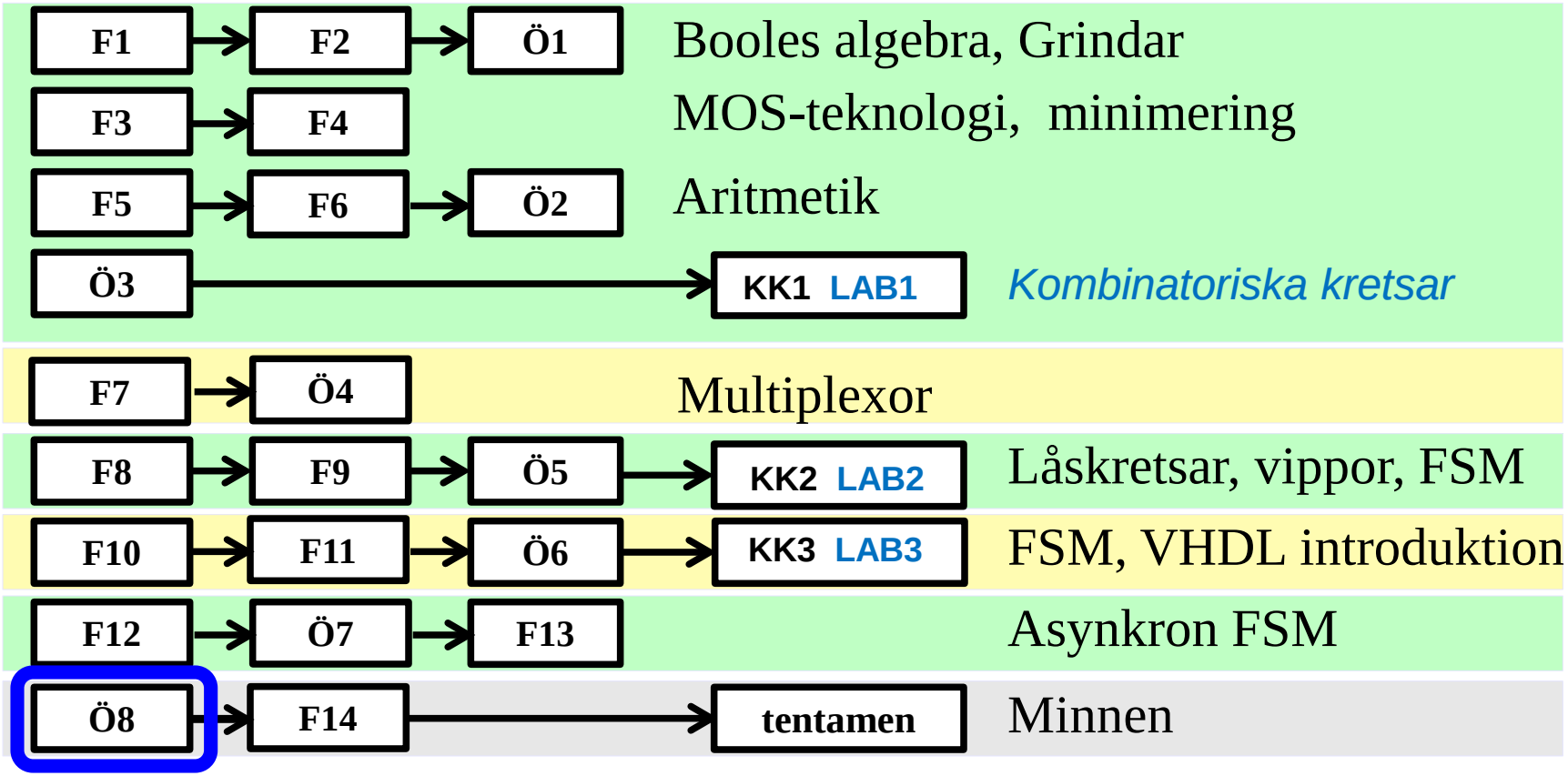
Avkodning av minnen (och I/O)

IE1204 Digital Design



Föreläsningar och övningar bygger på varandra! Ta alltid igen det Du missat!
Läs på i förväg – delta i undervisningen – arbeta igenom materialet efteråt!

William Sandqvist william@kth.se

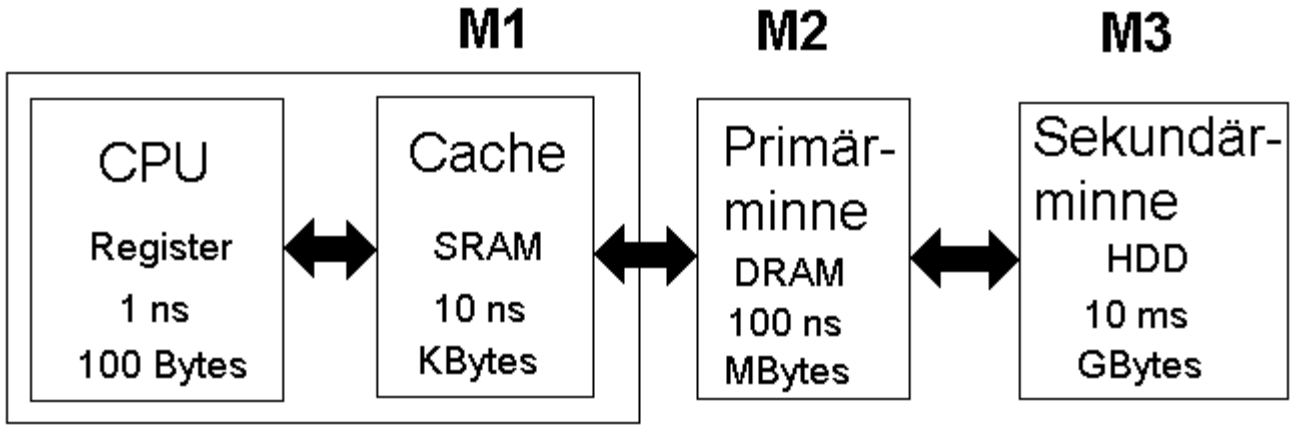
Minnesteknologier

Teknologi	Accesstid	Kostnad $/GB
SRAM	1 ns	1000
DRAM	50 ns	100
HDD	10 ms	1

Snabba minnen är dyra och billiga minnen är slöa!

Principiella siffror.

Minneshierarki



En tre nivåers minneshierarki. De snabbare minnestyperna används som ”buffertar” mot de långsammare.

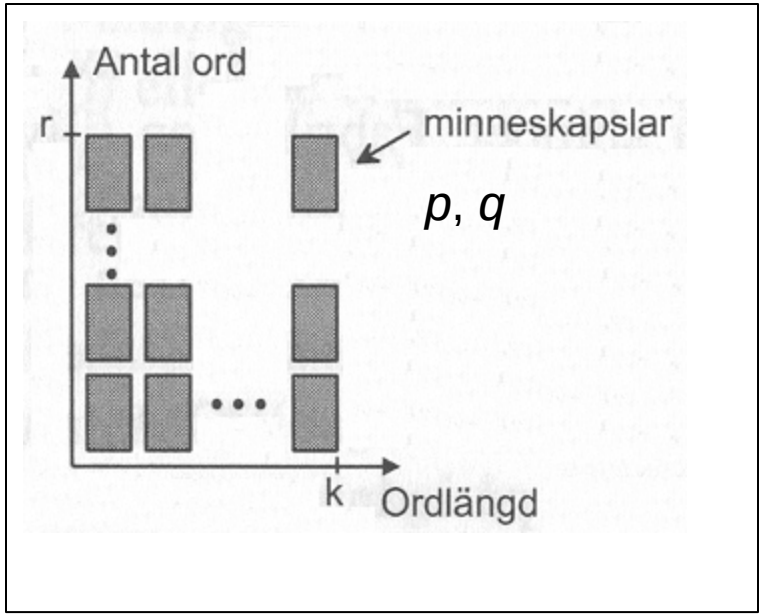
Principbild

Minne och minneskapslar

Minne:
N ord med ordlängden *M* bitar

Minneskapsel:
p ord med ordlängden *q* bitar

- Antalet kapselrader $r \leq N/p$
- Antalet kapselkolumner $k \geq M/q$
- Antalet kapslar $K = r \times k$



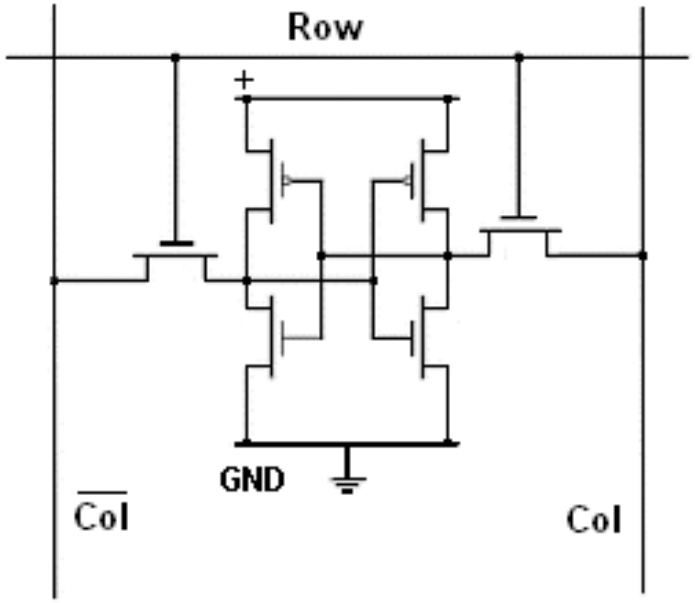
$$K = r \times k$$

N, *M*

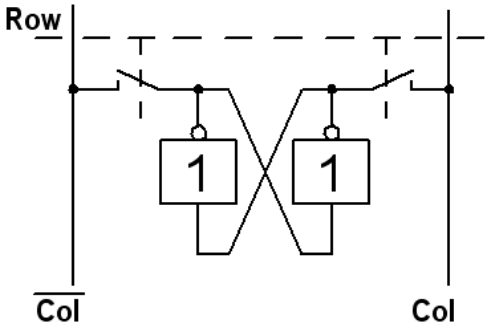
William Sandqvist william@kth.se

SRAM

Varje *bit* i ett CMOS SRAM består av en **låskrets** uppbyggd av *sex* MOS-transistorer.



Minnescellen är i princip en SR-latch.

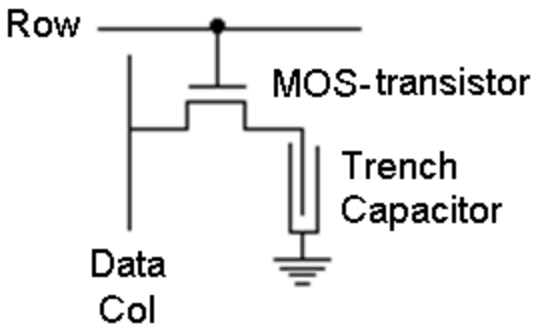


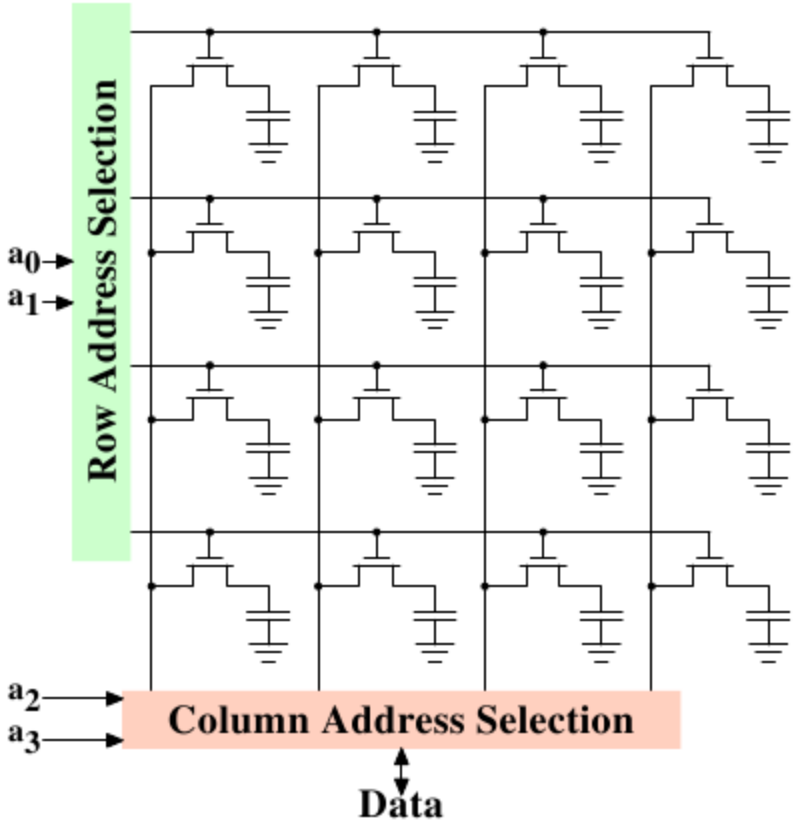
William Sandqvist william@kth.se

DRAM

Varje bit i ett DRAM består av *en* transistor och *en* minneskondensator.

En laddad kondensator läcker ut laddningen efter ett tag. Periodiskt måste alla kondensatorer undersökas och de som har laddning kvar måste då återladdas. Detta kallas för **Refresh**.
Det sköts av kretsar inuti minnet.

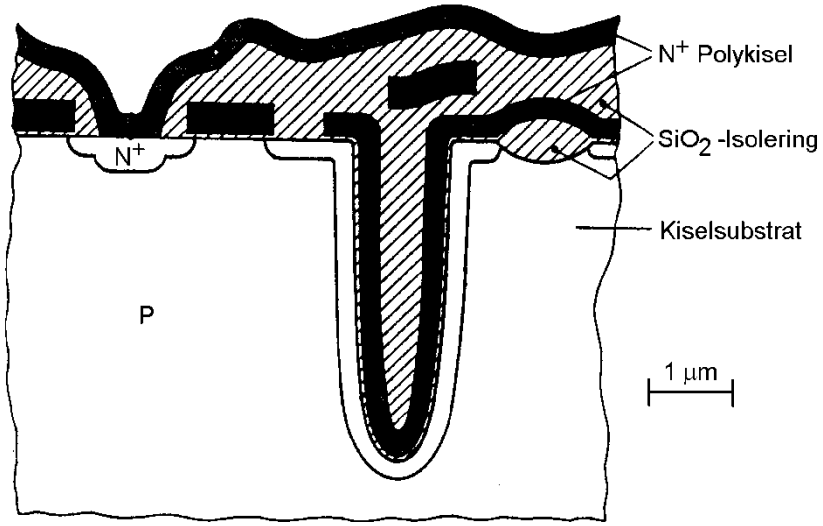


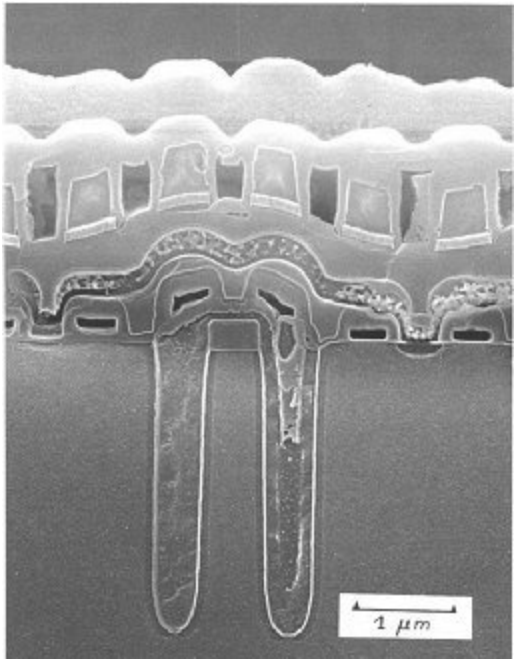


William Sandqvist william@kth.se

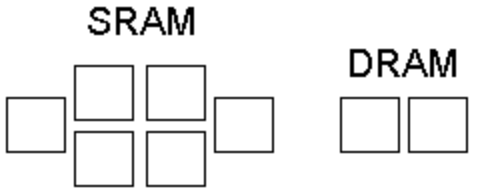
Kondensatorn byggs på djupet

Trench Capacitor (trench = dike)



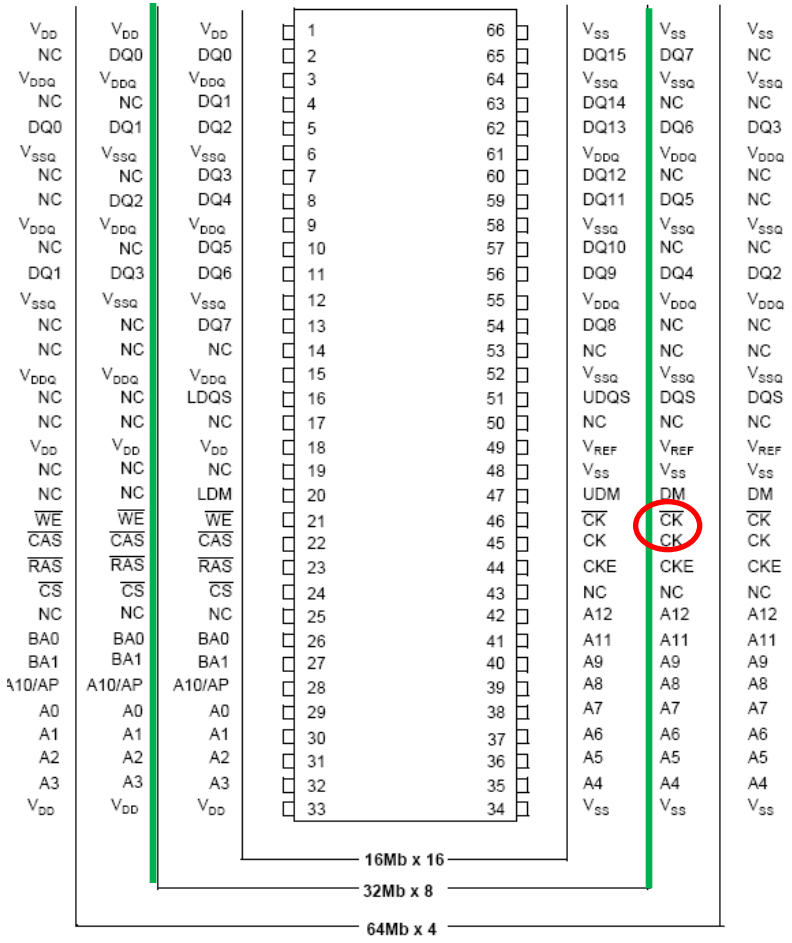


En bit i ett DRAM tar samma plats som *två* MOS-transistorer. En bit i SRAM som *sex* MOS-transistorer!



