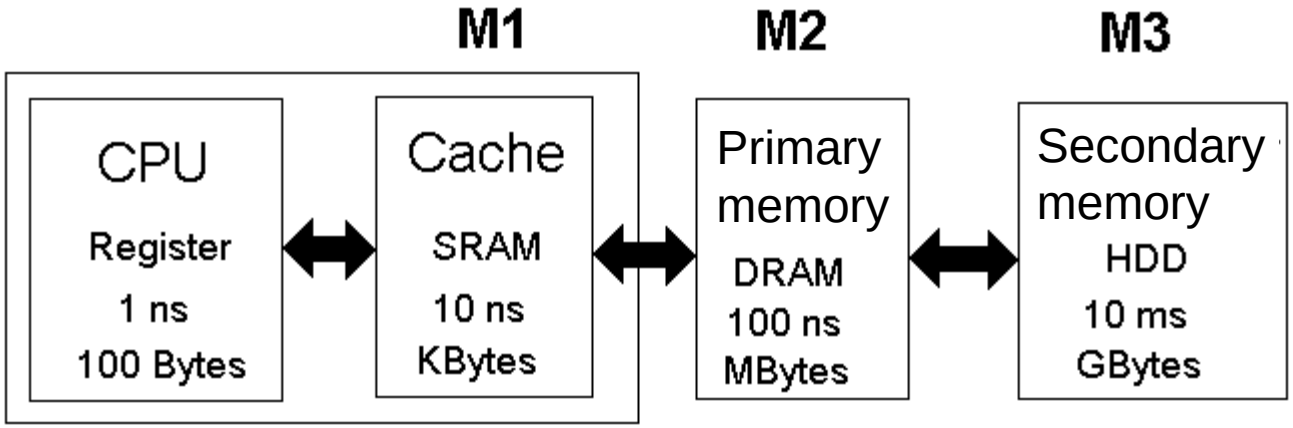
Memory technologies

Technologi	Access time	Cost $/GB
SRAM	1 ns	1000
DRAM	50 ns	100
HDD	10 ms	1

Fast memory is expensive and inexpensive memories are slow!

Principal figures.

Memory Hierarchy



A three-level memory hierarchy. The faster memory types are used as "buffers" against the slower.

Principle

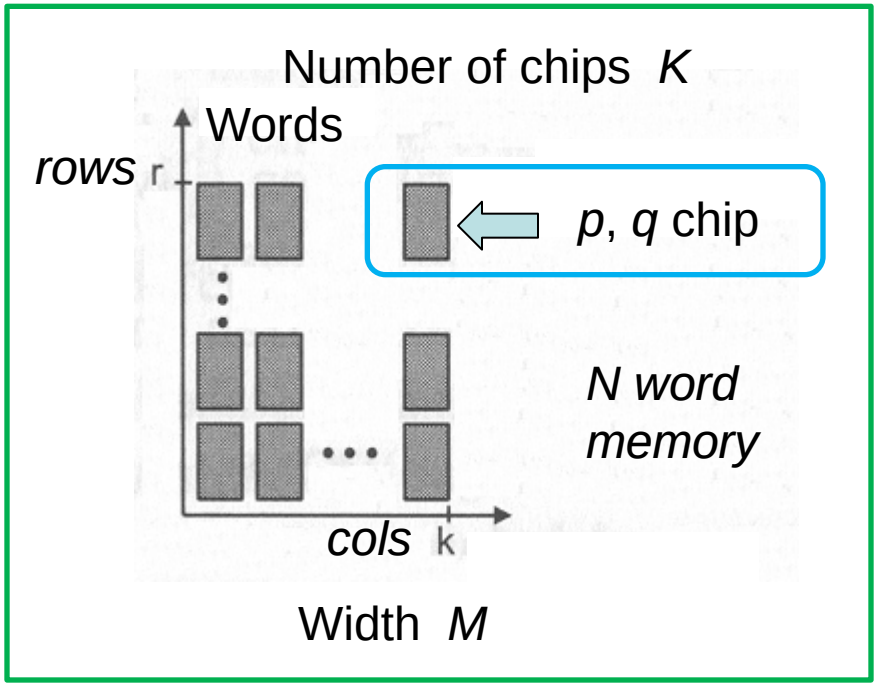
Memory and memory chips

Memory:
N words, width ***M*** bits

Memorychip:
p words, width ***q*** bits

- Number of rows $\boldsymbol{r} \leq N/p$
- Number of columns $\boldsymbol{k} \geq M/q$
- Number of chips $\boldsymbol{K} = r \times k$

$$K = r \times k$$

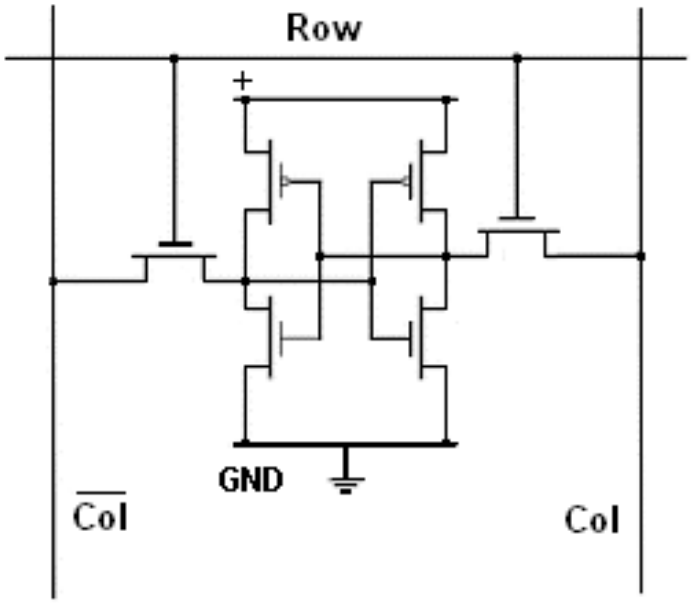


N, *M* - *memory*

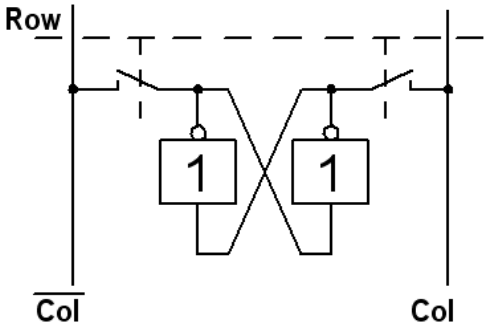
William Sandqvist william@kth.se

SRAM

Each bit in a CMOS SRAM consists of a latch circuit made up of six MOS transistors.



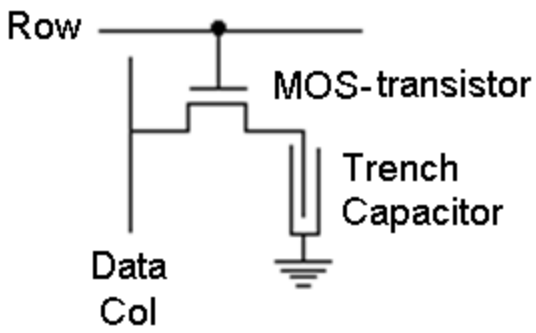
The memory cell is basically a SR-latch.

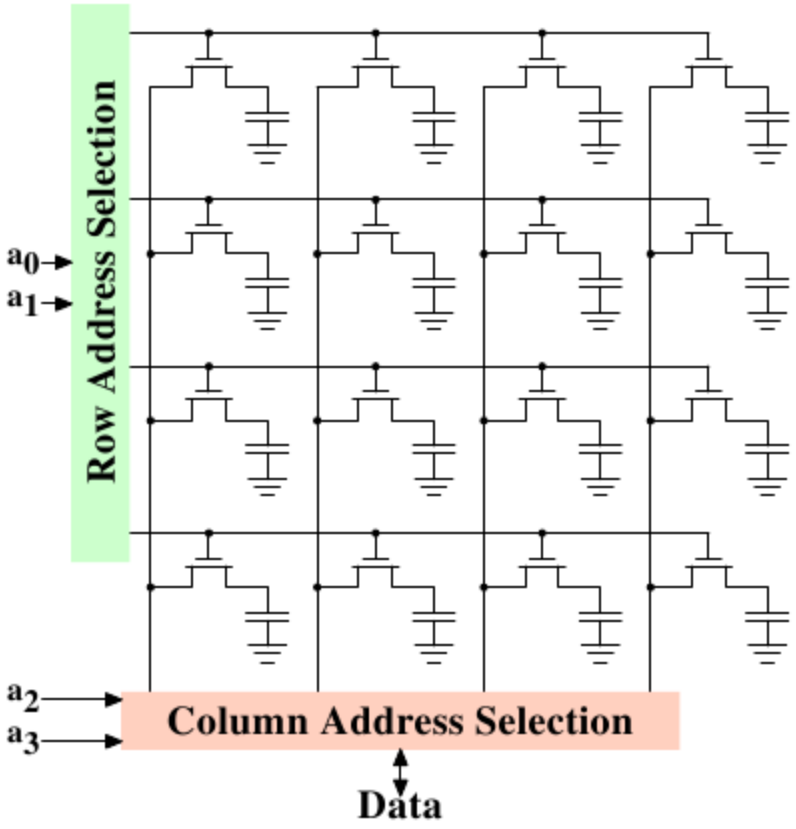


DRAM

Each bit in a DRAM consists of a transistor and a capacitor.

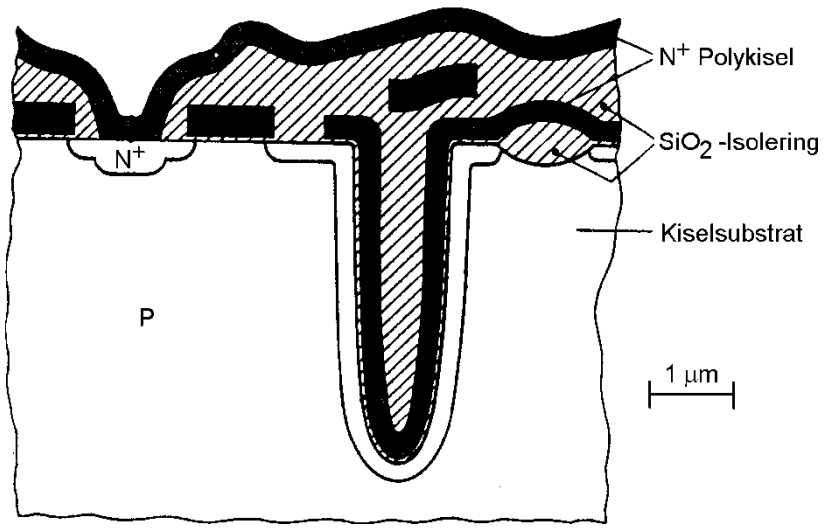
A charged capacitor leaks charge after a while. Periodically, all the capacitors must be searched and those who have charge left must then be reloaded. This is called **Refresh**. It is managed by circuitry within the memory.

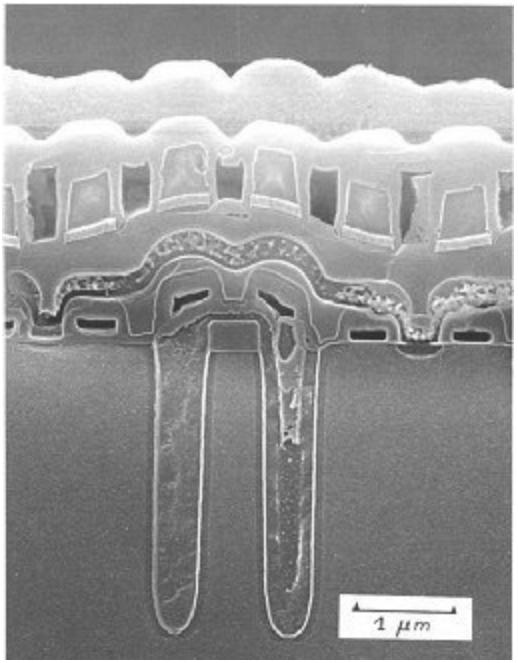




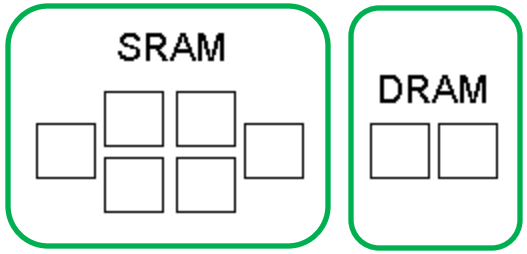
The capacitor is built on the depth

Trench Capacitor

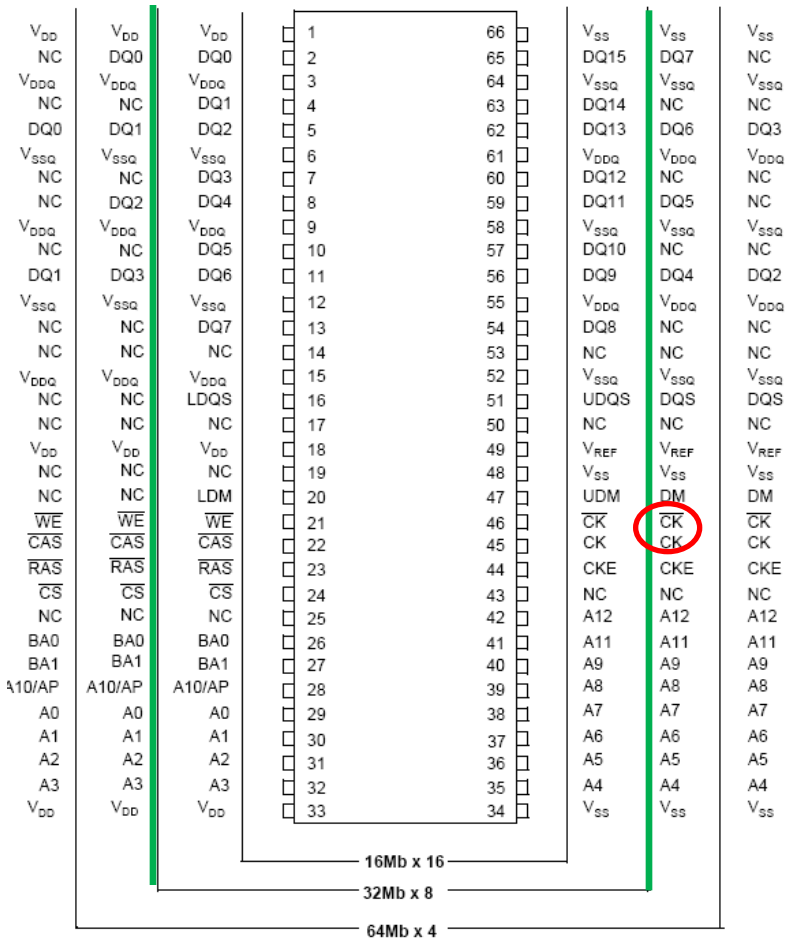




One bit in a DRAM takes the same place as two MOS transistors. One bit in the SRAM as six MOS transistors!



