10
$p_4 = BD$	5,7,13,15



f(A,B,C,D)

Urvalstabell

- Identifiera väsentliga primimplikatorer

no. var	PI	0	2	3	5	7	8	10	13	15
2	1		x	x						
2	2			x		x				
3	3	x	x				x	x		
3	4				x	x			x	x

Primimplikator	minterm
$p_1 = \overline{A}\overline{B}C$	2,3
$p_2 = \overline{A}CD$	3,7
$p_3 = \overline{B}\overline{D}$	0,2,8,10
$p_4 = BD$	5,7,13,15

Reduktion av primimplikatortabell

no. var	PI	0	2	3	5	7	8	10	13	15
2	1		x	x						
2	2			x		x				
3	3	x	x				x	x		
3	4				x	x			x	x

- Reducera bort väsentliga primimplikatorer från tabellen
  - p_3 och p_4 är väsentliga PI och kan strykas samt kolumner som täcks av dessa kan också strykas

no. var	PI	3
2	1	x
2	2	x

p_1 och p_2 är likvärdiga
Välj vilken som helst av dem

Förenklat logiskt uttryck

- p_3, p_4 och p_1 väljs som primimplikatorer

$$f(A,B,C,D) = \overline{B}\,\overline{D} + BD + \overline{A}\,\overline{B}C$$

- Alternativ lösning: p_3, p_4 och p_2 väljs som primimplikatorer

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

SLUT på Föreläsning 3

- Innehåll
  - Karnaugh-diagram för 5 och 6 variabler
  - Quine-McCluskey metoden

Kombinatoriska byggblock

- Innehåll
  - Aritmetiska operationer
    - Addition
    - Subtraktion
    - Multiplikation
  - Kodare och avkodare
  - Multiplexer och demultiplexer
  - Paritetskretsar

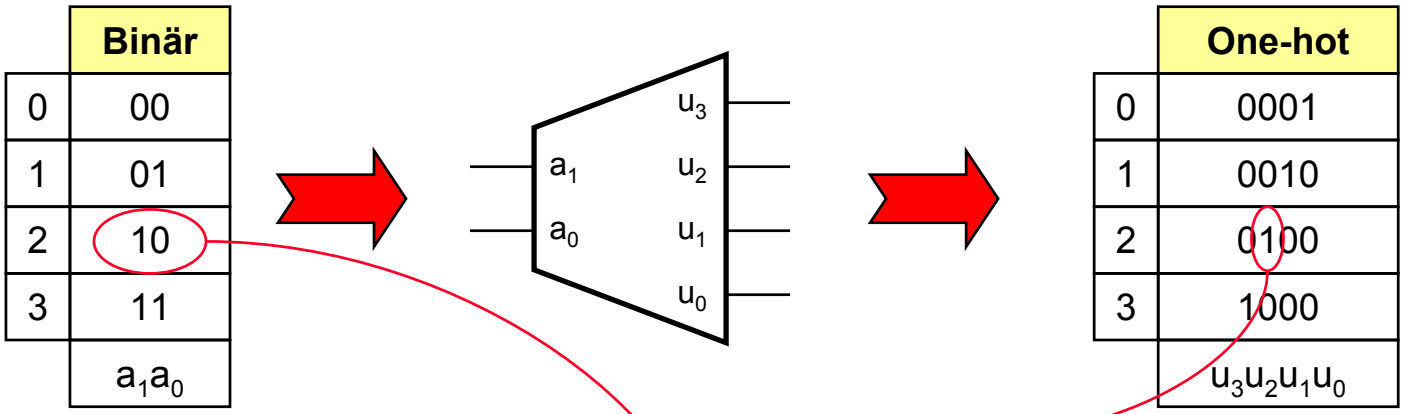
Introduktion

- En del kretsar används så ofta att de inte konstrueras på nytt utan återanvänds som ett grundläggande byggblock

- Exempel på sådana typer av byggblock
  - Kod-omvandlare
  - Multiplexer
  - Aritmetiska enheter (+, -, ·)

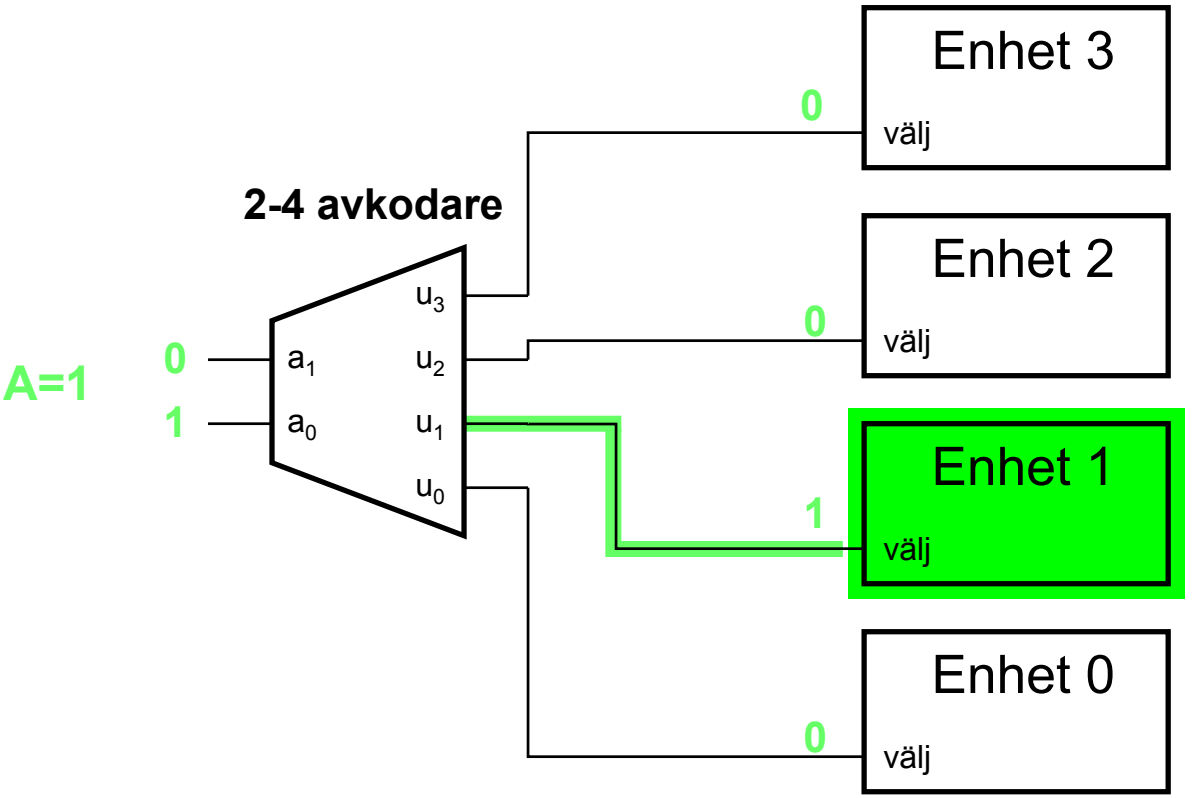
Binär avkodare

- Konverterar ett n-bitars binärt tal till ett 2^n-bitars *One-hot* kod



Det binära talet A (A={a_1,a_0}) sätter utsignalen u_A till ’1’ och övriga utsignaler till ’0’.

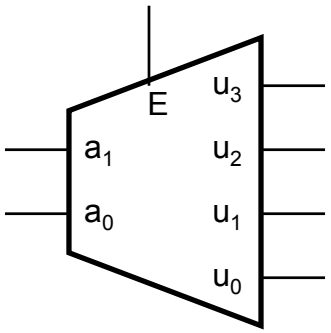
Exempel på användning av binär 2-4 avkodare



Väljer ut (adresserar) endast en enhet av flera

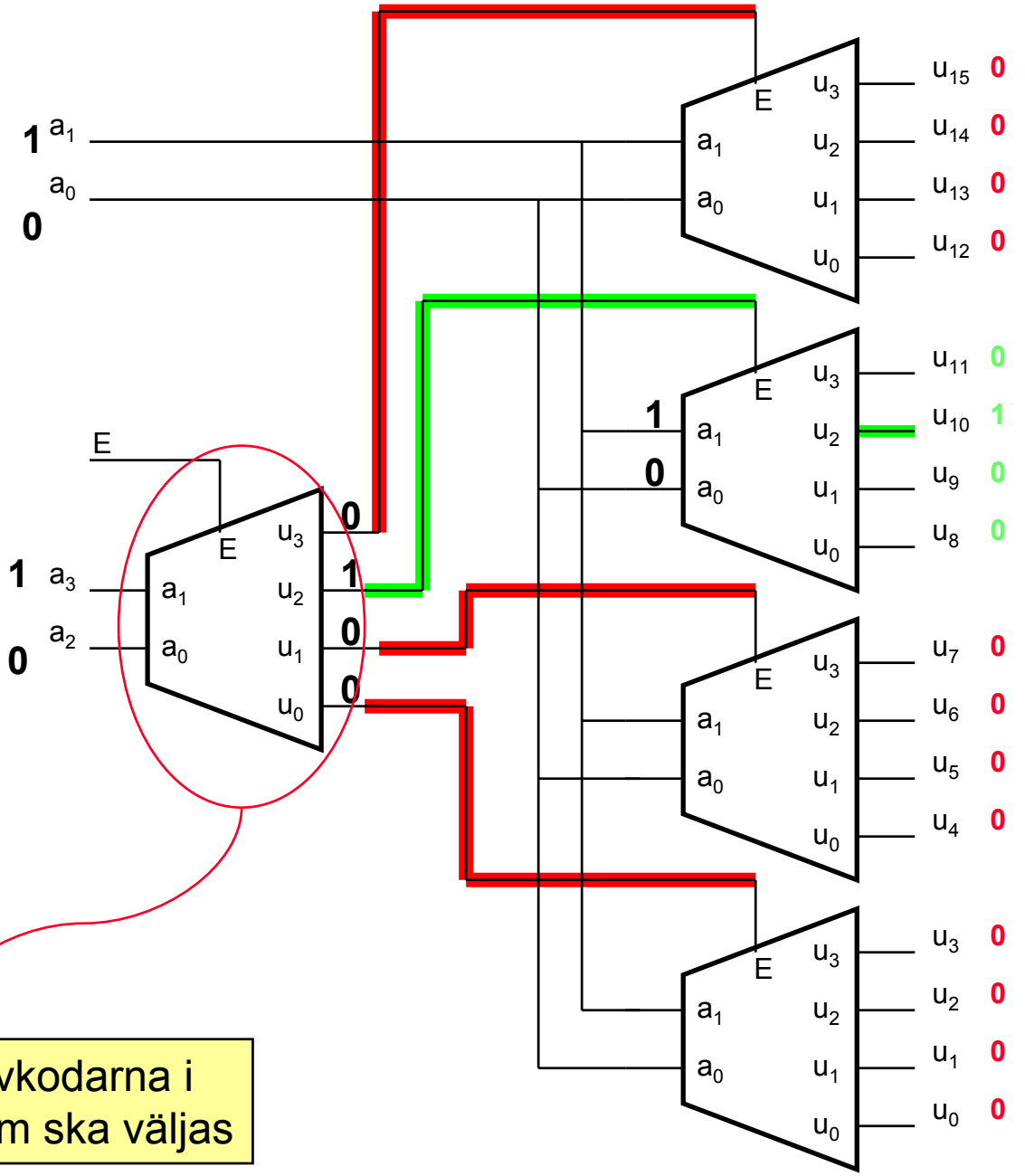
Avkodare med *enable*

- Med *enable*-signal kan man från 2-4 avkodare bygga större avkodare



E	a_1a_0	$u_3u_2u_1u_0$
0	00	0000
0	01	0000
0	10	0000
0	11	0000
1	00	0001
1	01	0010
1	10	0100
1	11	1000

4-16 avkodare

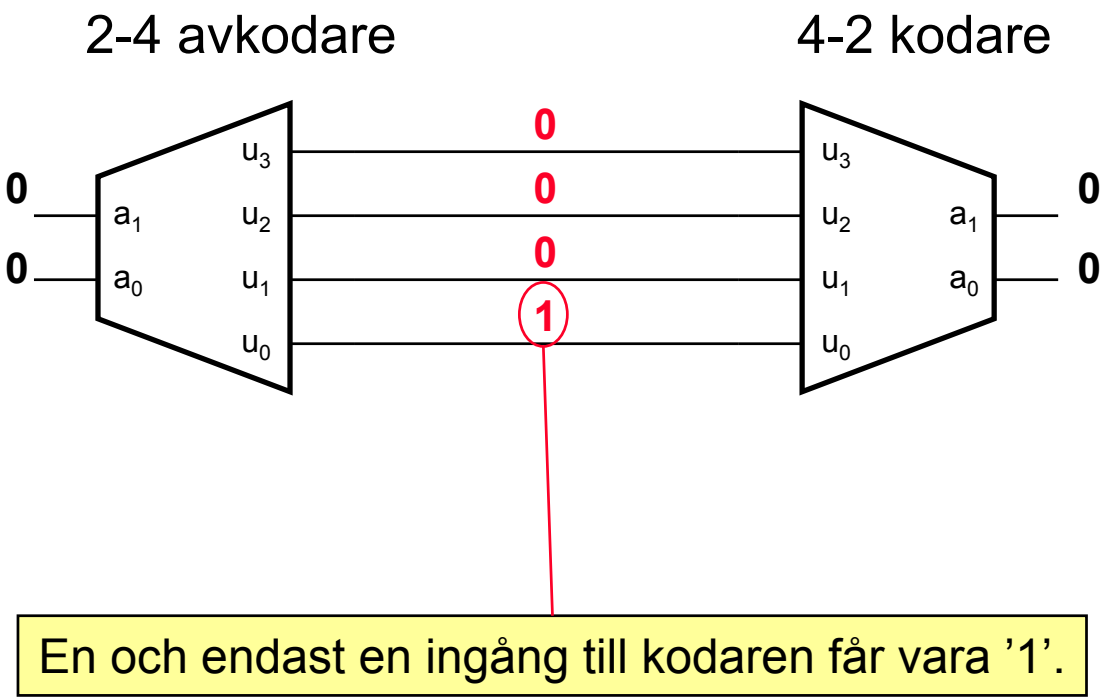


$A=10_{10}=1010_2 \Rightarrow$
$a_3 = 1$
$a_2 = 0$
$a_1 = 1$
$a_0 = 0$

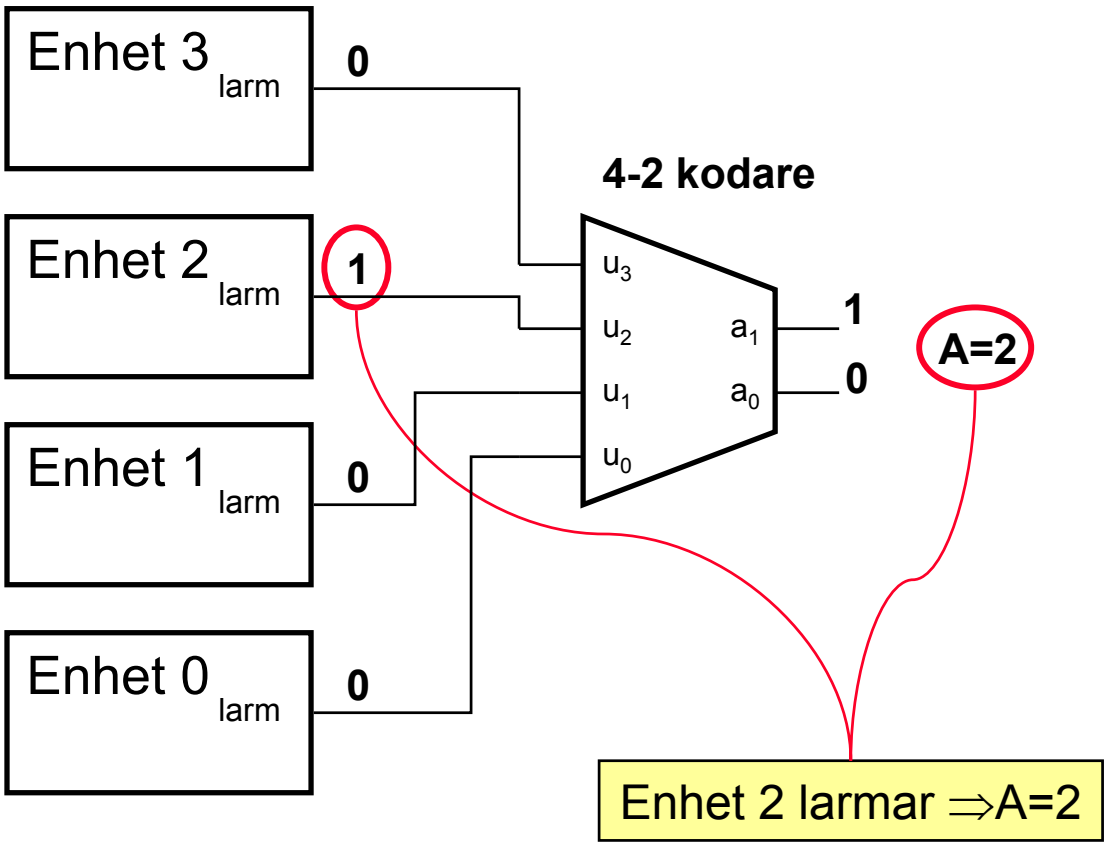
Väljer ut vilken av avkodarna i Det andra steget som ska väljas

Kodare

- Har ”invers” funktion av en avkodare



Exempel på användning av binär 4-2 kodare

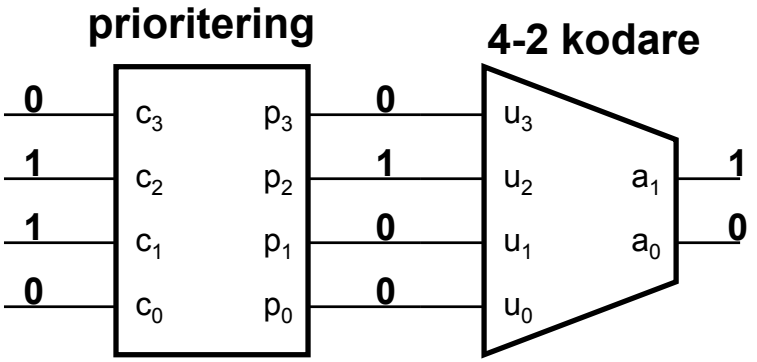


Vad händer om två enheter larmar samtidigt ?

Vad händer om ingen enhet larmar ?

Prioritetskodare

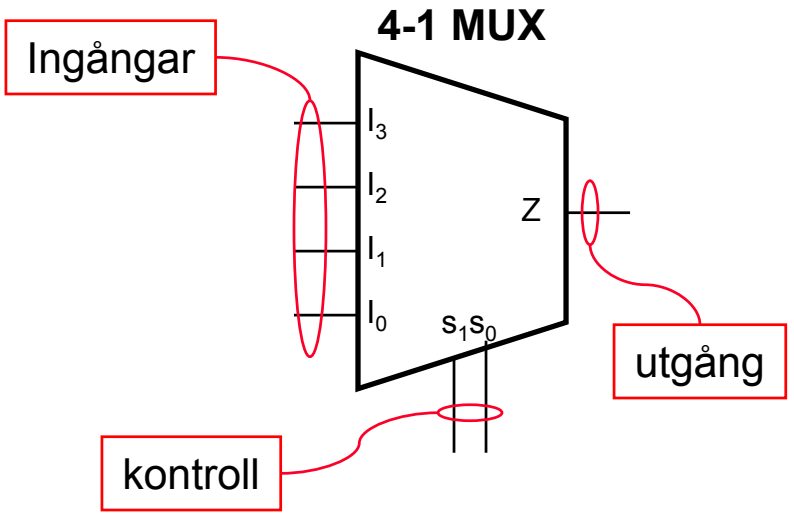
- Ingångarna har inbördes prioritet
  - När mer än en ingång är aktiv så genereras en kod för den ingång som har högst prioritet
  - Exempel:
    - Låt insignal 3 ha högst prioritet, 2 näst högst, o.s.v

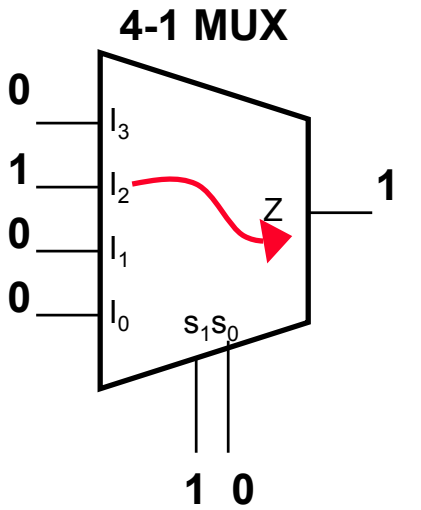


$p_3 = c_3$
$p_2 = p_2 \cdot c_3'$
$p_1 = p_1 \cdot c_3' \cdot c_2'$
$p_0 = p_0 \cdot c_3' \cdot c_2' \cdot c_1'$

Multiplexer

- Ett binärt tal (S) kontrollerar vilken av ingångarna (I_j) som ska bestämma utsignalens (Z) värde