William Sandqvist william@kth.se

Infineon HYB25D25640 256 Mbit SDRAM



Kapsel 256Mbit (32M×8)

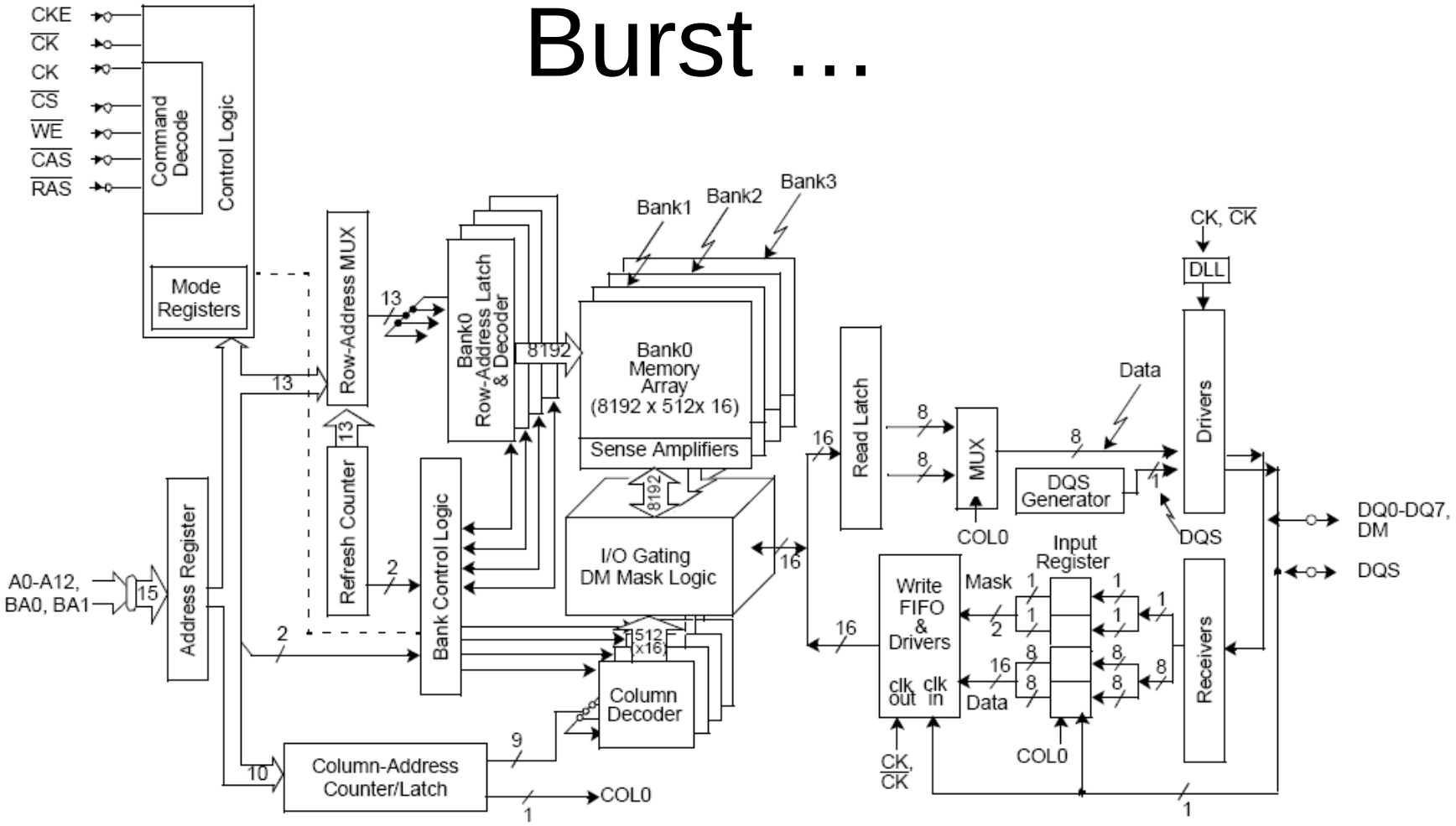
Synkront, använder bussklockan. Dubbel flanktriggat för dubbla datahastigheten ck+ck (även lägre effektförbrukning).

32M $2^5 \times 2^{20} = 2^{25}$, 25 adressbitar används. Tidsmultiplexad adressering, 13 bitar RAS (rader), 10 bitar CAS (kolumner), två bankbitar BA0 och BA1.

Burst kan vara 2, 4, 8 Byte i följd.

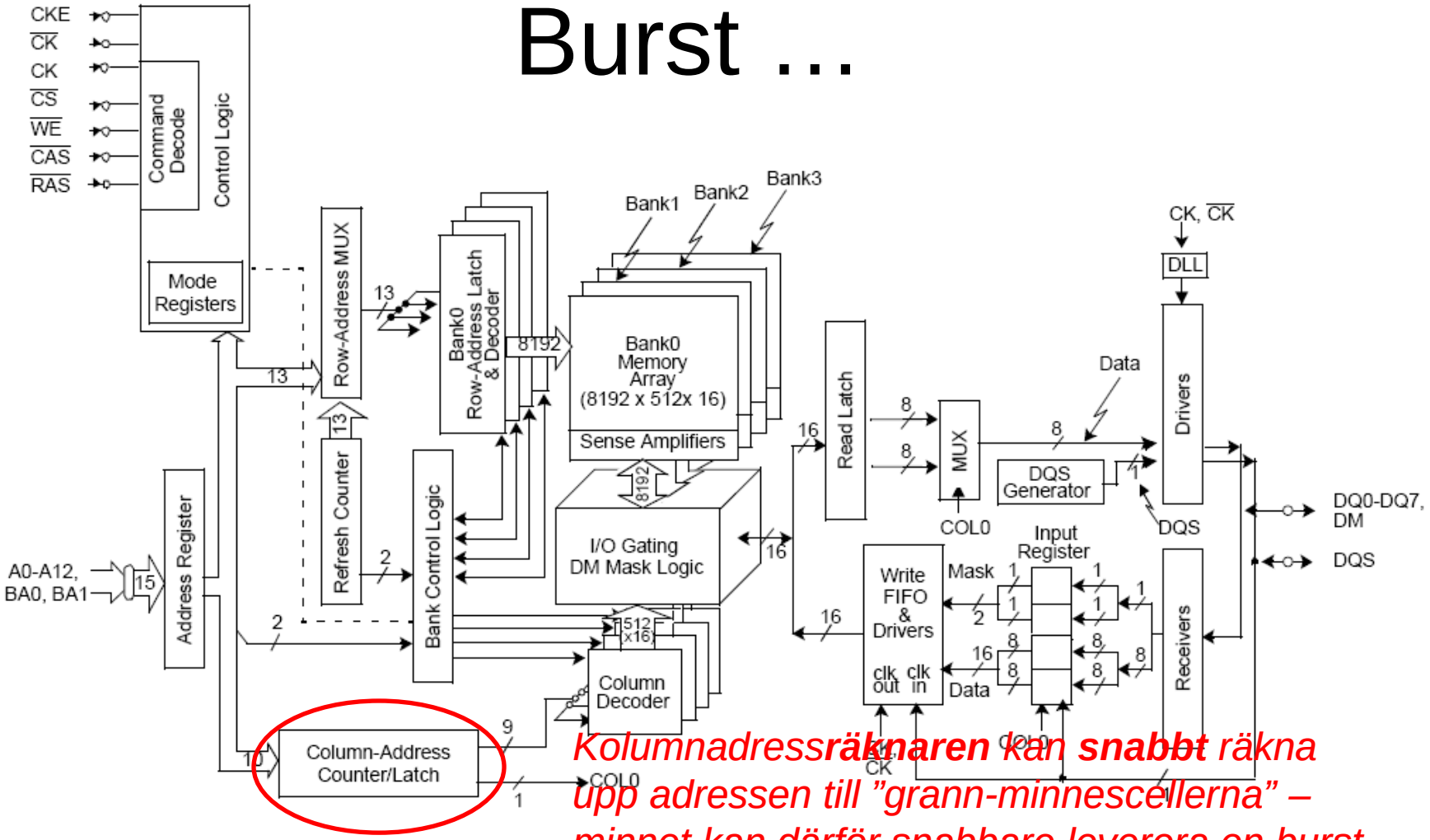
William Sandqvist william@kth.se

Burst ...



William Sandqvist william@kth.se

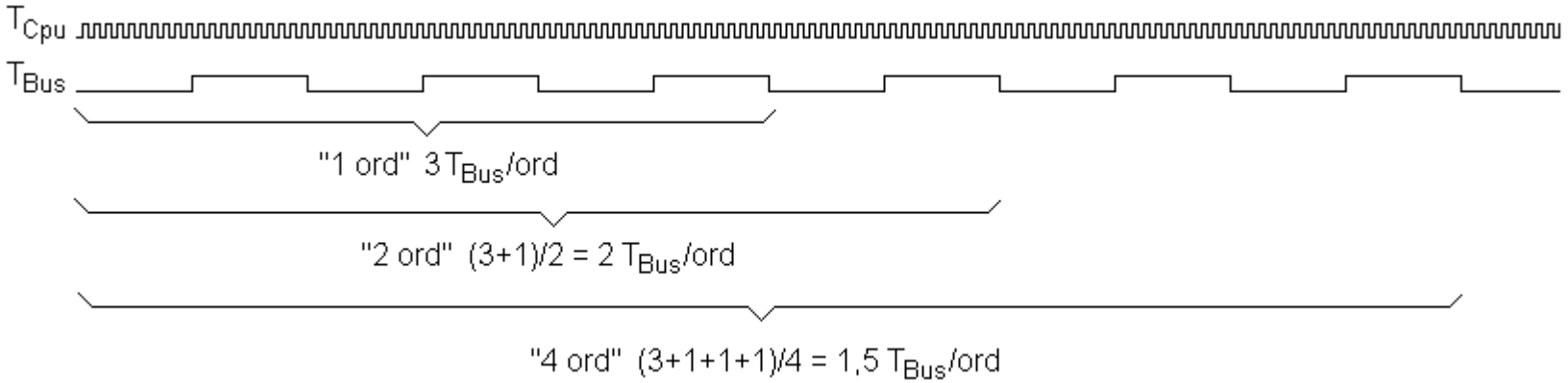
Burst …



*Kolumnadress**räknaren** kan **snabbt** räkna upp adressen till ”grann-minnescellerna” – minnet kan därför snabbare leverera en burst (skur) med flera Bytes i följd, än en enstaka random Byte.*

William Sandqvist william@kth.se

Burst ger snabbare medelaccess



- Att hämta **1** ”random” ord i minnet tar tre busscykler **3**T_{Bus}/ord
(varav 2 T_{BUS} är Waitstates)
- Att hämta en ”Burst” med **2** ord tar 3+1 busscykler, 4/2 = **2**T_{Bus}/ord
- Att hämta en ”Burst” med **4** ord tar 3+1+1+1 busscykler, 6/4 = **1,5**T_{Bus}/ord
- Att hämta en ”Burst” med **8** ord tar 3+1+1+1+1+1+1+1 busscykler, 10/8 = **1,25**T_{Bus}/ord

Det gäller dock att ha användning för alla hämtade ord – annars slösar man bussklockcykler med Burst-metoden!

Mer om detta i **Datorteknikkursen** i samband med cacheminnen.

William Sandqvist william@kth.se

ÖH **12.1** Dynamiskt minne

Kapsel
256Mbit (32M×8)

a) Hur många kapslar krävs för 256M×64?

William Sandqvist william@kth.se

ÖH **12.1** Dynamiskt minne

Kapsel
256Mbit (32M×8)

a) Hur många kapslar krävs för 256M×64?

Minne N = 256M M = 64 bitar. **Kapsel** p = 32M q = 8 bitar.

Antal kolumner $k = M/q = 64/8 = 8$.

Antalet rader $r = N/p = 256M/32M = 8$.

Antal kapslar $K = r \times k = 8 \times 8 = 64$.

William Sandqvist william@kth.se

512M×72 ?

Kapsel
256Mbit (32M×8)

b) Hur många kapslar krävs för 512M×72?

William Sandqvist william@kth.se

512M×72 ?

Kapsel
256Mbit (32M×8)

b) Hur många kapslar krävs för 512M×72?

Minne N = 512M M = 72 bitar. **Kapsel** p = 32M q = 8 bitar.

Antal kolumner $k = M/q = 72/8 = 9$.

Antalet rader $r = N/p = 512M/32M = 16$.

Antal kapslar $K = r \times k = 9 \times 1\ 6 = 144$.

William Sandqvist william@kth.se

512M×72 ?

Kapsel
256Mbit (32M×8)

b) Hur många kapslar krävs för 512M×72?

Minne N = 512M M = 72 bitar. **Kapsel** p = 32M q = 8 bitar.

Antal kolumner $k = M/q = 72/8 = 9$.

Antalet rader $r = N/p = 512M/32M = 16$.

Antal kapslar $K = r \times k = 9 \times 1\ 6 = 144$.

Den ”ovanliga” bitbredden 72 (= 64 + 8). De 8 *extra* bitarna används för att korrigera enkelfel, och för att kunna upptäcka dubbelfel.

(På så sätt kan även kapslar med något litet fel användas eftersom felet kan korrigeras. De kapslarna skulle annars behöva kasseras).

William Sandqvist william@kth.se

512M×72 ?

Kapsel
256Mbit (32M×8)

b) Hur många kapslar krävs för 512M×72?

Minne N = 512M M = 72 bitar. **Kapsel** p = 32M q = 8 bitar.

Antal kolumner $k = M/q = 72/8 = 9$.

Antalet rader $r = N/p = 512M/32M = 16$.

Antal kapslar $K = r \times k = 9 \times 1\ 6 = 144$.

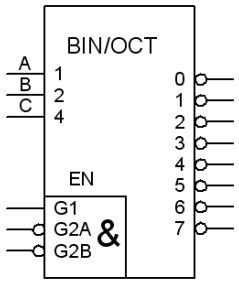
Den ”ovanliga” bitbredden 72 (= 64 + 8). De 8 *extra* bitarna används för att korrigera enkelfel, och för att kunna upptäcka dubbelfel.

(På så sätt kan även kapslar med något litet fel användas eftersom felet kan korrigeras. De kapslarna skulle annars behöva kasseras).

Eller så kommer ett bra minne att ”tåla” att några av minnescellerna ”slits ut” med tiden.

William Sandqvist william@kth.se

ÖH **12.2** ROM och SRAM



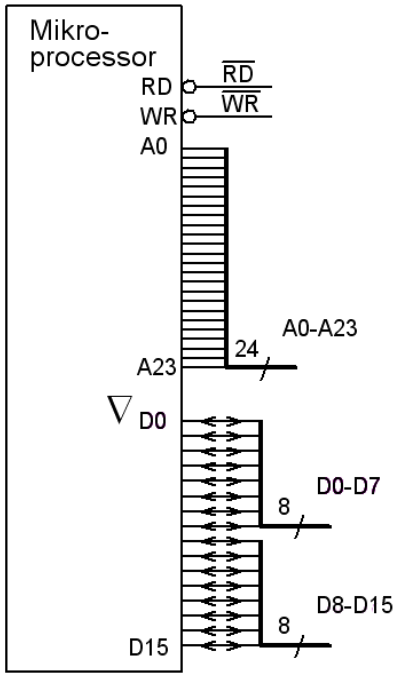
Läsminne:

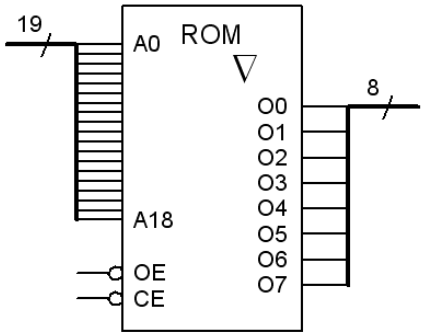
ROM 4M 512k × 8 bit

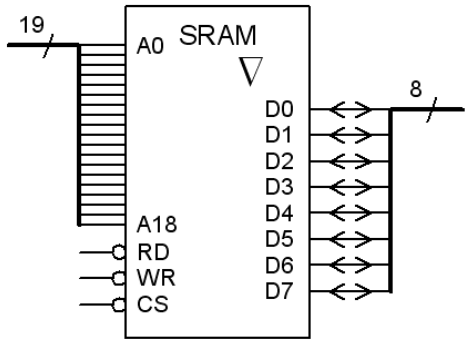
Läs/Skrivminne:

SRAM 4M 512k × 8 bit

Avkodare 3-to-8



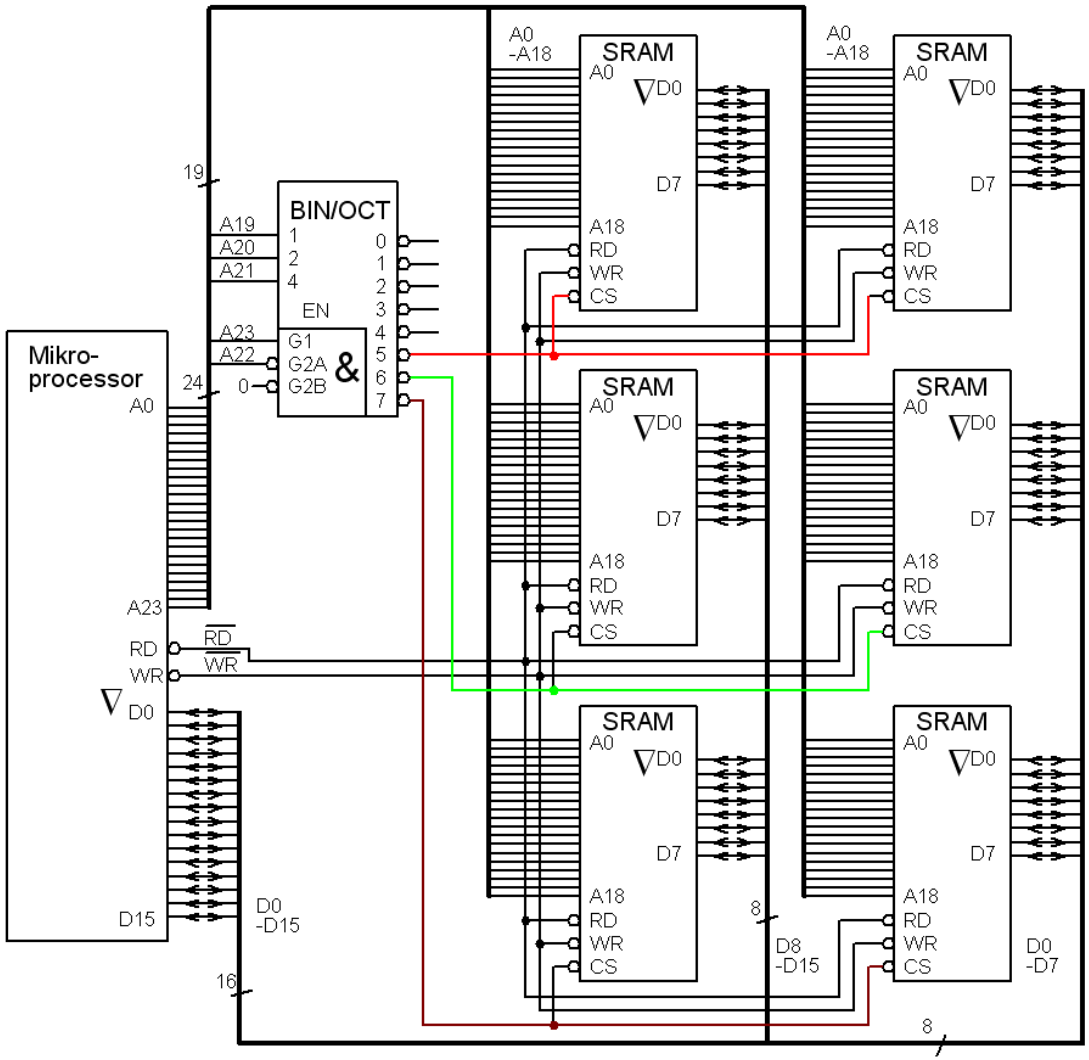




Antag att ROM och SRAM skall anslutas till en 16-bitars mikroprocessor som har 24 bitars adressering.

William Sandqvist william@kth.se

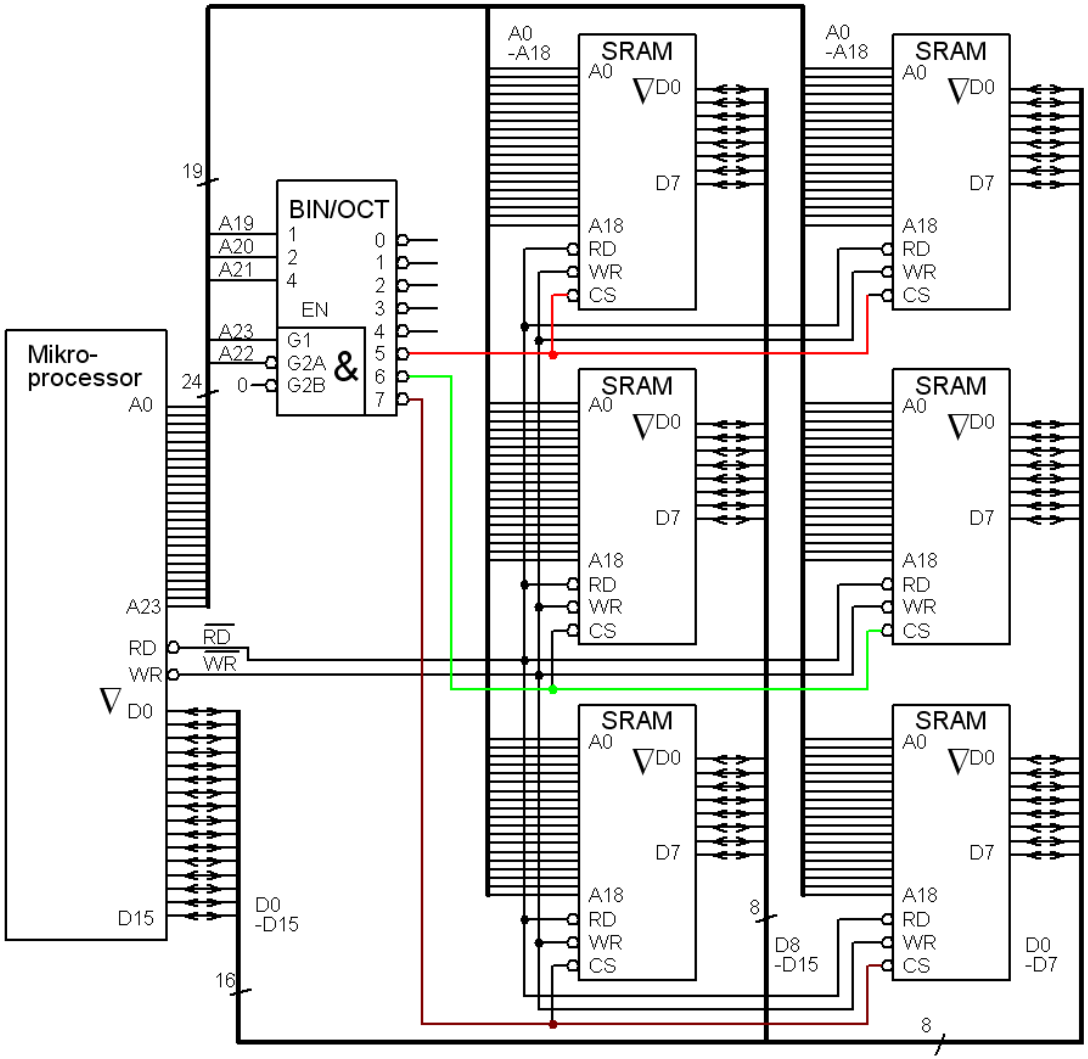
SRAM storlek?