Infineon HYB25D25640 256 Mbit SDRAM



Chip 256Mbit (32M×8)

Synchronously, using the bus clock. Double-edge triggered for double data rate ck + $\overline{ck}$ (even lower power).

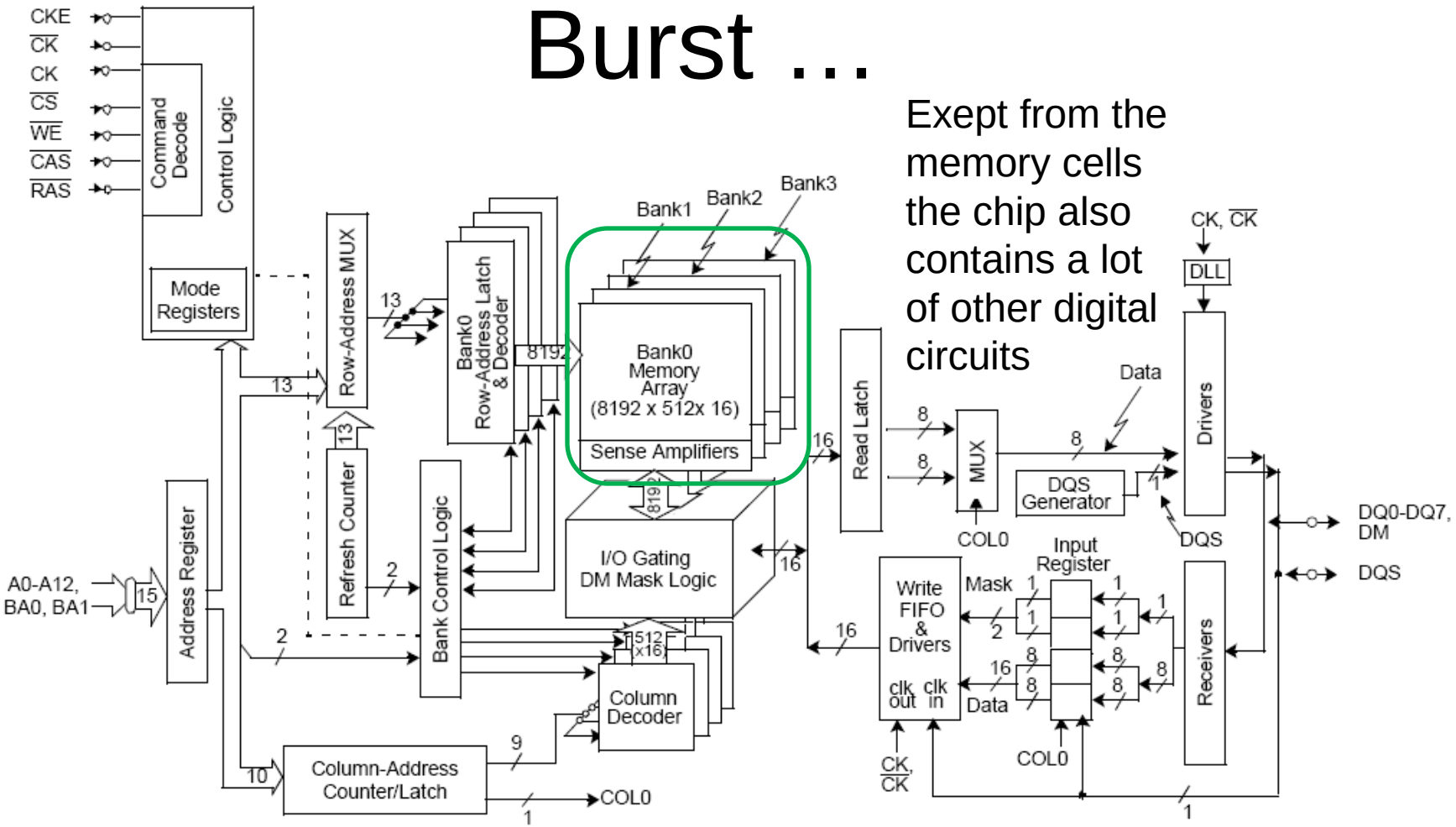
32M $2^5 \times 2^{20} = 2^{25}$, 25 address bits used. Time-multiplexed addressing, 13-bit RAS (row), 10 bit CAS (columns), two bank bits BA0 and BA1.

Burst can be 2, 4, 8 Bytes.

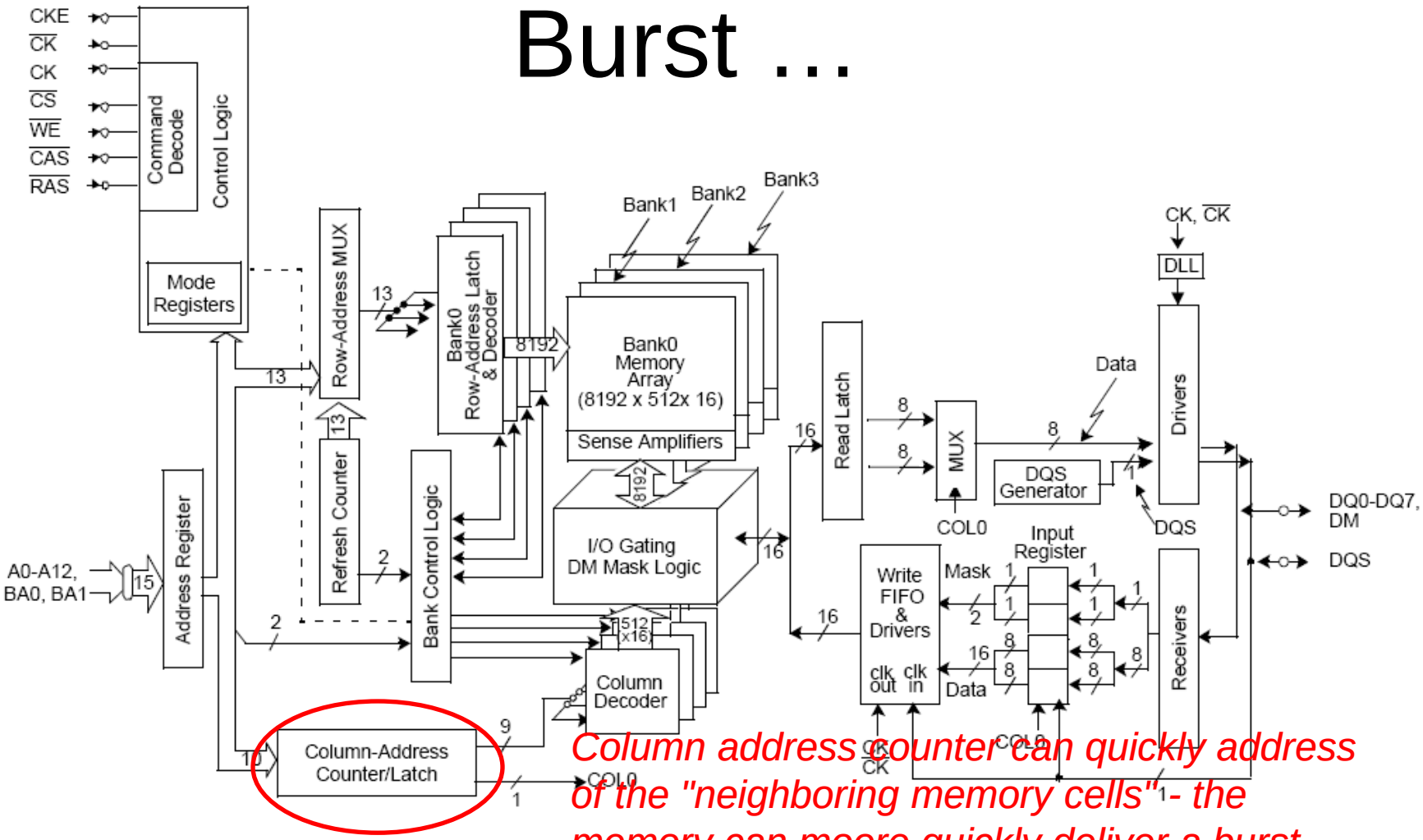
William Sandqvist william@kth.se

Burst ...

Exept from the memory cells the chip also contains a lot of other digital circuits



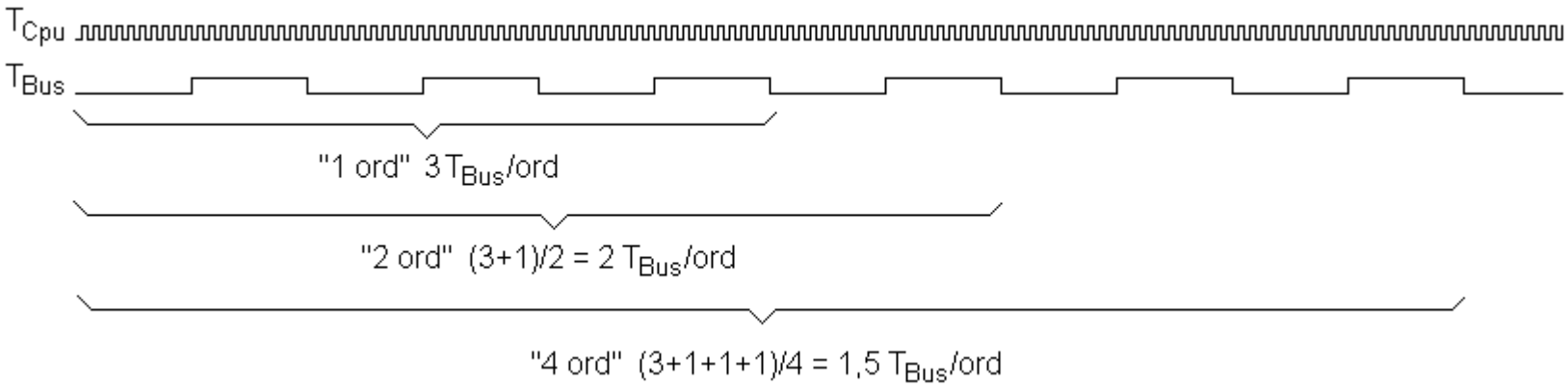
Burst ...



Column address counter can quickly address of the "neighboring memory cells"- the memory can moore quickly deliver a burst with several bytes in sequence, than an totaly random acess.

William Sandqvist william@kth.se

Burst provides faster average access



- To access **1** ”random” word in the memory takes three buscycles **3**T_{Bus}/word (2 T_{BUS} are Waitstates)
- To access a ”Burst” of **2** words takes 3+1 buscycles, 4/2 = **2**T_{Bus}/word
- To access a ”Burst” of **4** words takes 3+1+1+1 buscycles, 6/4 = **1,5**T_{Bus}/word
- To acess a ”Burst” of **8** words takes 3+1+1+1+1+1+1+1 cycles, 10/8 = **1,25**T_{Bus}/word

It's important to have proper use of all fetched words - otherwise you are wasting bus clock cycles with the Burst method!

More about this in the Computer Organization course, when reading about caches.

Ex **12.1** Dynamic Memory

Chip
256Mbit (32M×8)

a) How many chips are needed for 256M×64?

Ex **12.1** Dynamic Memory

Chip
256Mbit (32M×8)

a) How many chips are needed for 256M×64?

Memory N = 256M M = 64 bits. **Chip** p = 32M q = 8 bits.

Number of columns $k = M/q = 64/8 = 8$.

Number of rows $r = N/p = 256M/32M = 8$.

Number of chips $K = r \times k = 8 \times 8 = 64$.

512M×72 ?

Chip
256Mbit (32M×8)

b) How many chips are needed for 512M×72?

512M×72 ?

Chip
256Mbit (32M×8)

b) How many chips are needed for 512M×72?

Memory N = 512M M = 72 bits. **Chip** p = 32M q = 8 bits.

Number of columns $k = M/q = 72/8 = 9$.

Number of rows $r = N/p$ = 512M/32M = 16.

Number of chips $K = r \times k = 9 \times 16 = 144$.

512M×72 ?

Chip
256Mbit (32M×8)

b) How many chips are needed for 512M×72?

Memory N = 512M M = 72 bits. **Chip** p = 32M q = 8 bits.

Number of columns $k = M/q = 72/8 = 9$.

Number of rows $r = N/p$ = 512M/32M = 16.

Number of chips $K = r \times k = 9 \times 1\ 6 = 144$.

The "unusual" bit width 72 (= 64 + 8). The 8 extra bits are used for correcting single faults, and to detect double faults.

- (In this way, even capsules small errors could be used as the error can be corrected. They would otherwise have to be discarded).

512M×72 ?

Chip
256Mbit (32M×8)

b) How many chips are needed for 512M×72?

Memory N = 512M M = 72 bits. **Chip** p = 32M q = 8 bits.

Number of columns $k = M/q = 72/8 = 9$.

Number of rows $r = N/p$ = 512M/32M = 16.

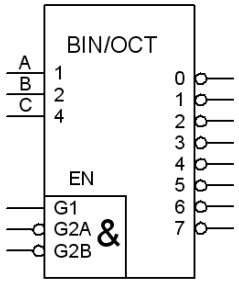
Number of chips $K = r \times k = 9 \times 1\ 6 = 144$.

The "unusual" bit width 72 (= 64 + 8). The 8 extra bits are used for correcting single faults, and to detect double faults.

(In this way, even capsules small errors could be used as the error can be corrected. They would otherwise have to be discarded).

- Or will a expensive memory be good even if some of the memory cells "wear out" over time.