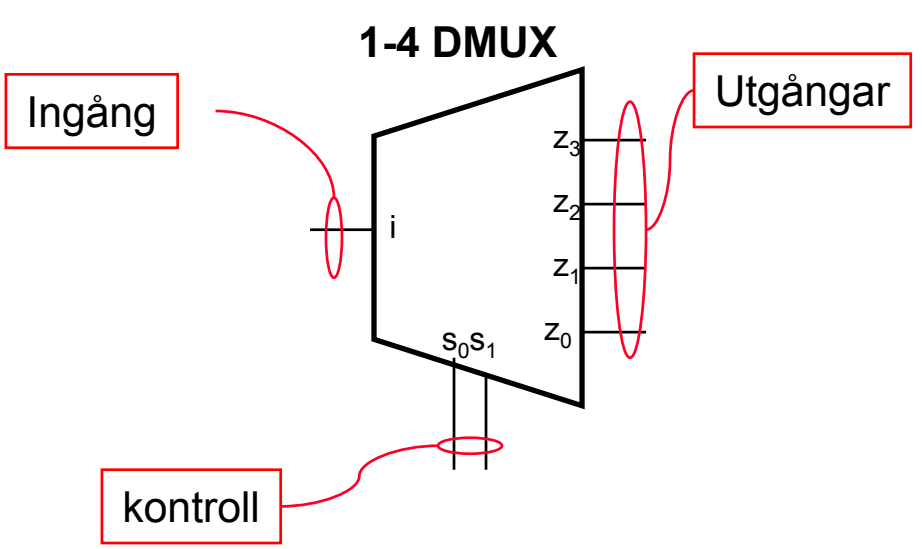


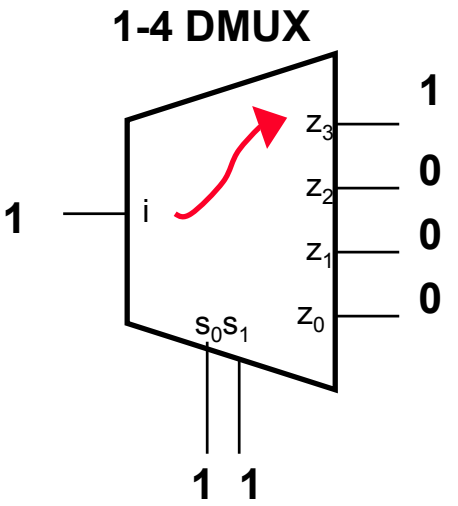


S=2 ⇒ värdet av I_2 visas på Z

Demultiplexer

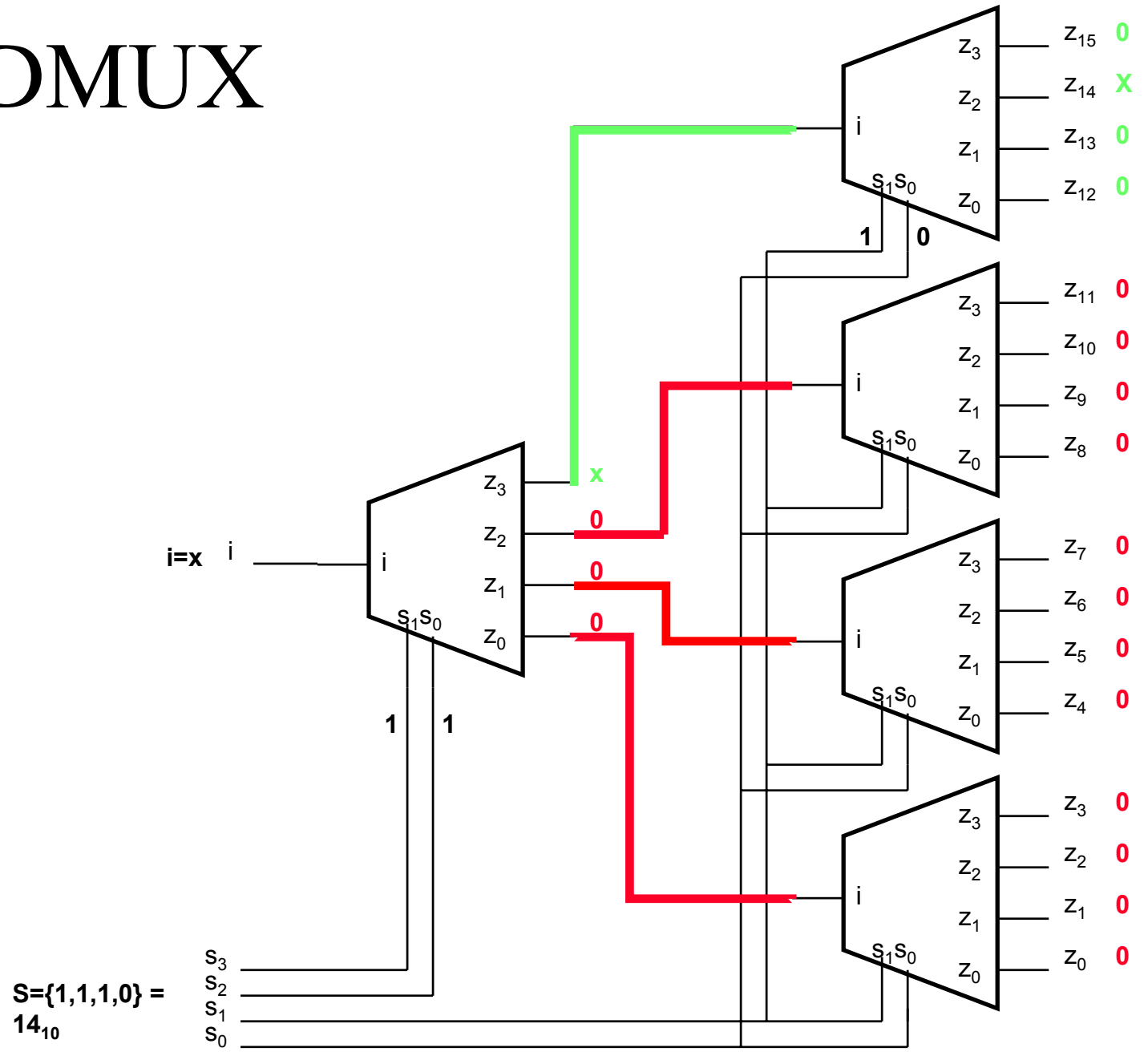
- Ett binärt tal (S) kontrollerar vilken av utgångarna (z_j) som ska tilldelas värdet av ingången (i)



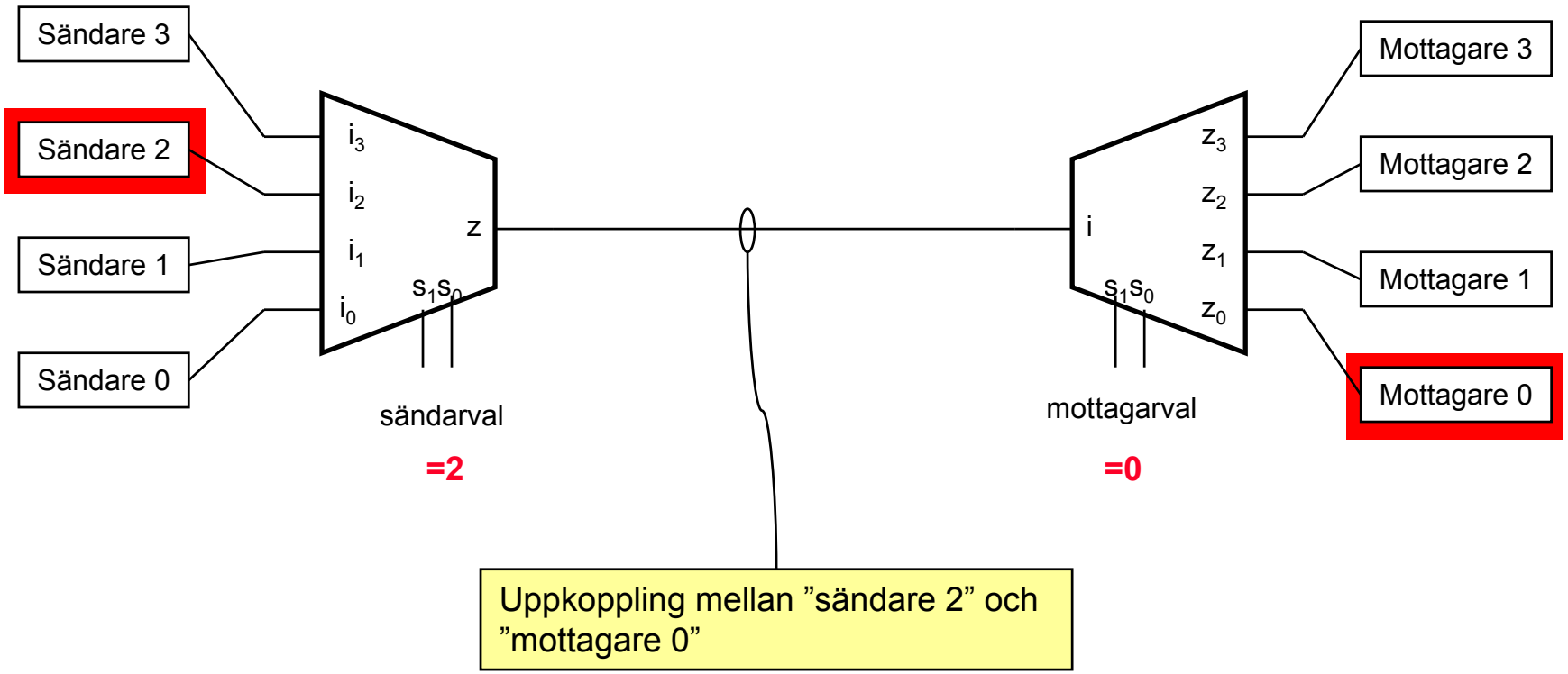


S=3 ⇒ värdet av i visas på z_3

1-16 DMUX


i=x
i
S={1,1,1,0} = 14₁₀
S3
S2
S1
S0
1
1
1
0
x
0
0
0
z15 0
z14 x
z13 0
z12 0
z11 0
z10 0
z9 0
z8 0
z7 0
z6 0
z5 0
z4 0
z3 0
z2 0
z1 0
z0 0

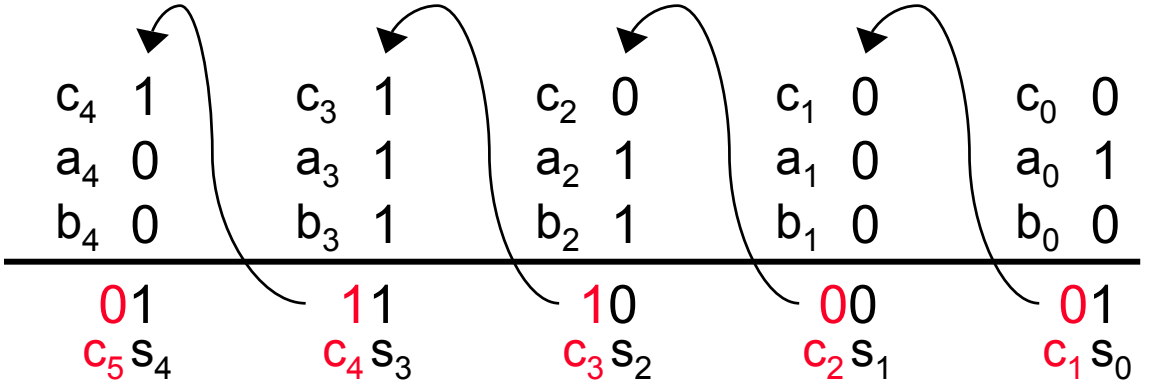
Exempel på användning av MUX och DMUX



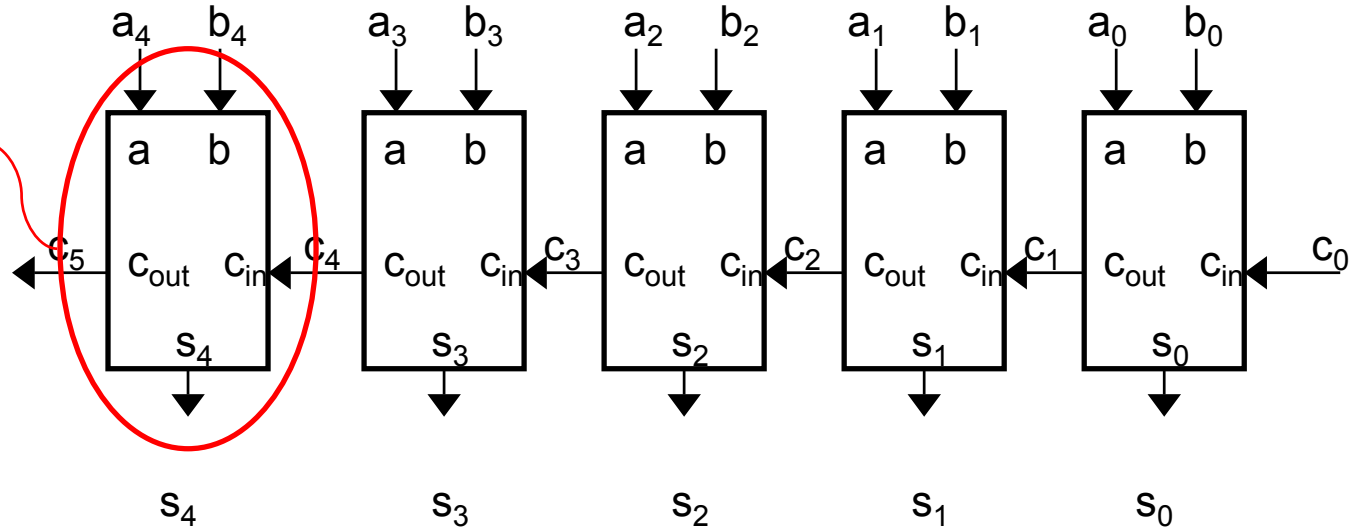
Addition

Minnessiffror (eng. carry bits)

```
  1 1 0 0 0
  01101
+ 01100
-------
  11001
```

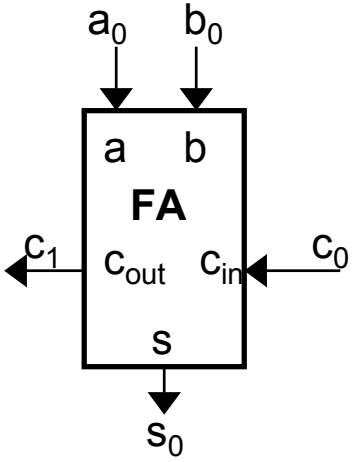


heladderare



Heladderare (*eng. Full-adder*)

- Funktion
  - Utför 1-bits addition
    - 1 bit för summa
    - 1 bit för *carry-out* (minnesbit)



a	b	c_{in}	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

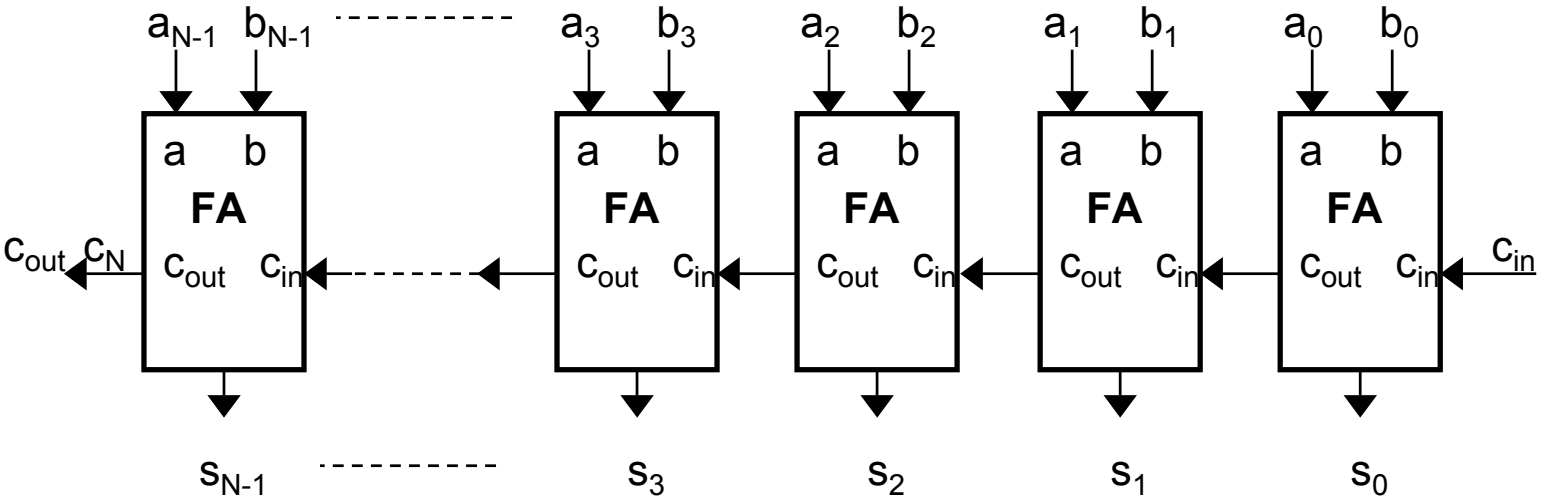
Minimering m.h.a K-diagram

$$c_{out} = ab + ac_{in} + bc_{in}$$

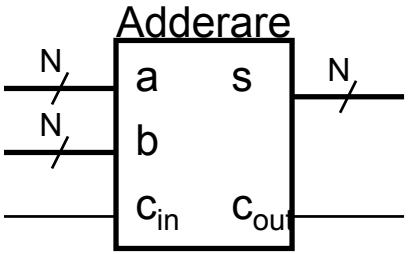
$$s = \bar{a}\bar{b}c_{in} + \bar{a}b\bar{c}_{in} + a\bar{b}\bar{c}_{in} + abc_{in}$$

Fler-bits adderare

- N-bitars adderare
  - Kaskadkopplade Full-adderare

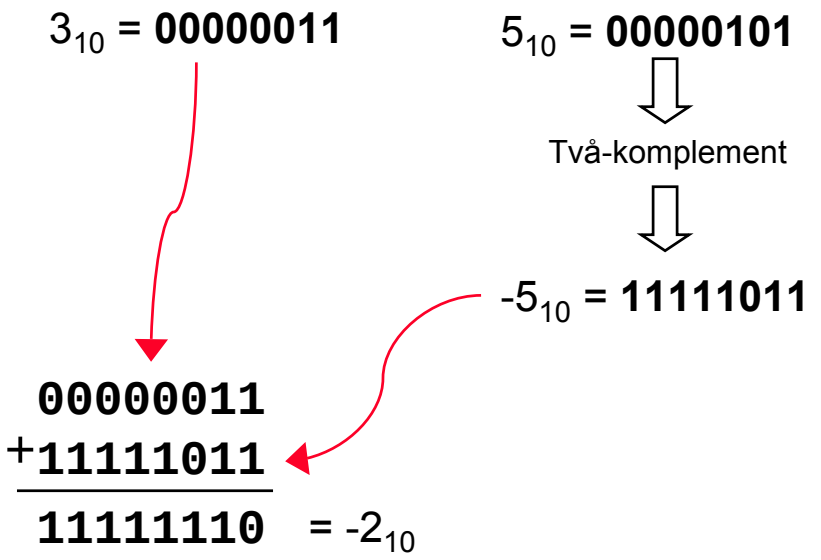


- Symbol:



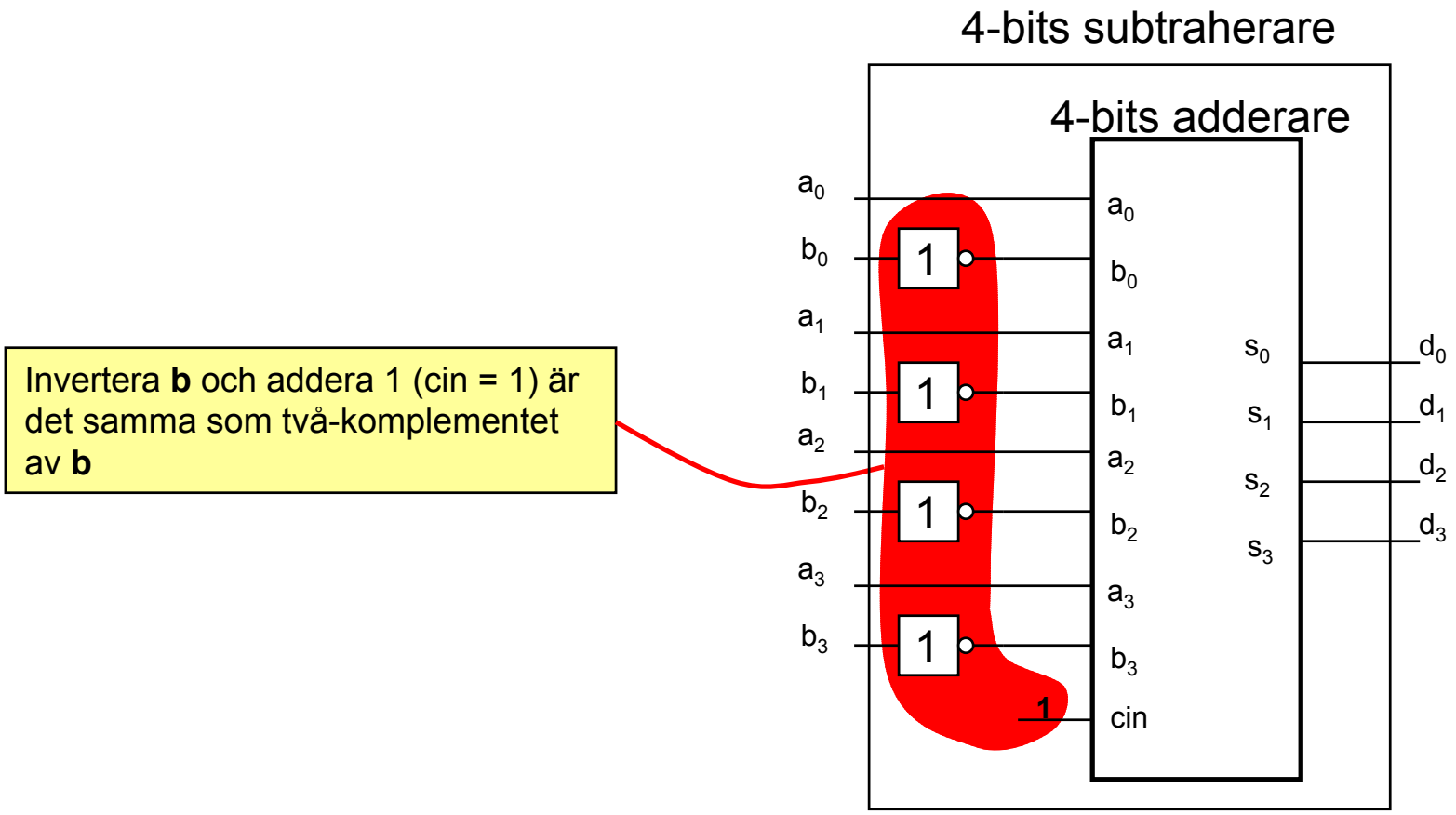
Subtraktion

- Subtraktion i två-komplement
  - Differens = subtraktor – subtrahend
  - Ta fram två-komplementet av subtrahenden
  - Addera subtraktorn och subtrahenden
    … Differens = subtraktor + (-subtrahend)
  - Exempel: Beräkna 3 – 5 som är 8-bitars tal