Hur stort är figurens SRAM, och vilket är adress-området uttryckt i hexadeci-mala siffror?

William Sandqvist william@kth.se

SRAM storlek?



Hur stort är figurens SRAM, och vilket är adress-området uttryckt i hexadecimala siffror?

- **Minneskapsel**:

p = 512k q = 8 bitar

- **Minne**:

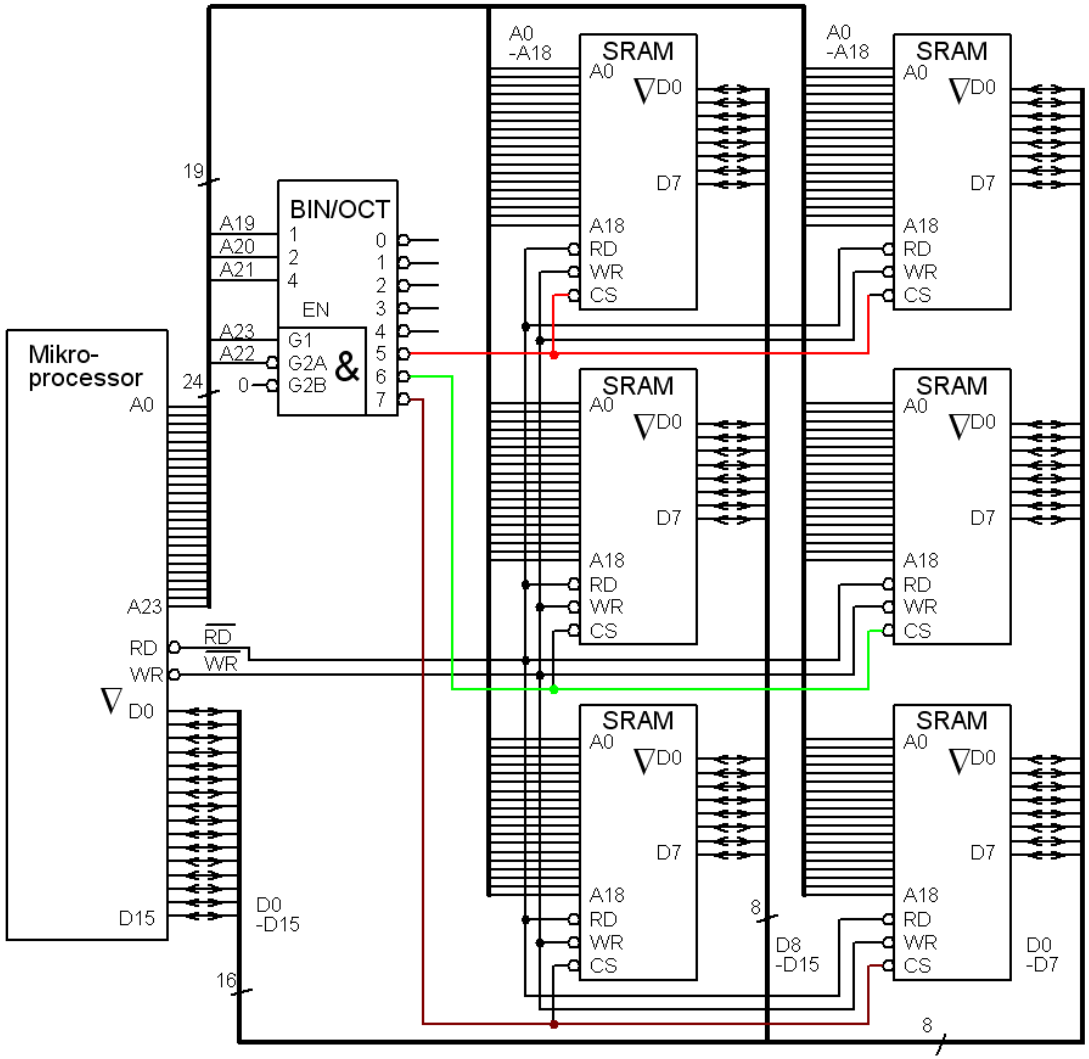
$r = 3$ $k = 2$ $K = 2 \times 3 = 6$

$M = k \times q = 2 \times 8 = 16$ bitar

$N = \mathrm{p} \times \mathrm{r} = 512\mathrm{k} \times 3 = 1{,}5\mathrm{M}$

William Sandqvist william@kth.se

SRAM storlek?



Hur stort är figurens SRAM, och vilket är adress-området uttryckt i hexadecimala siffror?

- **Minneskapsel**:

p = 512k q = 8 bitar

- **Minne**:

$r = 3 \quad k = 2 \quad K = 2 \times 3 = 6$

$M = k \times q = 2 \times 8 = 16$ bitar

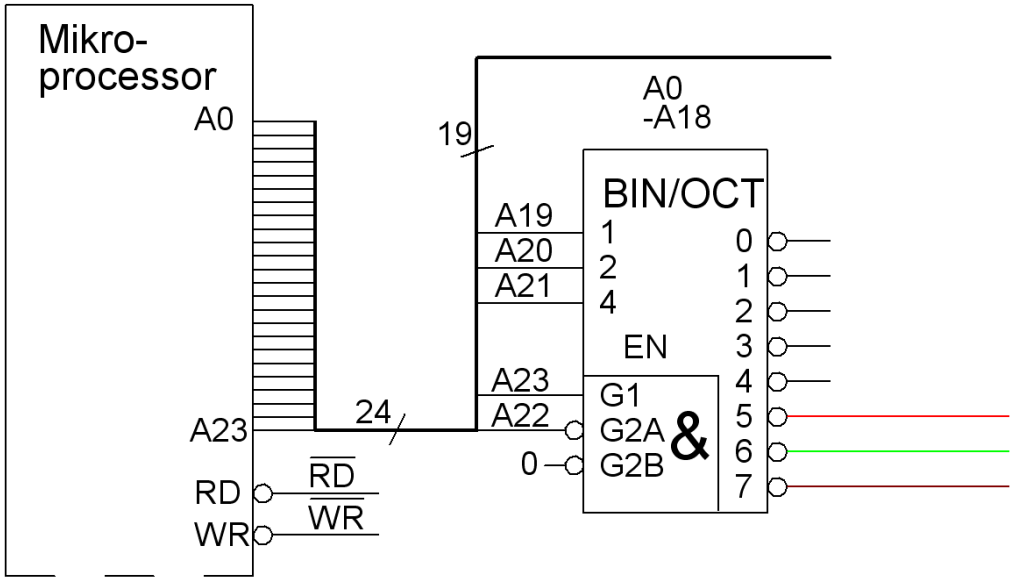
$N = \mathrm{p} \times \mathrm{r} = 512\mathrm{k} \times 3 = 1{,}5\mathrm{M}$

$$\overline{RD} = \overline{RD}$$

$$\overline{WR} = \overline{WR}$$

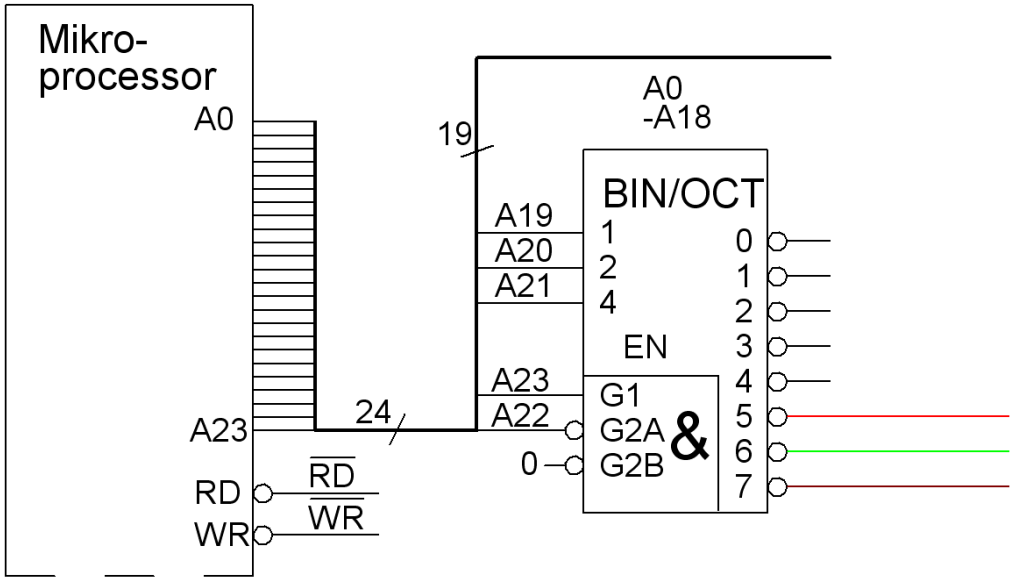
William Sandqvist william@kth.se

SRAM adressområde?



William Sandqvist william@kth.se

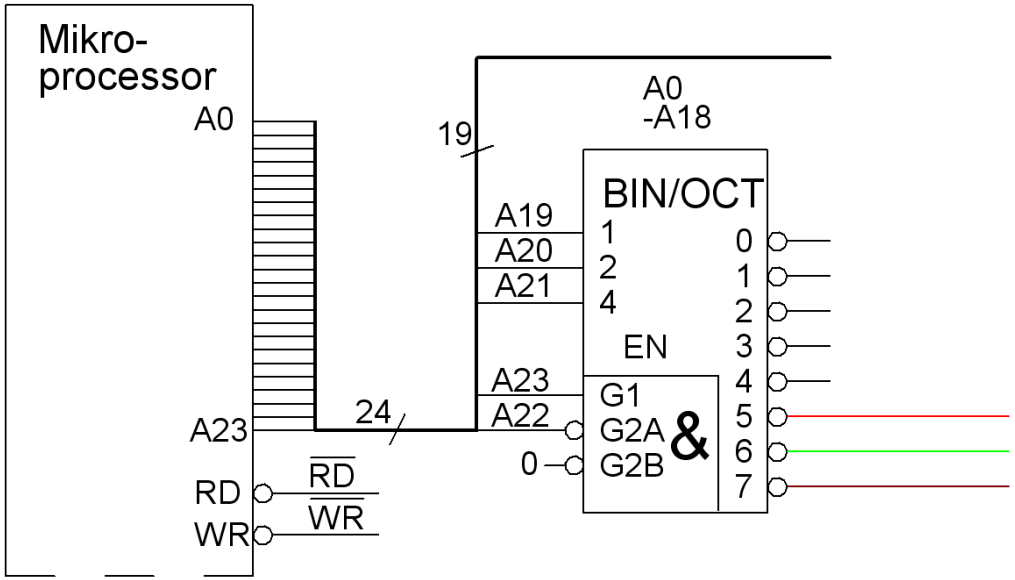
SRAM adressområde?



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

William Sandqvist william@kth.se

SRAM adressområde?



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

SRAM adressområde: A80000 - BFFFFF

William Sandqvist william@kth.se

Byt adressområde! ?

Ändra till adressområde 980000 – AFFFFF ?

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

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

William Sandqvist william@kth.se

Byt adressområde! ?

Ändra till adressområde 980000 – AFFFFF ?

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

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

William Sandqvist william@kth.se

Byt adressområde! ?

Ändra till adressområde 980000 – AFFFFF ?

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

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

”10|011” → ”3”
”10|101” → ”5”

William Sandqvist william@kth.se

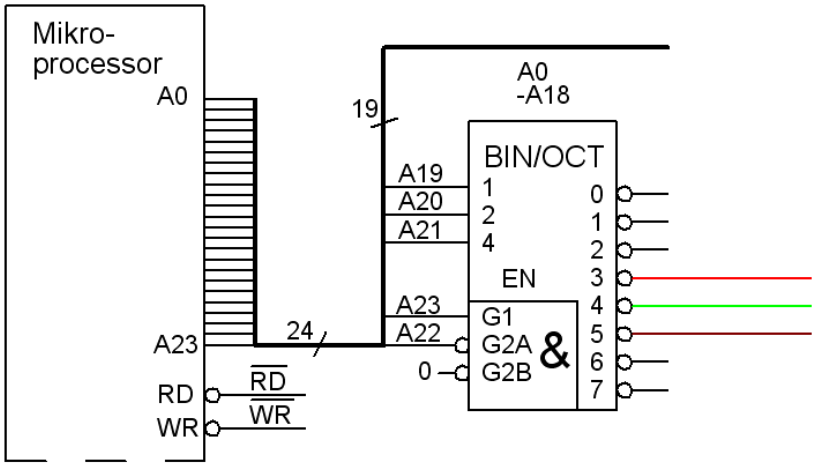
Byt adressområde! ?

Ändra till adressområde 980000 – AFFFFF ?

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

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

”10|011” → ”3”
”10|101” → ”5”



William Sandqvist william@kth.se

Byt adressområde! ?

Ändra till adressområde 480000 – 5FFFFF ?

William Sandqvist william@kth.se

Byt adressområde! ?

Ändra till adressområde 480000 – 5FFFFF ?

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

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

William Sandqvist william@kth.se

Byt adressområde! ?

Ändra till adressområde 480000 – 5FFFFF ?

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

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

”01|001” → ”1”
”01|011” → ”3”

William Sandqvist william@kth.se

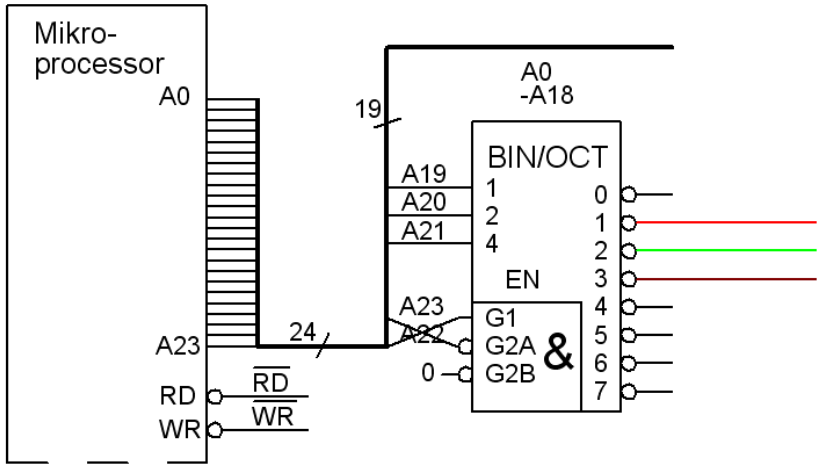
Byt adressområde! ?

Ändra till adressområde 480000 – 5FFFFF ?

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

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

”01|001” → ”1”
”01|011” → ”3”



William Sandqvist william@kth.se

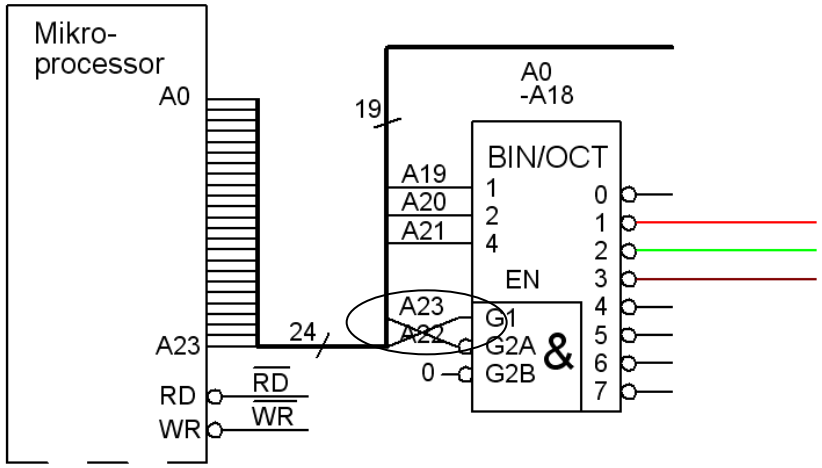
Byt adressområde! ?

Ändra till adressområde 480000 – 5FFFFF ?

480000
01|00|1000|0000|0000|0000|0000|

5FFFFF
01|01|1111|1111|1111|1111|1111|

”01|001” → ”1”
”01|011” → ”3”



William Sandqvist william@kth.se

ROM 00 00 00…?

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

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

William Sandqvist william@kth.se

ROM 00 00 00…?

Oftast läser en processor sin första instruktion från adress 0. Då måste det finnas ett läsminne på den adressen.
Antag ett ROM-minne 2M $\times$ 16 bitar adressområde 000000 … och framåt.
ROM Chip 512k$\times$8.