Ex **12.2** ROM and SRAM



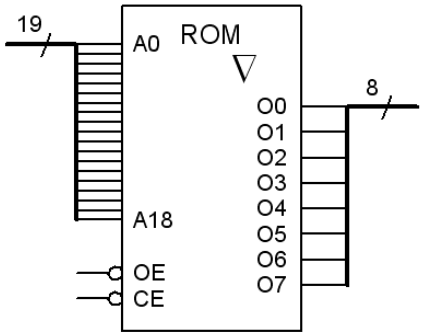
Decoder 3-to-8

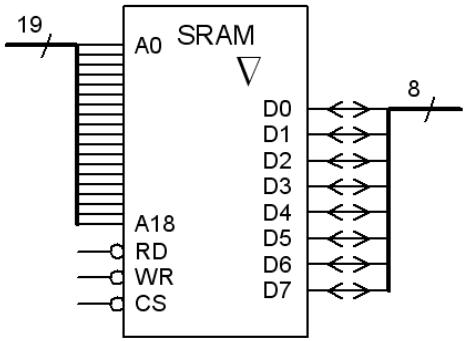
ROM:

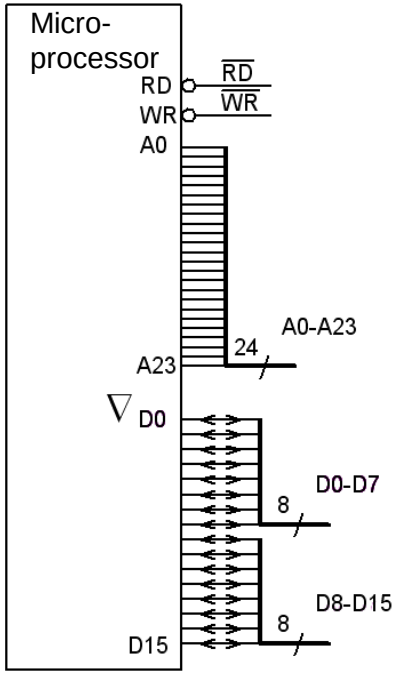
ROM 4M 512k × 8 bit

RAM:

SRAM 4M 512k × 8 bit



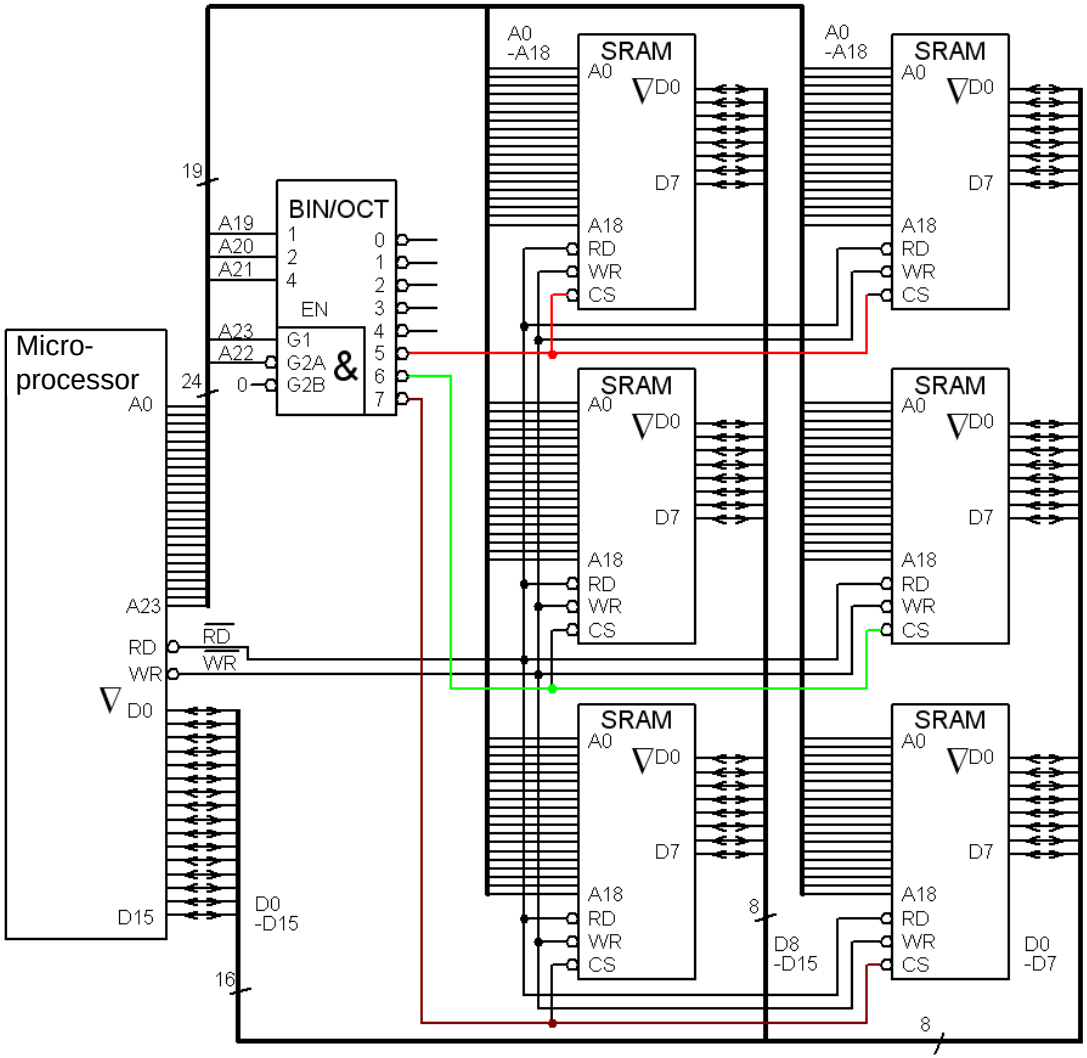




Suppose that the ROM and the SRAM is to be connected to a **16**-bit microprocessor having **24** bit addressing.

William Sandqvist william@kth.se

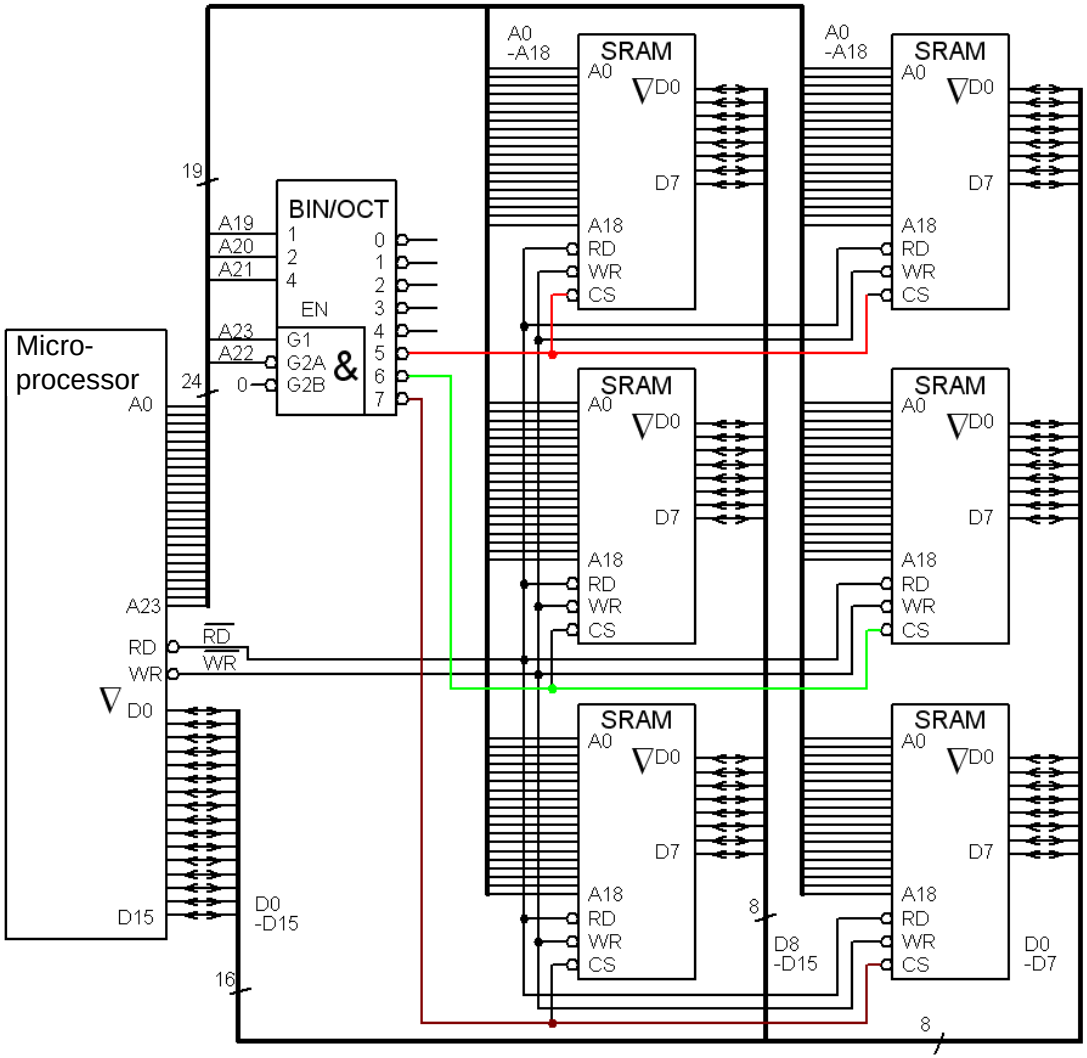
SRAM size?



How big is the figure SRAM, and which is the address area expressed in hexadecimal numbers

William Sandqvist william@kth.se

SRAM size?



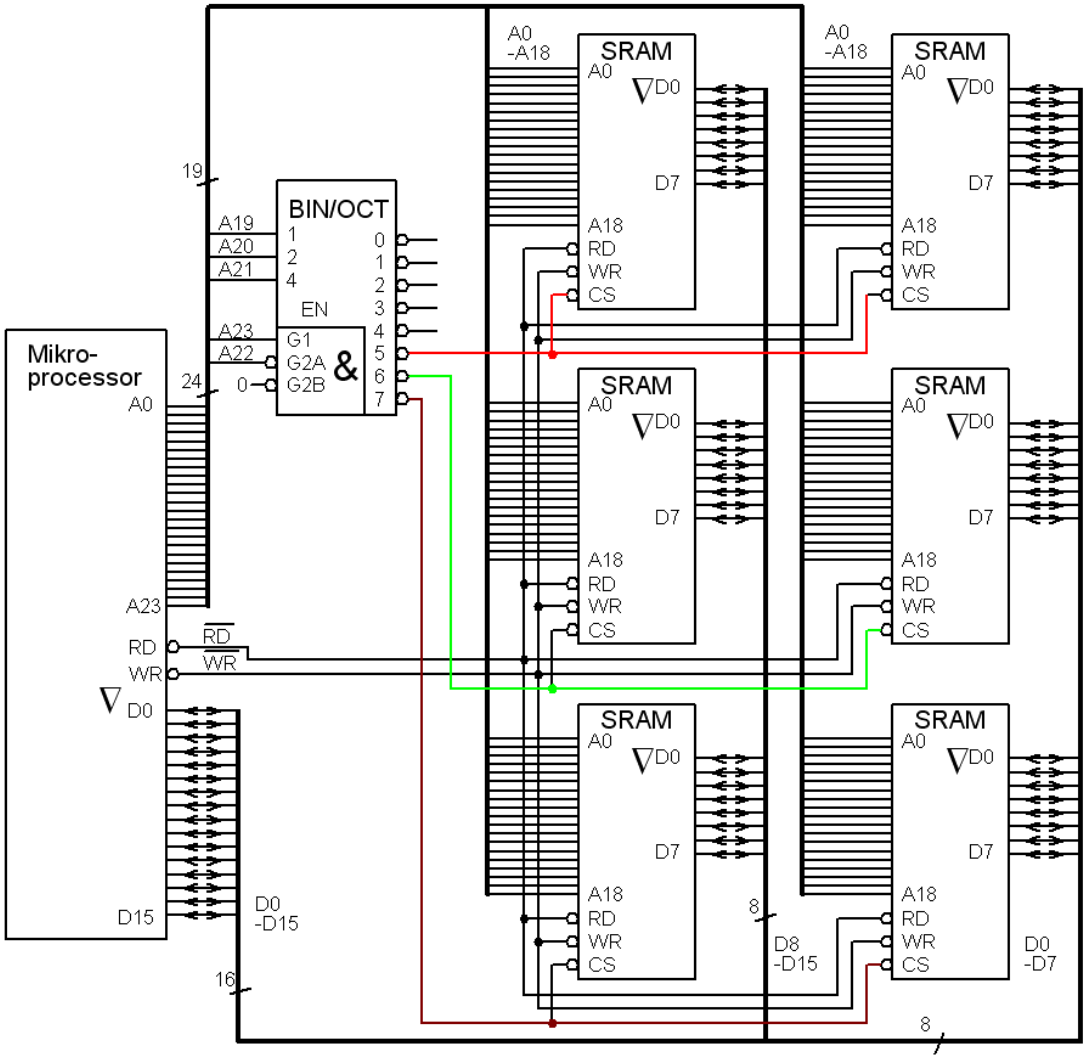
How big is the figure SRAM, and which is the address area expressed in hexadecimal numbers

- **Chip**:
  p = 512k q = 8 bits
- **Memory**:
  $r = 3$ $k = 2$ $K = 2 \times 3 = 6$

$M = k \times q = 2 \times 8 = 16$ bits
$N = \mathrm{p} \times \mathrm{r} = 512\mathrm{k} \times 3 = 1{,}5\mathrm{M}$

William Sandqvist william@kth.se

SRAM Control?