Subtraherare

- 4-bitars subtraherare: d = a - b

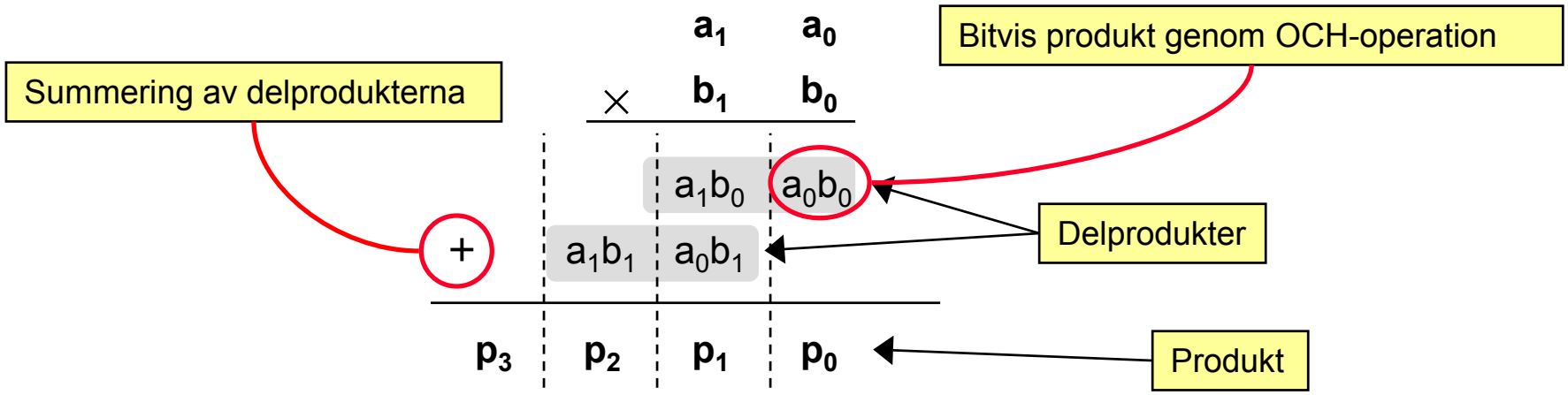


Binär multiplikation

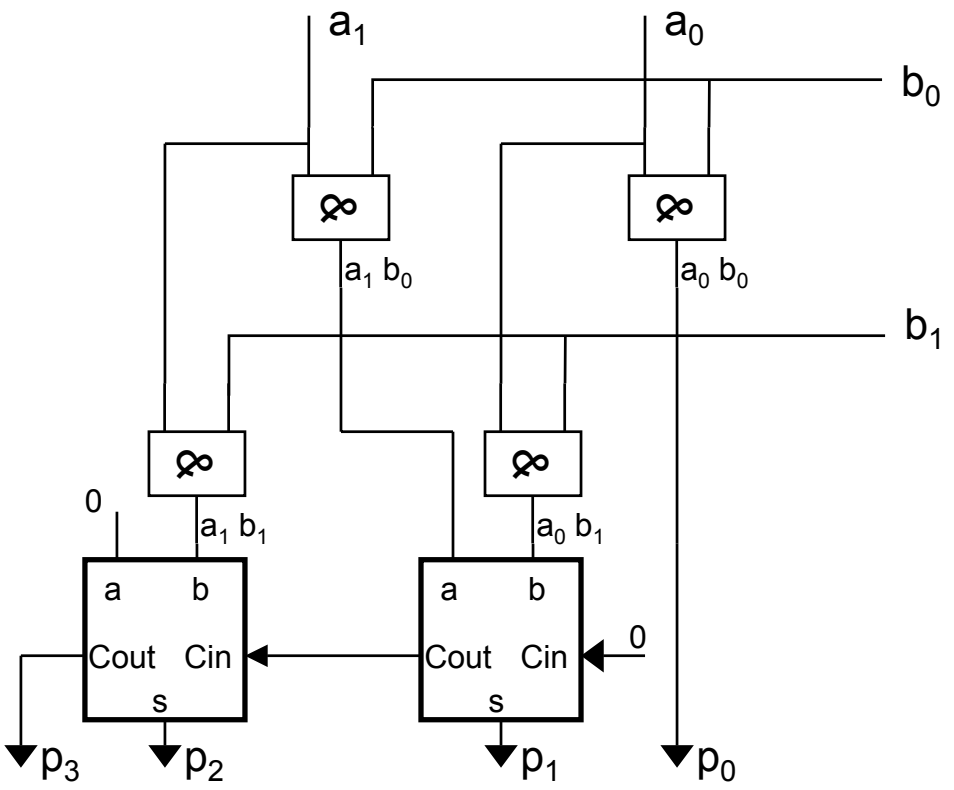
- Exempel: Utför multiplikationen 2 × 3

```
  10
×11
  10
 10
110   = 6
```

- Operationer i en multiplikation

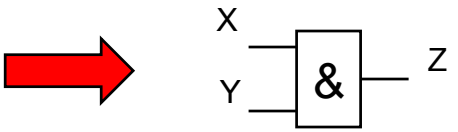


2 ×2 bitars multiplikator



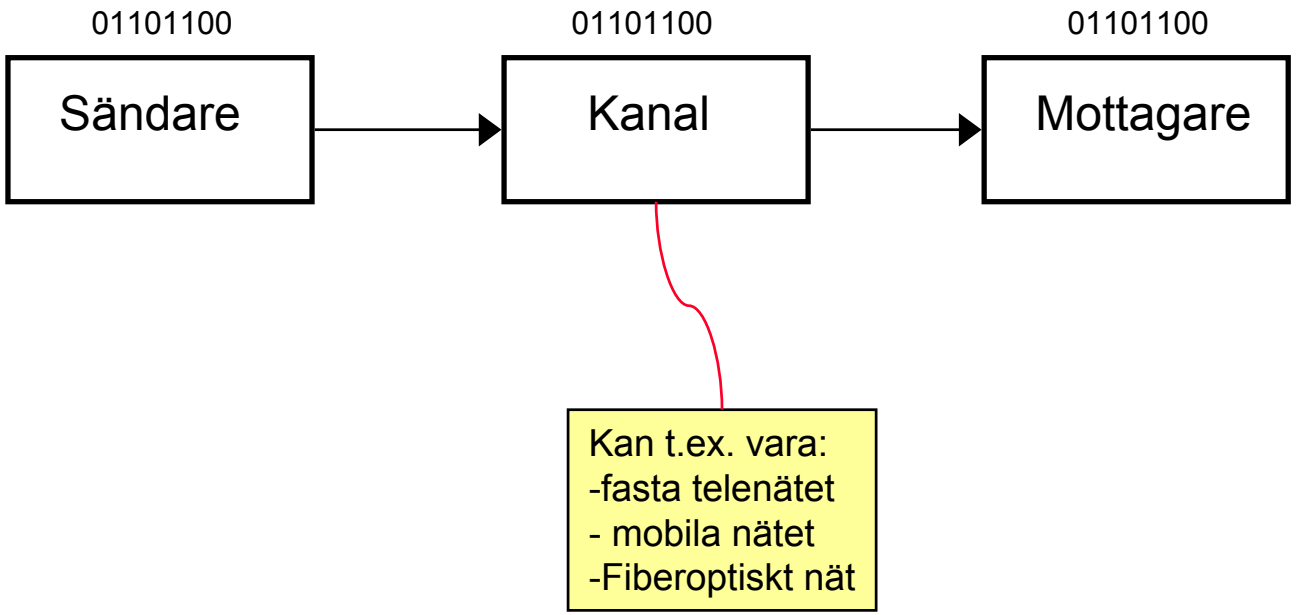
1-bits multiplikation

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1



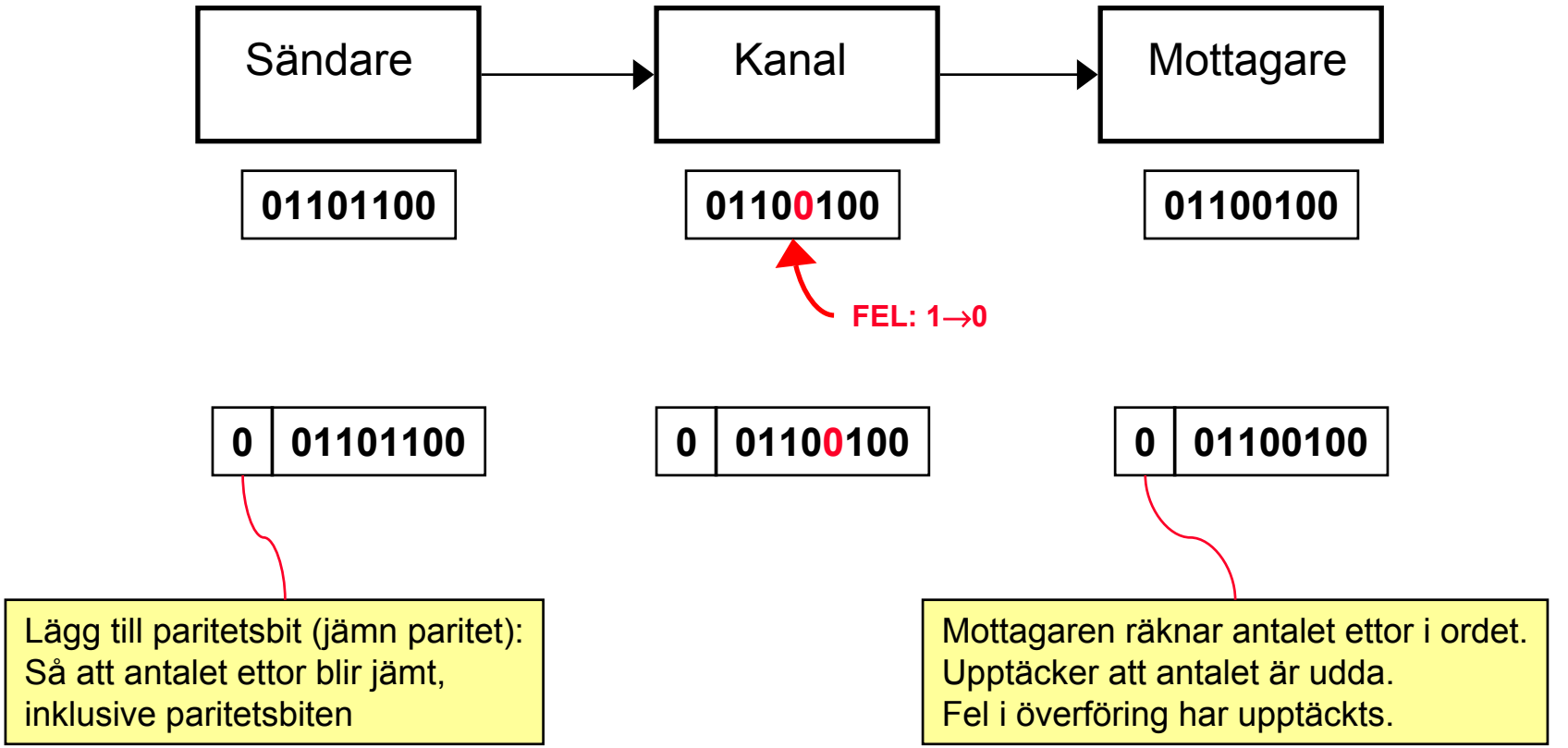
Felupptäckande koder

- Modell för informationsöverföring



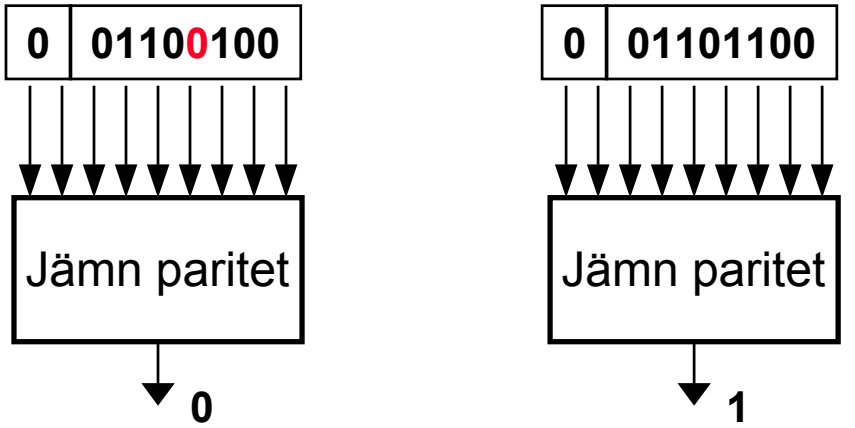
Felupptäckande koder

- Paritetsbit som felupptäckande kod

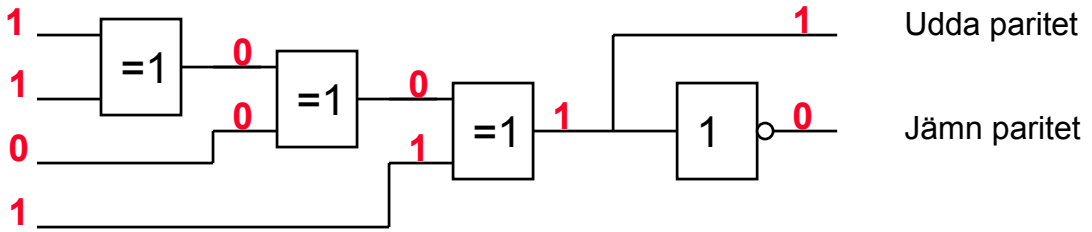


Paritetskretsar

- Krets för jämn paritet
  - Ger ’1’ ut om antalet 1:or in är jämnt



- Exempel på 4-bitars paritetskrets med xor-grind



SLUT på Föreläsning 5

- Innehåll
  - Kodare och avkodare
  - Multiplexer och demultiplexer
  - Aritmetiska operationer
    - Addition
    - Subtraktion
    - Multiplikation
  - Paritetskretsar

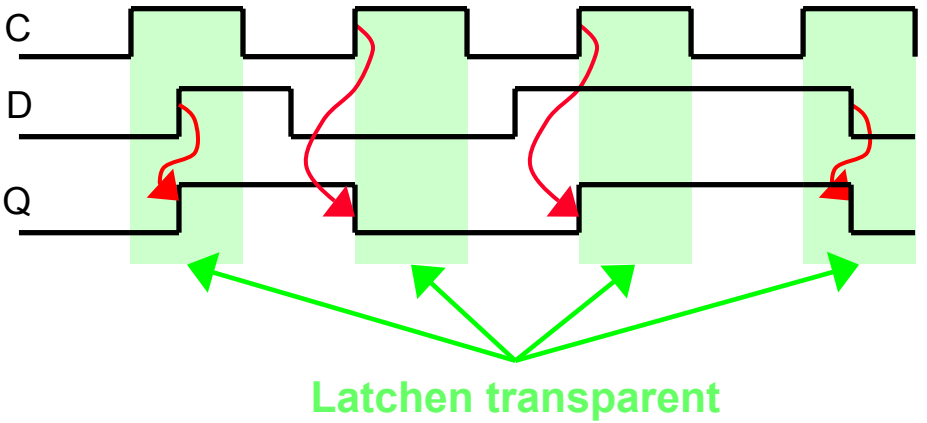
Minnen

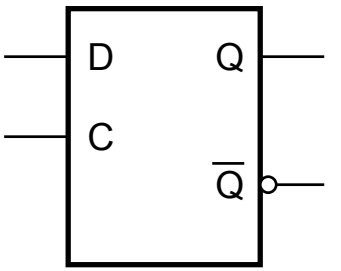
- Innehåll
  - Minneselement
    - Låskrets (*eng. latch*)
    - Vippa (*eng. Flip-flop*)
    - Register
  - Random-Access Memory (RAM)
  - Read-Only Memory (ROM)

Minneselement

- Tillåter lagring av binära variabler för framtida beräkningar
  - Latch
    - Dess värde ändras vid klocksignalens aktiva nivå (t.ex C=1)
    - Grundläggande bistabilt minneselement
    - Lagrar 1 bit
  - Vippa (eng. Flip-flop)
    - Dess värde ändras endast vid klocksignalens flank
    - Lagrar 1 bit
    - Används t.ex i tillståndsmaskiner och register
  - Register
    - En grupp vippor som klockas med samma klocka
    - Lagrar en binär variabel som består av flera bitar

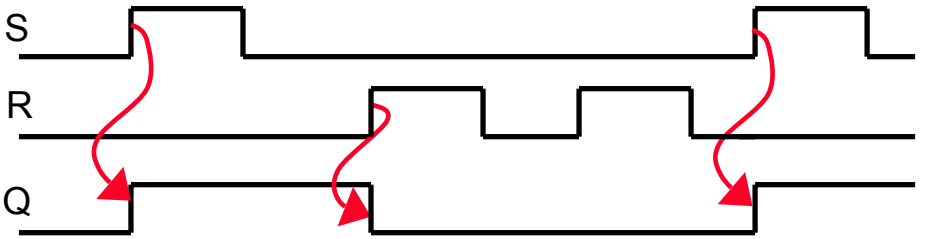
D-latch – funktion

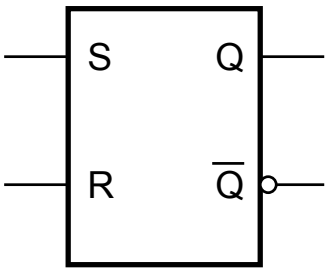




- En D-latch kan vara i ett av två lägen
  - Transparent – Utgången (Q) följer värdet på ingången (D)
  - Lås – Värdet på utgången (Q) bibehålls och är oberoende av ingången (D)
- Ingången C kontrollerar i vilket läge latchen befinner sig i

SR-latch – funktion

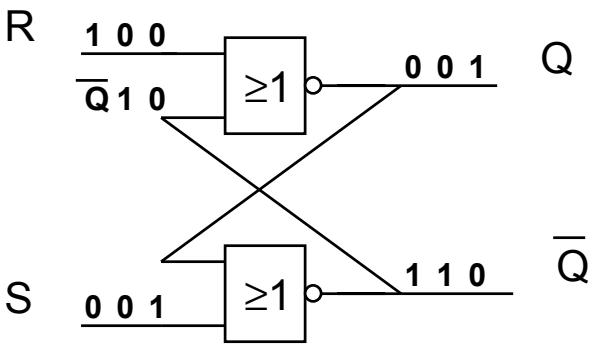


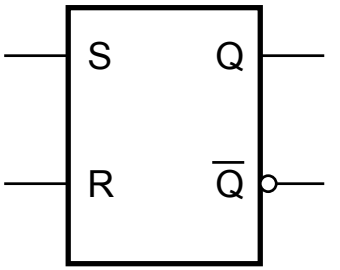


- En SR-latch har två ingångar som aktiverar latchens operationer
  - Set: Ingången S sätter latchen till 1 (Q=1)
  - Reset: Ingången R sätter latchen till 0 (Q=0)

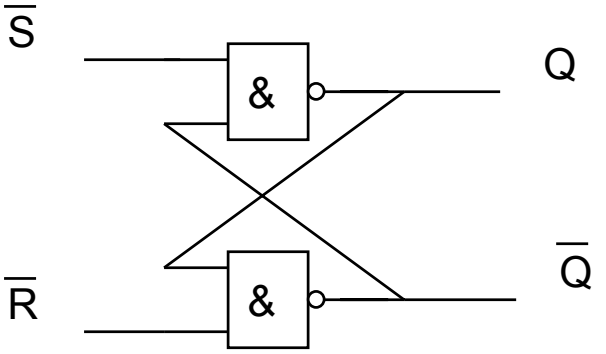
SR-latch – uppbyggnad

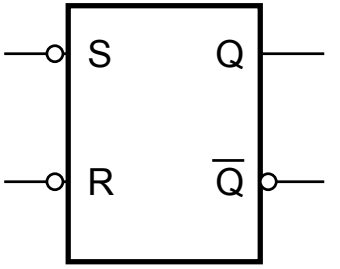
S	R	Q^+	Operation
0	0	Q	Behåll Q
1	0	1	Set Q → 1
0	1	0	Reset Q → 0
1	1	0	Otillåten komb.