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

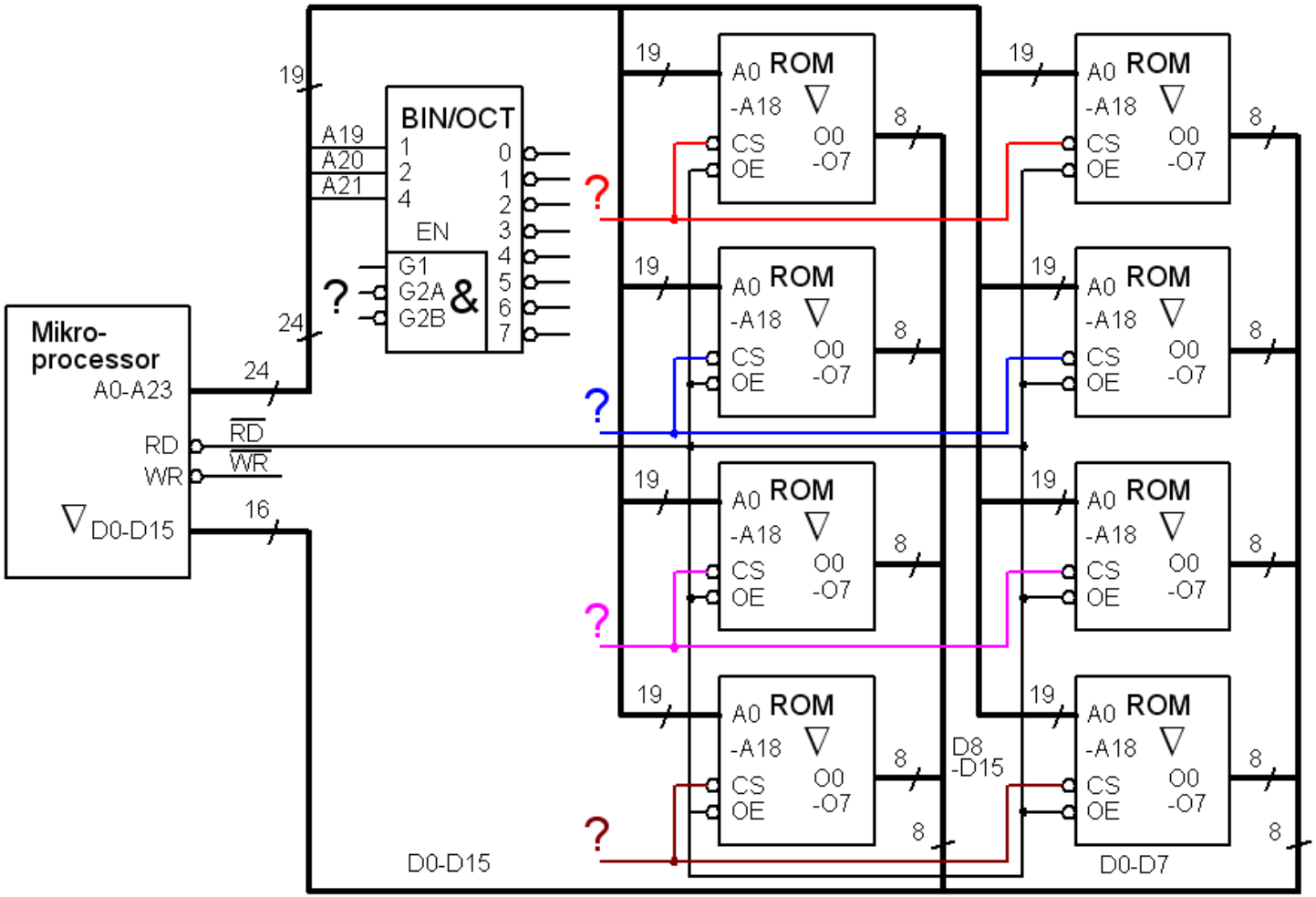
Minne:
N = 2 M (4·512k) ordlängden M = 16 bitar

Minneskapsel:
p = 512 k ordlängden q = 8 bitar

- Antalet kapselrader $r \leq N/p = 4 \cdot 512k/512k = 4$
- Antalet kapselkolumner $k \geq M/q = 16/8 = 2$
- Antalet kapslar $K = r \times k = 4 \times 2 = 8$

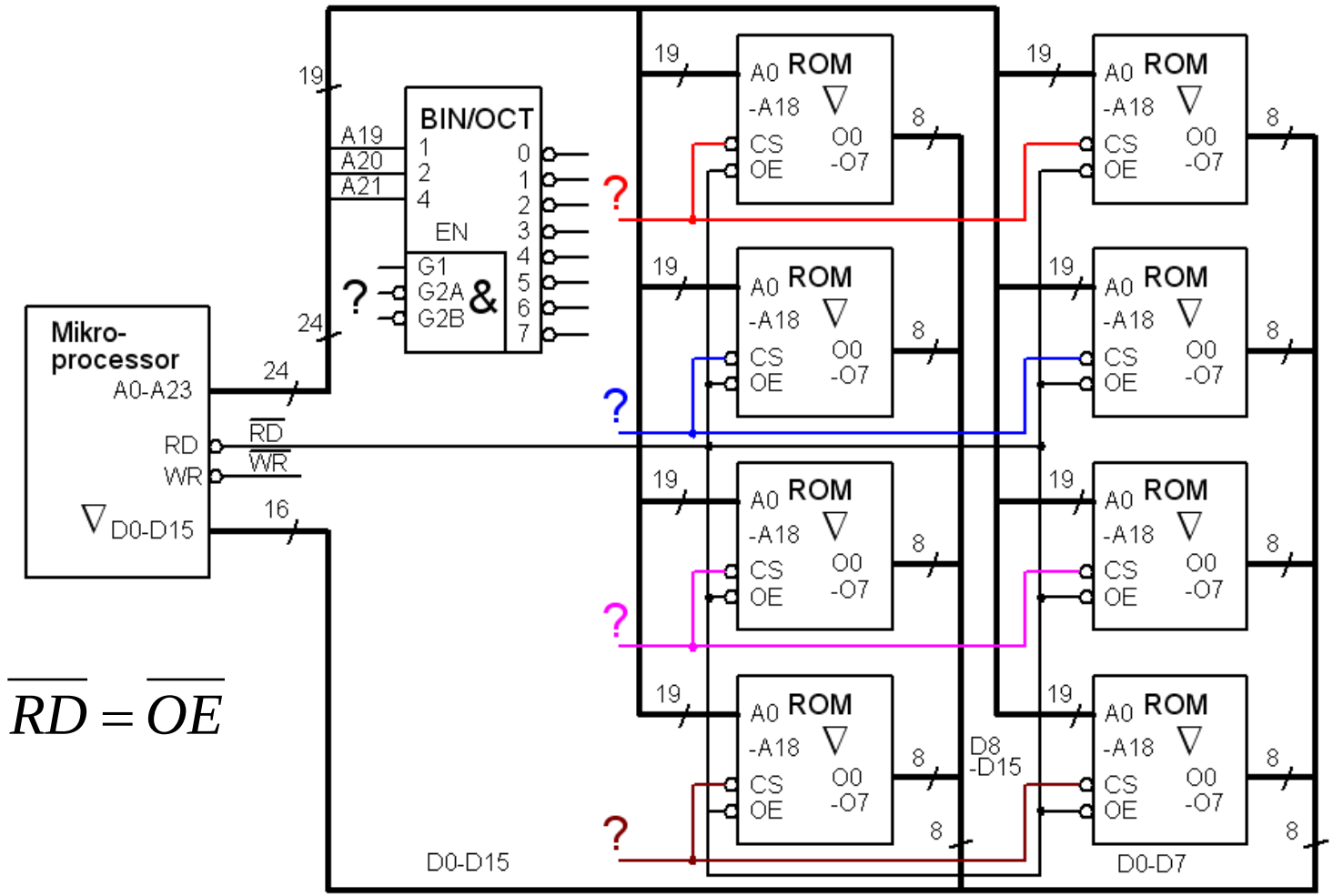
William Sandqvist william@kth.se

ROM anslutning?



William Sandqvist william@kth.se

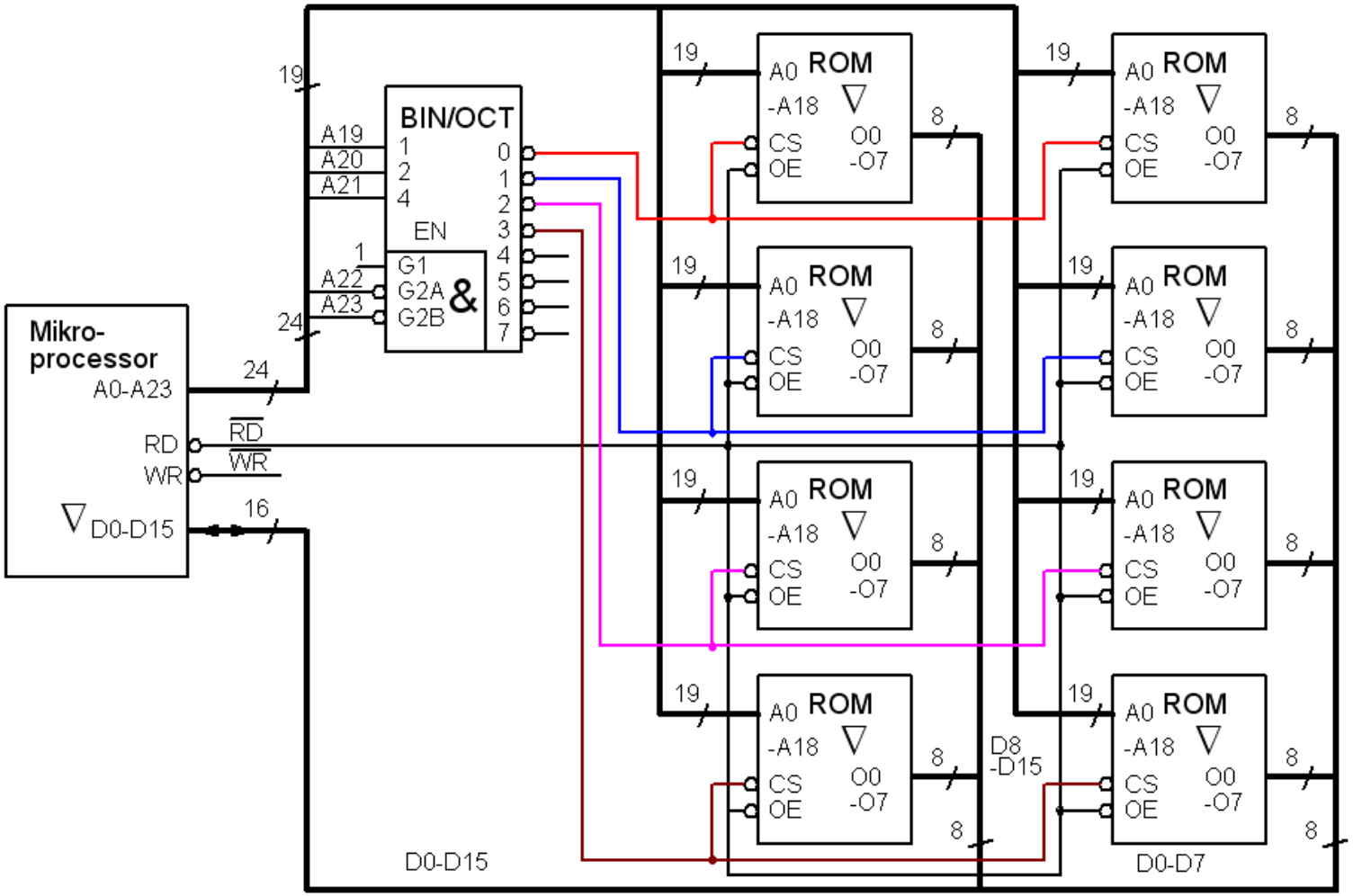
ROM anslutning?



$$\overline{RD} = \overline{OE}$$

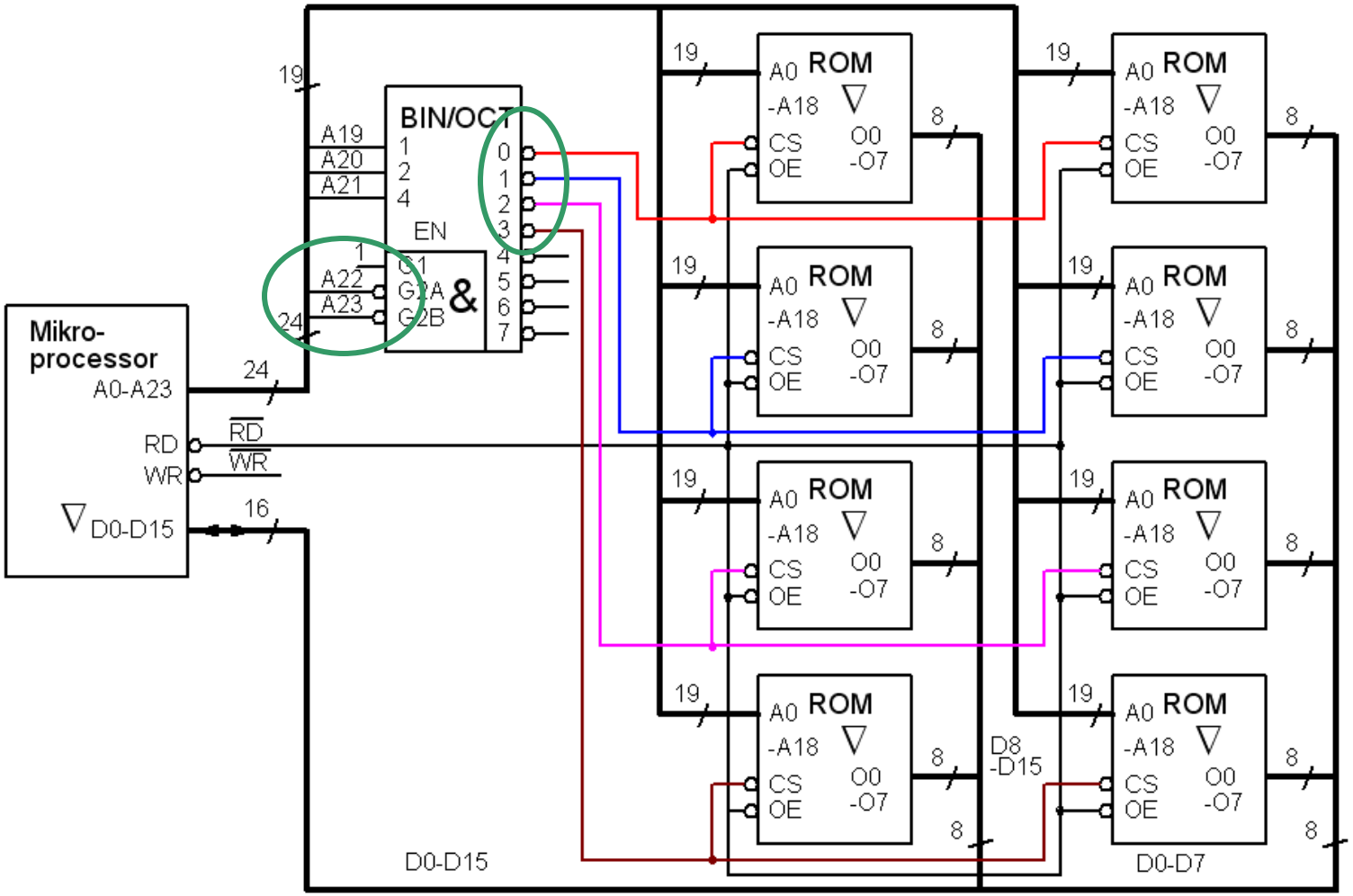
William Sandqvist william@kth.se

Decoder anslutning?



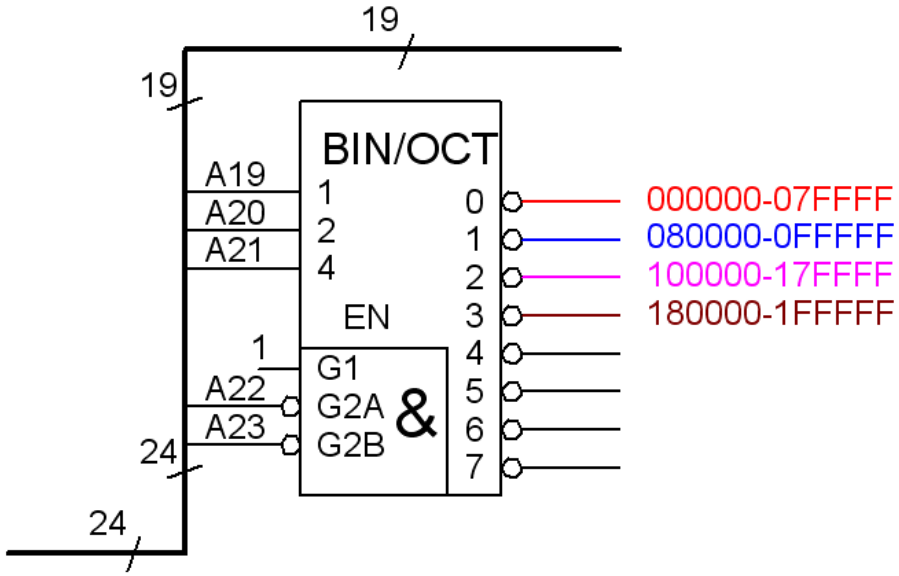
William Sandqvist william@kth.se

Decoder anslutning?



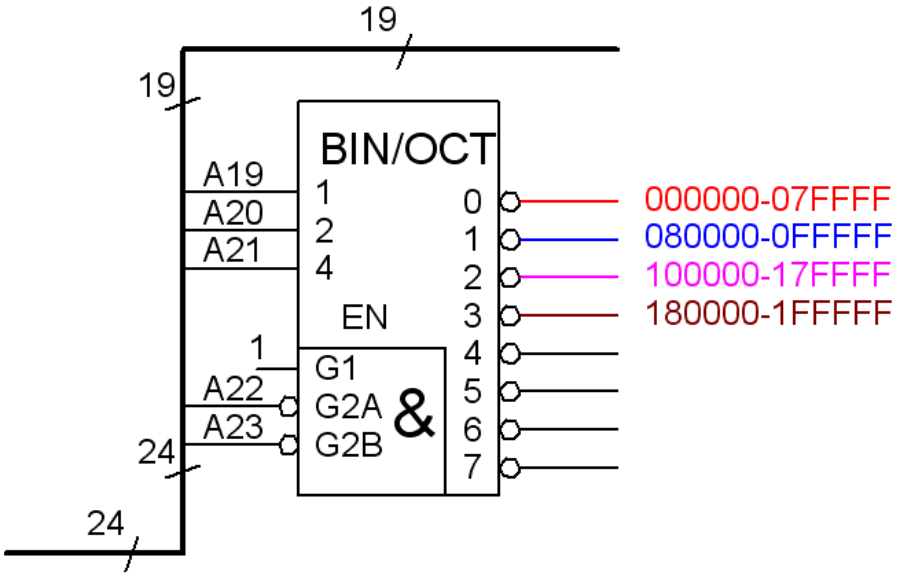
William Sandqvist william@kth.se

Decoder ROM adresser?



William Sandqvist william@kth.se

Decoder ROM adresser?



00ab|cmmm|mmmm|mmmm|mmmm|mmmm

00**00**|**0**000|0|0|0|0 - 00**00**|**0**111|F|F|F|F 000000-07FFFF

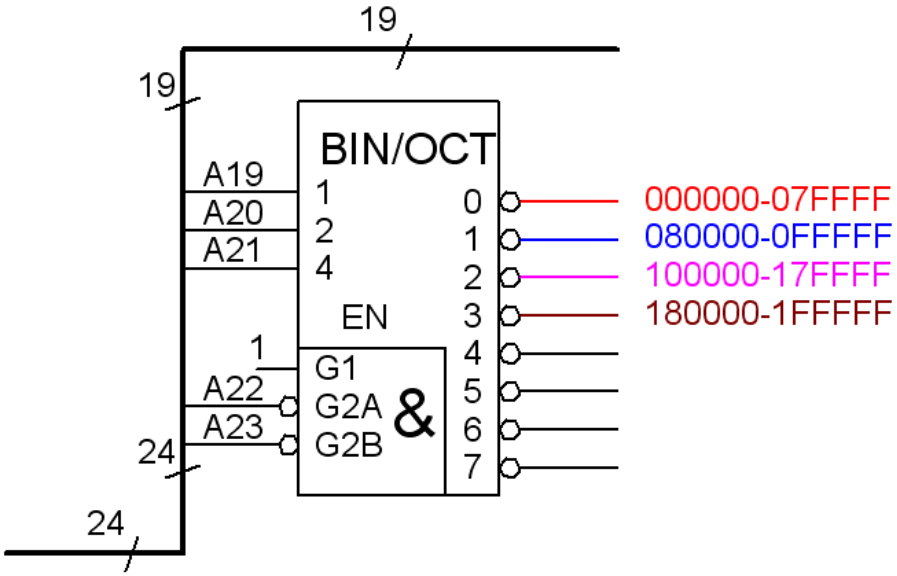
00**00**|**1**000|0|0|0|0 - 00**00**|**1**111|F|F|F|F 080000-0FFFFF

00**01**|**0**000|0|0|0|0 - 00**01**|**0**111|F|F|F|F 100000-17FFFF

00**01**|**1**000|0|0|0|0 - 00**01**|**1**111|F|F|F|F 180000-1FFFFF

William Sandqvist william@kth.se

Decoder ROM adresser?



ROM

00ab|cmmm|mmmm|mmmm|mmmm|mmmm

00**00**|**0**000|0|0|0|0 - 00**00**|**0**111|F|F|F|F 000000-07FFFF

00**00**|**1**000|0|0|0|0 - 00**00**|**1**111|F|F|F|F 080000-0FFFFF

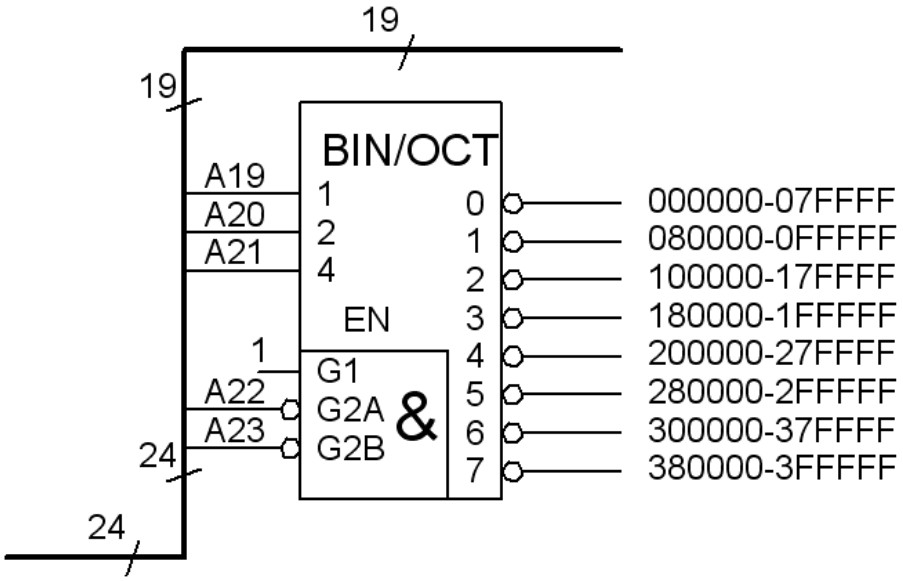
00**01**|**0**000|0|0|0|0 - 00**01**|**0**111|F|F|F|F 100000-17FFFF

00**01**|**1**000|0|0|0|0 - 00**01**|**1**111|F|F|F|F 180000-1FFFFF

Totalt ROM 000000 – 1FFFFF

William Sandqvist william@kth.se

Decoder SRAM+I/O adresser?