How big is the figure SRAM, and which is the address area expressed in hexadecimal numbers

- **Chip**:
  p = 512k q = 8 bits
- **Memory**:
  $r = 3$ $k = 2$ $K = 2 \times 3 = 6$

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

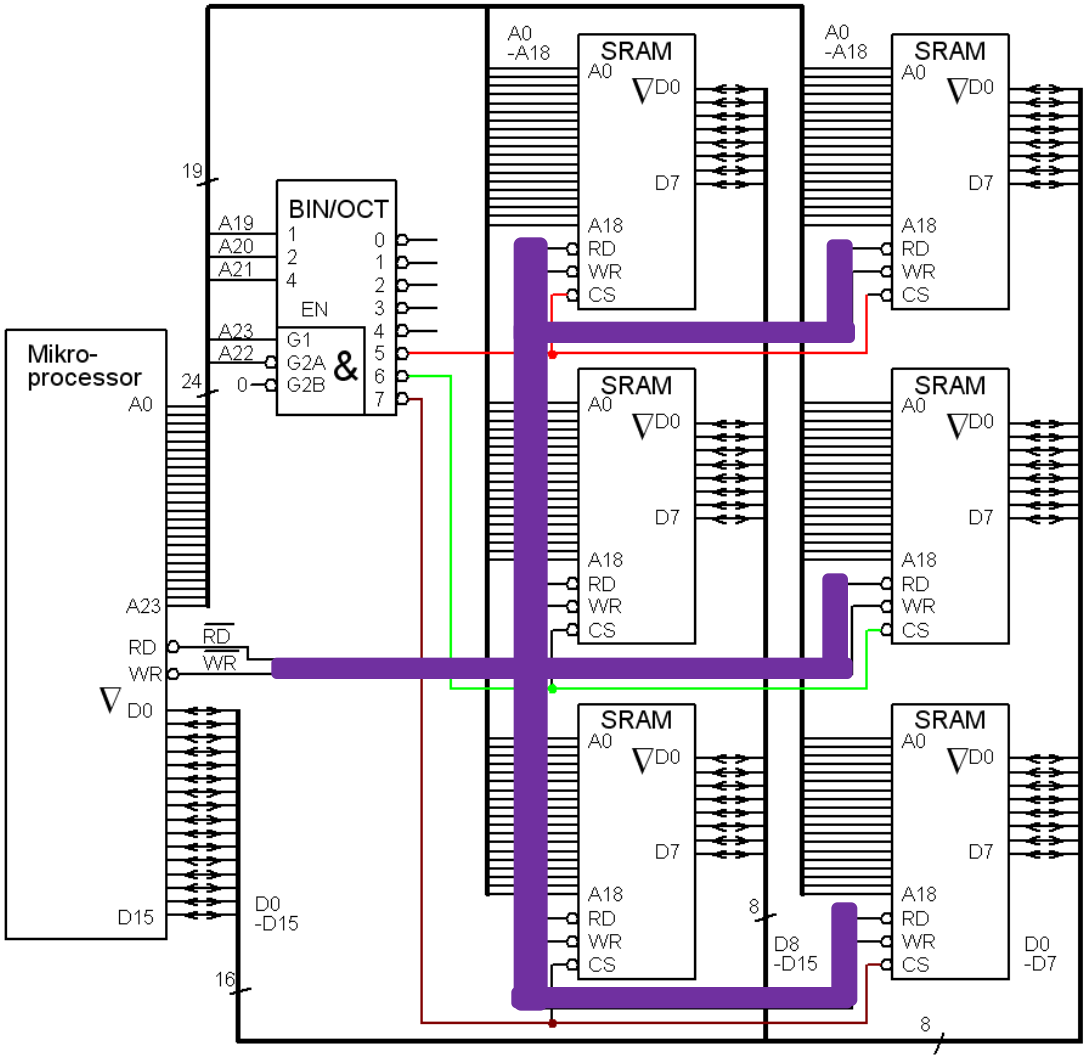
$N = \mathrm{p} \times \mathrm{r} = 512\mathrm{k} \times 3 = 1{,}5\mathrm{M}$

$$\overline{RD} = ?$$

$$\overline{WR} = ?$$

William Sandqvist william@kth.se

SRAM Control?



How big is the figure SRAM, and which is the address area expressed in hexadecimal numbers

- **Chip**:
  p = 512k q = 8 bits
- **Memory**:
  $r = 3$ $k = 2$ $K = 2 \times 3 = 6$

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

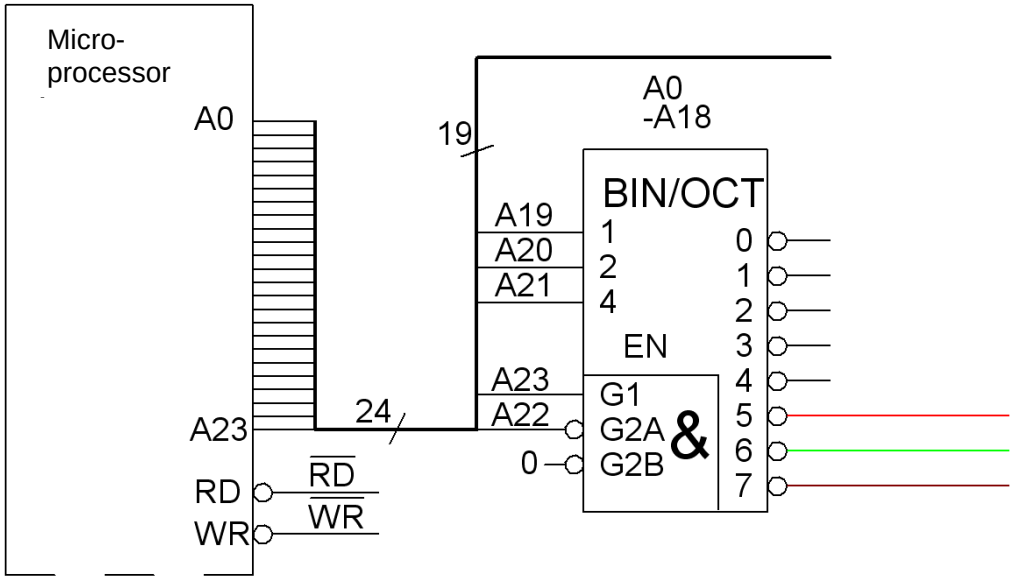
$N = \mathrm{p} \times \mathrm{r} = 512\mathrm{k} \times 3 = 1{,}5\mathrm{M}$

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

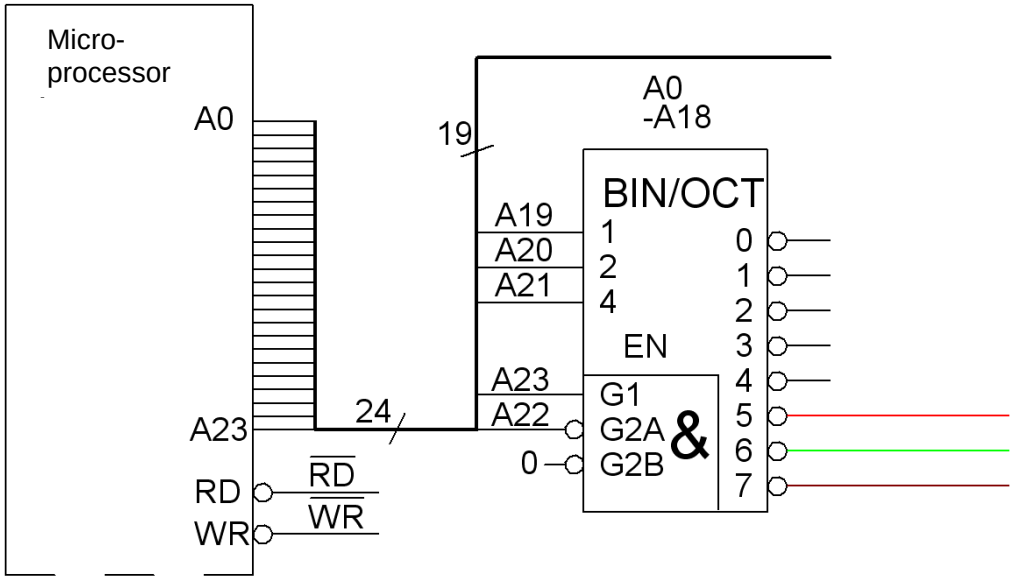
$$\overline{WR} = \overline{WR}$$

William Sandqvist william@kth.se

SRAM address range?

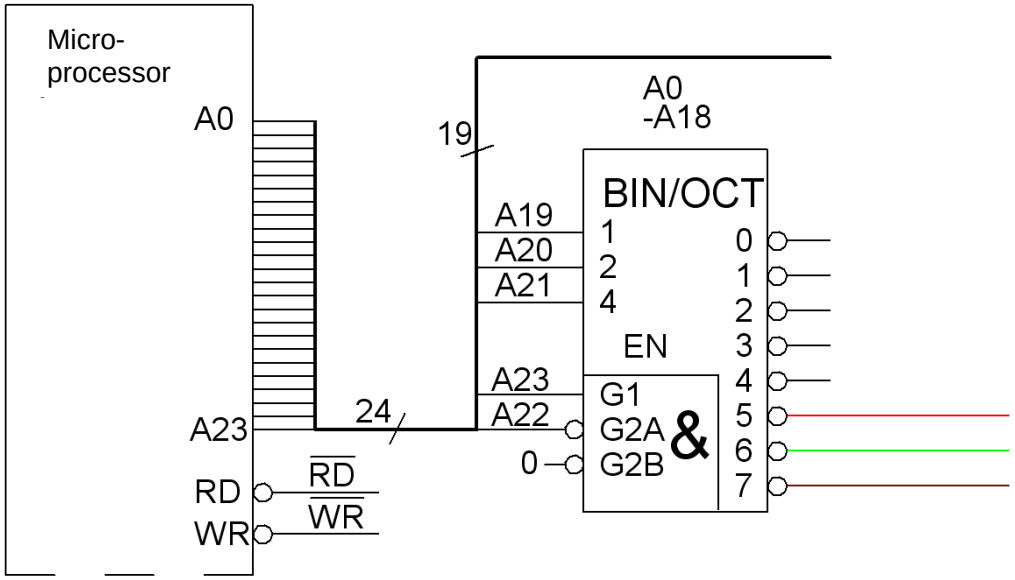


SRAM address range?



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			

SRAM address range?



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			

SRAM address range: A80000 - BFFFFF

William Sandqvist william@kth.se

Change the address range! ?

Change the address range to 980000 – AFFFFF ?

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

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

Change the address range! ?

Change the address range to 980000 – AFFFFF ?

980000

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

AFFFFF

1010|1111|1111|1111|1111|1111|

Change the address range! ?

Change the address range to 980000 – AFFFFF ?

980000

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

AFFFFF

1010|1111|1111|1111|1111|1111|

”10|011” → ”3”

”10|101” → ”5”

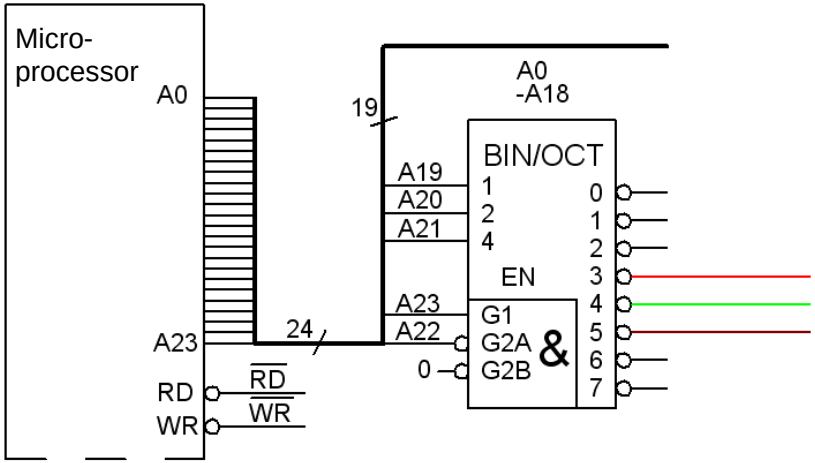
Change the address range! ?

Change the address range to 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

Change the address range! ?

Change the address range to 480000 – 5FFFFF ?

Change the address range! ?

Change the address range to 480000 – 5FFFFF ?

480000

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

5FFFFF

0101|1111|1111|1111|1111|1111|

Change the address range! ?

Change the address range to 480000 – 5FFFFF ?

480000
01**00**|**1**000|0000|0000|0000|0000|

5FFFFF
01**01**|**1**111|1111|1111|1111|1111|

”01|001” → ”1”
”01|011” → ”3”

William Sandqvist william@kth.se

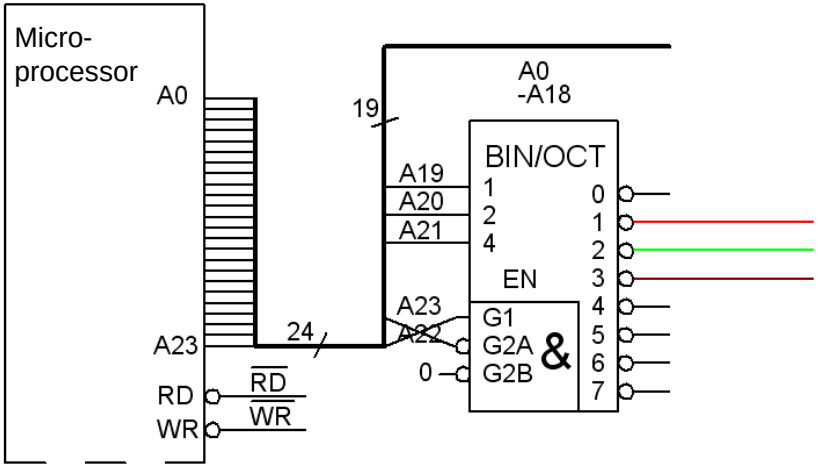
Change the address range! ?

Change the address range to 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

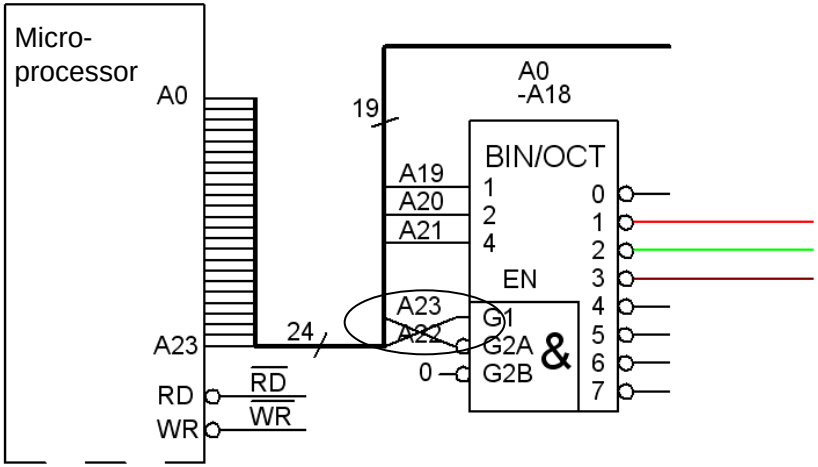
Change the address range! ?

Change the address range to 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…?

Most often a processor reads its first instruction from address 0, then there must be a ROM at that address. Suppose a ROM 2M × 16 bitar address range 000000 … and forward. ROM Chip 512k×8.

- How many chips are needed?
- How is the decoder connected?
- How are the memory chips connected?
- Which is the address area for the ROM expressed in hexadecimal numbers.

ROM 00 00 00…?

Most often a processor reads its first instruction from address 0, then there must be a ROM at that address. Suppose a ROM 2M $\times$ 16 bit address range 000000 … and forward. ROM Chip 512k$\times$8.