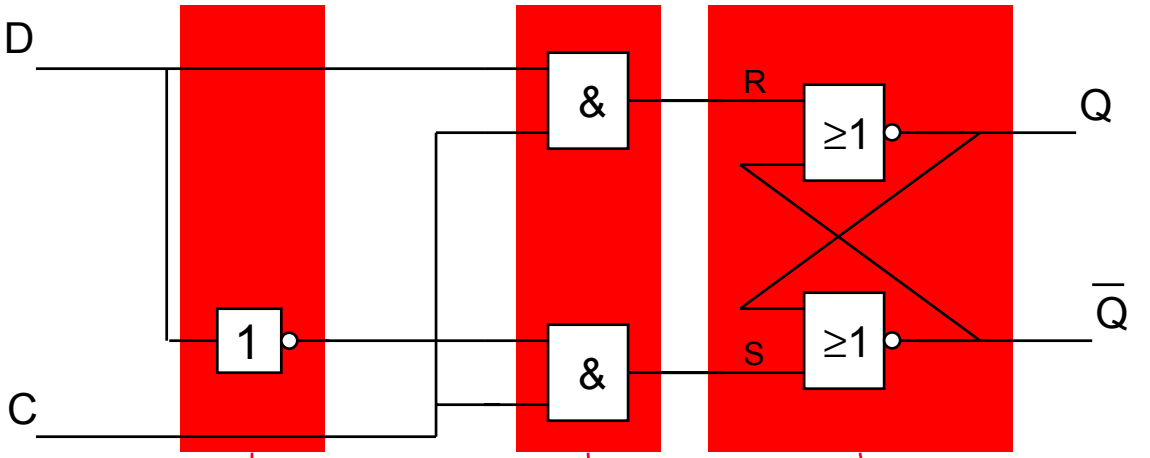


S'	R'	Q^+	Operation
0	0	Q	Otillåten komb.
1	0	1	Reset Q → 0
0	1	0	Set Q → 1
1	1	0	Behåll Q





D-latch – uppbyggnad

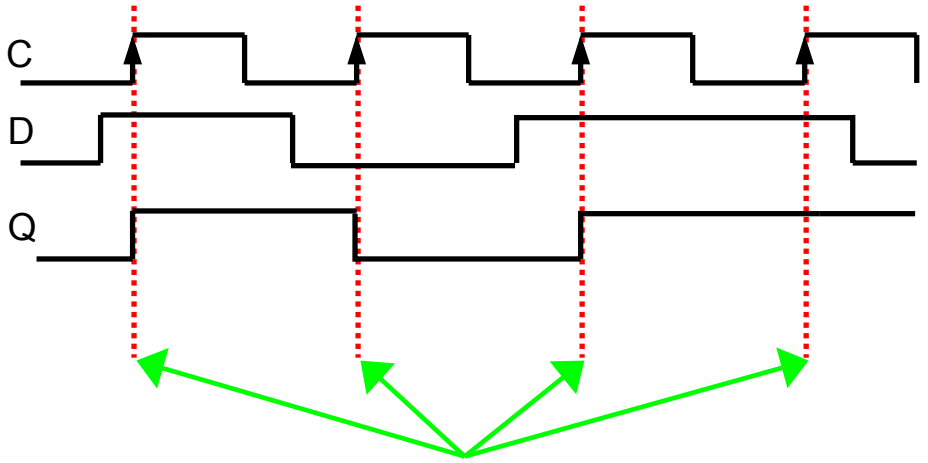


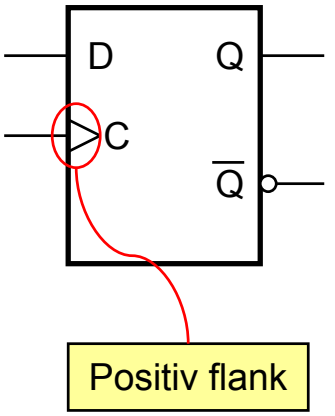
SR-latch

Inverteraren skapar S- och R-signaler som alltid är varandras invers:
S=D och R=D'

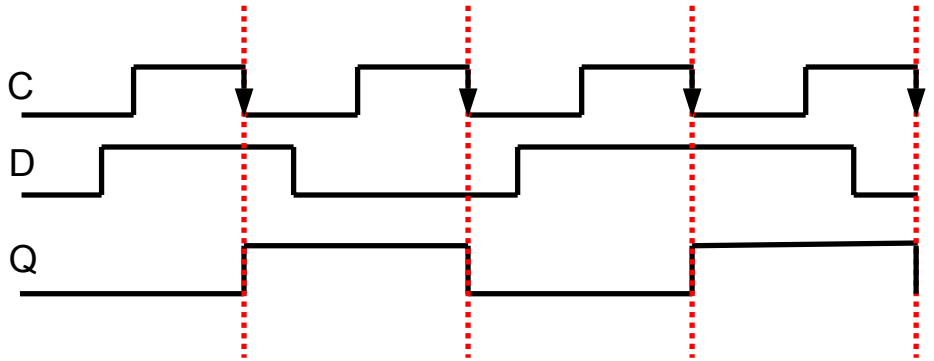
OCH-grindar som gör att då C=0 ändras inte latchens innehåll:
C=0 $\rightarrow$S=0; R=0 $\rightarrow$ $Q^+ = Q$

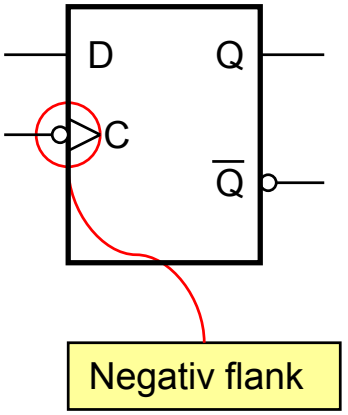
D-vippa – funktion



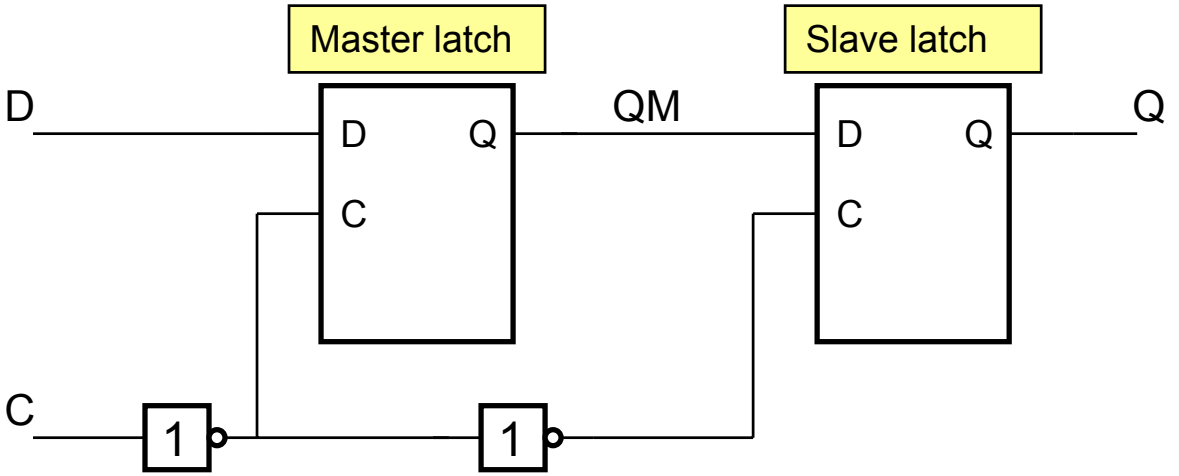


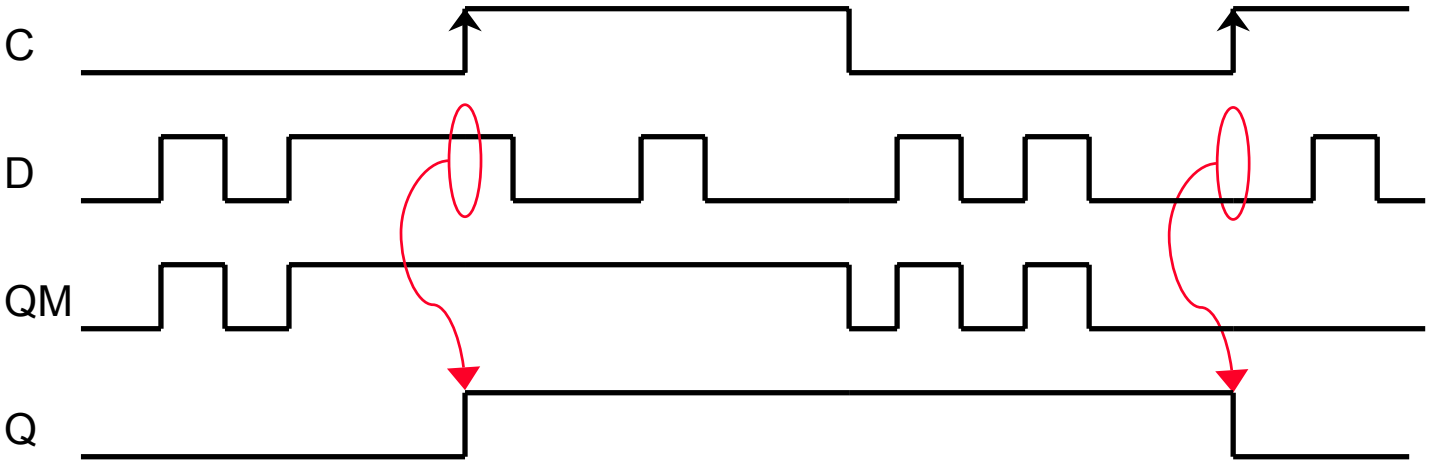
D-vippan ”samplar” värdet på
D-ingången vid klocksignalens flank





D-vippa – uppbyggnad

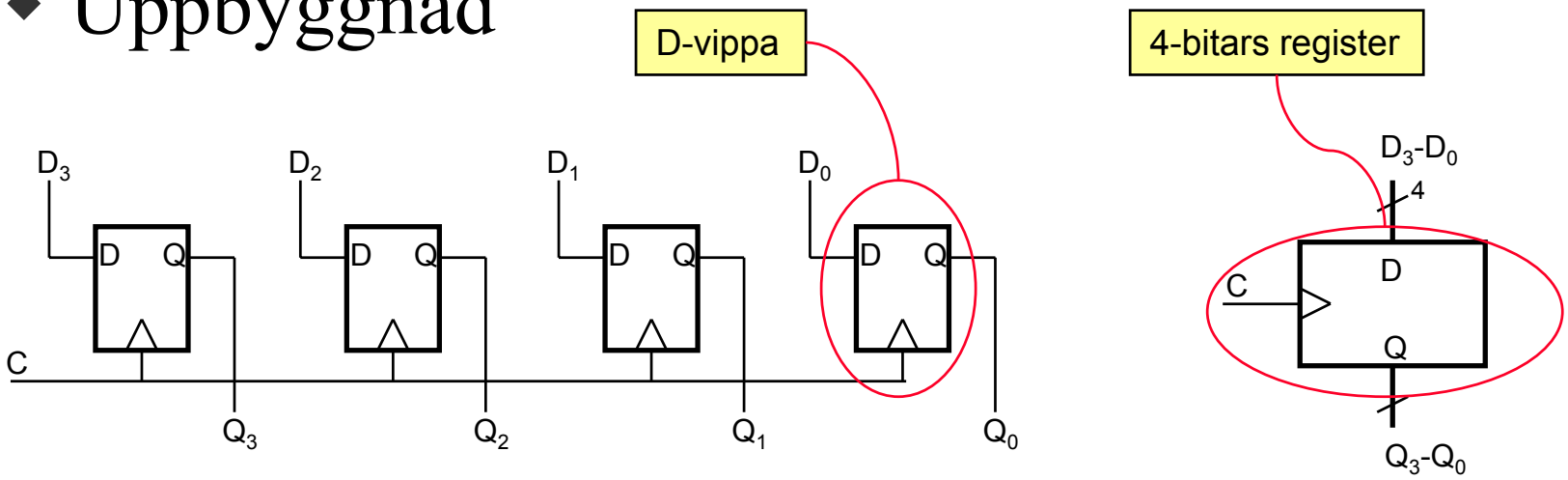




Master Transparent	Master Låser	Master Transparent	Master Låser
Slave Låser	Slave Transparent	Slave Låser	Slave Transparent

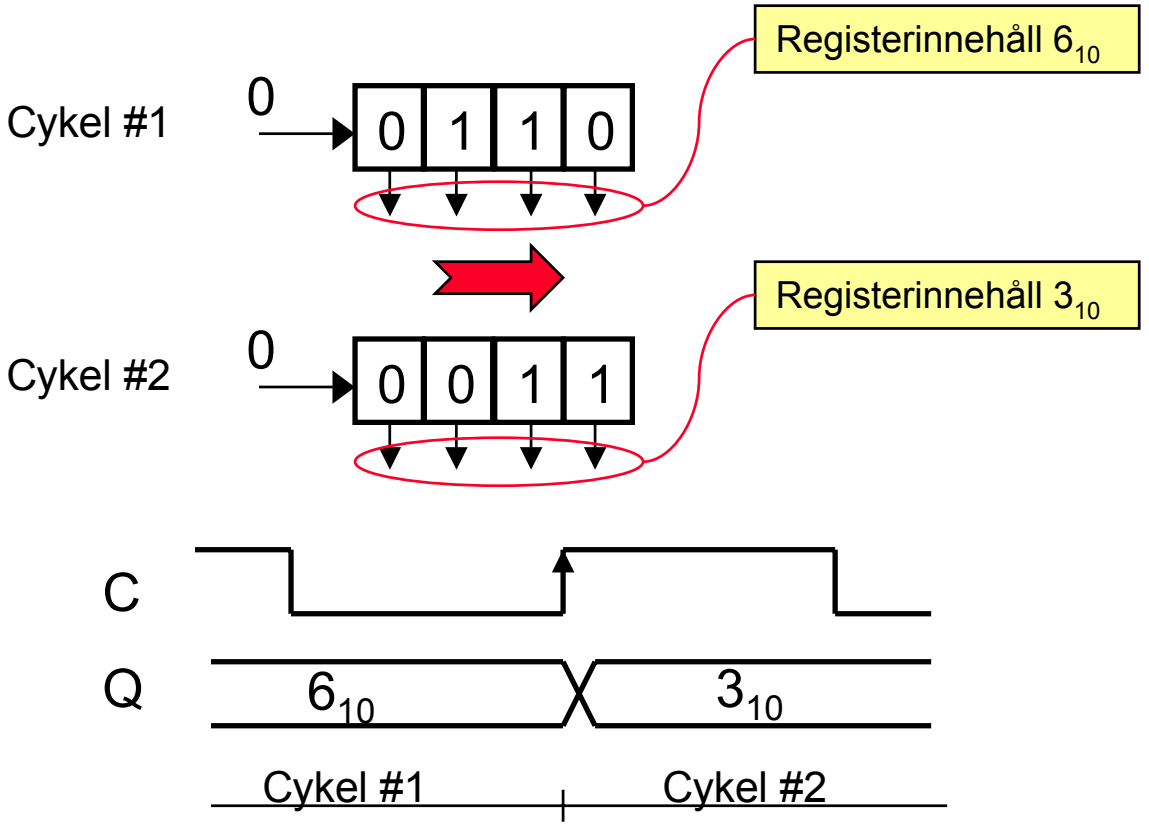
Register

- Funktion
  - Lagrar ett n-bitars binärt tal
  - Bitarna i talet skrivs till registret och läses från registret parallellt
- Uppbyggnad

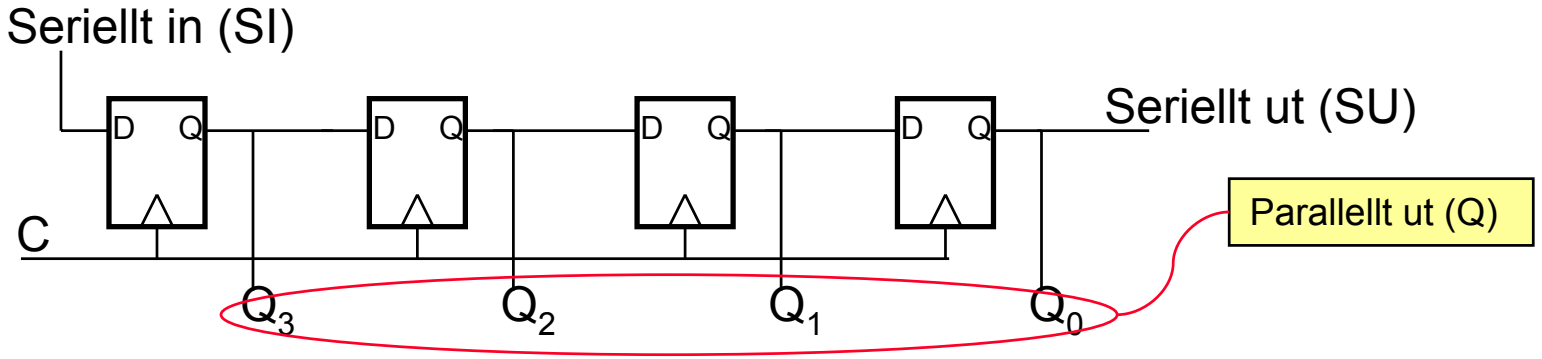


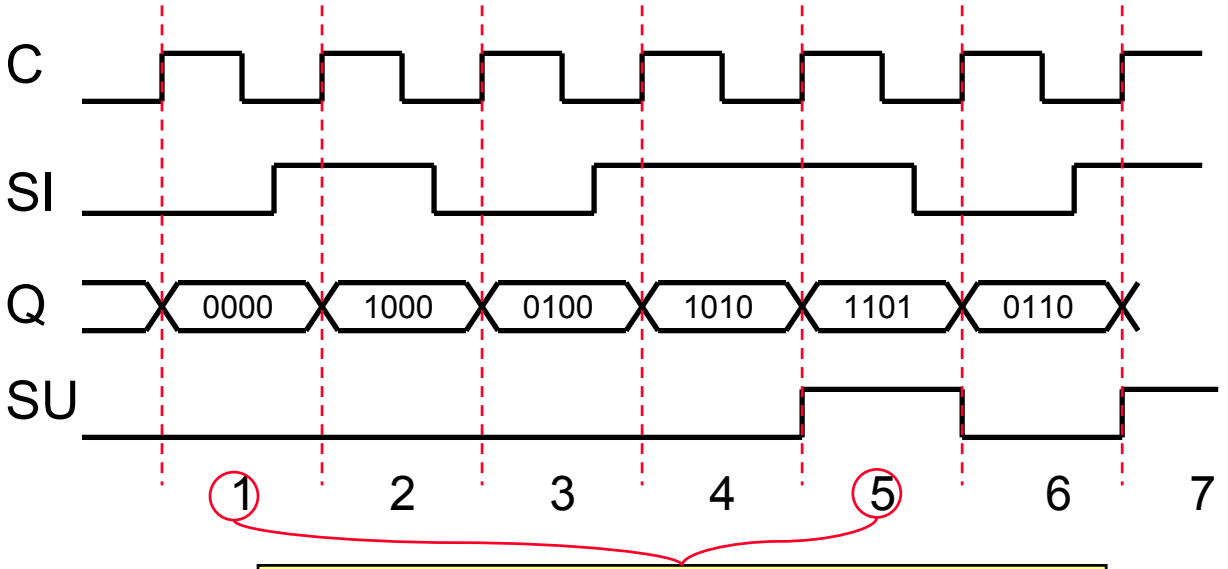
Skiftregister – funktion

- Funktion
  - Flyttar talet i ett register i ”sidled” åt vänster eller höger



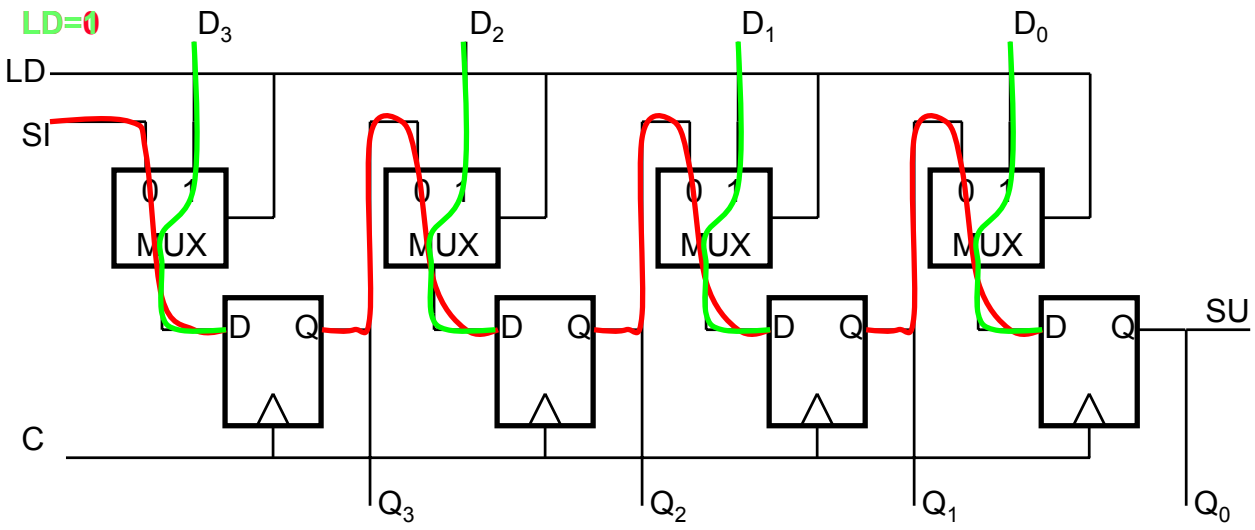
Skiftregister – uppbyggnad





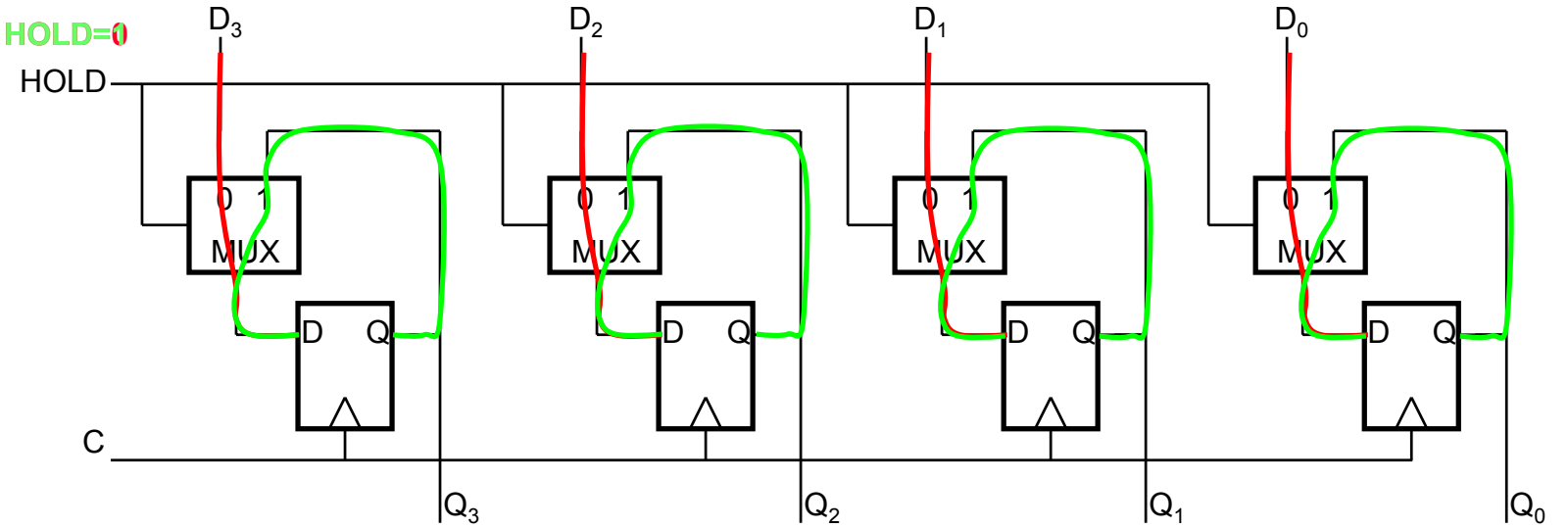
Skiftregister med parallell laddning

- Kan ställas i två lägen
  - Skifta innehållet (LD=0)
  - Ladda in ett binärt tal parallellt (LD=1)