00ab|cmmm|mmmm|mmmm|mmmm|mmmm

00**10**|**0**000|0|0|0|0 - 00**10**|**0**111|F|F|F|F 200000-27FFFF

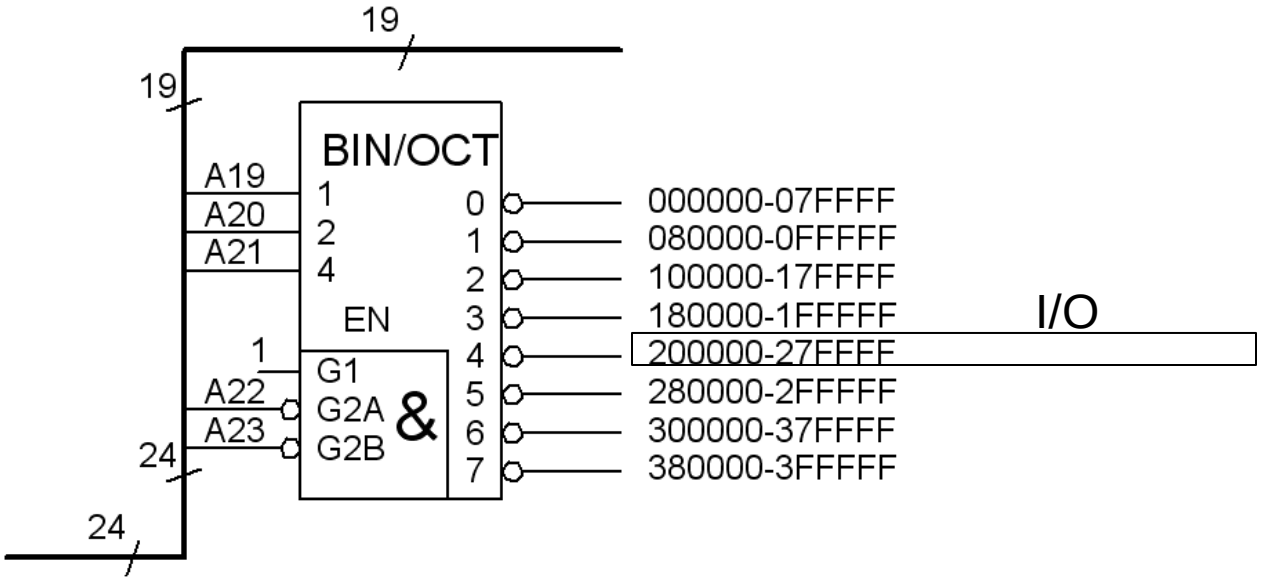
00**10**|**1**000|0|0|0|0 - 00**10**|**1**111|F|F|F|F 280000-2FFFFF

00**11**|**0**000|0|0|0|0 - 00**11**|**0**111|F|F|F|F 300000-37FFFF

00**11**|**1**000|0|0|0|0 - 00**11**|**1**111|F|F|F|F 380000-3FFFFF

William Sandqvist william@kth.se

Decoder SRAM+I/O adresser?



00ab|cmmm|mmmm|mmmm|mmmm|mmmm

00**10**|**0**000|0|0|0|0 - 00**10**|**0**111|F|F|F|F 200000-27FFFF

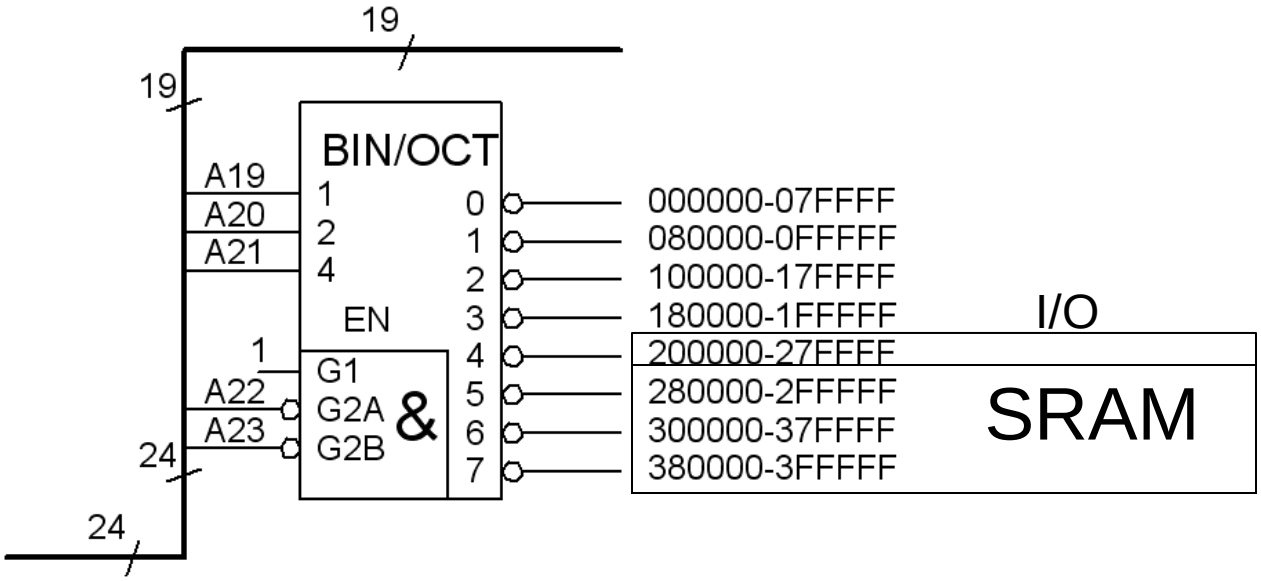
00**10**|**1**000|0|0|0|0 - 00**10**|**1**111|F|F|F|F 280000-2FFFFF

00**11**|**0**000|0|0|0|0 - 00**11**|**0**111|F|F|F|F 300000-37FFFF

00**11**|**1**000|0|0|0|0 - 00**11**|**1**111|F|F|F|F 380000-3FFFFF

William Sandqvist william@kth.se

Decoder SRAM+I/O adresser?



00ab|cmmm|mmmm|mmmm|mmmm|mmmm

00**10**|**0**000|0|0|0|0 - 00**10**|**0**111|F|F|F|F 200000-27FFFF

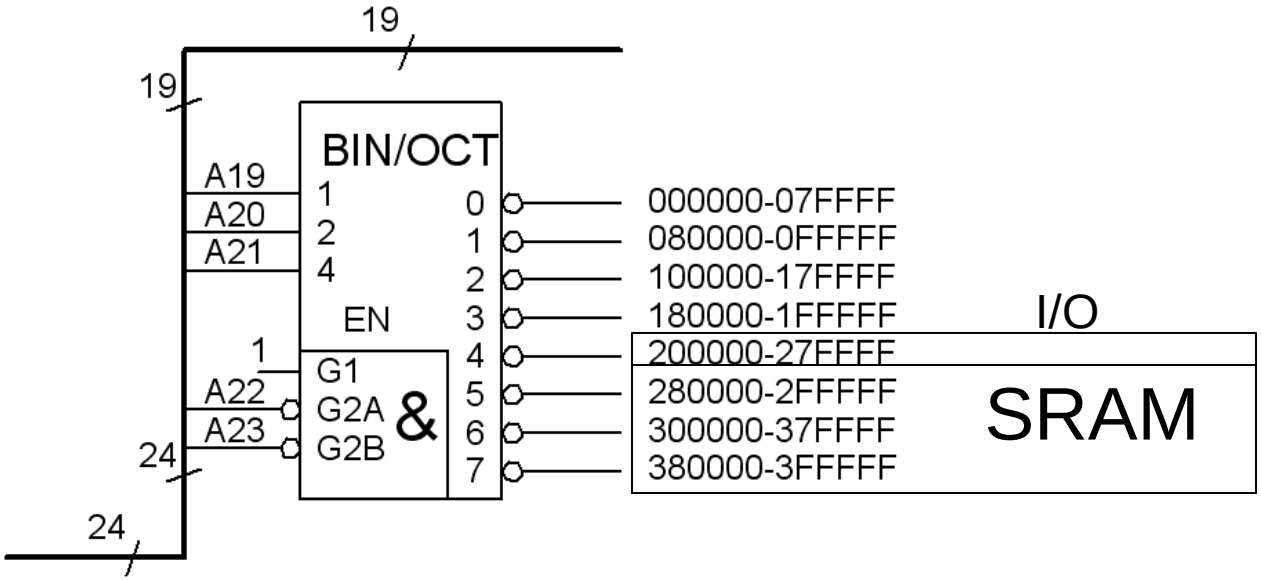
00**10**|**1**000|0|0|0|0 - 00**10**|**1**111|F|F|F|F 280000-2FFFFF

00**11**|**0**000|0|0|0|0 - 00**11**|**0**111|F|F|F|F 300000-37FFFF

00**11**|**1**000|0|0|0|0 - 00**11**|**1**111|F|F|F|F 380000-3FFFFF

William Sandqvist william@kth.se

Decoder SRAM+I/O adresser?



00ab|cmmm|mmmm|mmmm|mmmm|mmmm

00**10**|**0**000|0|0|0|0 - 00**10**|**0**111|F|F|F|F 200000-27FFFF

00**10**|**1**000|0|0|0|0 - 00**10**|**1**111|F|F|F|F 280000-2FFFFF

00**11**|**0**000|0|0|0|0 - 00**11**|**0**111|F|F|F|F 300000-37FFFF

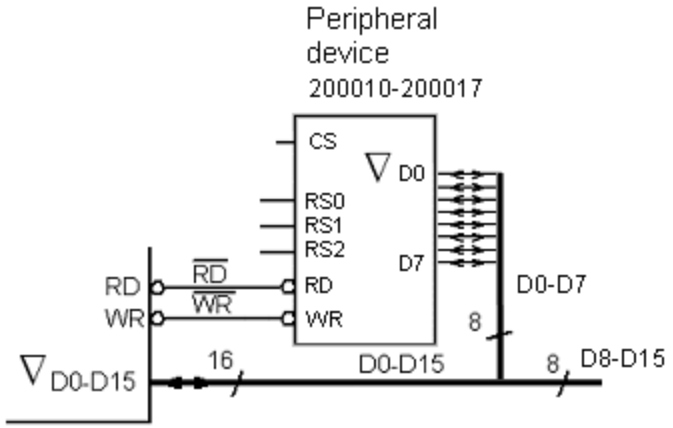
00**11**|**1**000|0|0|0|0 - 00**11**|**1**111|F|F|F|F 380000-3FFFFF

Möjliga SRAM+I/O adresser 200000 – 3FFFFF

William Sandqvist william@kth.se

ÖH **12.3** Input/Output

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

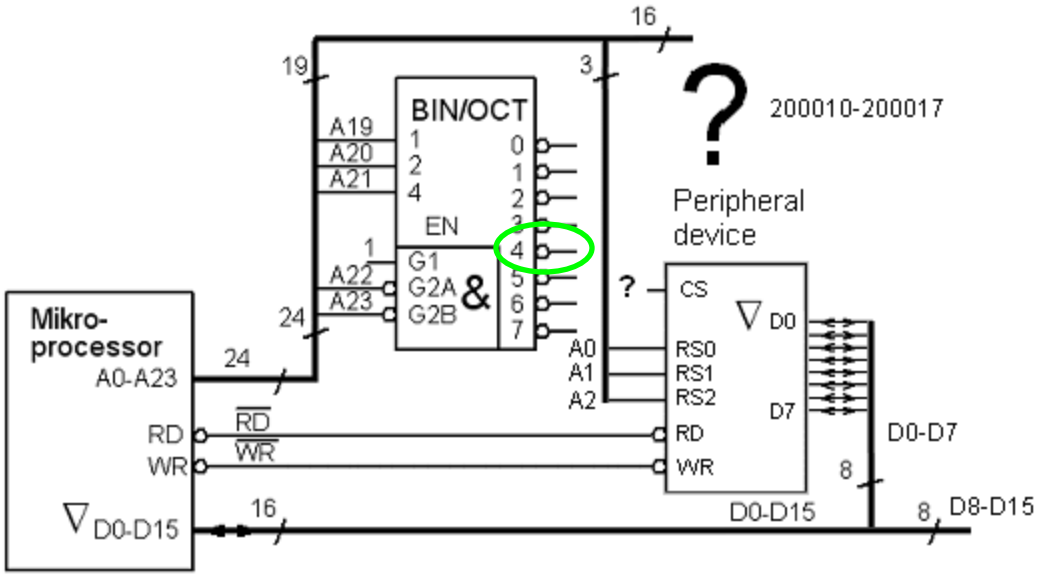


Periferikretsen kopplas in som ett litet RAM-minne. Bara de 8 minst signifikanta databitarna används. CS *Chip Select enablar chippet.*

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

William Sandqvist william@kth.se

ÖH **12.3** Input/Output



I/O adresser, på avkodarens utgång ”4”, 200000 – 27FFFF enligt tidigare uppgift.

William Sandqvist william@kth.se

Avkodningen

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

Avkodarens utgång ”4”

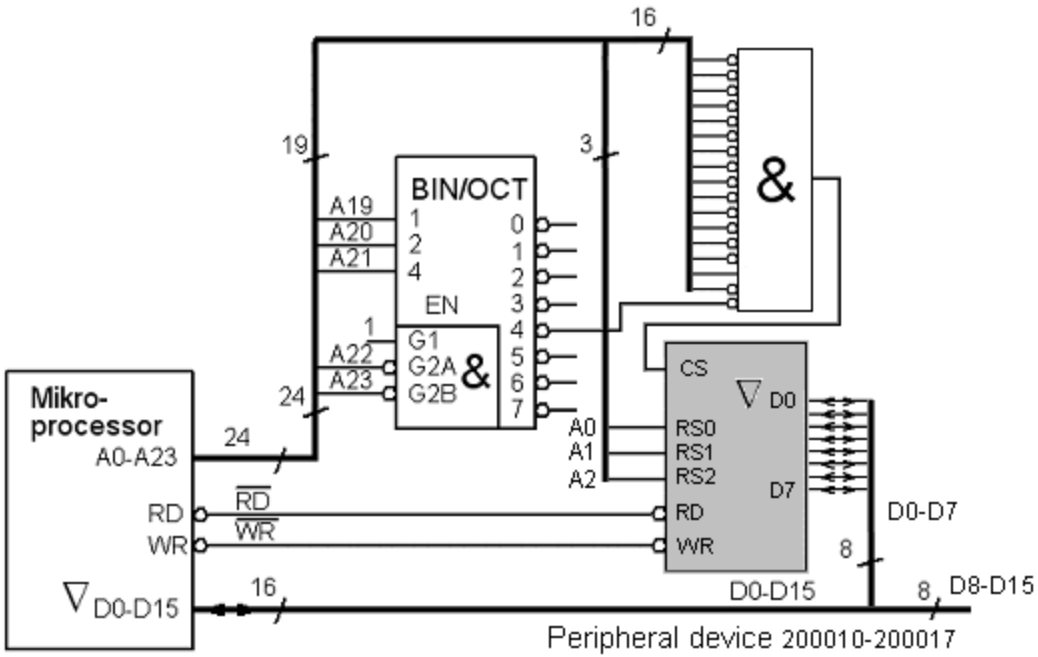
$RS_2RS_1RS_0$

Kvar att avkoda:

$$\overline{A}_{18} \cdot \overline{A}_{17} \cdot \overline{A}_{16} \cdot \overline{A}_{15} \cdot \overline{A}_{14} \cdot \overline{A}_{13} \cdot \overline{A}_{12} \cdot \overline{A}_{11} \cdot \overline{A}_{10} \cdot \overline{A}_{9} \cdot \overline{A}_{8} \cdot \overline{A}_{7} \cdot \overline{A}_{6} \cdot \overline{A}_{5} \cdot A_{4} \cdot \overline{A}_{3}$$

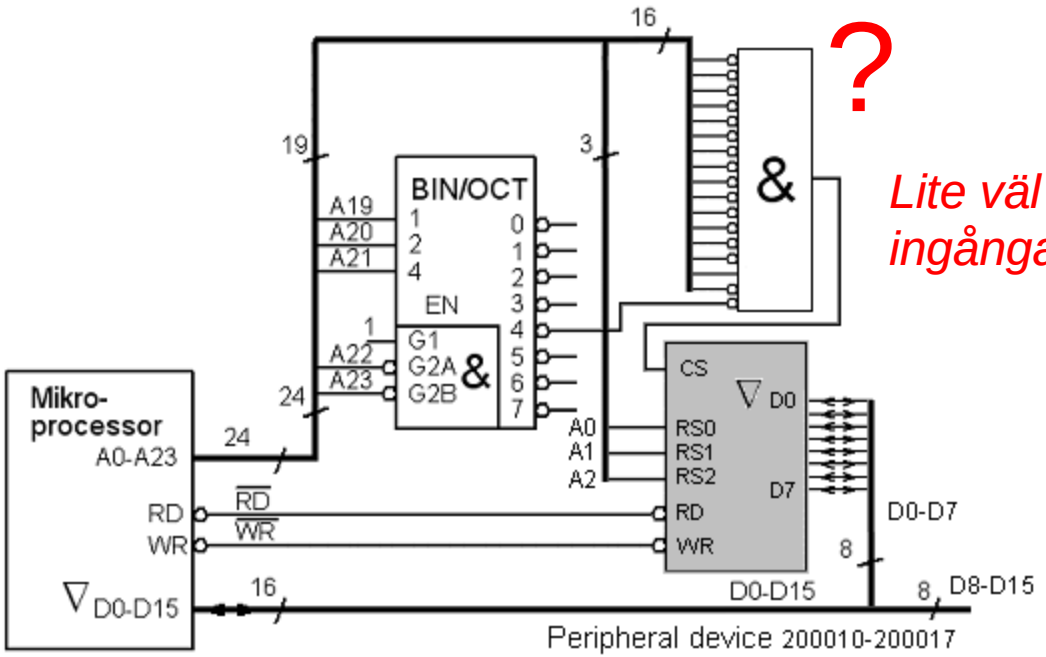
William Sandqvist william@kth.se

Anslutningarna



William Sandqvist william@kth.se

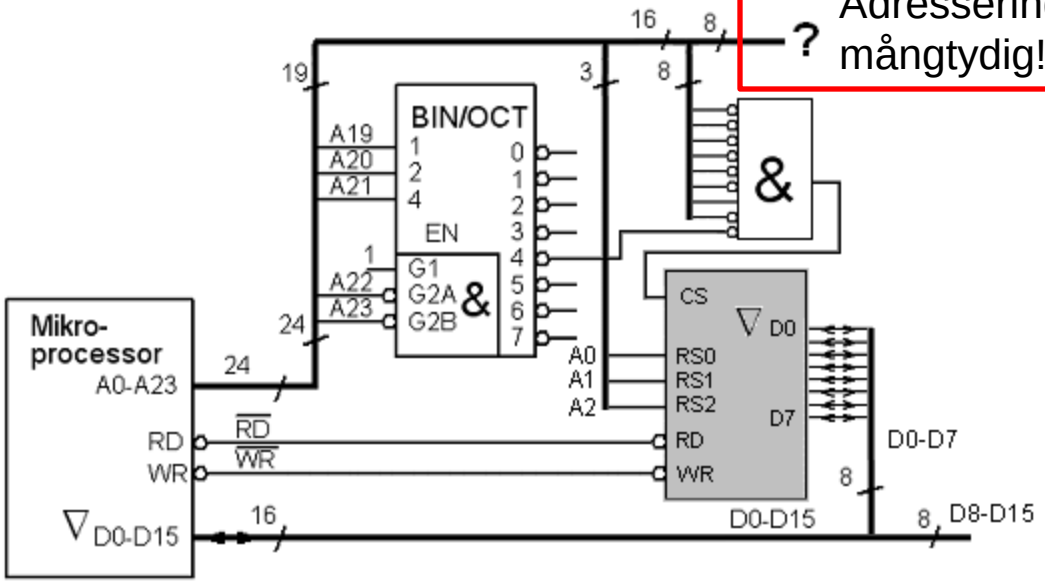
Anslutningarna



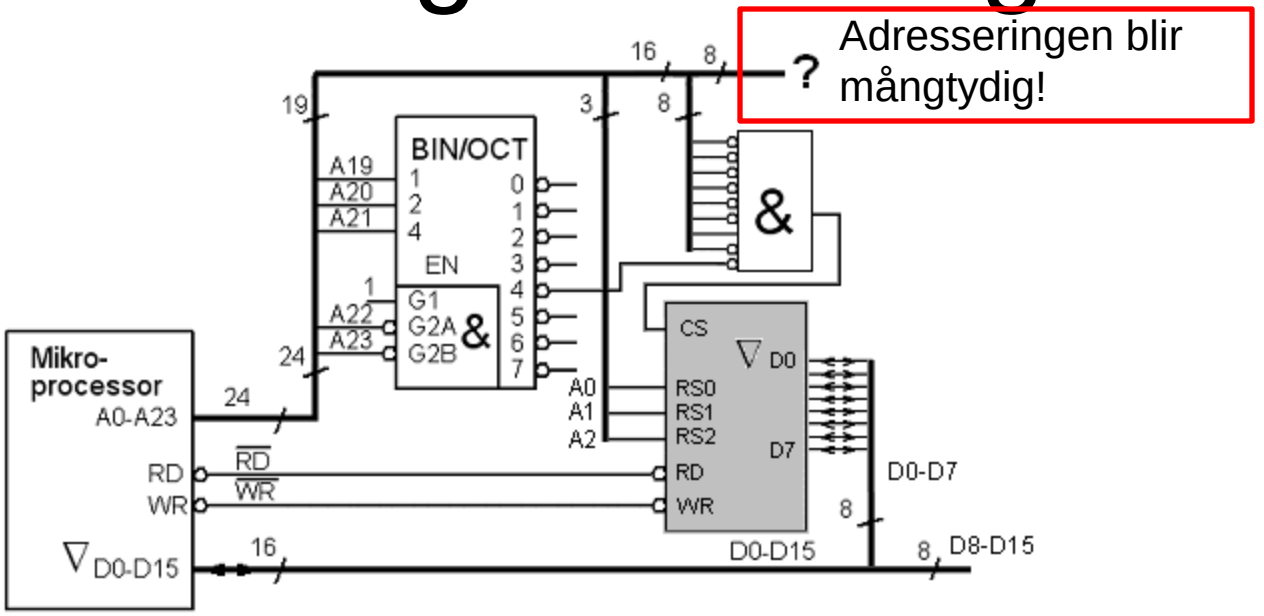
Lite väl många ingångar?