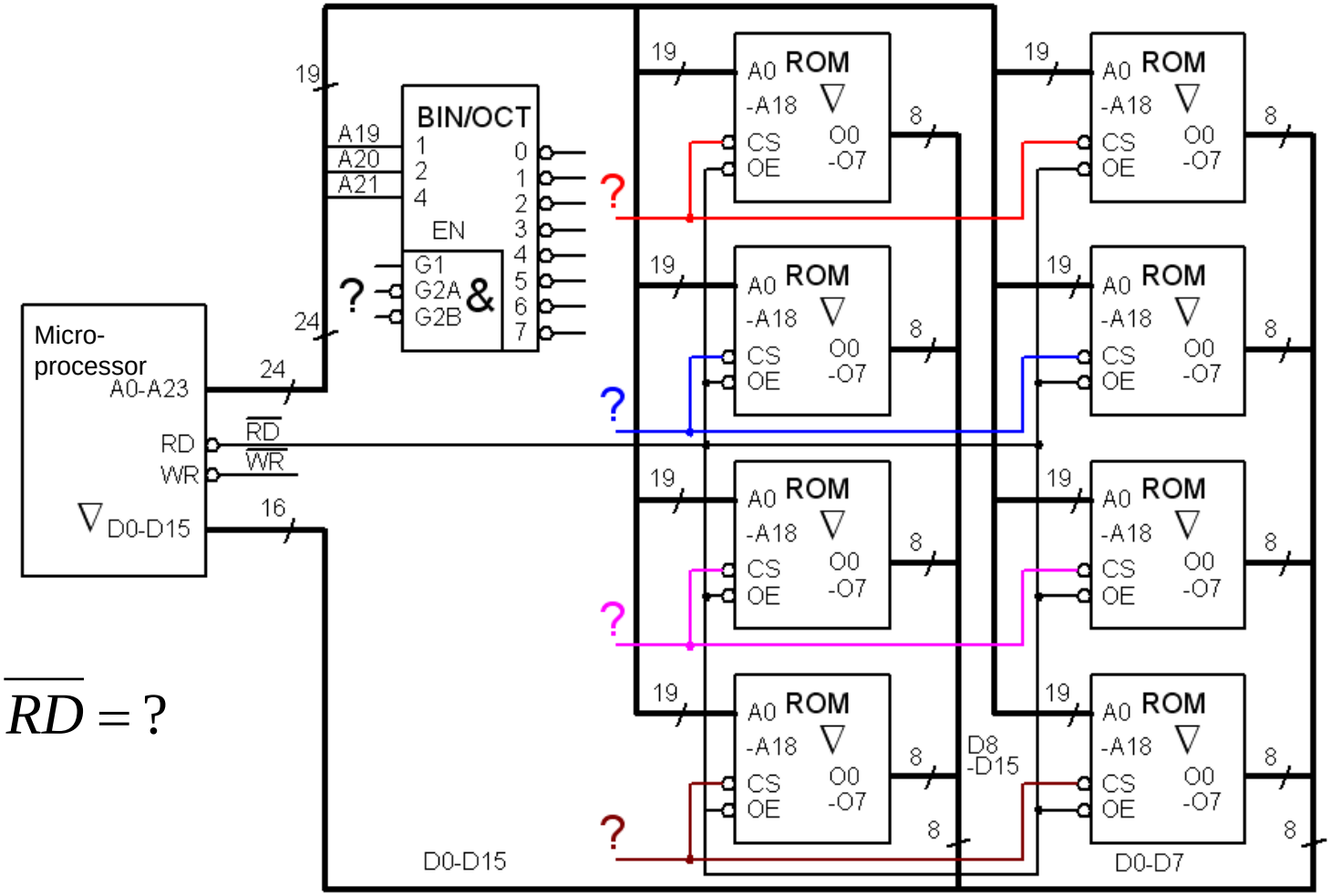
- How many chips are needed?
- How is the decoder connected?
- How are the memory chips connected?
- Which is the address area for the ROM expressed in hexadecimal numbers.

Memory:
N = 2 M (4·512k) word is M = 16 bitar

Memory chip:
p = 512 k word is q = 8 bitar

- Number of rows $r \leq N/p$ = 4·512k/512k = 4
- Number of columns $k \geq M/q$ = 16/8 = 2
- Number of chips $K = r \times k = 4 \times 2 =$ **8**

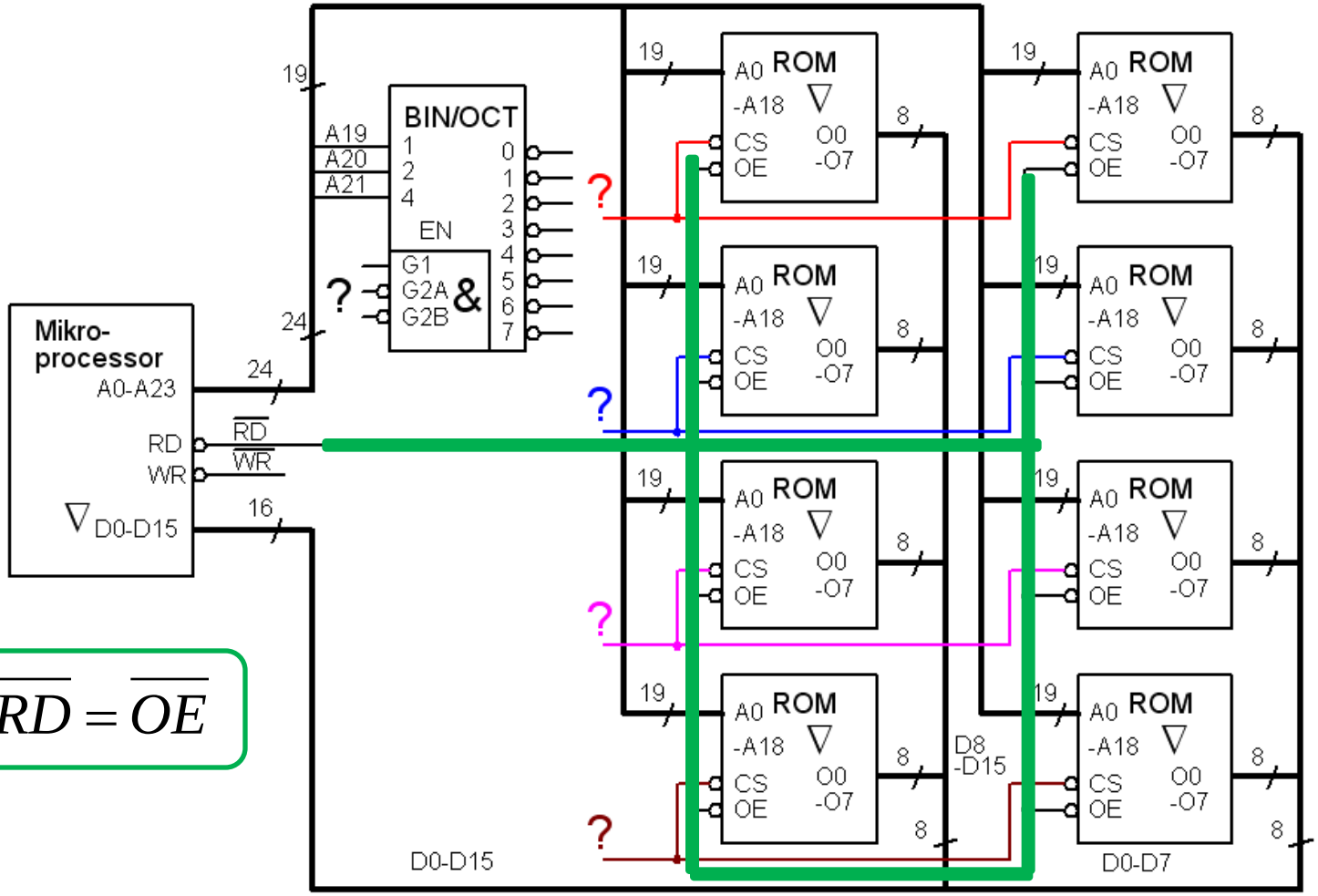
ROM Control connections?



$$\overline{RD} = ?$$

William Sandqvist william@kth.se

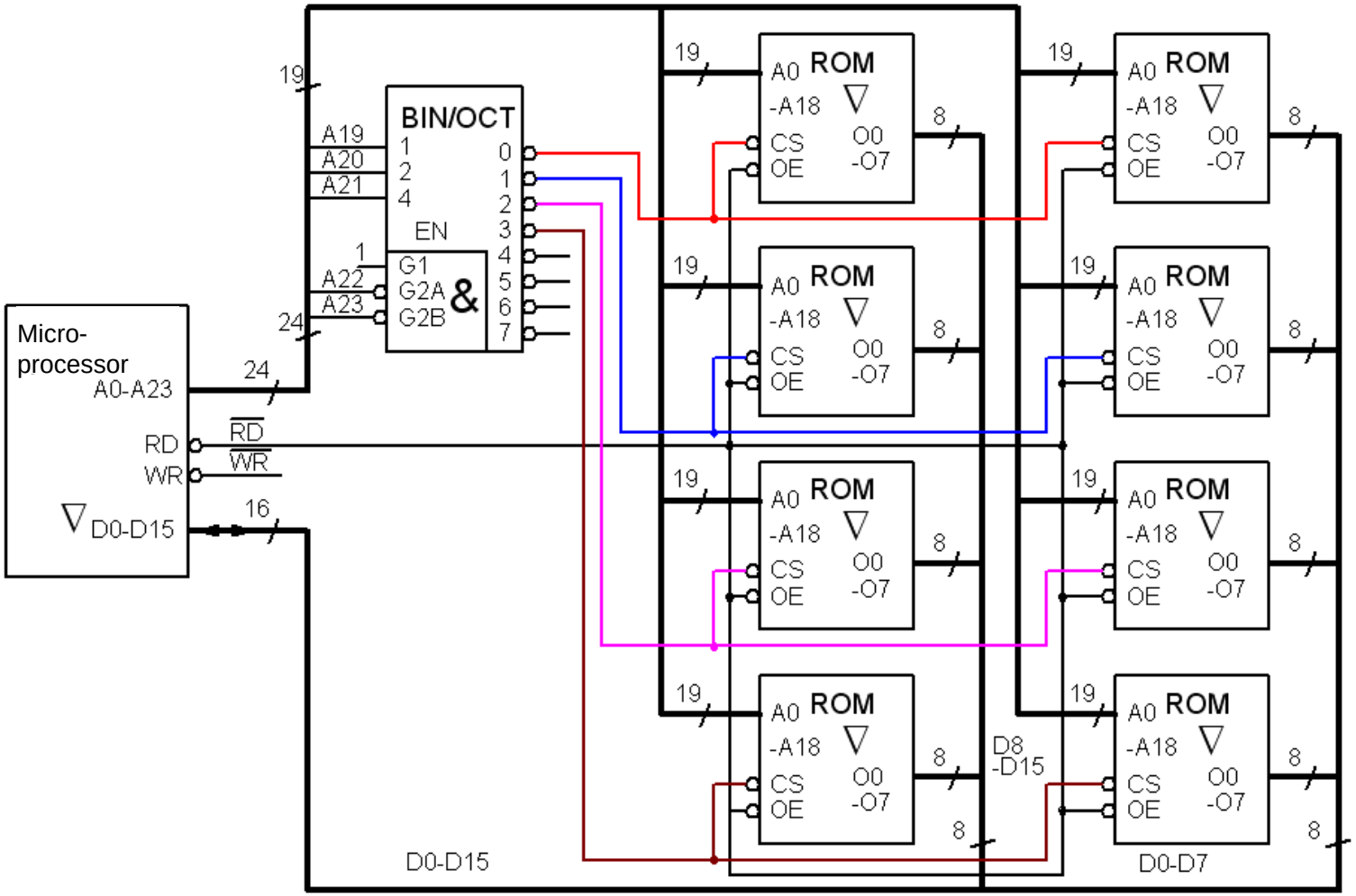
ROM Control connections?



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

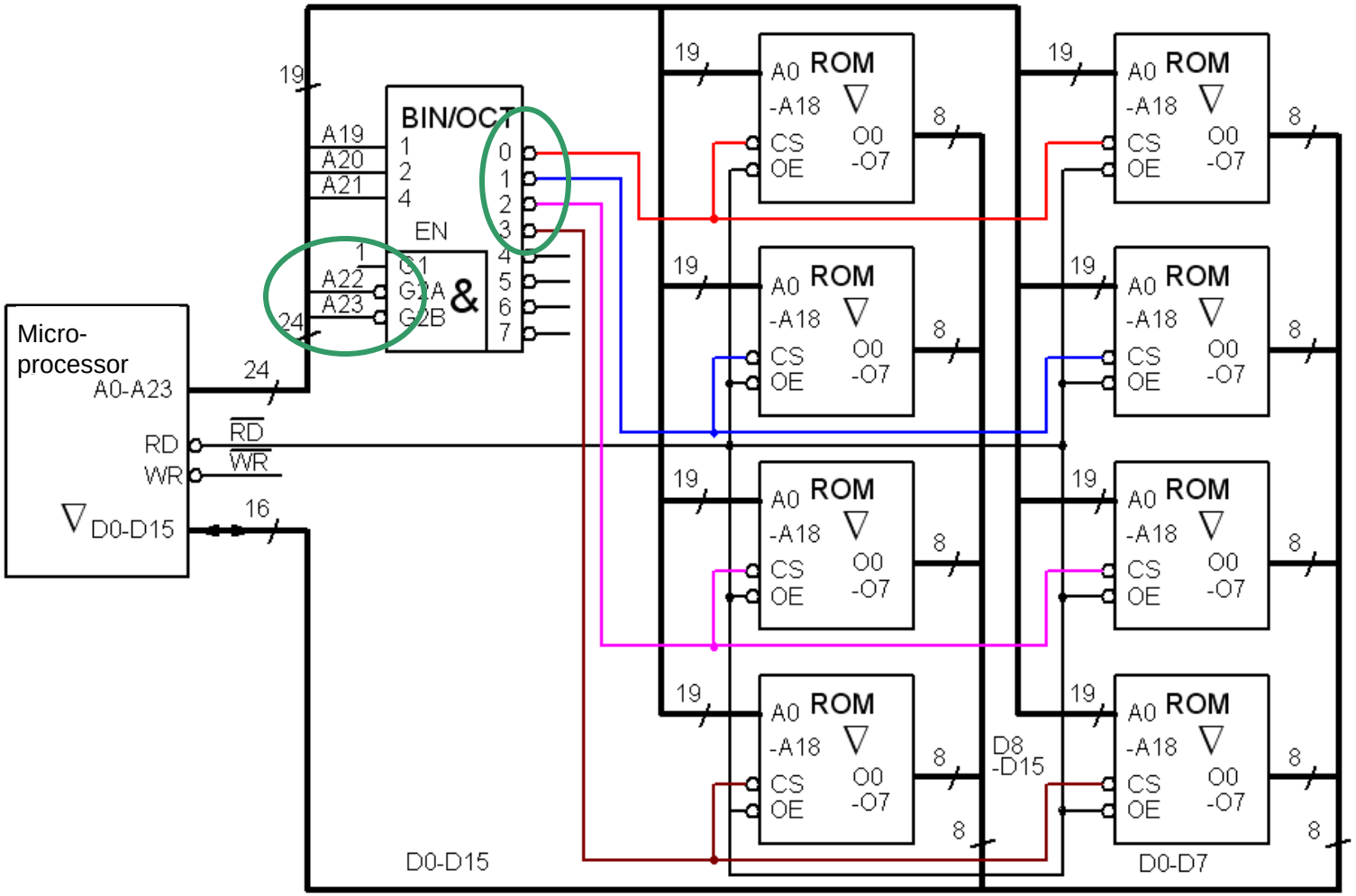
William Sandqvist william@kth.se

Decoder ROM connection?