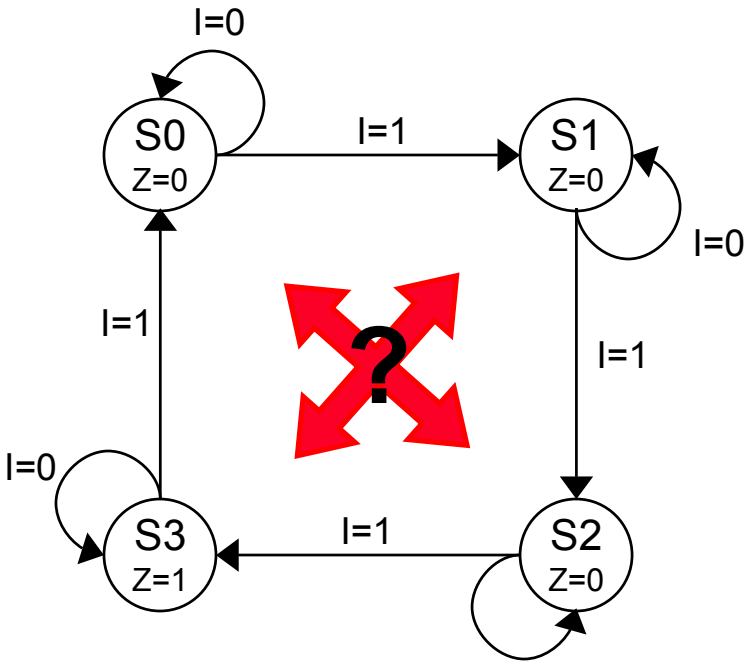
Register med ”håll”-funktion

- Registret kan ställas i två lägen
  - Skriv in nytt värde i registret (hold=0)
  - Behåll det tidigare värdet (hold=1)



Initiering av minneselement

- Motivation
  - Då spänningen slås på är vippornas tillstånd okänt – de kan antingen vara 1 eller 0.
  - För en tillståndsmaskin innebär det att start-tillståndet är okänt



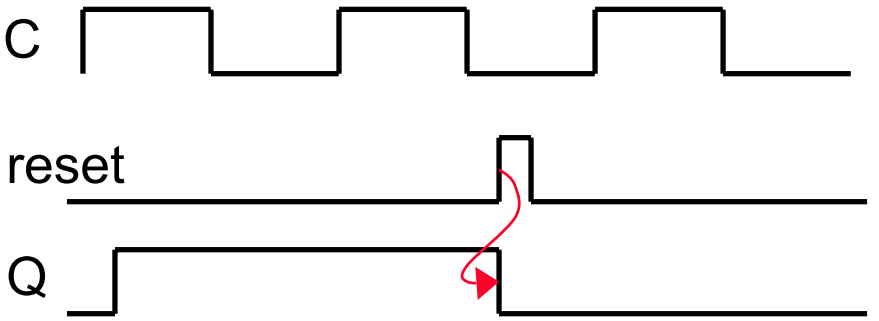
Tillståndskodning

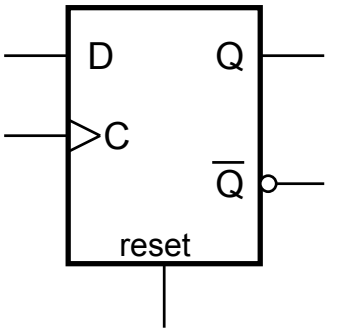
Tillstånd (S)	Binär (Q)
S0	00
S1	01
S2	10
S3	11
	q_1q_0

Vid uppstart kan Vipporna ha vilket tillstånd som helst

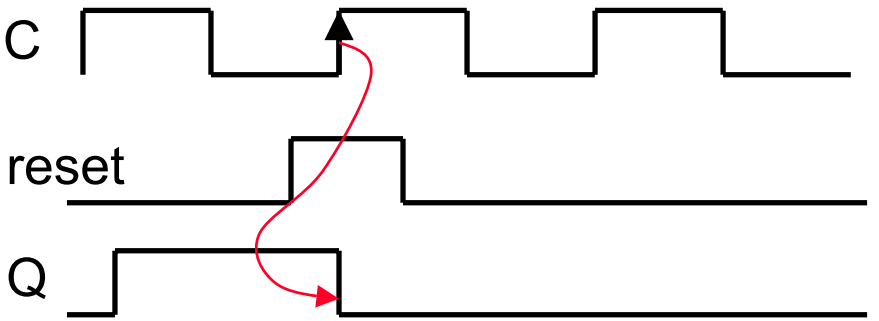
D-vippa med reset-signal

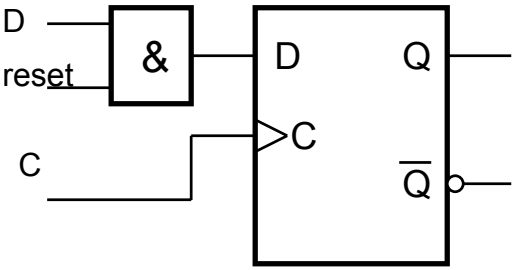
- Asynkron reset
  - Nollställer vippans innehåll oberoende av klockan



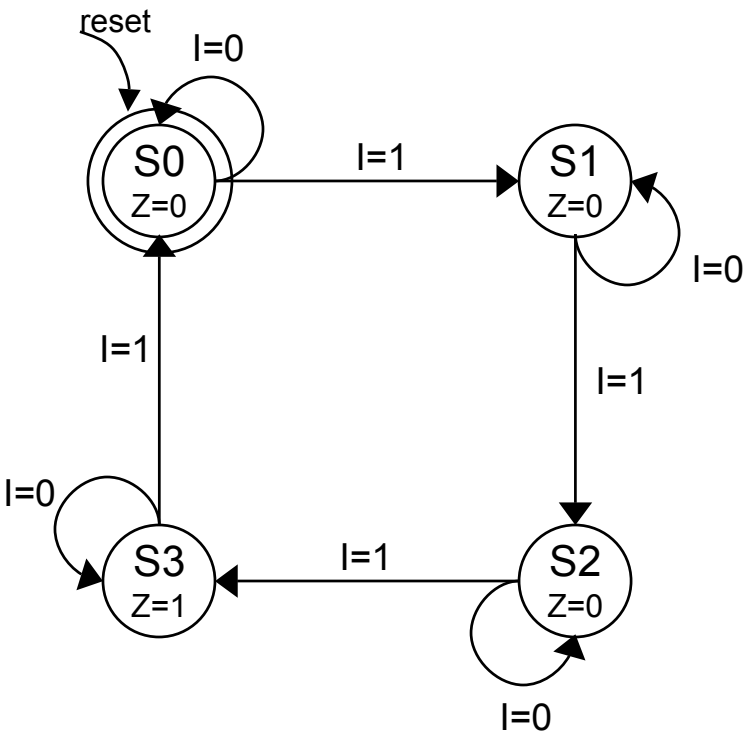


- Synkron reset
  - Nollställer vippans innehåll vid klockans aktiva flank





Initiering av en tillståndsmaskin



Tillståndskodning

Tillstånd (S)	Binär (Q)
S0	00
S1	01
S2	10
S3	11
	q_1q_0

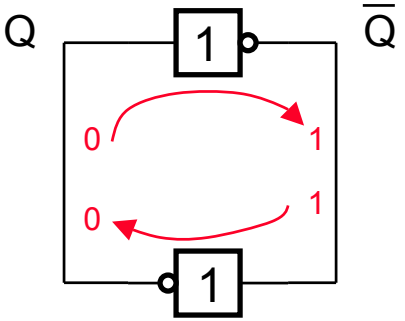
Vid uppstart tvinga maskinen till ett definierat tillstånd

Random-Access Memory (RAM)

- Generellt
  - Stora matriser av minnesceller som kan skrivas och läsas
  - Används t.ex i datorers internminne
    - Exekverande program
    - Data för snabb åtkomst

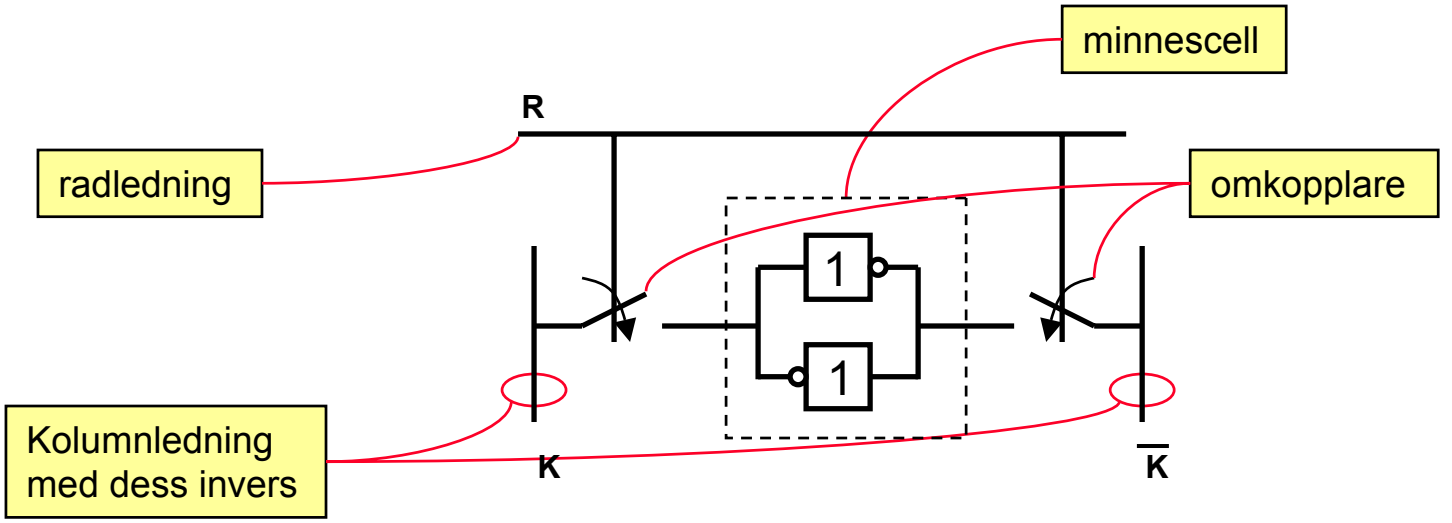
RAM minnescell

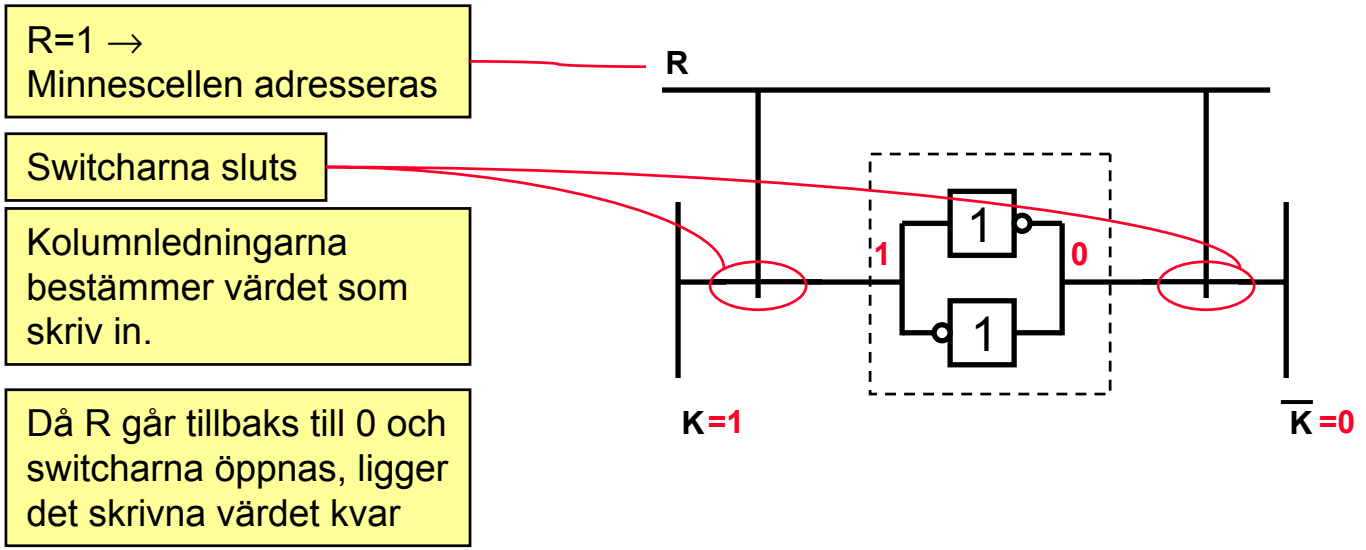
- Funktion
  - Håller 1 bits information statiskt, d.v.s data ligger kvar ända tills nytt data skrivs in
- Uppbyggnad
  - Konstrueras med två sammankopplade inverterare



Latch, jämför med nor2-latchen

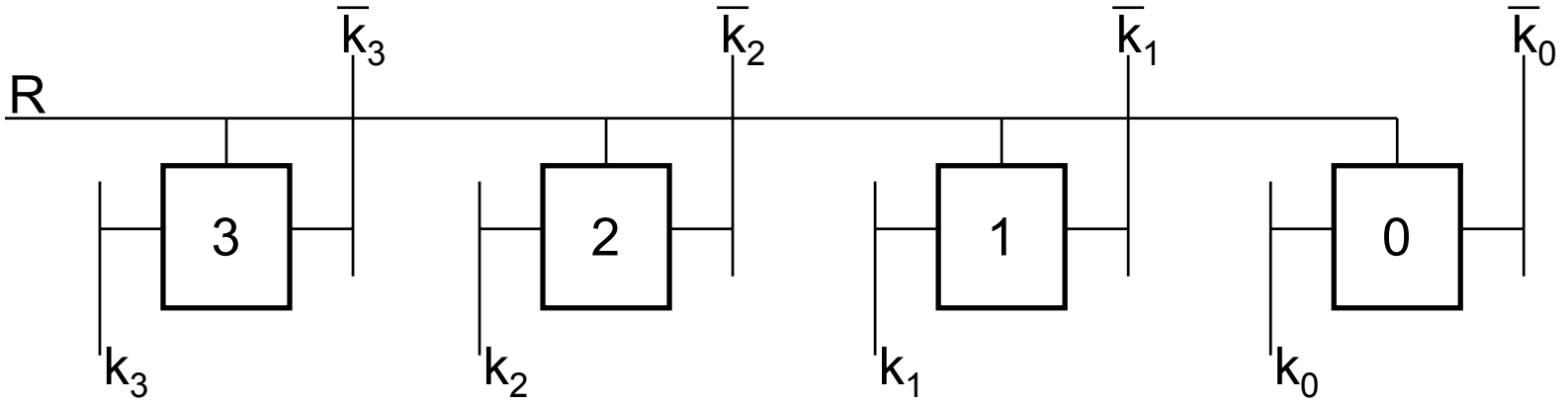
Skrivning i minnescell





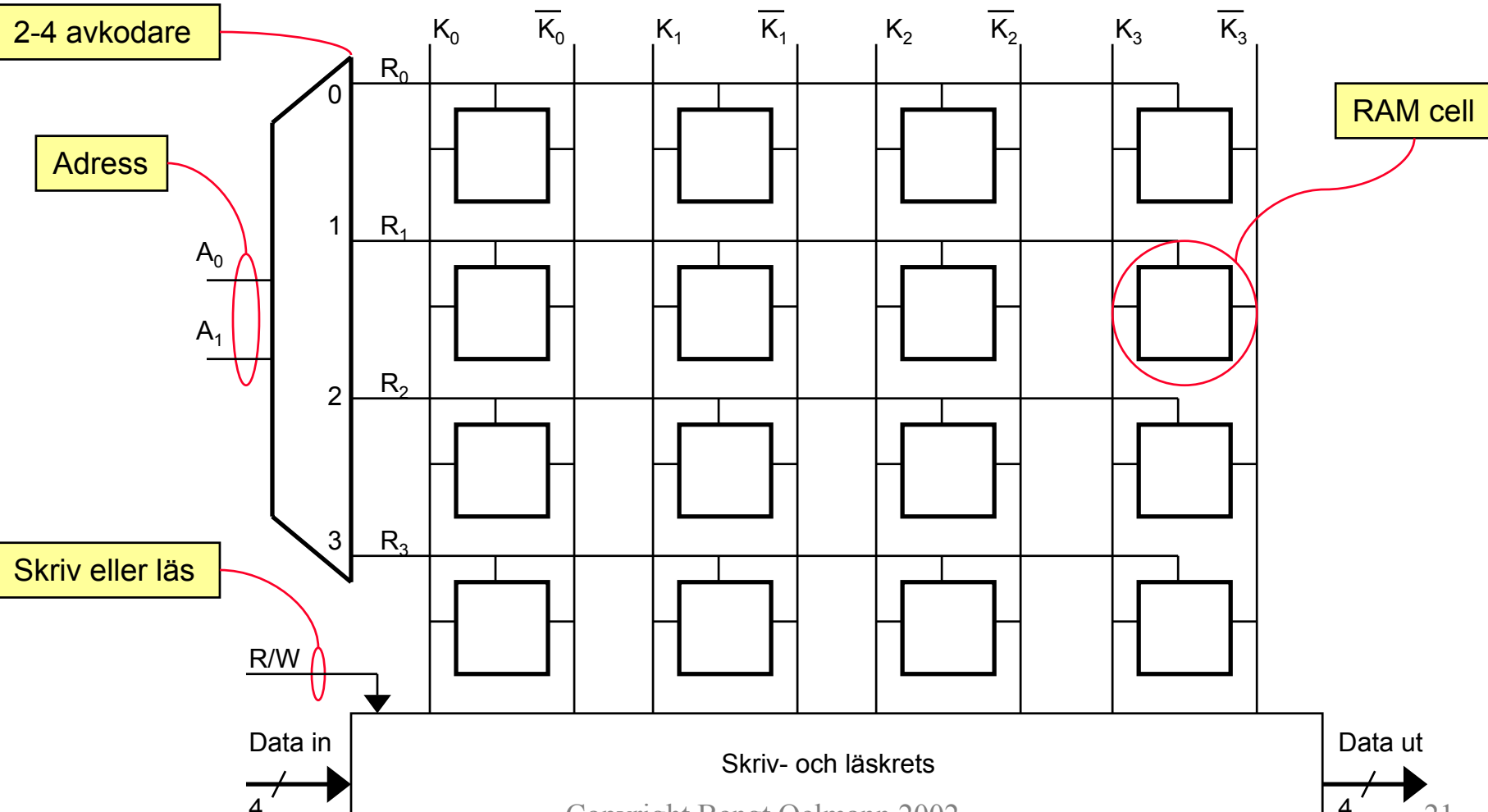
Matriser med RAM-celler

- Exempel
  - 1×4 bitars minne, d.v.s ett binärt tal som består av 4-bitar kan lagras
  - Funktion
    - R=0: minnescellerna isoleras från kolumnledningarna
    - R=1: samtliga minnesceller i ordet kan kommas åt antigen för läsning eller skrivning via $k_3 – k_0$