William Sandqvist william@kth.se

Ofullständig avkodning?



Ofullständig avkodning?



För fullständig avkodning använde vi en &-grind med 17 ingångar! I bland gör man en **ofullständig avkodning**. Man struntar då i att ta med alla adress-signalerna och kan därmed använda en grind med färre ingångar.

I/O-enhetens adressering blir mångtydig, den kan adresseras med många olika adresser, men den som skriver programkoden bestämmer ju själv vilka adresser det är som används. Huvudsaken är att man ser till att I/O-enhetens adresser *inte kolliderar* med någon annan enhets adresser.

William Sandqvist william@kth.se

volatile ?

William Sandqvist william@kth.se

`volatile` ?

Eftersom I/O-enheter *inte* är riktiga minnen – det kan verka som om innehållet kan ändras ”av sig självt” – så kan man vid programmeringen av processorn behöva ”hjälpa” kompilatorn att förstå detta genom att deklarera dem som **`volatile`** (= flyktiga) i sina datorprogram.

Detta kommer Du att möta i Datorteknik-kursen.

William Sandqvist william@kth.se

Repetition inför tentamen

ÖH 6.10 Kombinatoriskt nät med 5 variabler

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

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

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

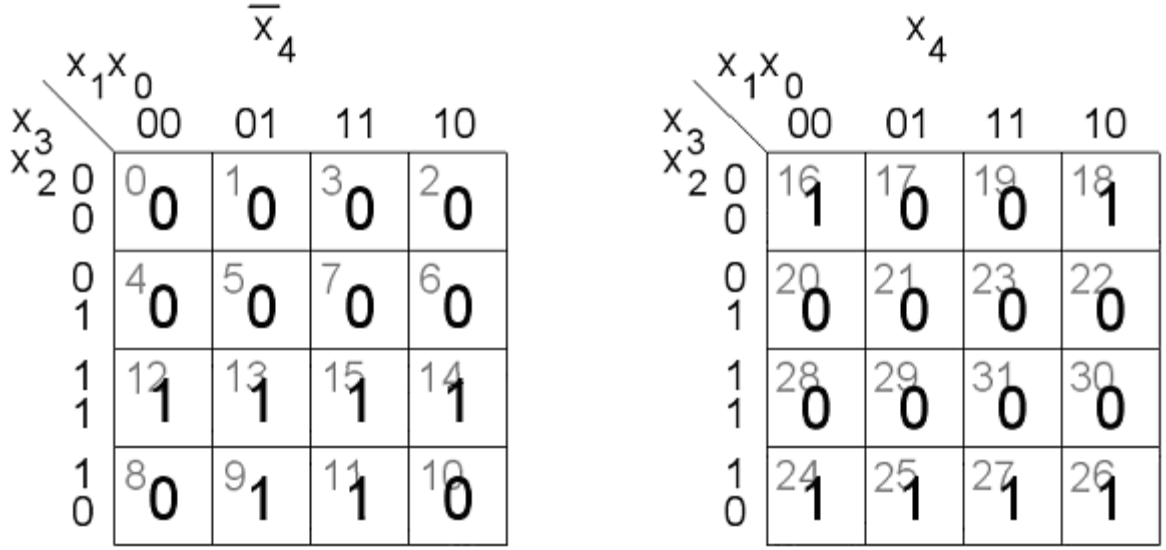
	x_4	x_3	x_2	x_1	x_0	f
16	1	0	0	0	0	1
17	1	0	0	0	1	0
18	1	0	0	1	0	1
19	1	0	0	1	1	0
20	1	0	1	0	0	0
21	1	0	1	0	1	0
22	1	0	1	1	0	0
23	1	0	1	1	1	0
24	1	1	0	0	0	1
25	1	1	0	0	1	1
26	1	1	0	1	0	1
27	1	1	0	1	1	1
28	1	1	1	0	0	0
29	1	1	1	0	1	0
30	1	1	1	1	0	0
31	1	1	1	1	1	0

William Sandqvist william@kth.se

6.10 Kombinatoriskt nät med 5 variabler

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

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

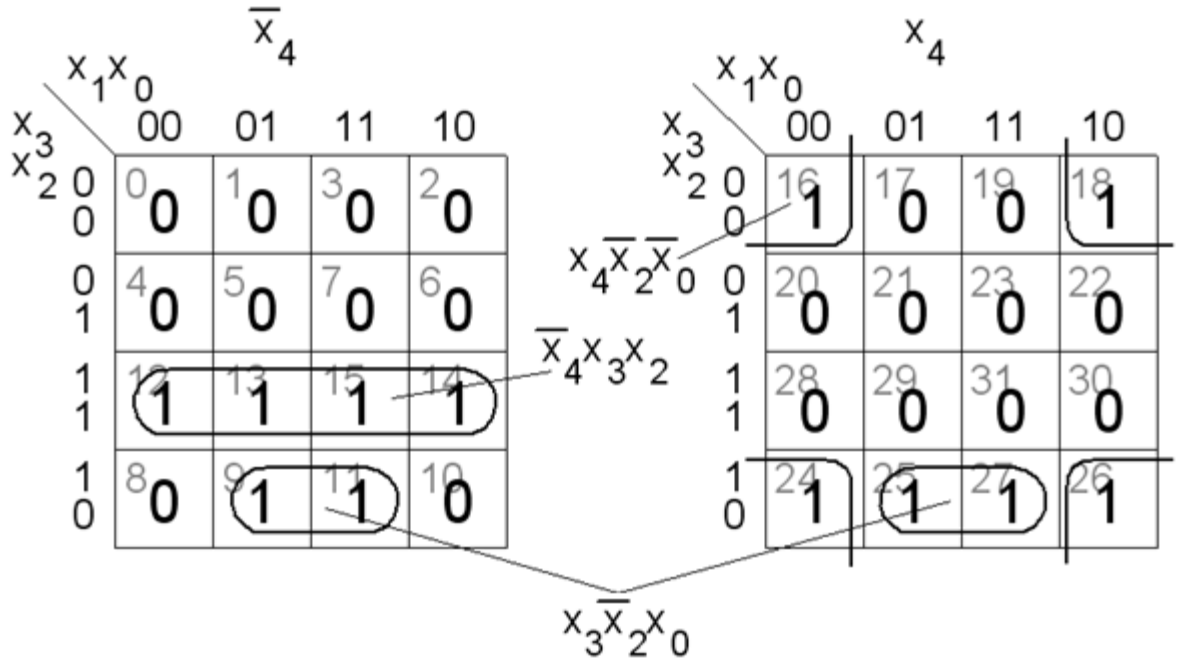


William Sandqvist william@kth.se

6.10 Kombinatoriskt nät med 5 variabler

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

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

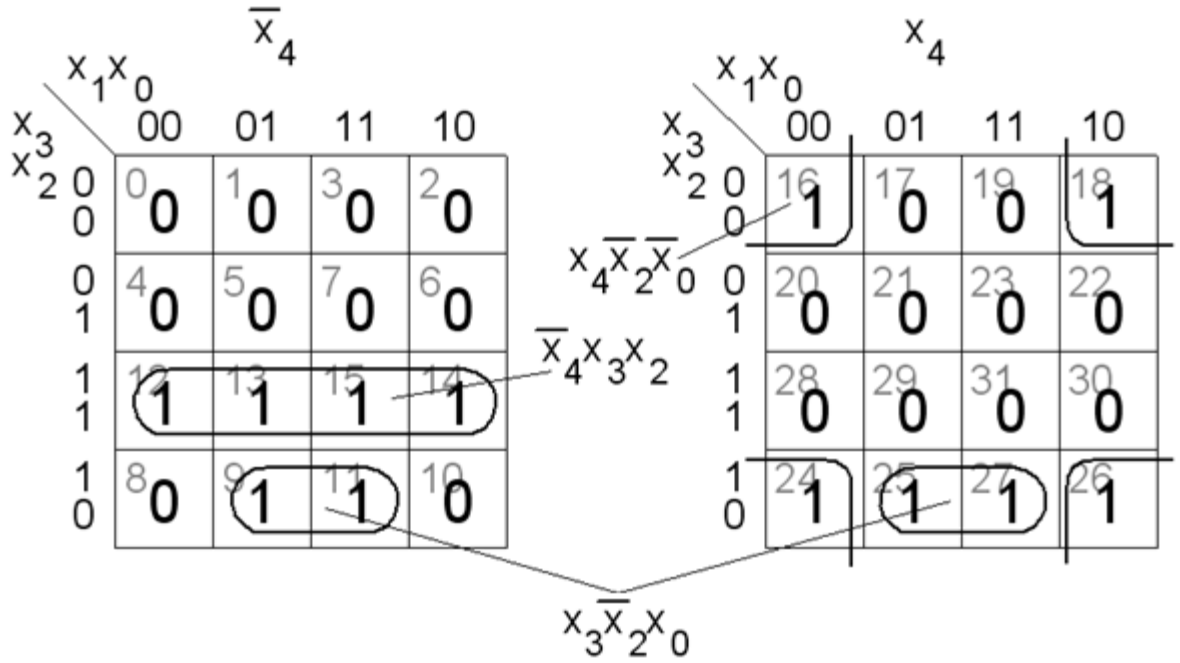


William Sandqvist william@kth.se

6.10 Kombinatoriskt nät med 5 variabler

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

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



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

William Sandqvist william@kth.se

6.10 Kombinatoriskt nät med 5 variabler

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

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

$\overline{x}_4$

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

x_4

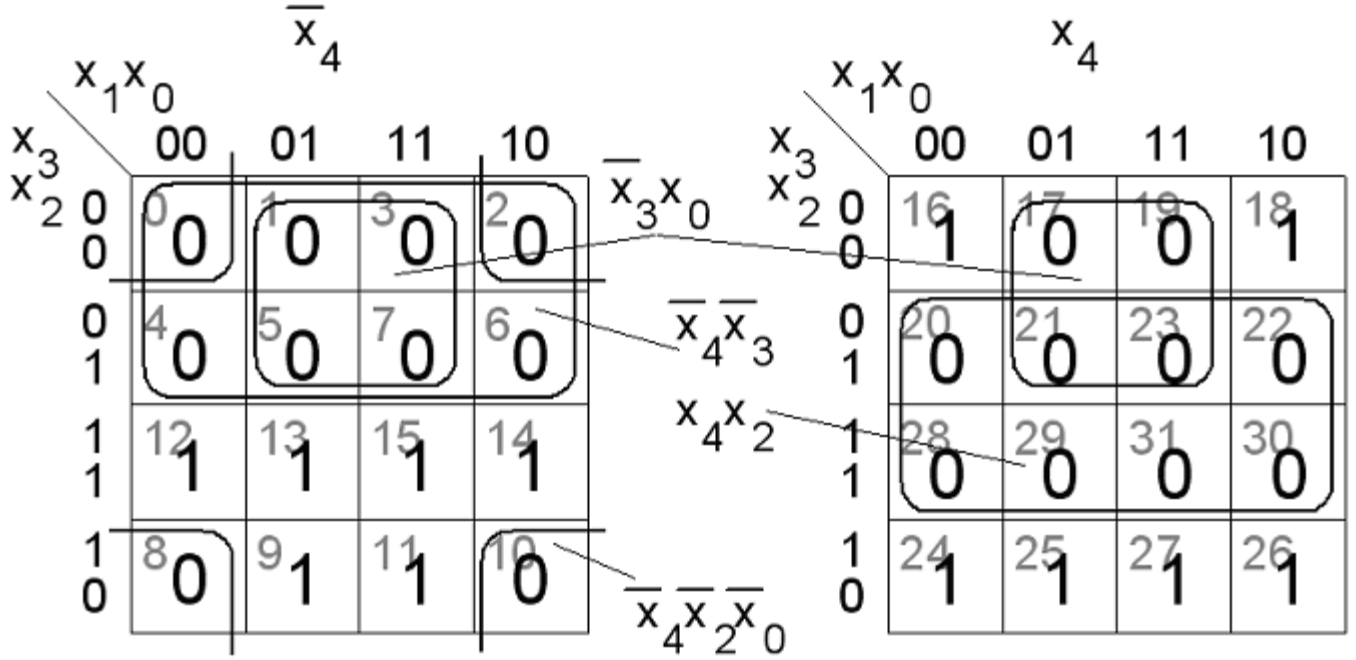
x_3x_2 \ x_1x_0	00	01	11	10
00	16 1	17 0	19 0	18 1
01	20 0	21 0	23 0	22 0
11	28 0	29 0	31 0	30 0
10	24 1	25 1	27 1	26 1

William Sandqvist william@kth.se

6.10 Kombinatoriskt nät med 5 variabler

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

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

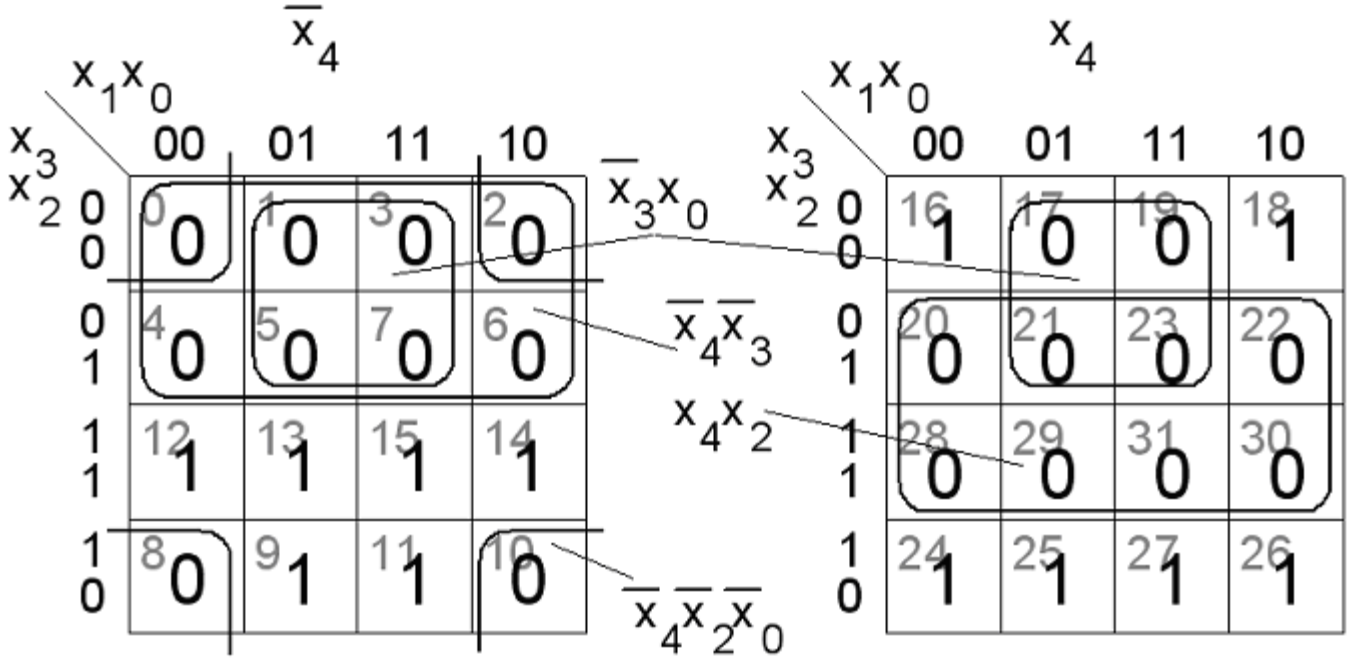


William Sandqvist william@kth.se

6.10 Kombinatoriskt nät med 5 variabler

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

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

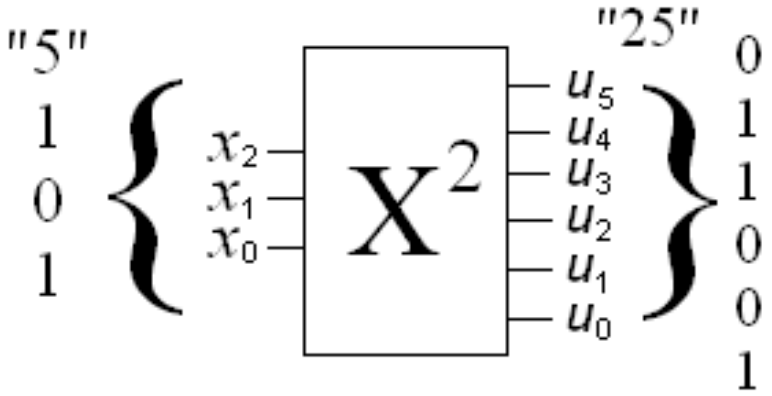


$$\overline{f} = \overline{x_4}\,\overline{x_3} + \overline{x_3}\,x_0 + x_4\,x_2 + \overline{x_4}\,\overline{x_2}\,\overline{x_0}$$