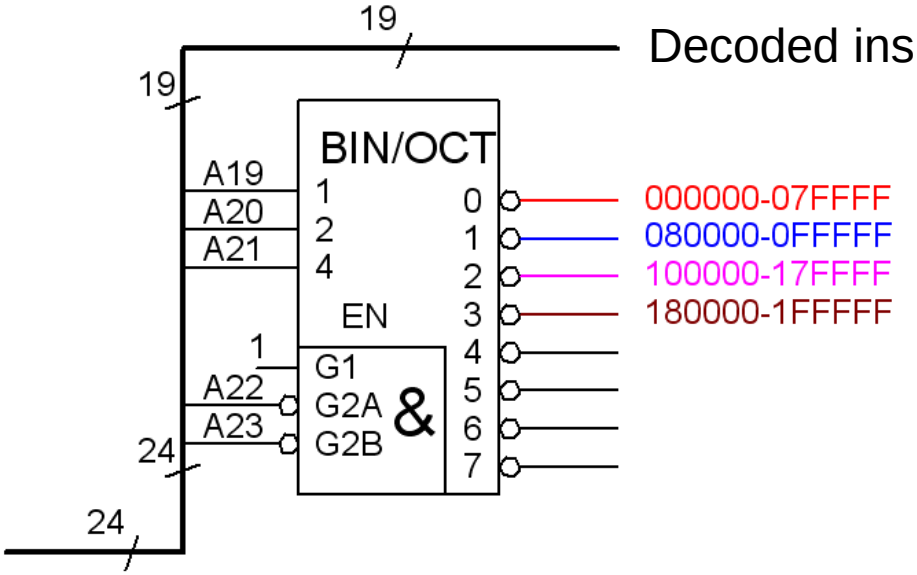
William Sandqvist william@kth.se

Decoder connection?



William Sandqvist william@kth.se

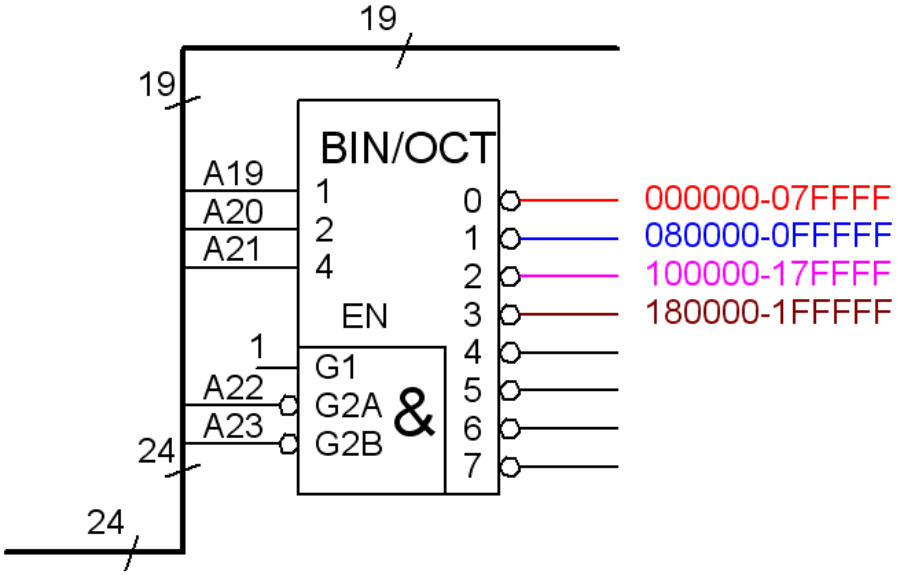
Decoder ROM adresses?



Decoded inside memory chips

Four rows with memory chips

Decoder ROM adresses?



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

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

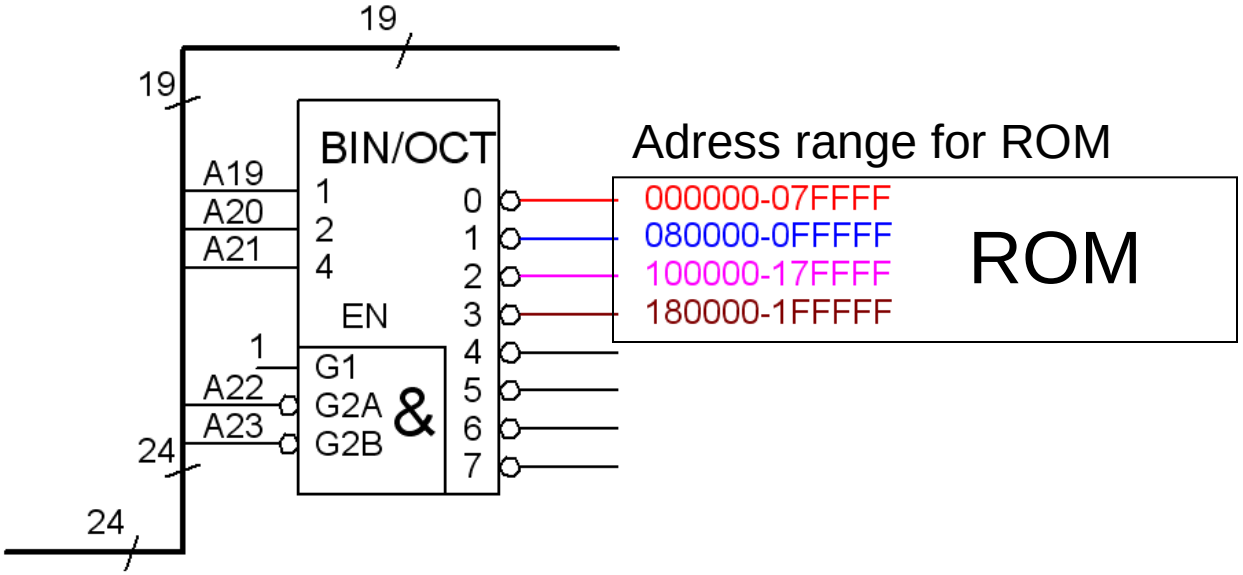
00**00**|**1**000|0|0|0|0 - 00**00**|**1**111|F|F|F|F 080000-0FFFFF

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 adresses?



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

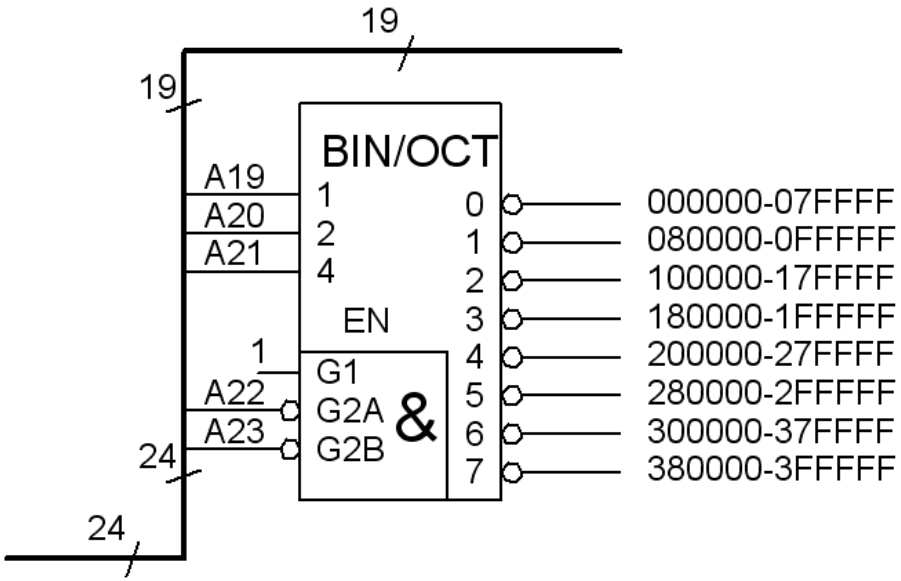
00**01**|**0**000|0|0|0|0 - 00**01**|**0**111|F|F|F|F 100000-17FFFF

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

Totaly ROM 000000 – 1FFFFF

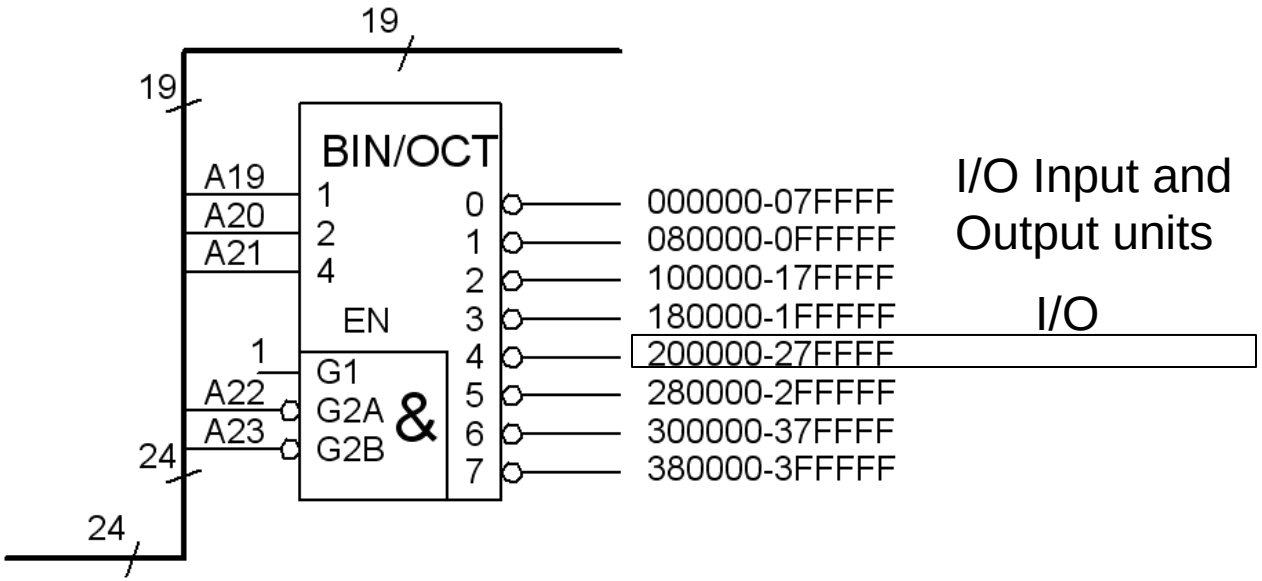
William Sandqvist william@kth.se

Decoder SRAM+I/O adresses?



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

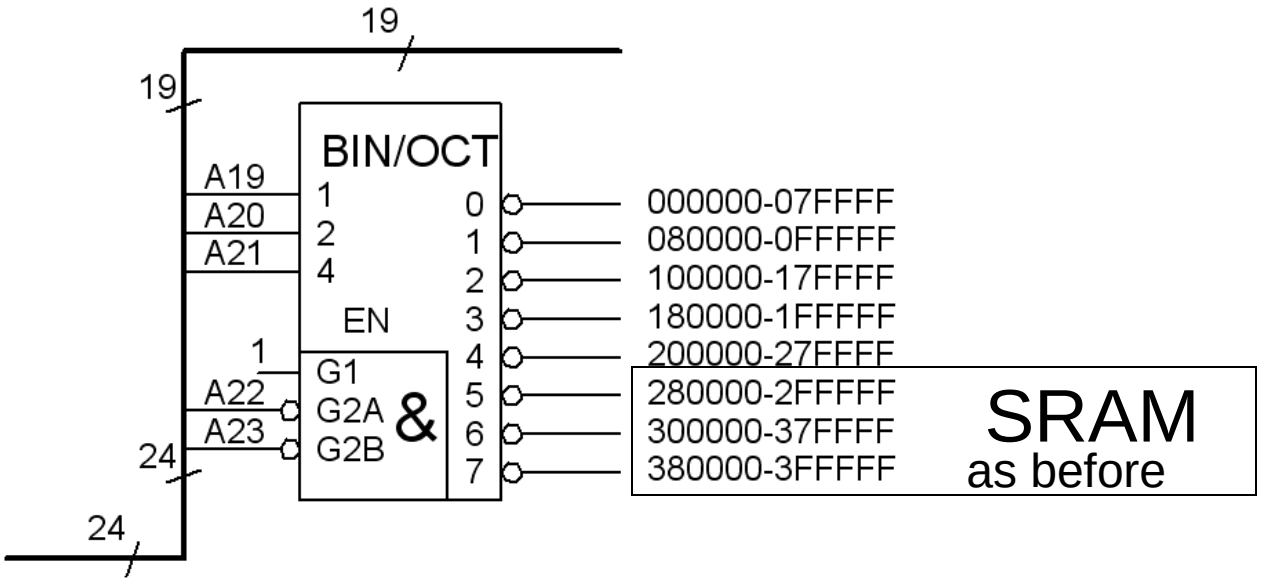
Decoder SRAM+I/O adresses?



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

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

Decoder SRAM+I/O adresses?