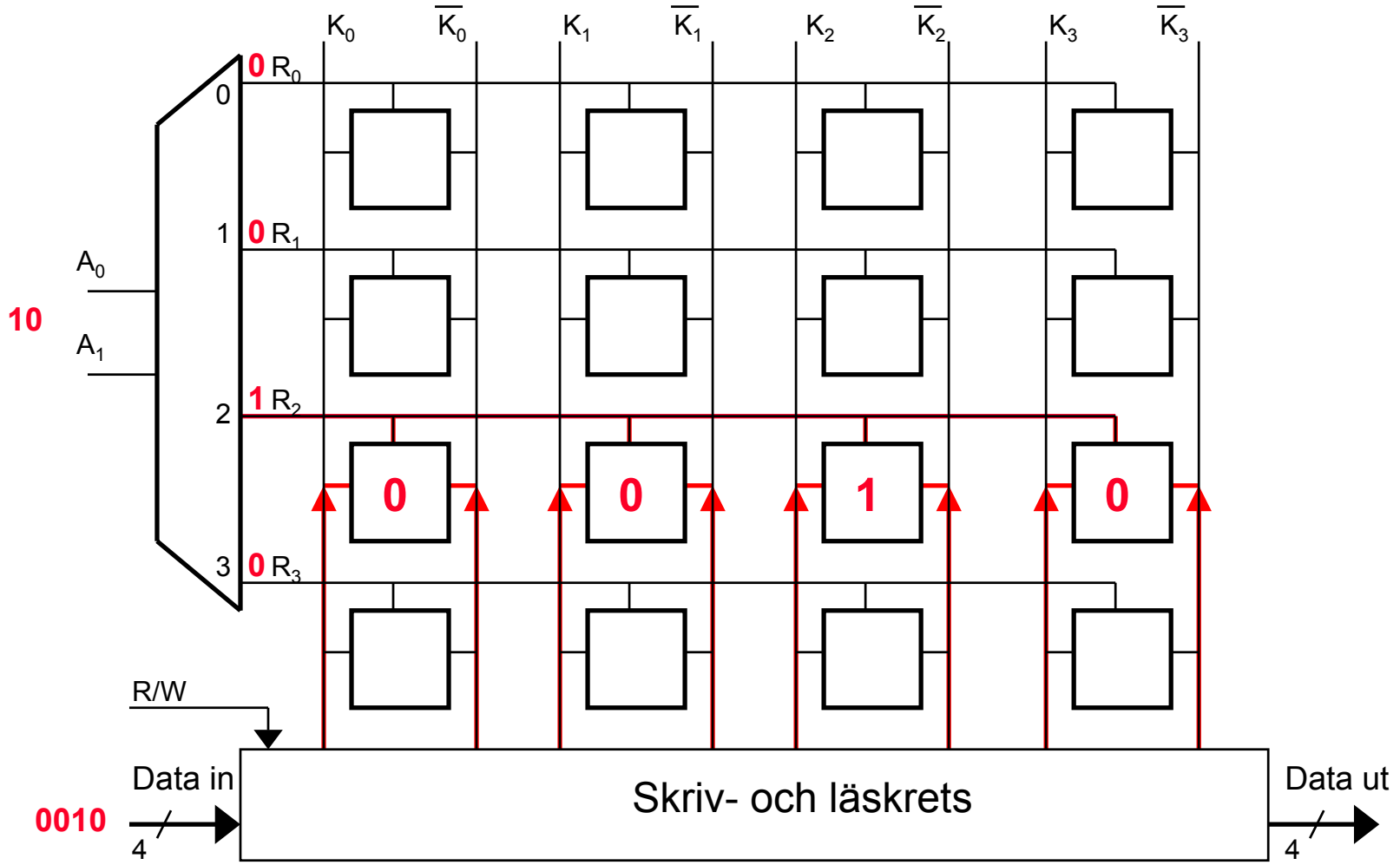


Matris med RAM-celler

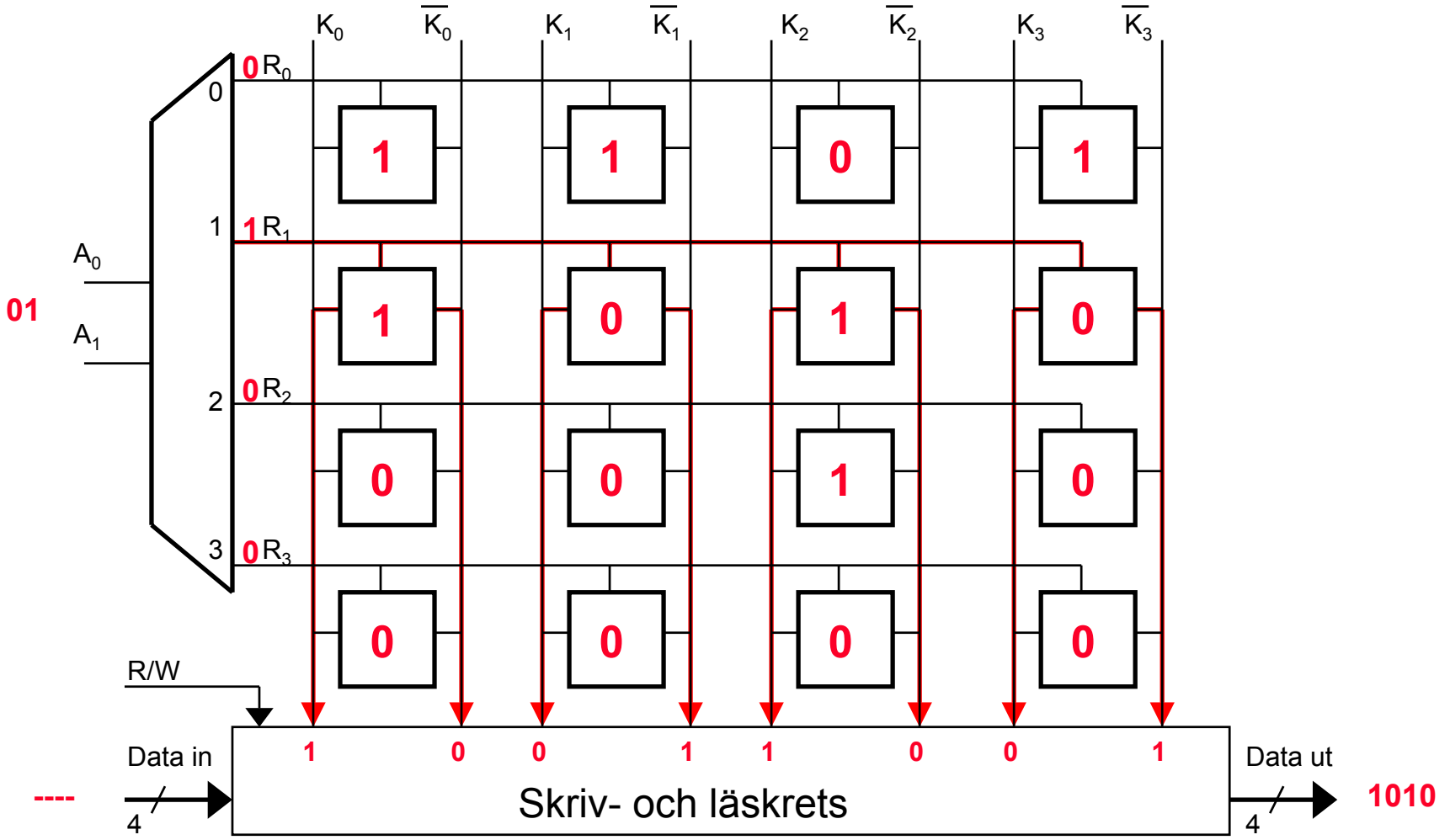
- Exempel
  - 4×4 bitars minne (4 binära tal á 4 bitar)



Skrivning i RAM



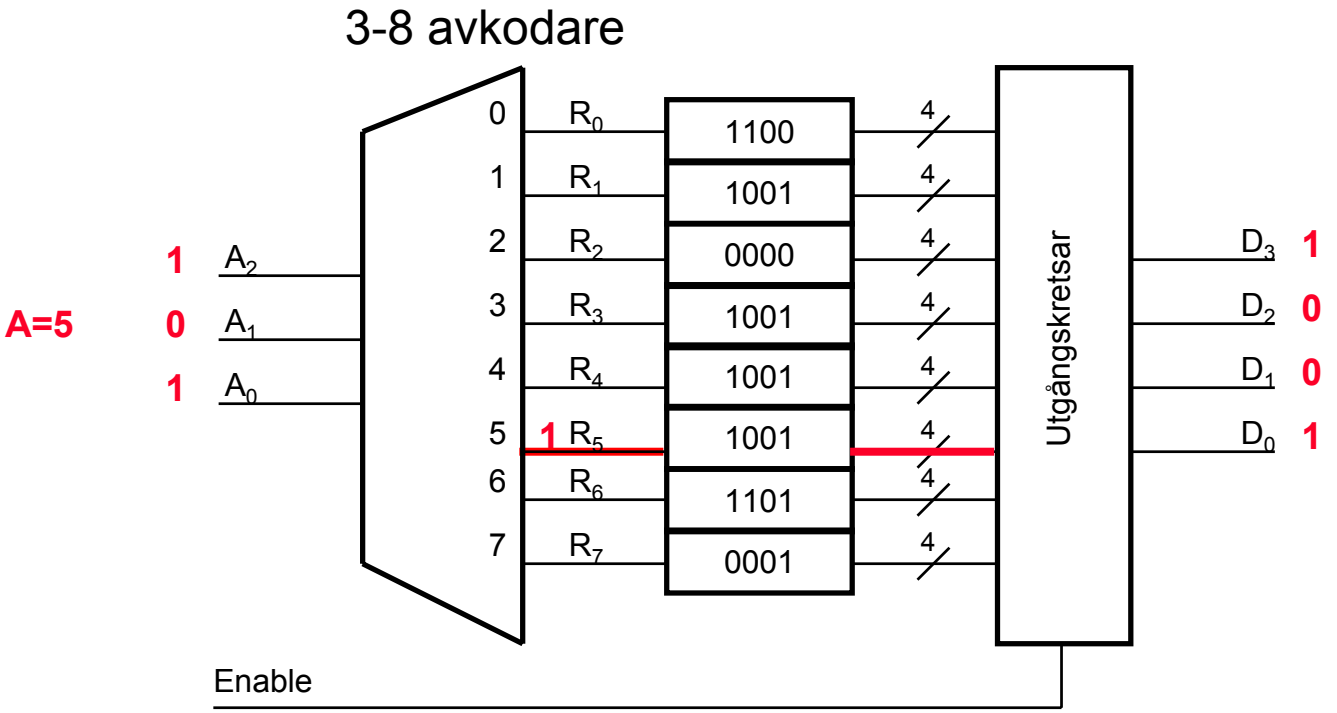
Läsning i RAM



Read-Only Memory (ROM)

- Generellt
  - Permanent minne
  - Innehållet bestäms vid tillverkning
  - Icke-flyktigt minne – data finns kvar om spänningen försvinner
  - Fungerar som en tabell

ROM – uppbyggnad och funktion



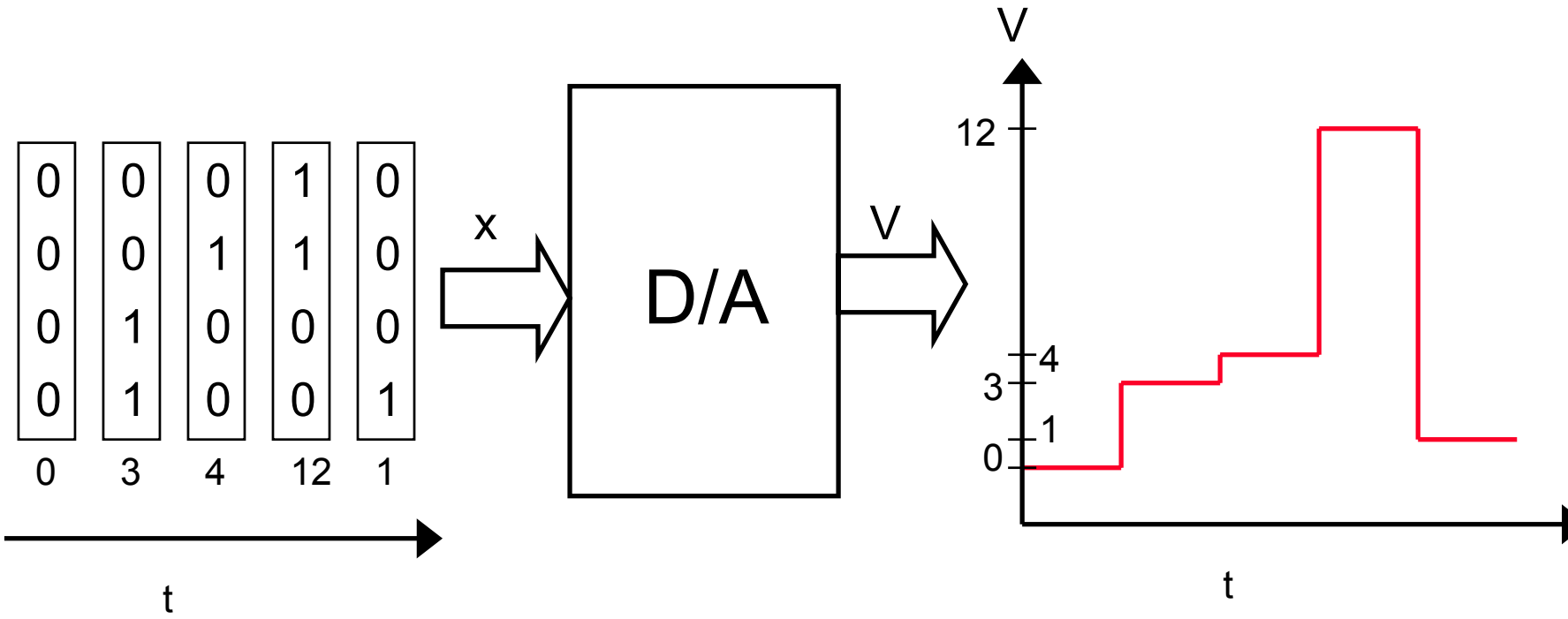
SLUT på Föreläsning 6.1

- Innehåll
  - Minneselement
    - Låskrets (eng latch)
    - Vippa (eng. Flip-flop)
    - Register
  - Random-Access Memory (RAM)
  - Read-Only Memory (ROM)

DA- och AD-omvandling

- Innehåll
  - Digital-till-analog omvandling
    - Spänningsdelare
    - Viktade resistorer
    - R-2R resistorstege
  - Analog-till-digital omvandling
    - Nivåramp
    - Successiv approximation
    - Flash

Digital till Analog omvandling



Digital till Analog omvandling

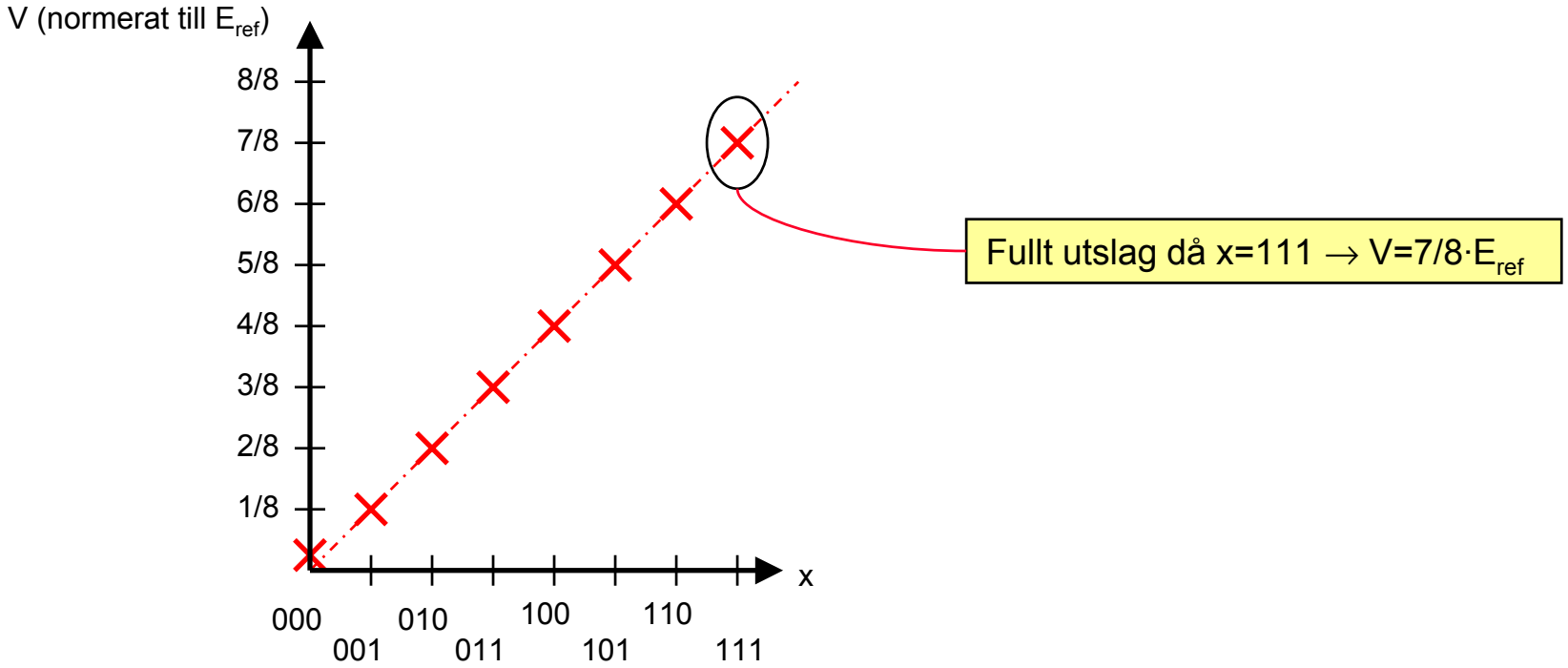
- Omvandling av det binära talet x till spänningen V.

$$x = 0.x_1x_2x_3 \quad \text{där } 0 \le x \le 7/8$$

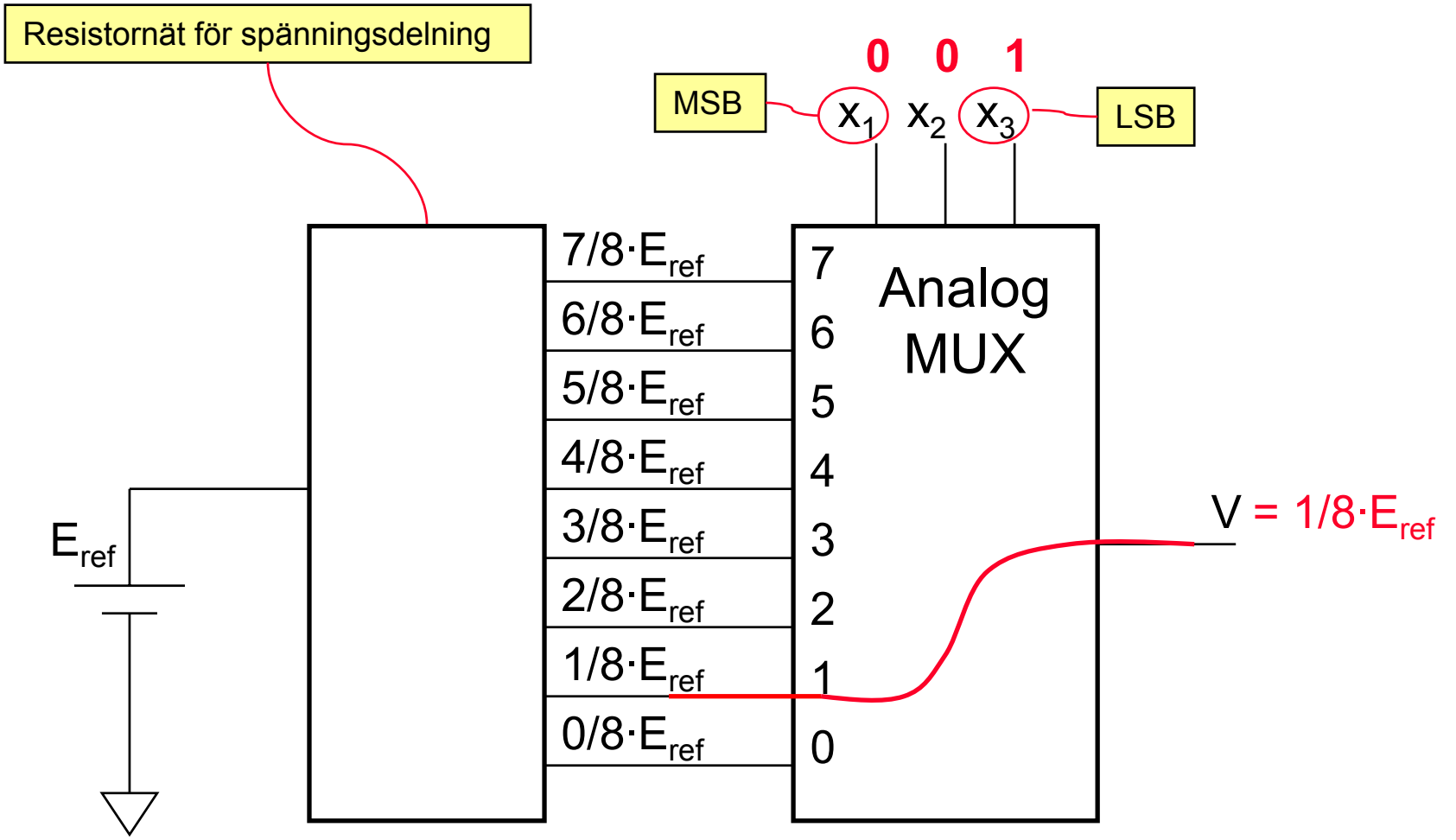
$$V = (x_1 \cdot 2^{-1} + x_2 \cdot 2^{-2} + x_3 \cdot 2^{-3}) \times E_{ref}$$

där E_{ref} är en referensspänning.