William Sandqvist william@kth.se

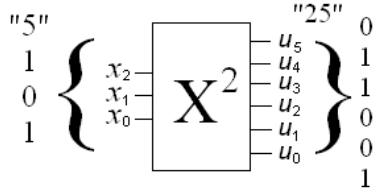
ÖH 8.1 Binär kvadrerare

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



William Sandqvist william@kth.se

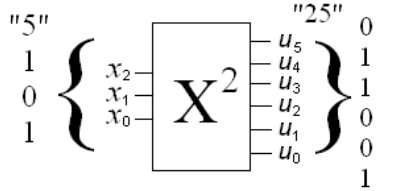
8.1 Sanningstabell



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

William Sandqvist william@kth.se

8.1 Karnaughdiagram



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

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

$u_5 = x_2 x_1$

x_2 \ x_1x_0	00	01	11	10
0	0	0	0	0
1	0	0	1	1

$u_4 = x_2 x_0 + x_2 \bar{x}_1$

x_2 \ x_1x_0	00	01	11	10
0	0	0	0	0
1	1	1	1	0

$u_3 = \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 x_0$

x_2 \ x_1x_0	00	01	11	10
0	0	0	1	0
1	0	1	0	0

$u_2 = x_1 \bar{x}_0$

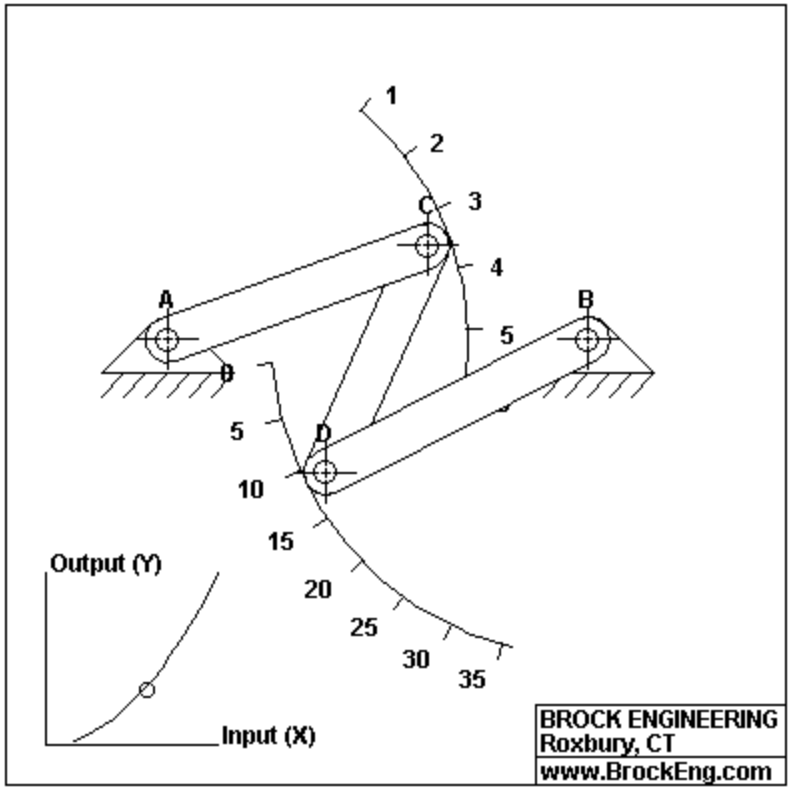
x_2 \ x_1x_0	00	01	11	10
0	0	0	0	1
1	0	0	0	1

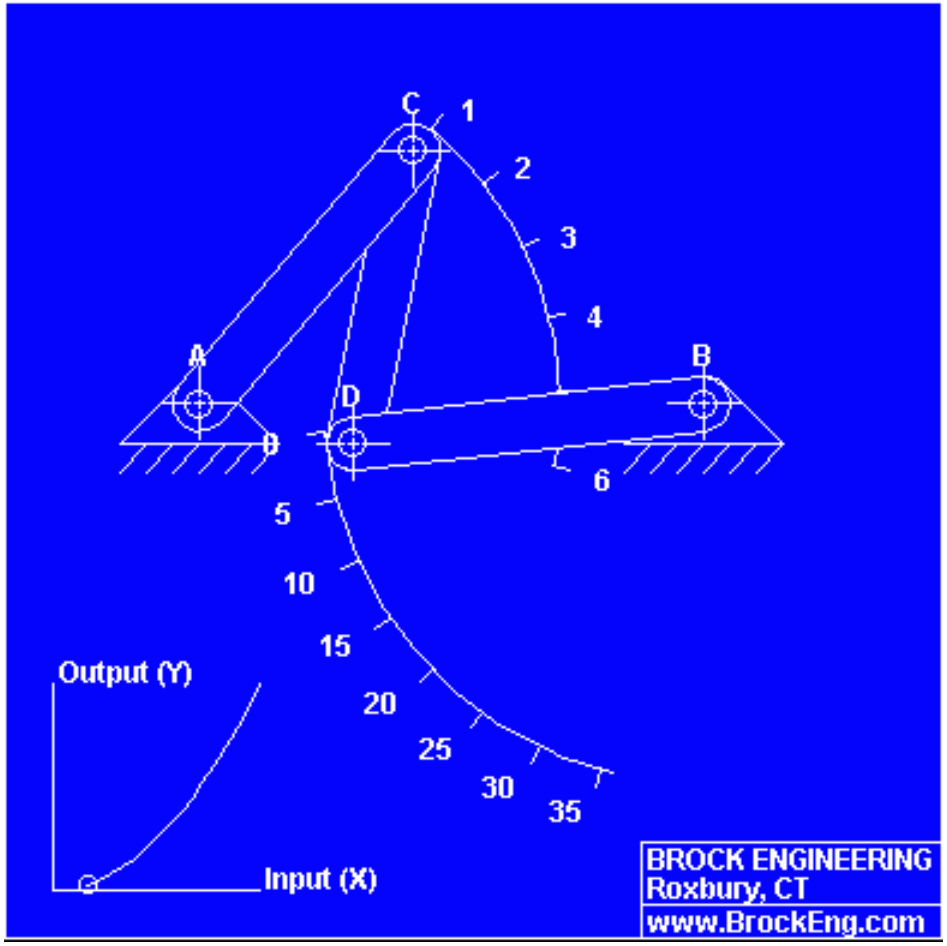
$u_1 = 0$

$u_0 = x_0$

William Sandqvist william@kth.se

Mekanisk ”kvadrerare”





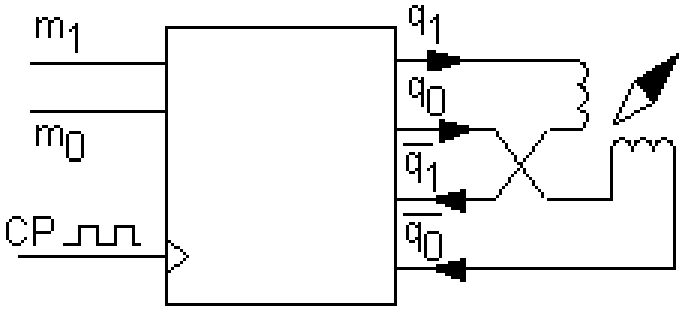
Brock institute for advaced studies function generator

ÖH 10.9 Stegmotorstyrning

En stegmotor är en digital komponent som drivs med pulser.

Stegmotorer brukar anslutas till räknare som räknar Gray-kod.

Figurens räknare har dessutom en mode-ingång, m_1m_0.

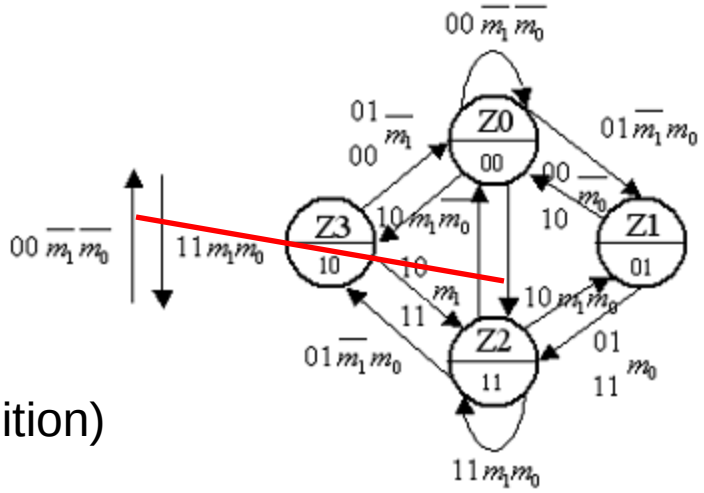


$m_1m_0 = 00 \rightarrow$ 0-ställning (fix position)

$m_1m_0 = 01 \rightarrow$ upp-räkning (cw)

$m_1m_0 = 10 \rightarrow$ ner-räkning (ccw)

$m_1m_0 = 11 \rightarrow$ 1-ställning (annan fix position)



William Sandqvist william@kth.se

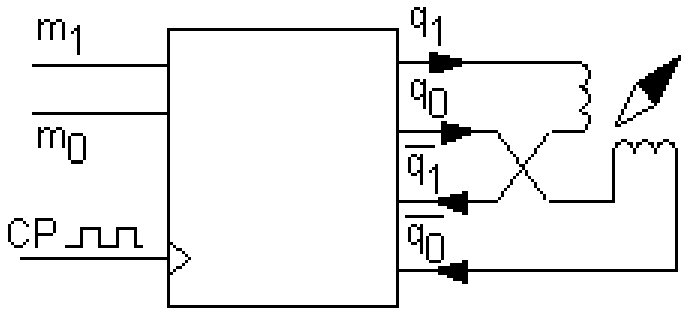
10.9 Tillståndsdiagrammet

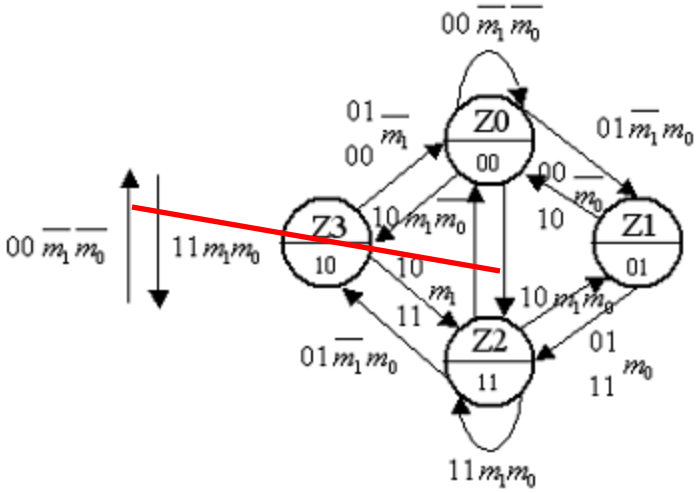
$m_1m_0 = 00 \rightarrow$ 0-ställning (fix position)

$m_1m_0 = 01 \rightarrow$ upp-räkning (cw)

$m_1m_0 = 10 \rightarrow$ ner-räkning (ccw)

$m_1m_0 = 11 \rightarrow$ 1-ställning (annan fix position)

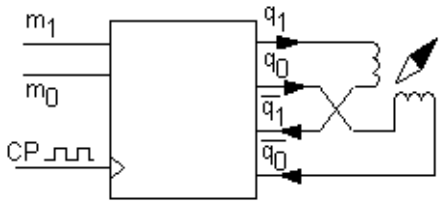


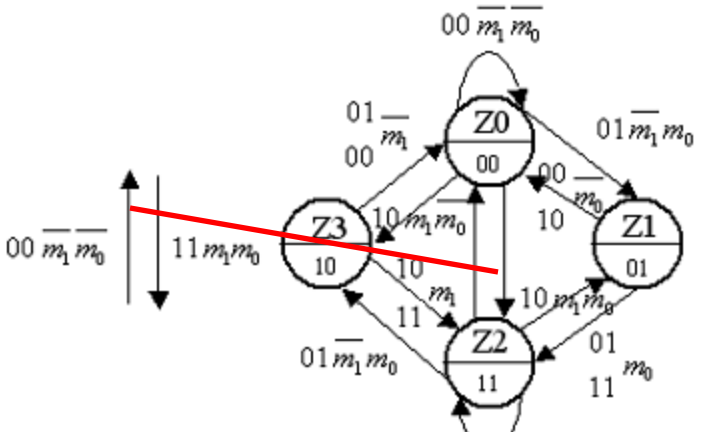


Ibland skriver man boolska vilkor i stället för siffror vid pilarna. I figuren används både vilkor och siffror.

William Sandqvist william@kth.se

10.9 Tillståndstabell och tillståndsavkodare





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

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

q_1^+

0	0	1	1
0	1	1	0
0	1	1	0
0	0	1	1

q_0^+

0	1	1	0
0	1	1	0
0	0	1	1
0	0	1	1

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

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

William Sandqvist william@kth.se

BV 10.5

One aproach for implementing integer division is to perform repeated subtraction as indicated in pseudo-code.

```
Q = 0;
R = A
While ((R – B) ≥ 0) do
R = R – B;
Q = Q + 1;
End while;
```

a) Give an ASM chart that represents the pseudo-code.
b) Show the datapath circuit corresponding to part (a).
c) Give the ASM chart for the control circuit corresponding to part (b).

William Sandqvist william@kth.se

Algorithmic State Machine

ASM metoden består av följande steg:

1. Skapa en algoritm, med *pseudokod*, som beskriver kretsens önskade funktion.

2. Omvandla pseudokoden till ett *ASM diagram*.

3. Designa ett *Dataflödes-schema* (*datapath*) utifrån ASM diagrammet.

4. Skapa ett *detaljerat ASM diagram* utifrån dataflödes-schemat.

5. Designa styrlogik utifrån det detaljerade ASM-diagrammet.

William Sandqvist william@kth.se

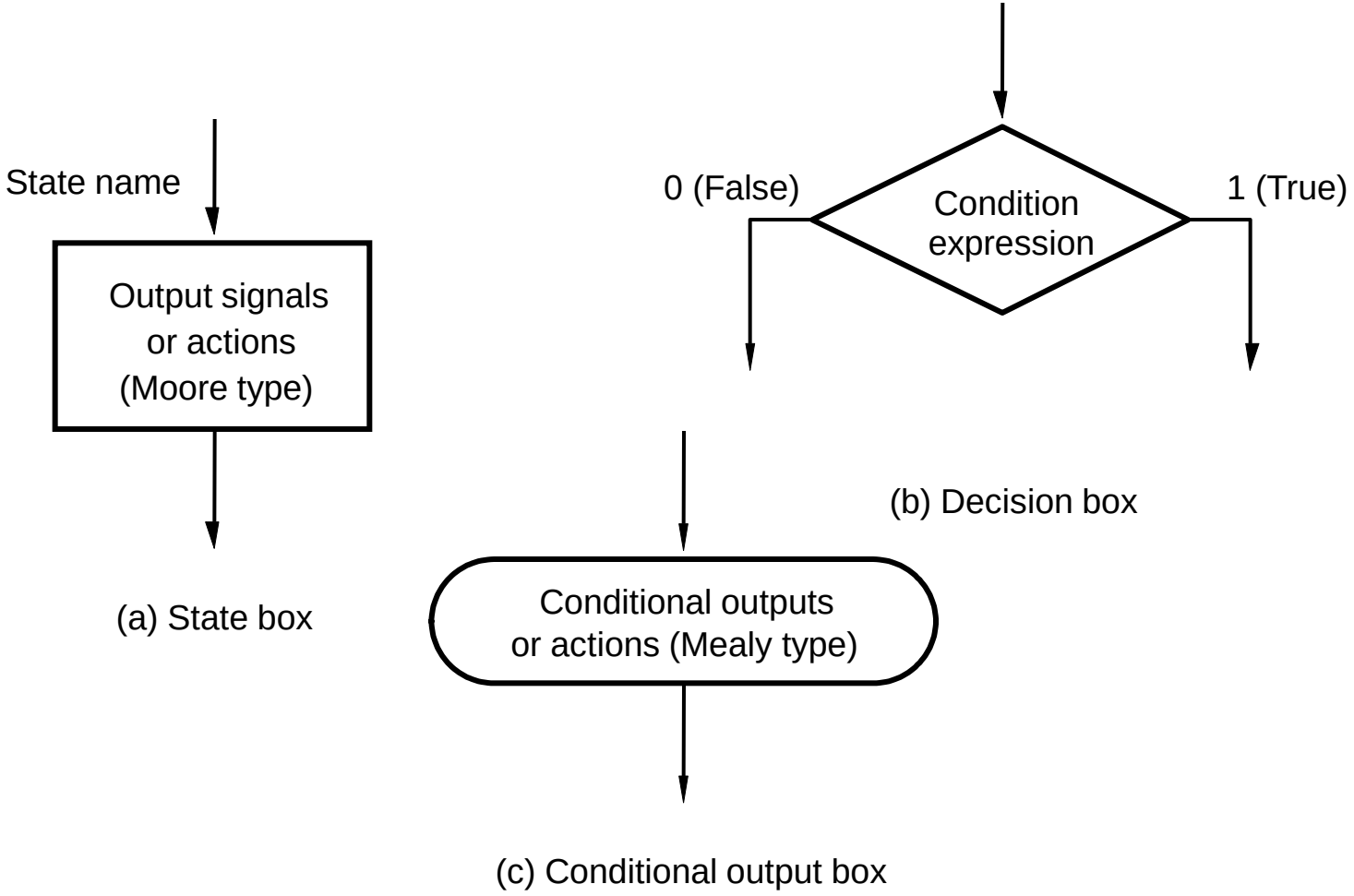


Figure 8.86. Elements used in ASM charts.

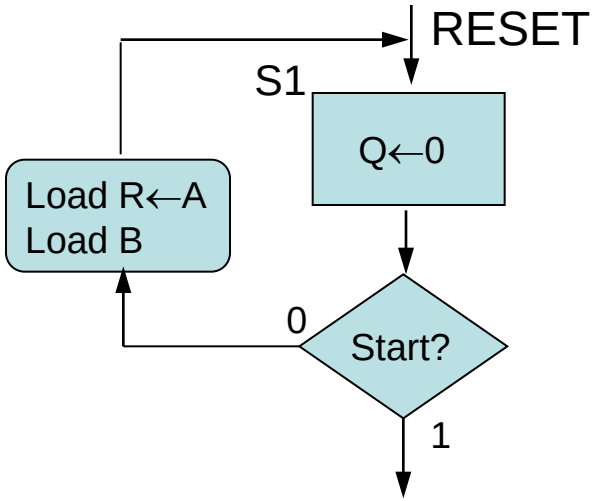
William Sandqvist william@kth.se

BV 10.5 ASM chart

Q = 0;
R = A
While ((R – B) $\geq$ 0) do
R = R – B;
Q = Q + 1;
End while;

William Sandqvist william@kth.se

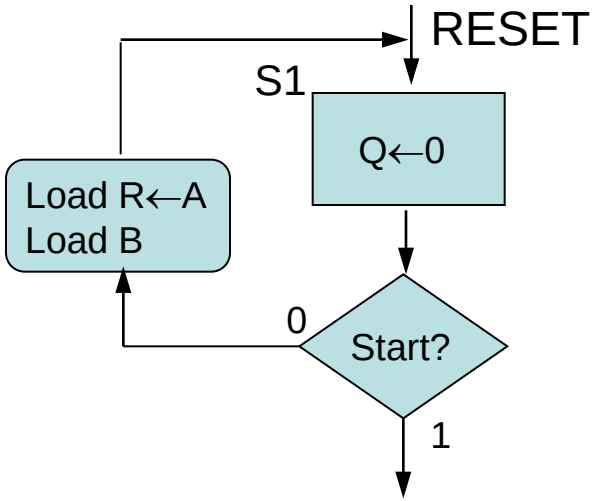
BV 10.5 ASM chart



Q = 0;
R = A
While ((R – B) ≥ 0) do
R = R – B;
Q = Q + 1;
End while;

William Sandqvist william@kth.se

BV 10.5 ASM chart

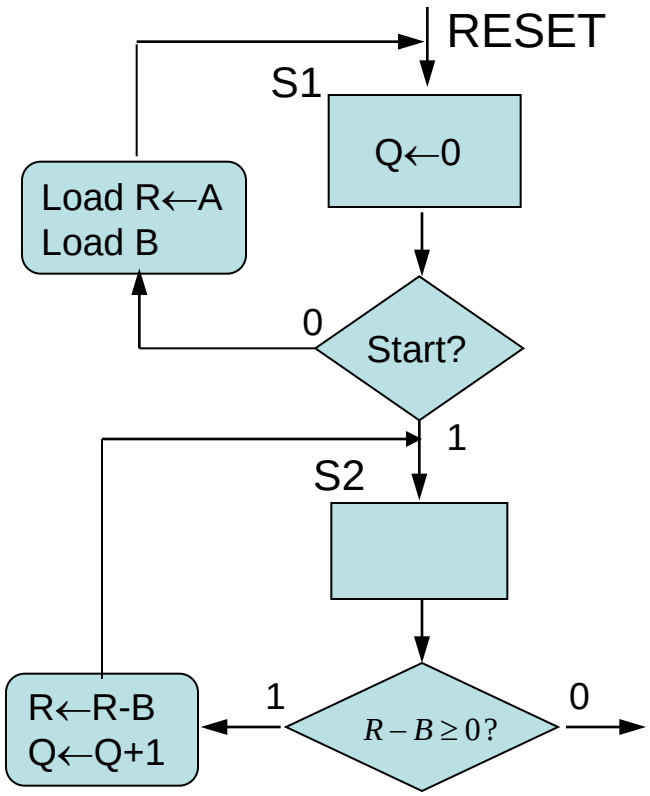


Vänta på start!