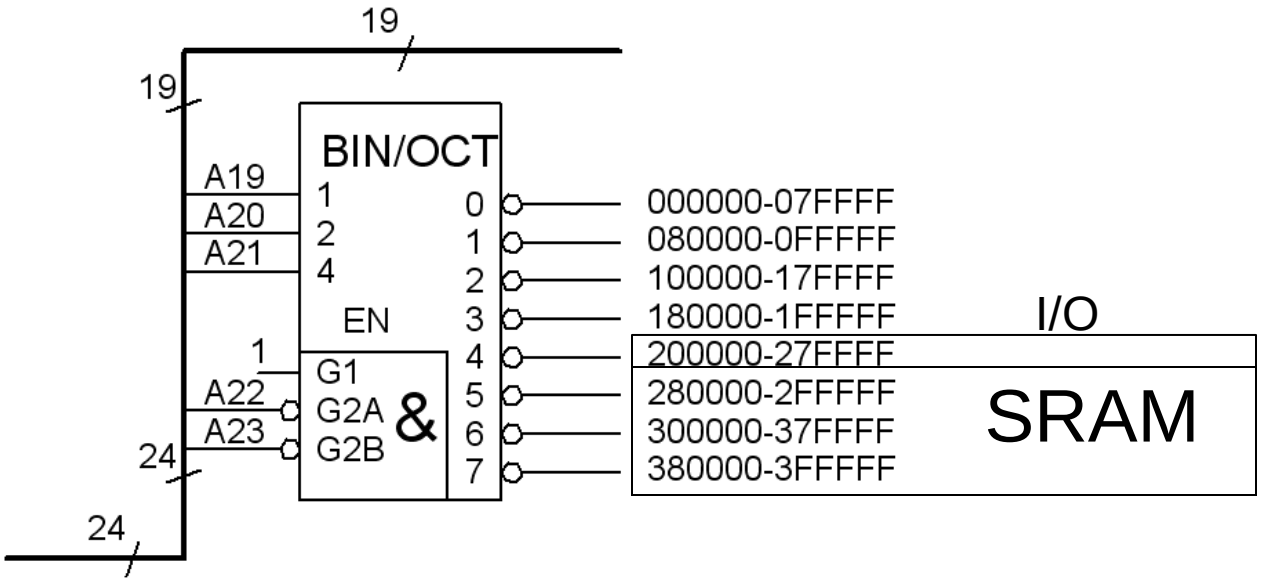


00ab|cmmm|mmmm|mmmm|mmmm|mmmm

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

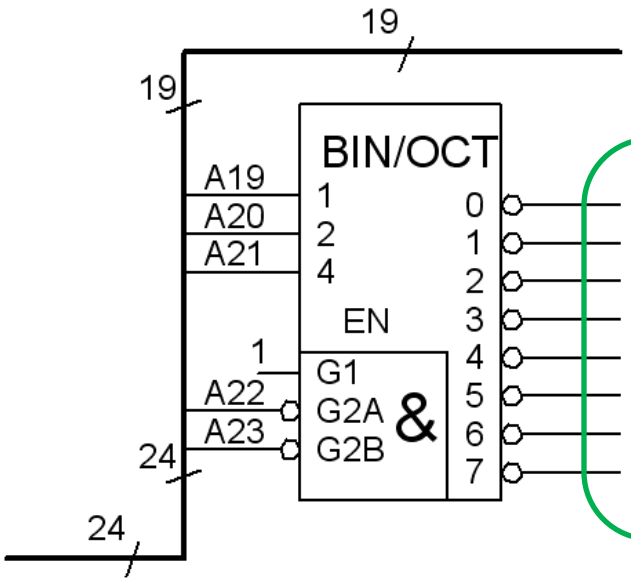
00**10**|**1**000|0|0|0|0 - 00**10**|**1**111|F|F|F|F 280000-2FFFFF

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

- Possible SRAM+I/O adresser 200000 – 3FFFFF

Memory map

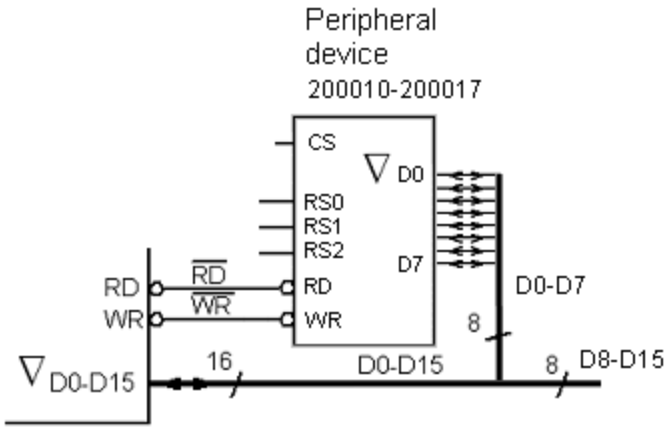


The **memory map** for this example.

000000-07FFFF	ROM
080000-0FFFFF	
100000-17FFFF	
180000-1FFFFF	
200000-27FFFF	I/O
280000-2FFFFF	RAM
300000-37FFFF	
380000-3FFFFF	

Ex **12.3** Input/Output

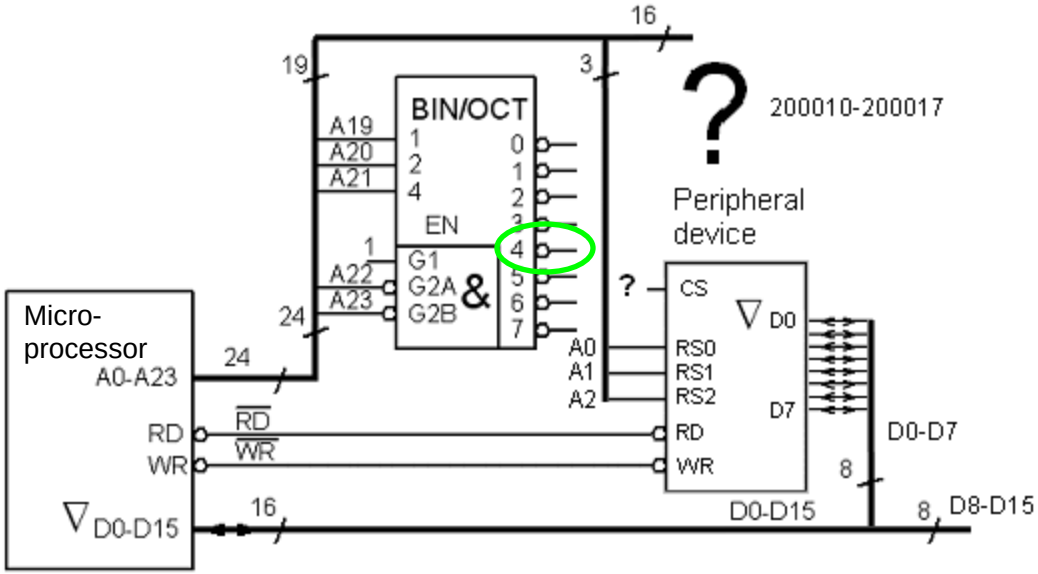
Peripherals, I/O, are often connected to a CPU as if they were memory chips (though with only a few "memory cells"). Eg. a real time clock chip - keeps track of time and date. It is controlled/read from the 8 built-in registers.



Peripheral circuit connected as a small RAM. Only the 8 least significant bits of data are used. CS Chip Select enables the chip.

Connect a 8 register memory-mapped peripheral device (I/O) to a CPU. The CPU has 16-bit data bus (only 8 bits are used by the chip), and a 24 bit address bus. Use a 3:8-decoder and if needed gates. The peripheral device must be connected so that it can register addresses **`0x200010 … 0x200017`**.

Ex **12.3** Input/Output



I/O adresses, at the decoder output ”4”, 200000 – 27FFFF according to the earlier task.

Decoding

```
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
```

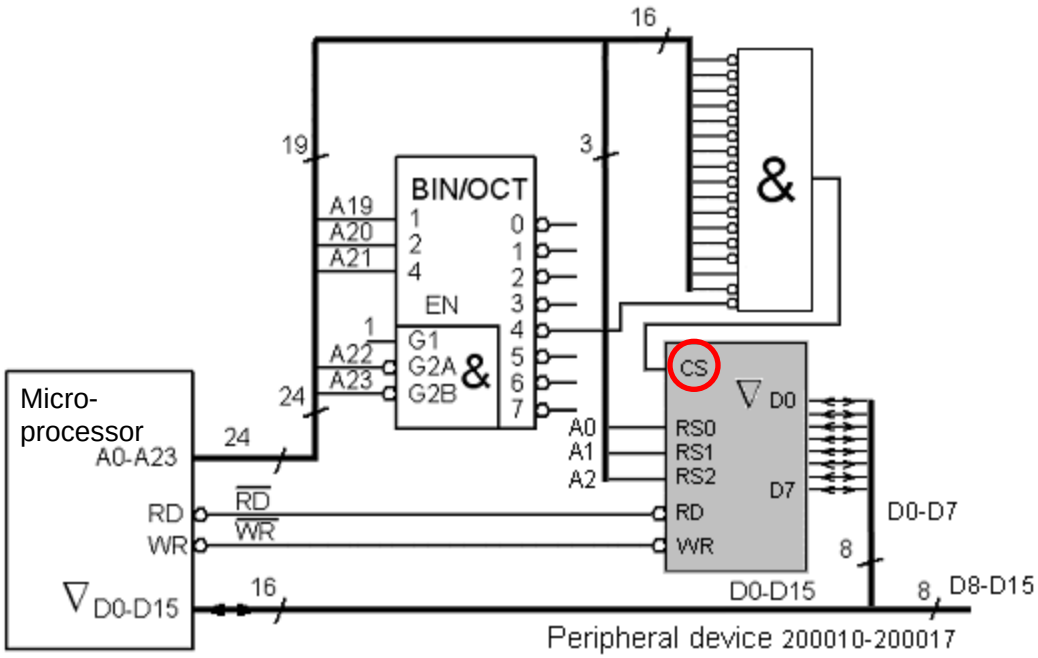
Decoder output ”4”

$RS_2RS_1RS_0$

Remains to decode:

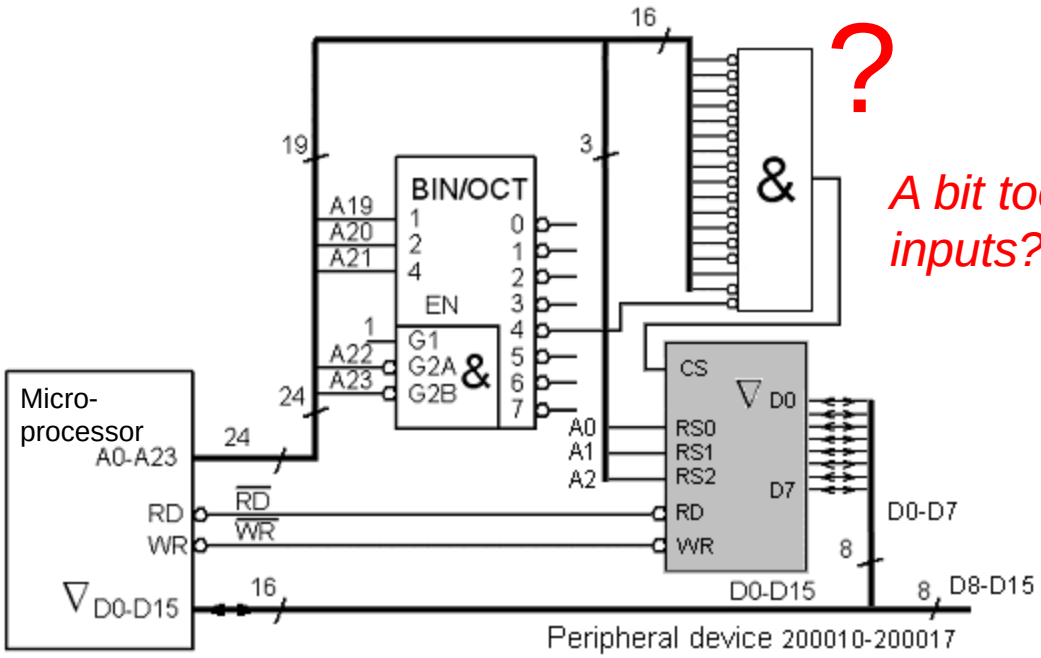
$$\overline{A}_{18} \cdot \overline{A}_{17} \cdot \overline{A}_{16} \cdot \overline{A}_{15} \cdot \overline{A}_{14} \cdot \overline{A}_{13} \cdot \overline{A}_{12} \cdot \overline{A}_{11} \cdot \overline{A}_{10} \cdot \overline{A}_{9} \cdot \overline{A}_{8} \cdot \overline{A}_{7} \cdot \overline{A}_{6}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