Referensspänningen är en skalfaktor

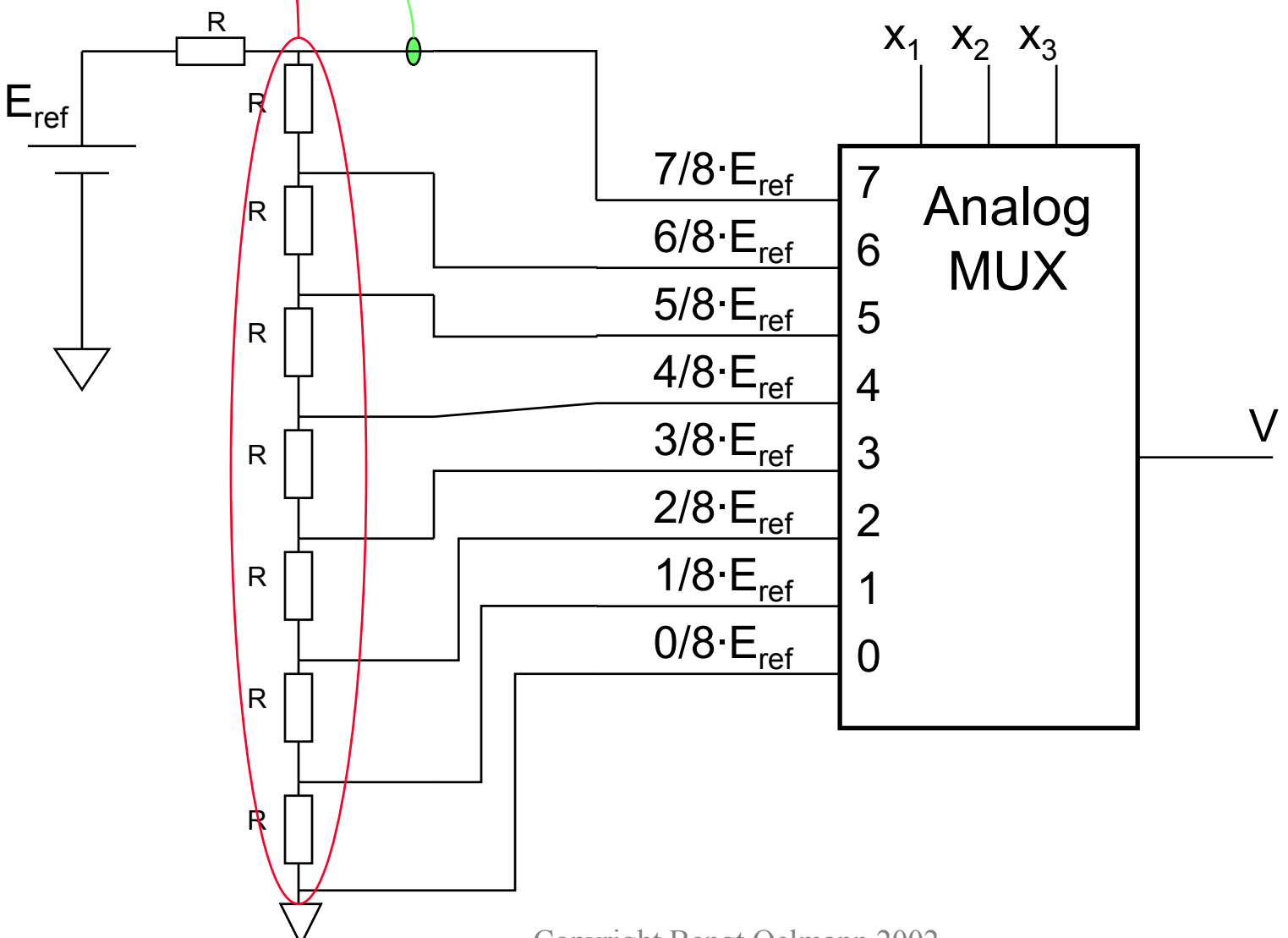


Princip för DA med spänningsdelning

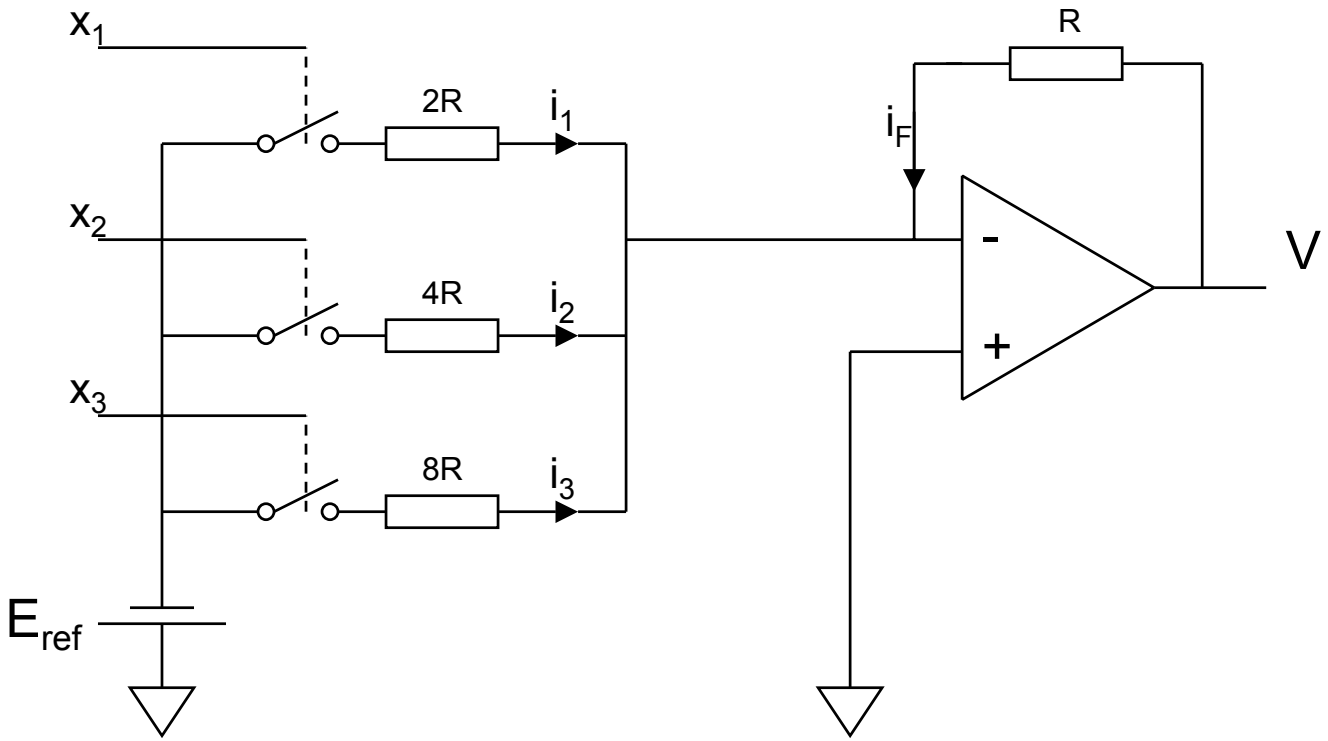


DA med spänningsdelning

$$V = \frac{7R}{8R} E_{ref}$$



DA med viktade resistorer

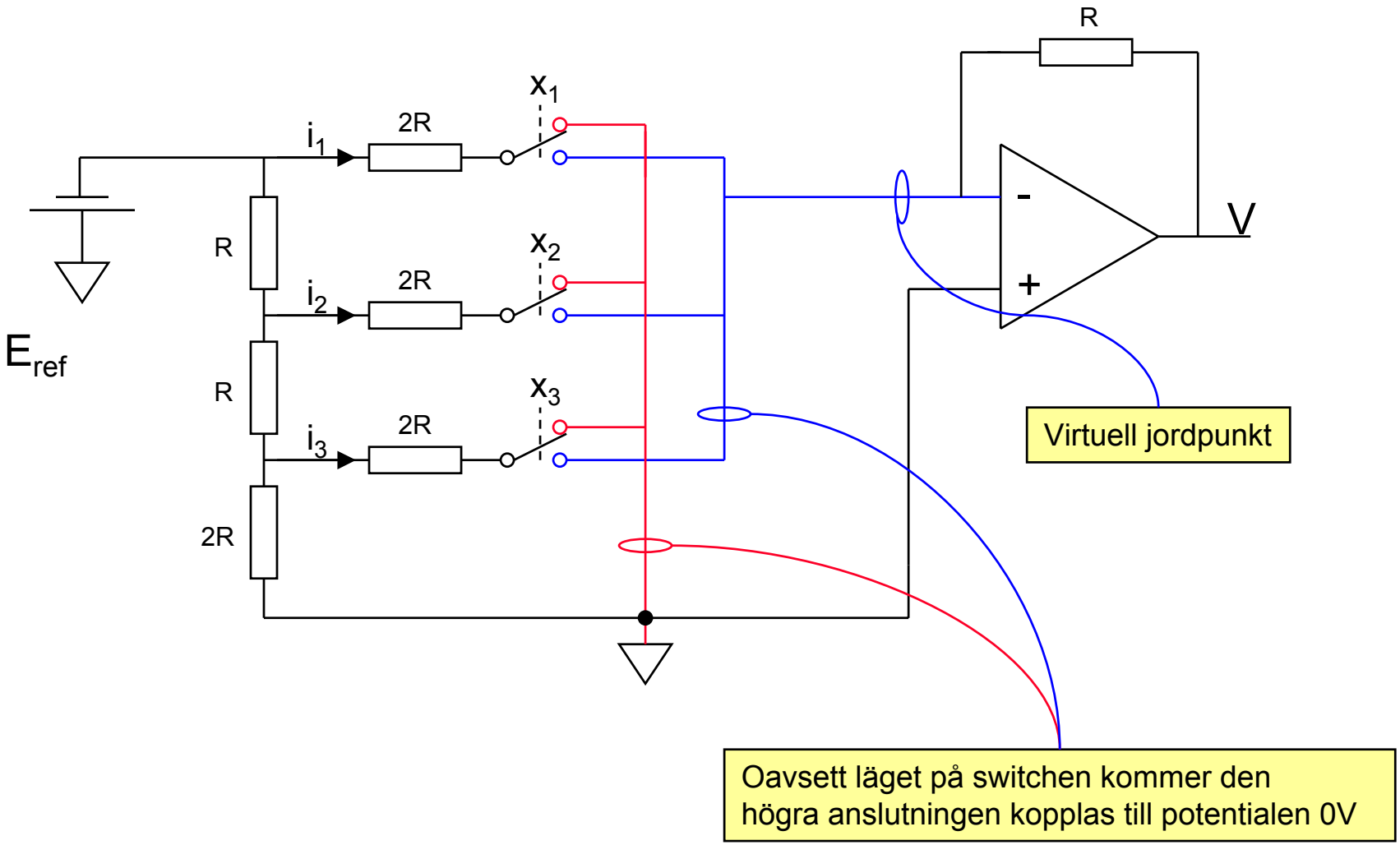


$$i_1 = 2 \cdot i_2 = 4 \cdot i_3$$

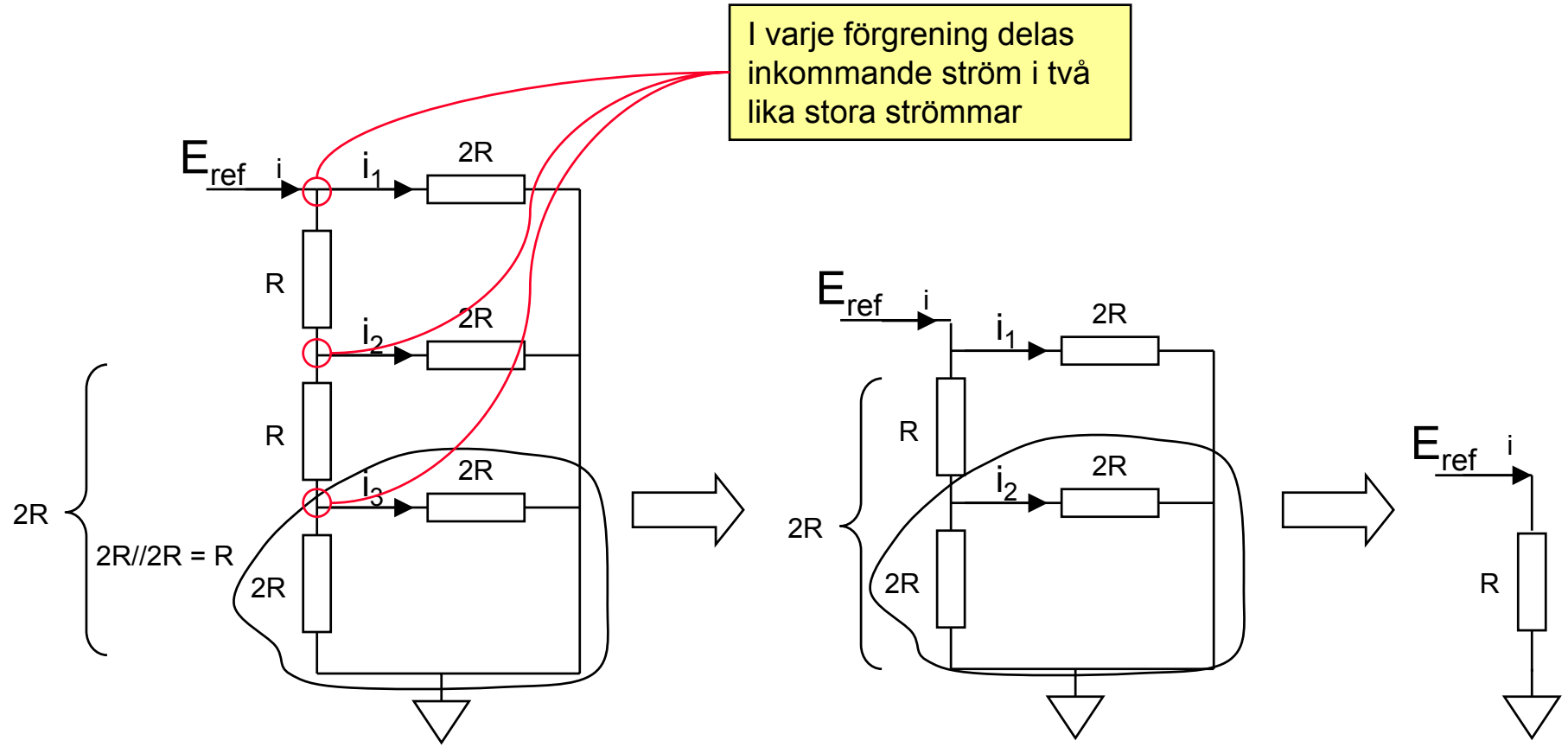
Viktade resistorer ger viktade strömmar som summeras vid OP:n

$$V = (x_1 \cdot 2^{-1} + x_2 \cdot 2^{-2} + x_3 \cdot 2^{-3}) \cdot E_{ref}$$

DA med R-2R resistorstege

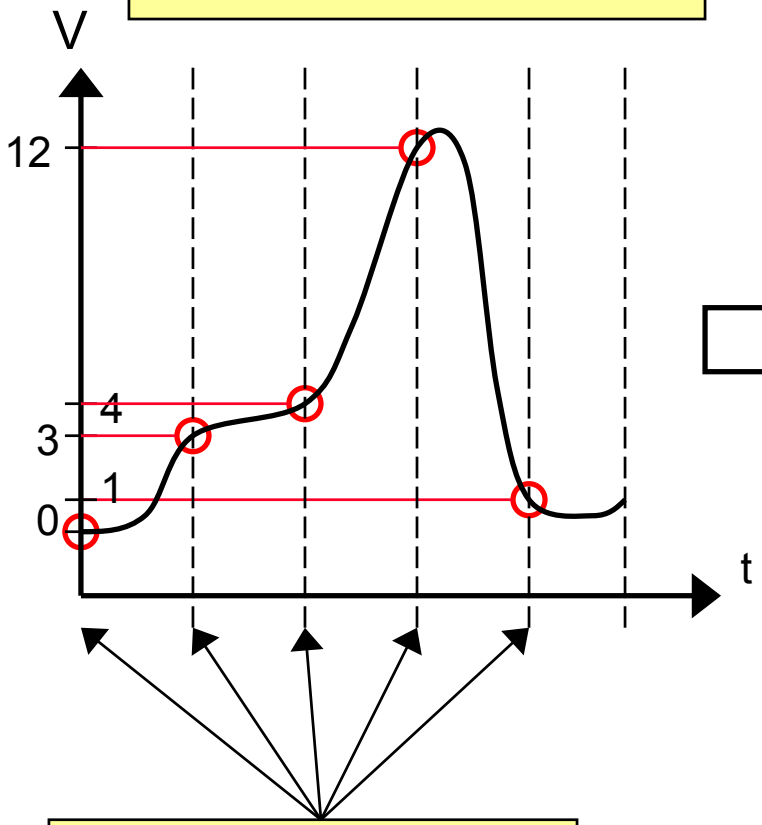


Strömmarna i en R-2R resistorstege

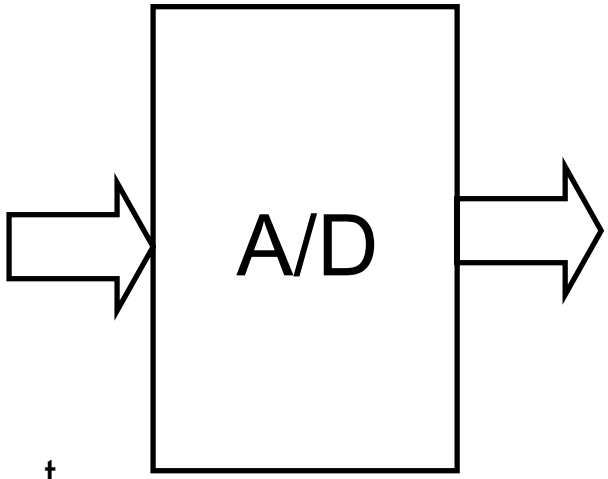


Analog till Digital omvandling

Kvantisering: det analoga värdet tilldelas ett siffervärde



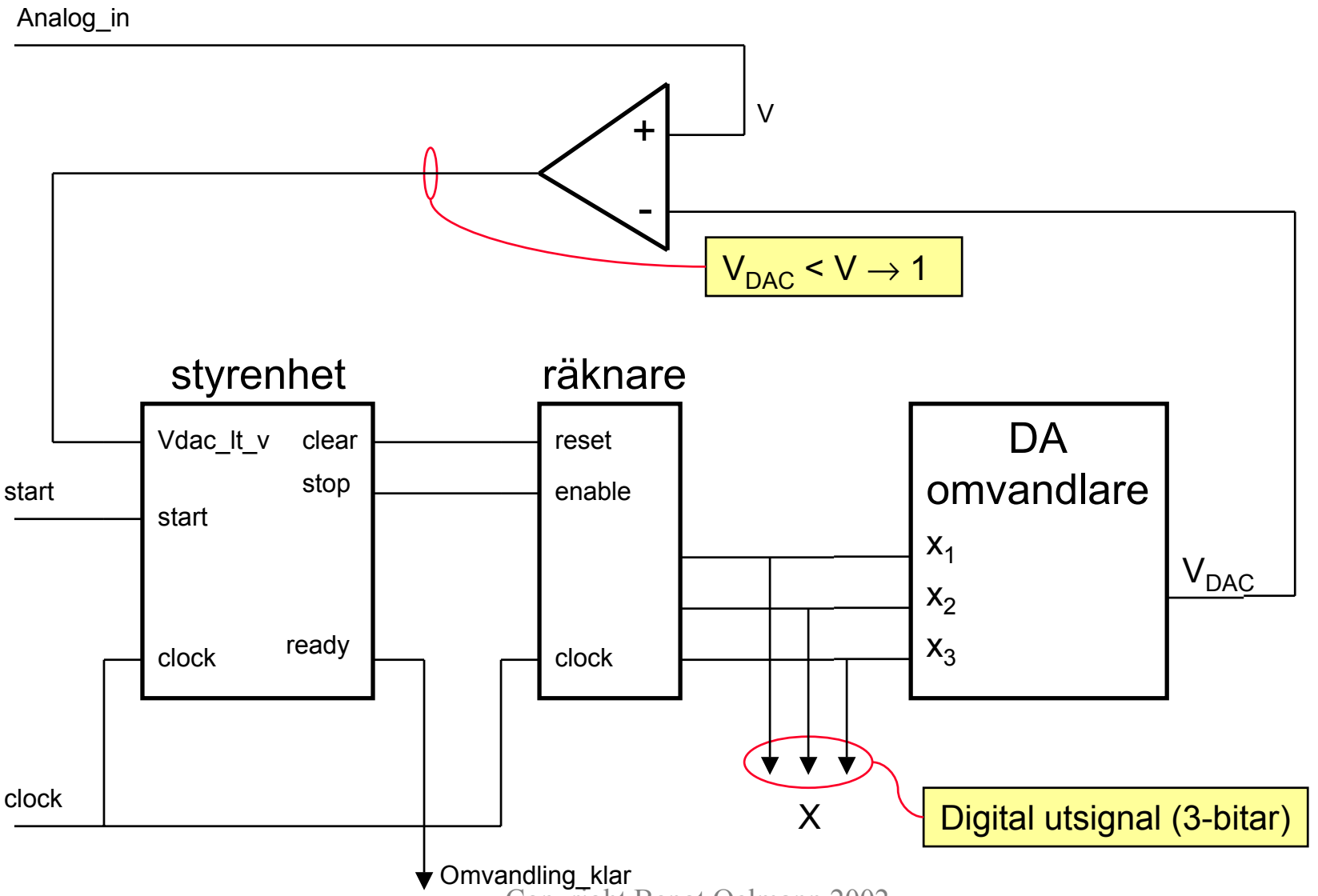
Sampling med konstant frekvens (samplingsfrekvens)



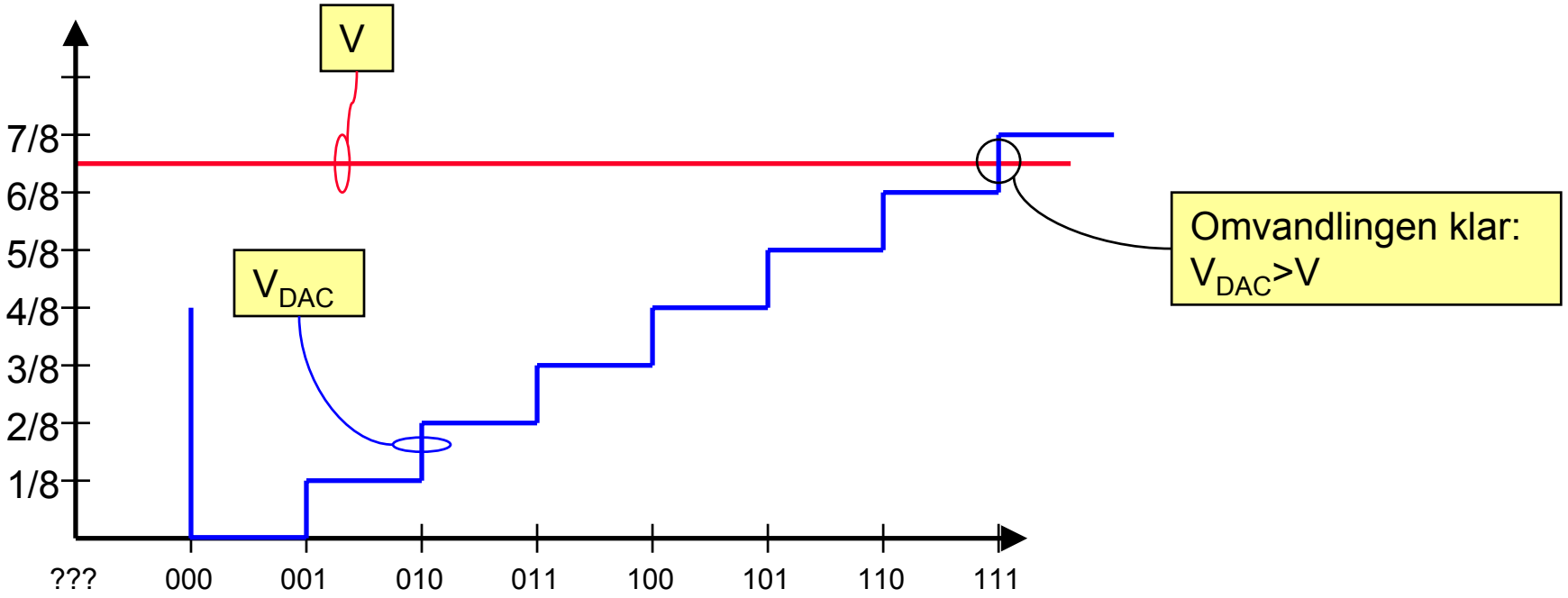
0	0	0	1	0
0	0	1	1	0
0	1	0	0	0
0	1	0	0	1
0	3	4	12	1