Q = 0;
R = A
While ((R – B) ≥ 0) do
R = R – B;
Q = Q + 1;
End while;

William Sandqvist william@kth.se

BV 10.5 ASM chart



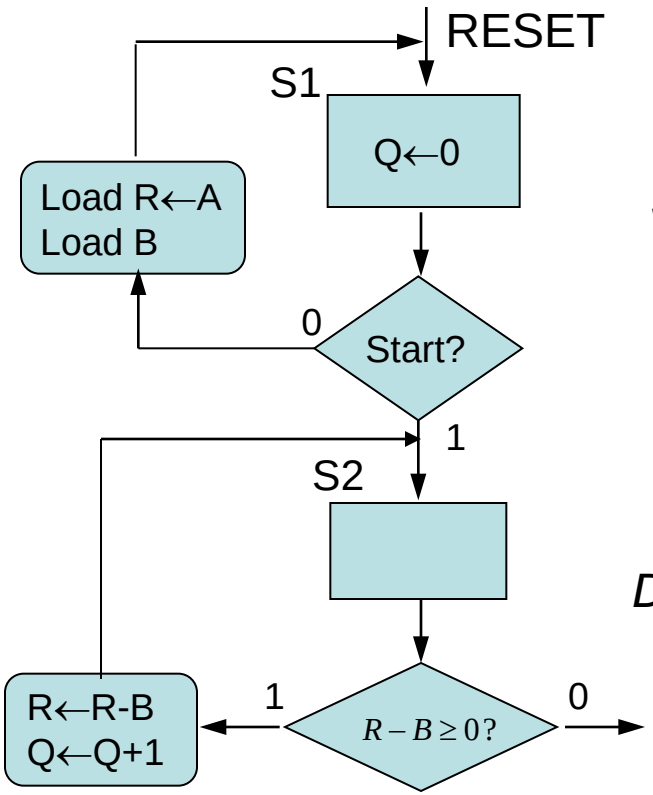
Vänta på start!

Q = 0;
R = A
While ((R – B) ≥ 0) do
R = R – B;
Q = Q + 1;
End while;

William Sandqvist william@kth.se

BV 10.5 ASM chart

Q = 0;
R = A
While ((R – B) ≥ 0) do
R = R – B;
Q = Q + 1;
End while;



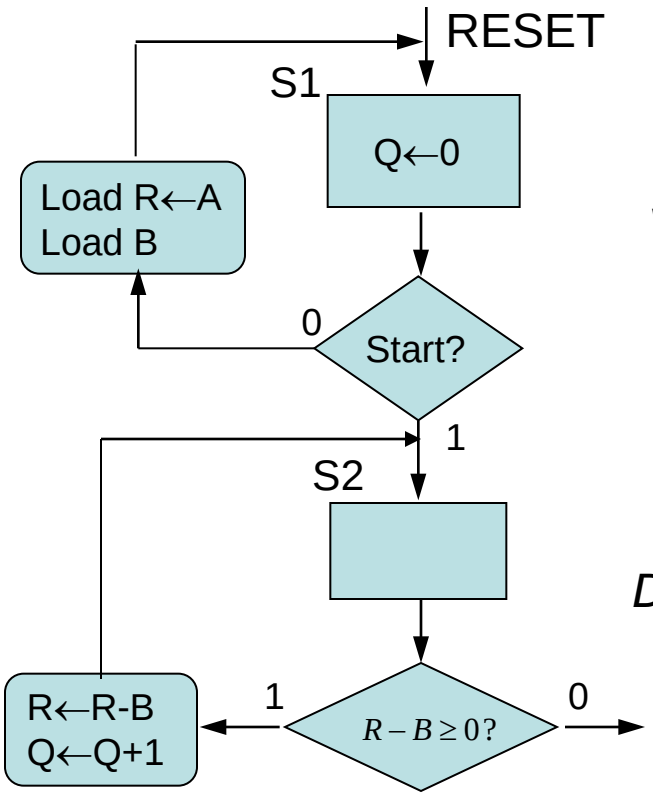
Vänta på start!

Dividera!

William Sandqvist william@kth.se

BV 10.5 ASM chart

Q = 0;
R = A
While ((R – B) ≥ 0) do
R = R – B;
Q = Q + 1;
End while;



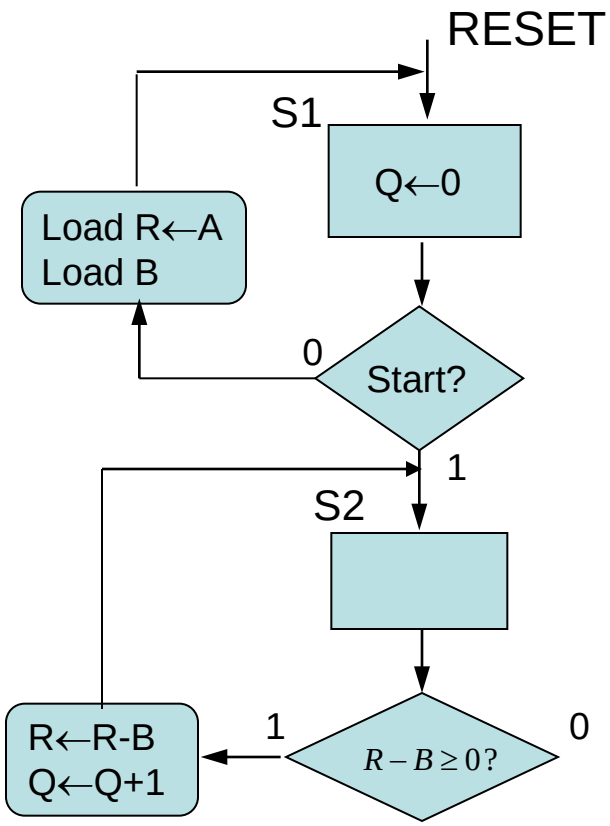
Vänta på start!

Dividera!

Färdigt?

William Sandqvist william@kth.se

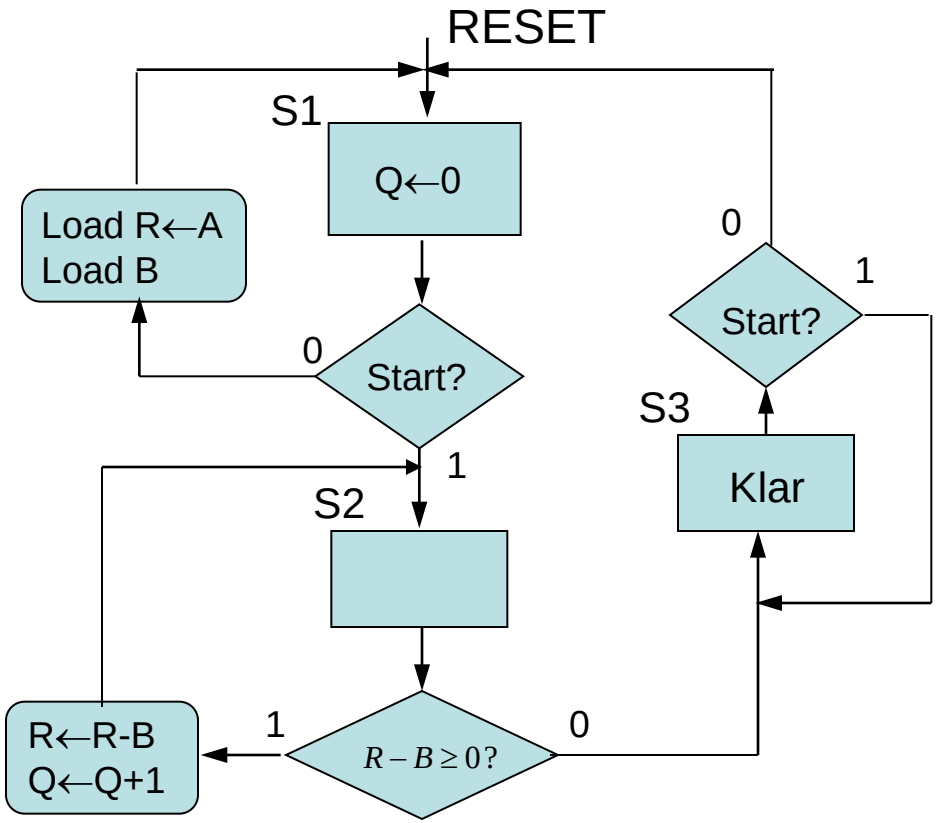
BV 10.5 ASM chart




```
Q = 0;
R = A
While ((R – B) ≥ 0) do
R = R – B;
Q = Q + 1;
End while;
```

William Sandqvist william@kth.se

BV 10.5 ASM chart

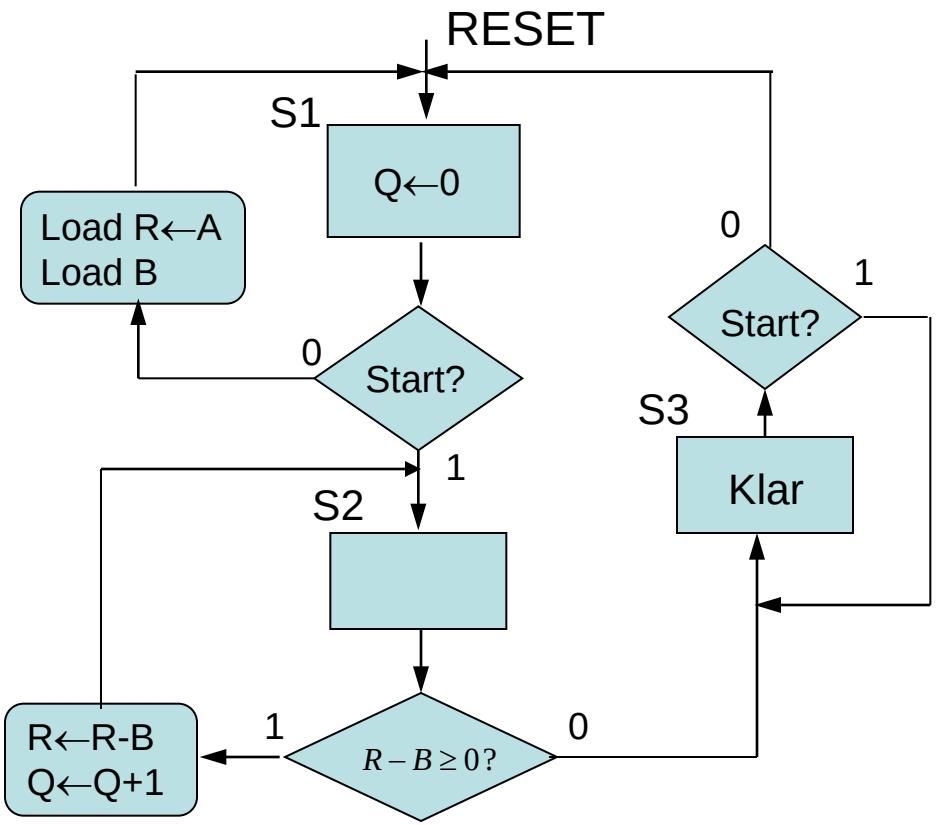


Q = 0;
R = A
While ((R – B) ≥ 0) do
R = R – B;
Q = Q + 1;
End while;

William Sandqvist william@kth.se

BV 10.5 ASM chart

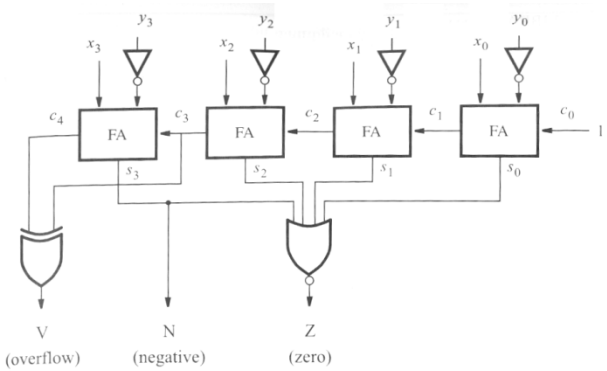
Q = 0;
R = A
While ((R – B) ≥ 0) do
R = R – B;
Q = Q + 1;
End while;



Vänta på att Start släpps, för omstart.

William Sandqvist william@kth.se

Kommer Du ihåg?



$$X - Y$$

$$V = c_4 \oplus c_3 \qquad N = s_3$$

$$Z = \overline{(s_3 + s_2 + s_1 + s_0)}$$

$$X = Y \quad \Rightarrow \quad Z = 1$$

$$X < Y \quad \Rightarrow \quad N \oplus V$$

$$X \leq Y \quad \Rightarrow \quad Z + N \oplus V$$

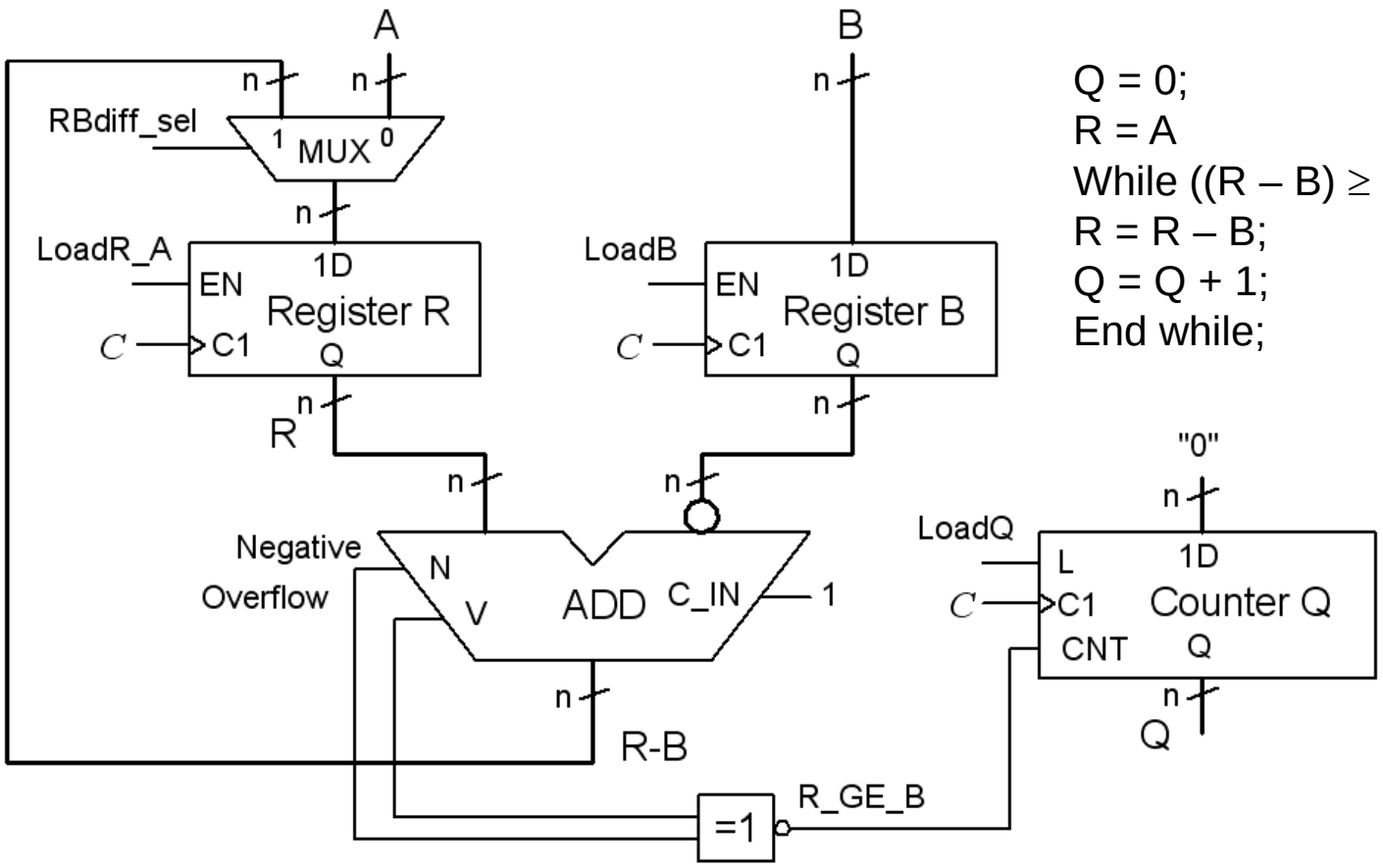
$$X > Y \quad \Rightarrow \quad \overline{Z + N \oplus V} = \overline{Z} \cdot (\overline{N \oplus V})$$

$$X \geq Y \quad \Rightarrow \quad \overline{N \oplus V}$$

Så här kan en dator göra de vanligaste jämförelserna …

William Sandqvist william@kth.se

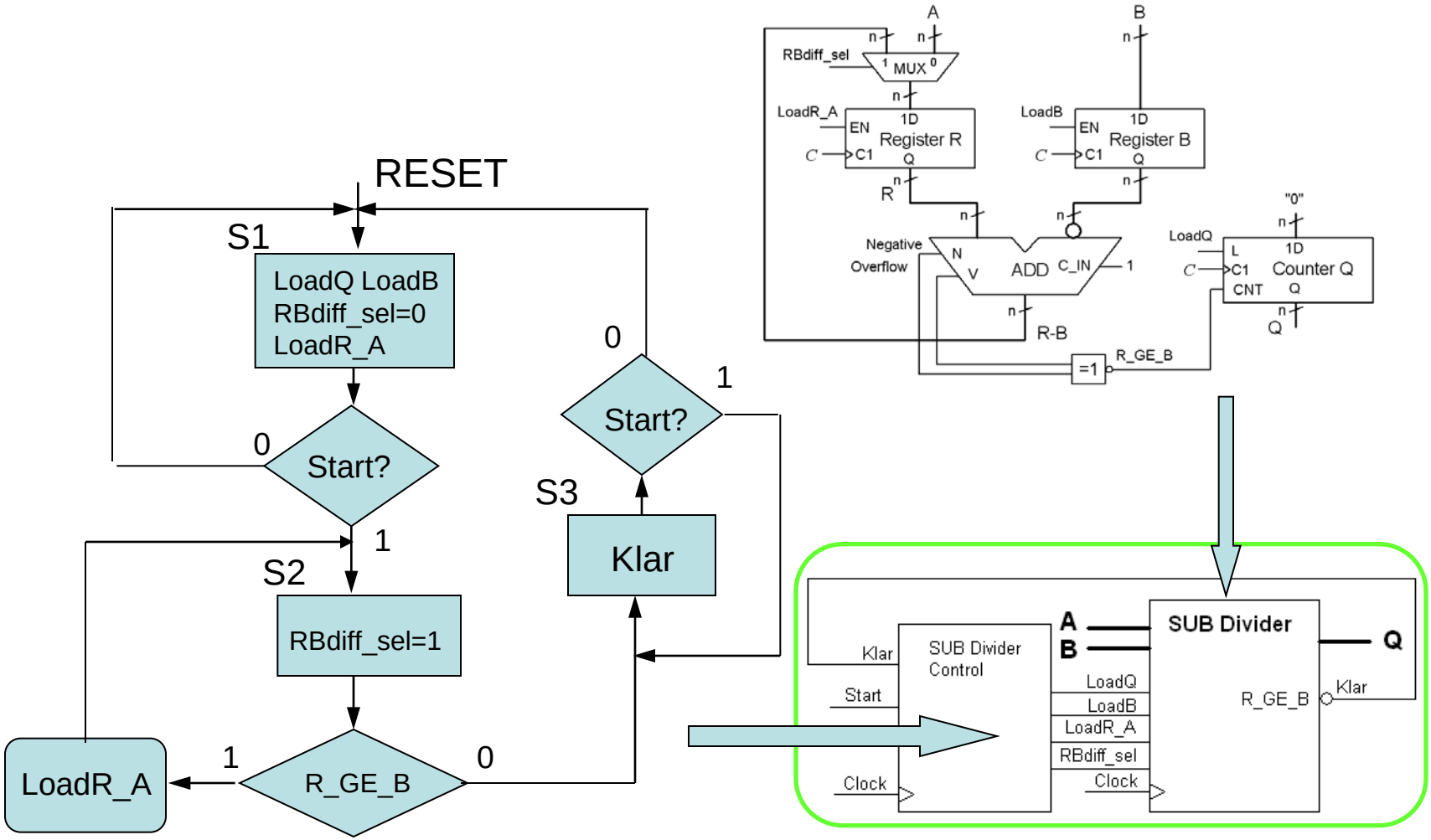
BV 10.5 datapath circuit



Q = 0;
R = A
While ((R – B) ≥ 0) do
R = R – B;
Q = Q + 1;
End while;

William Sandqvist william@kth.se

BV 10.5 ASM control



William Sandqvist william@kth.se

William Sandqvist william@kth.se