Resultatet är en serie tidsdiskreta värden som är kvantiserade

AD-omvandlare – nivåramp



Omvandlingstid – nivåramp

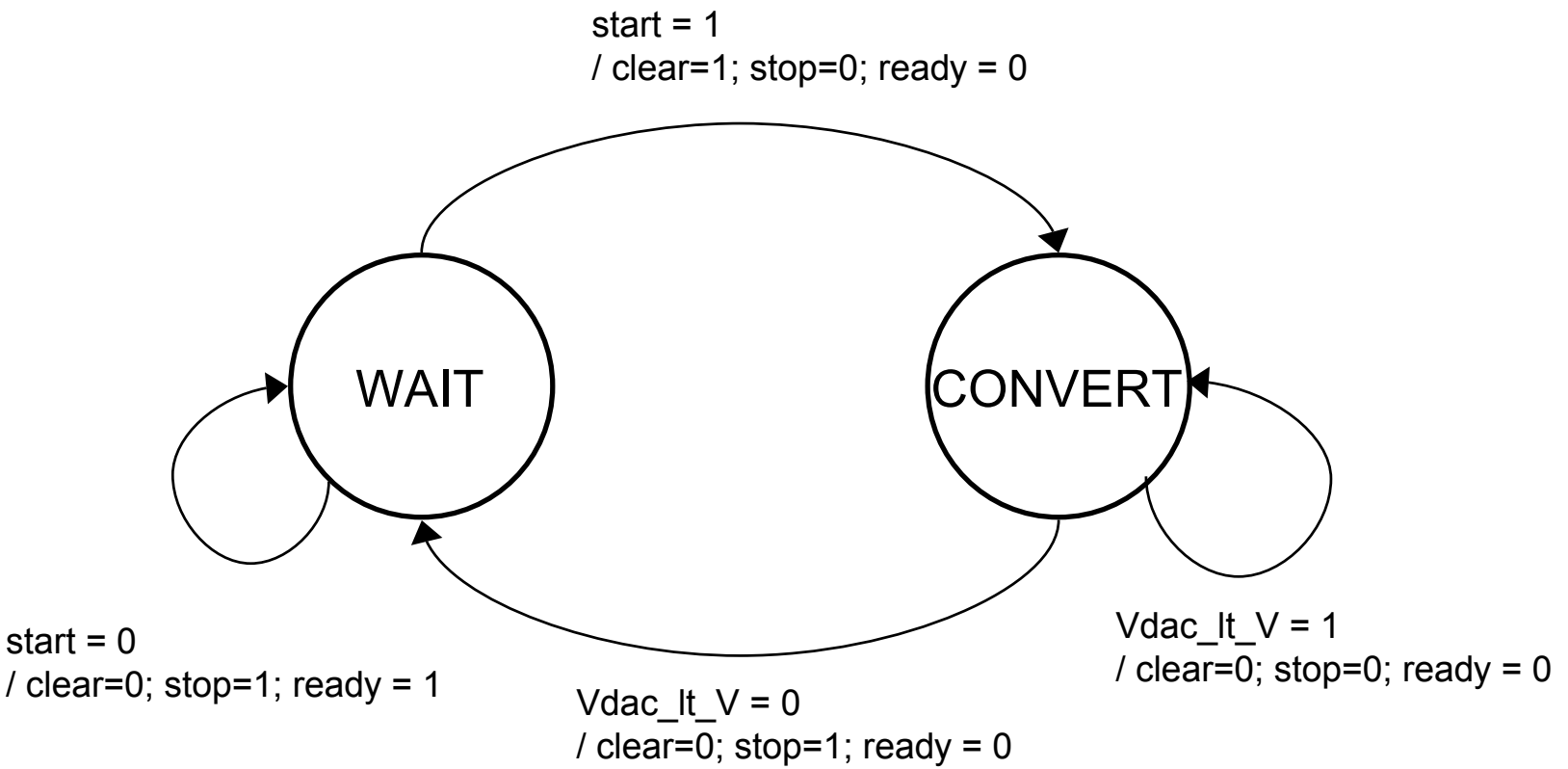


Omvandlingstiden varierar:
-tiden är kortare för omvandling av små spänningar
-Maximal omvandlingstid är:

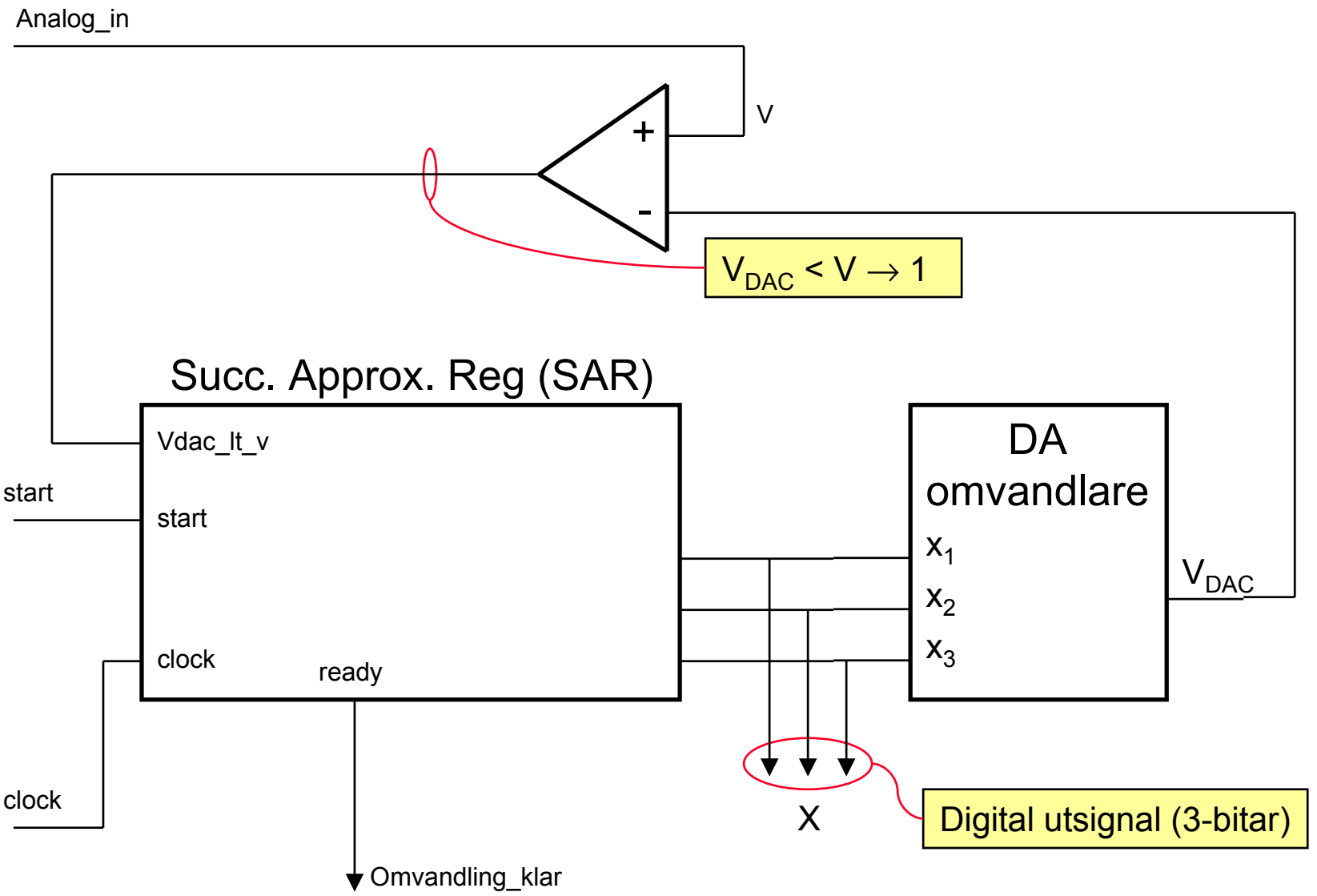
$$t_c = 2^n \cdot T_{clock}$$

Styrenhet – nivåramp

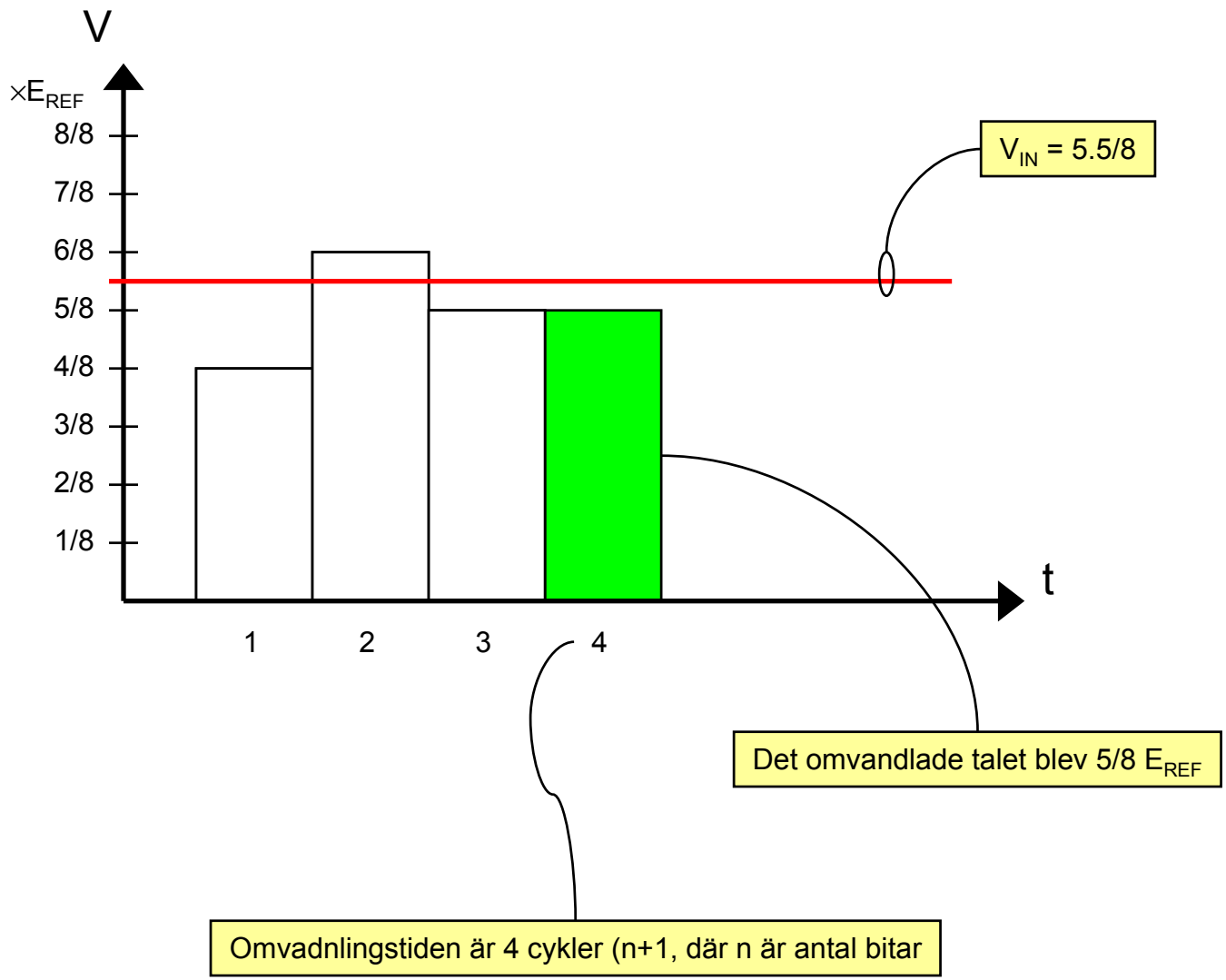
- Styrenheten är en tillståndsmaskin
- Tillståndsgraf



AD-omvandlare – successiv approximation

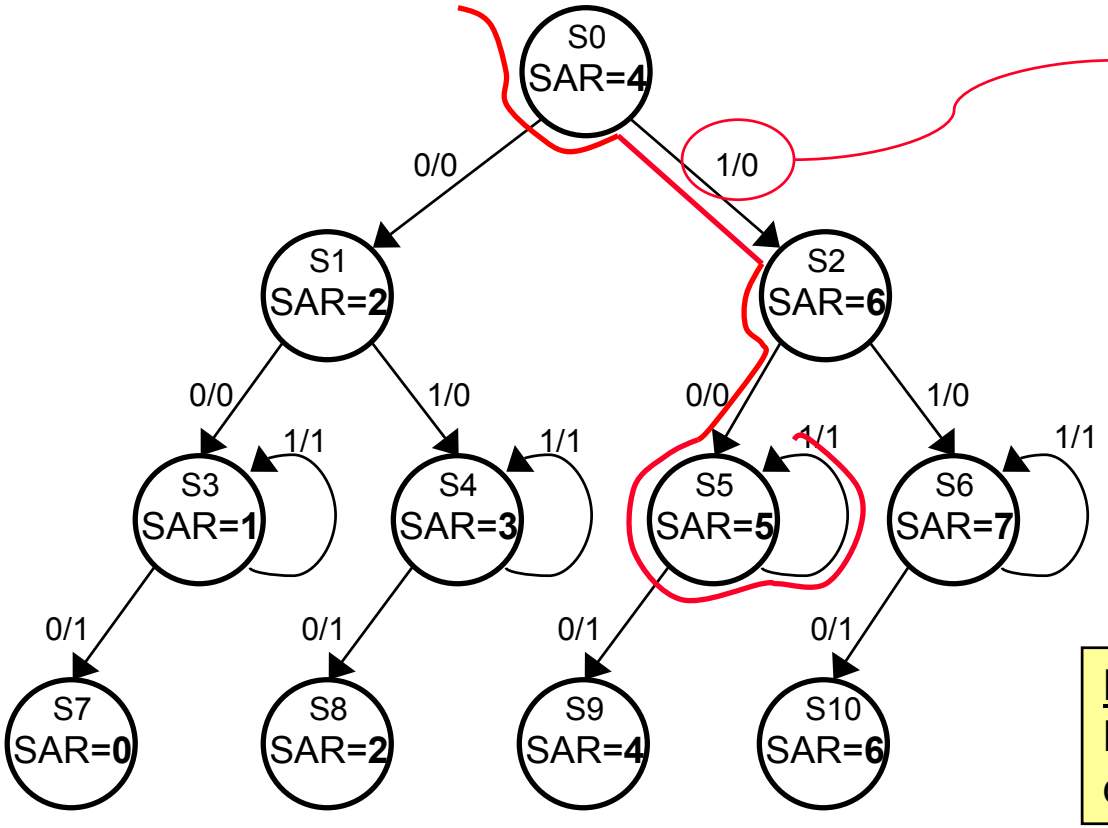


Omvandling med succ. approximation



Succ. Appr. Register (SAR)

- En tillståndsmaskin som tar fram ett digitalt tal enligt intervall-halveringsmetod

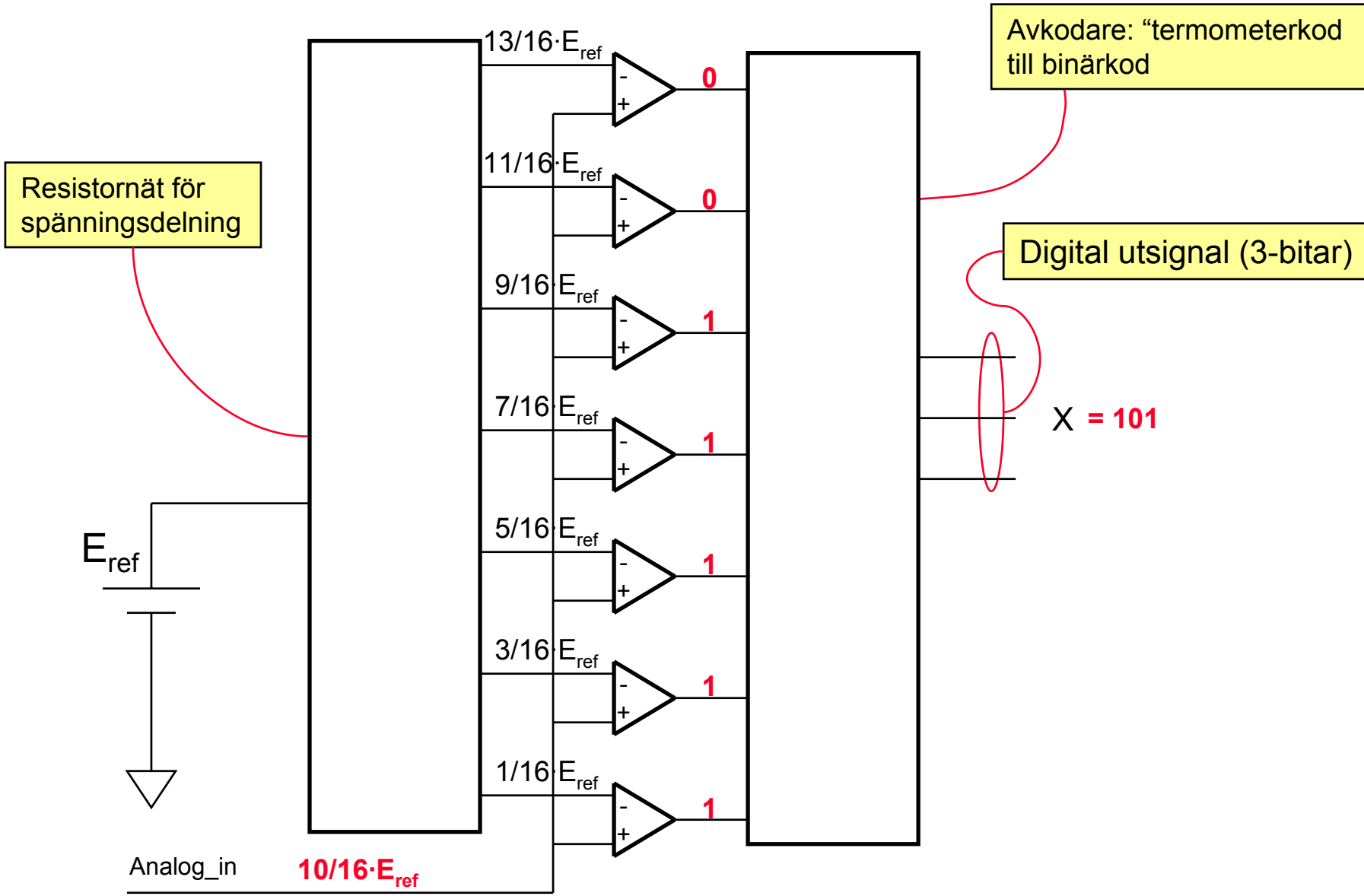


Format: Vdac_lt_V / ready

Om $V_{dac} < V$: Vdac_lt_V = 1
Annars: Vdac_lt_V = 0

Exempel: omvandling där
Det analoga värdet motsvarar det digitala talet 5.

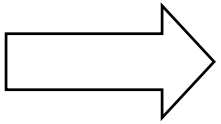
AD-omvandlare – Flash



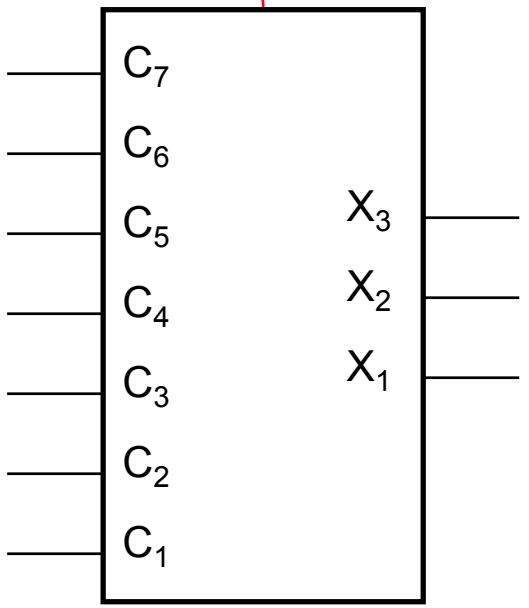
Avkodare – termometer till binärkod

"termometerkod"

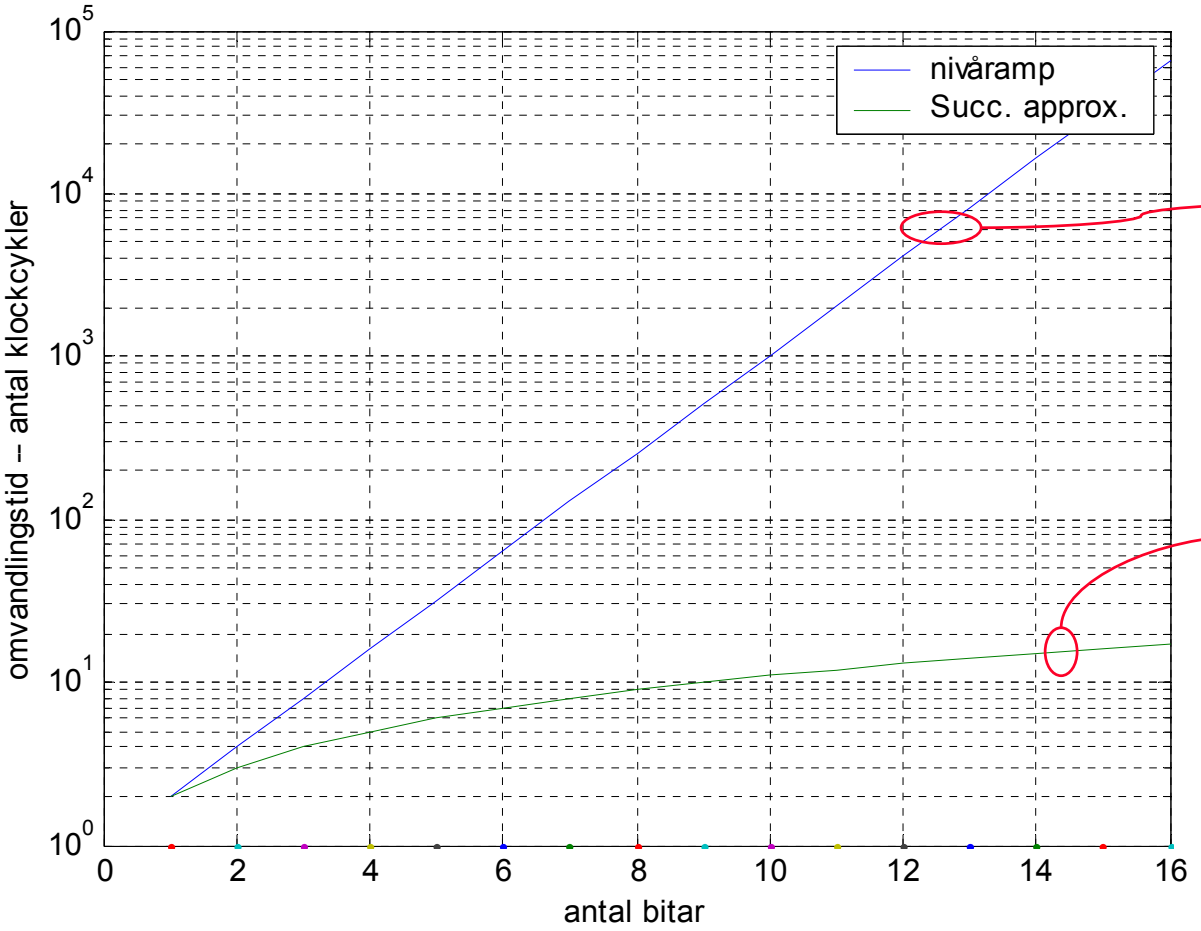
$C_7 - C_1$	X_1-X_3
0000000	000
0000001	001
0000011	010
0000111	011
0001111	100
0011111	101
0111111	110
1111111	111



Kombinatorisk logik



Jämförelse – Omvandlingstider



$$t_c = 2^n$$

$$t_c = n+1$$

$$t_{c-flash} = 1$$

Jämförelse – komplexitet

- Nivå-ramp och succ. approximation är av samma komplexitet

- Flash omvandlaren kräver n-1 komparatorer för en n-bitars omvandlare
  - Ex. En 10 bitars omvandlare kräver 1023 komparatorer

SLUT på Föreläsning 6.2

- Innehåll
  - Digital-till-analog omvandling
    - Spänningsdelare
    - Viktade resistorer
    - R-2R resistorstege
  - Analog-till-digital omvandling
    - Nivåramp
    - Successiv approximation
    - Flash