Connections

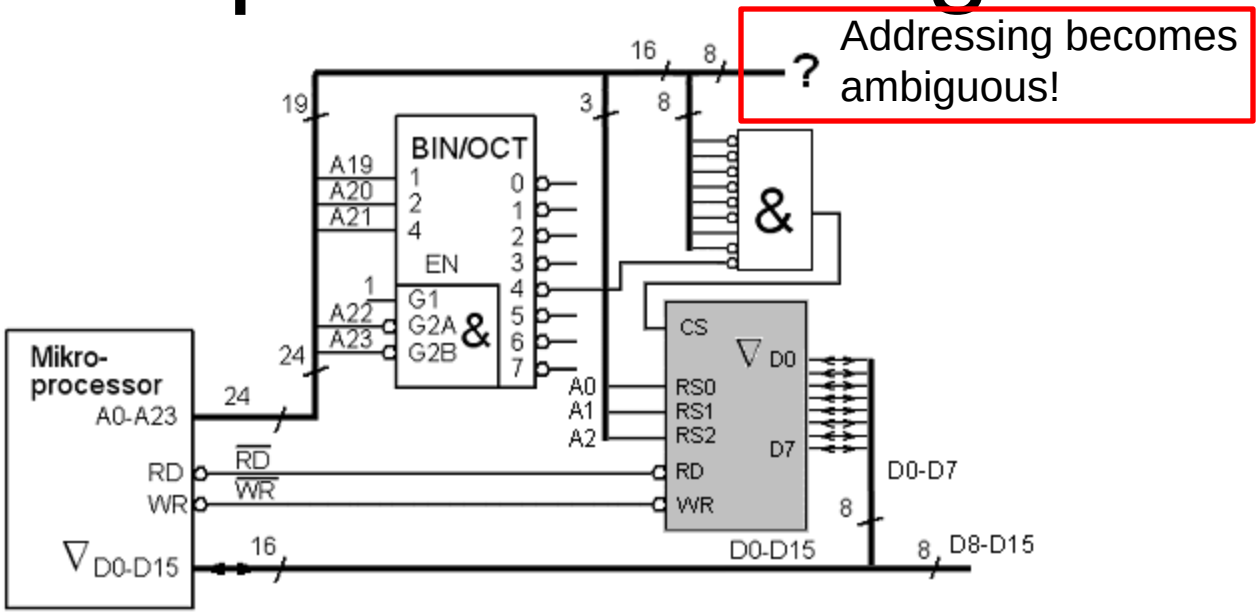


William Sandqvist william@kth.se

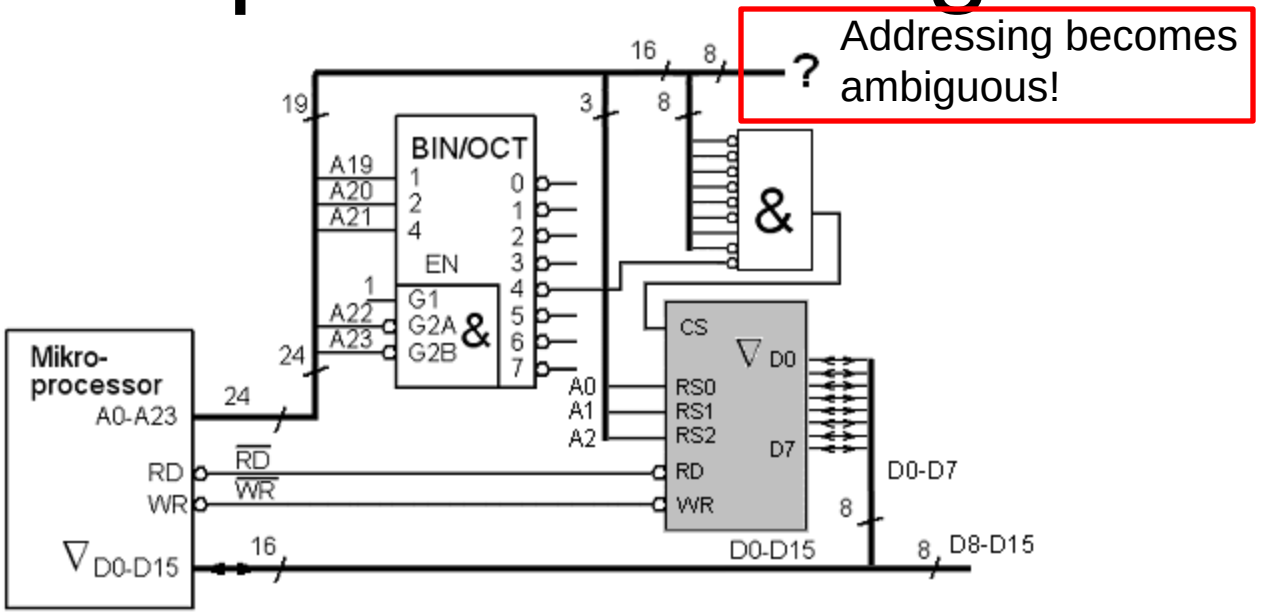
Connections



incomplete decoding?



incomplete decoding?



For full decoding, we used a &-gate with 17 inputs! Sometimes you make a partial decoding. Then you omits address signals and thus can use a gate with fewer inputs.

I/O device addressing is ambiguous, it can be addressed with many different addresses, but the one who writes the program code determines which addresses to use. The main thing is to ensure that the I/O device addresses do not collide with any other device addresses.

volatile ?

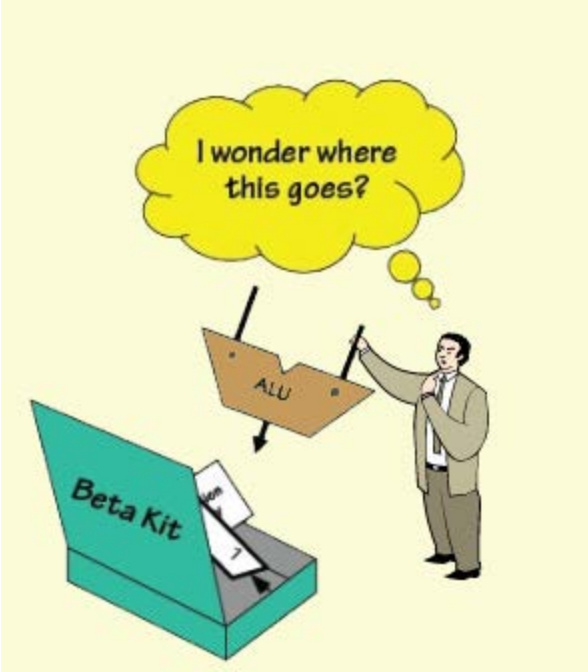
William Sandqvist william@kth.se

`volatile` ?

Since the I/O devices are not true memories - it can seem as if the content can be changed "by itself" - so when you write computer programs you need to "help" the compiler to understand this. It could be done by declaring these adresses as **`volatile`**.

This, you will meet in Computer Engineering course.

How to construct a bigger Digital system?



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).

Algorithmic State Machine

ASM method consists of the following steps:

1. Creating an algorithm in pseudo code, which describes the desired circuit function.

2. Transform the pseudocode to an *ASM diagram*.

3. Design a *data flow diagram* from the ASM diagram.

4. Create a detailed *ASM diagram* from the data flow diagram.

5. Design the control logic based on the detailed ASM chart.

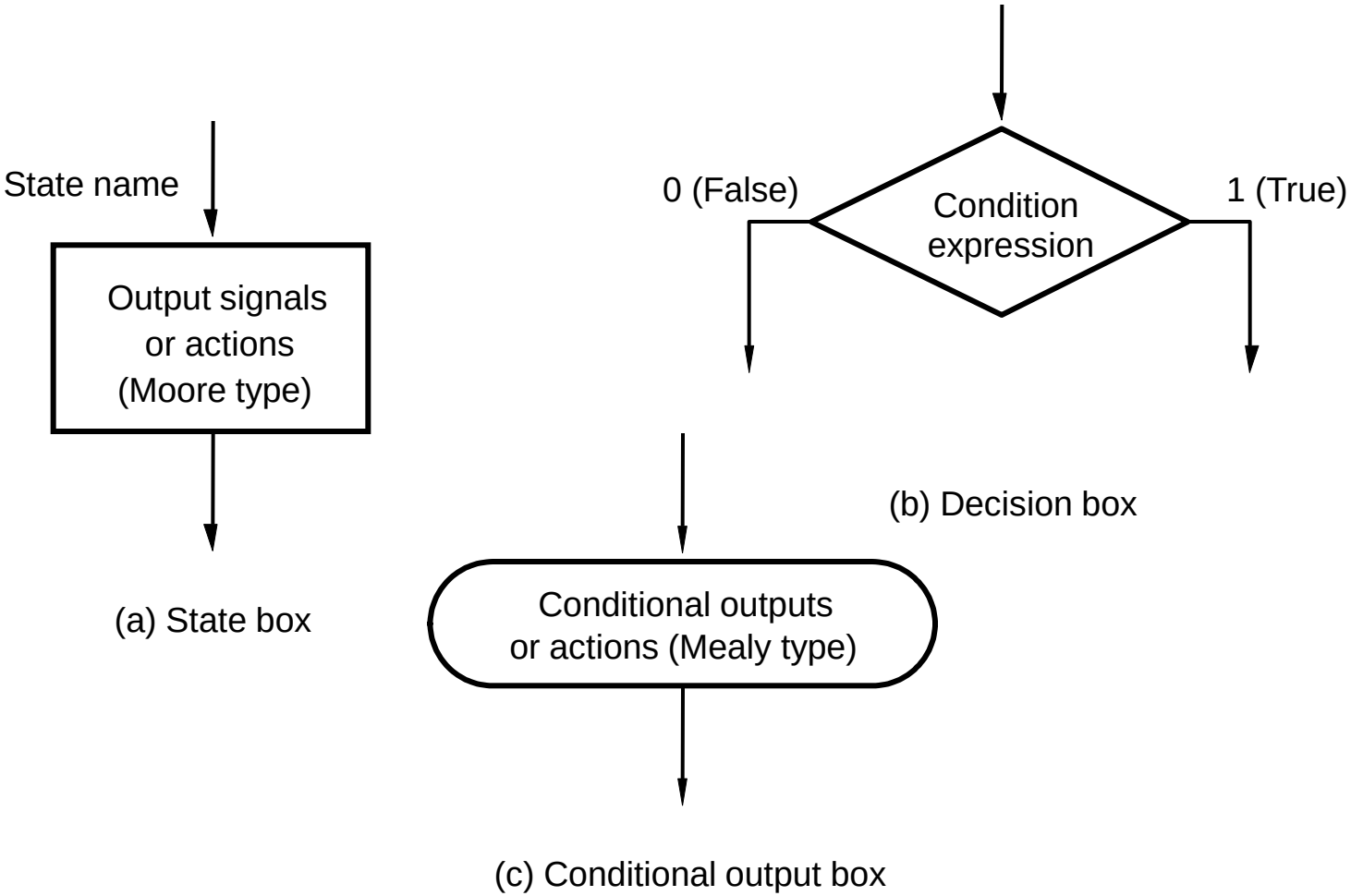
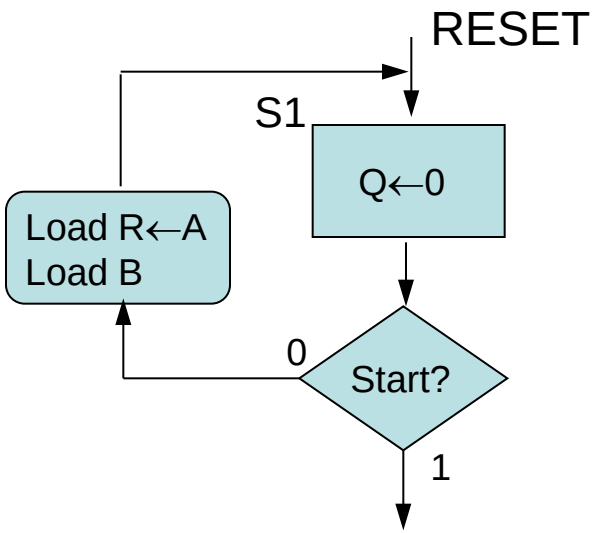


Figure 8.86. Elements used in ASM charts.

BV 10.5 ASM chart

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

BV 10.5 ASM chart

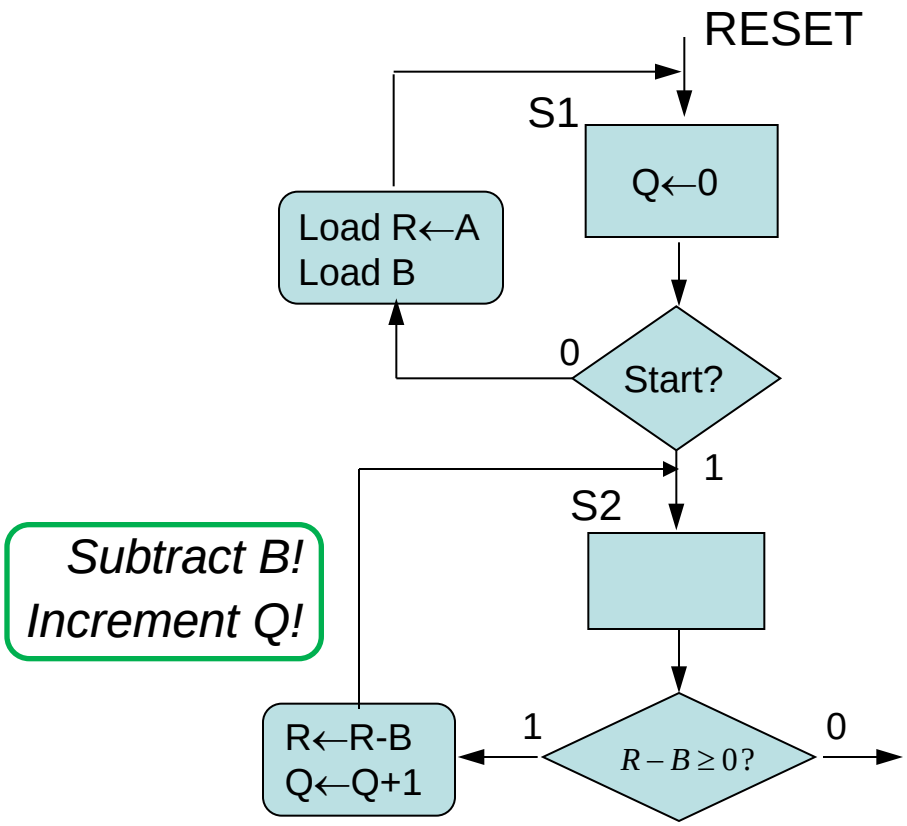


Wait for start (1)!

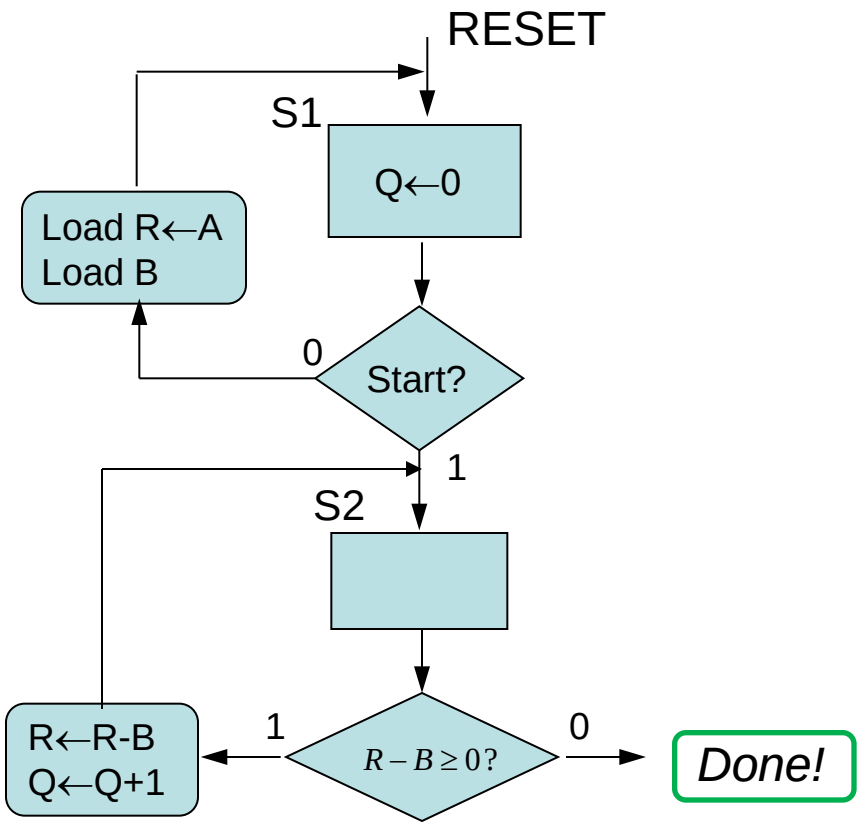
Q = 0;
R = A
While ((R – B) ≥ 0) do
R = R – B;
Q = Q + 1;
End while;

BV 10.5 ASM chart

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



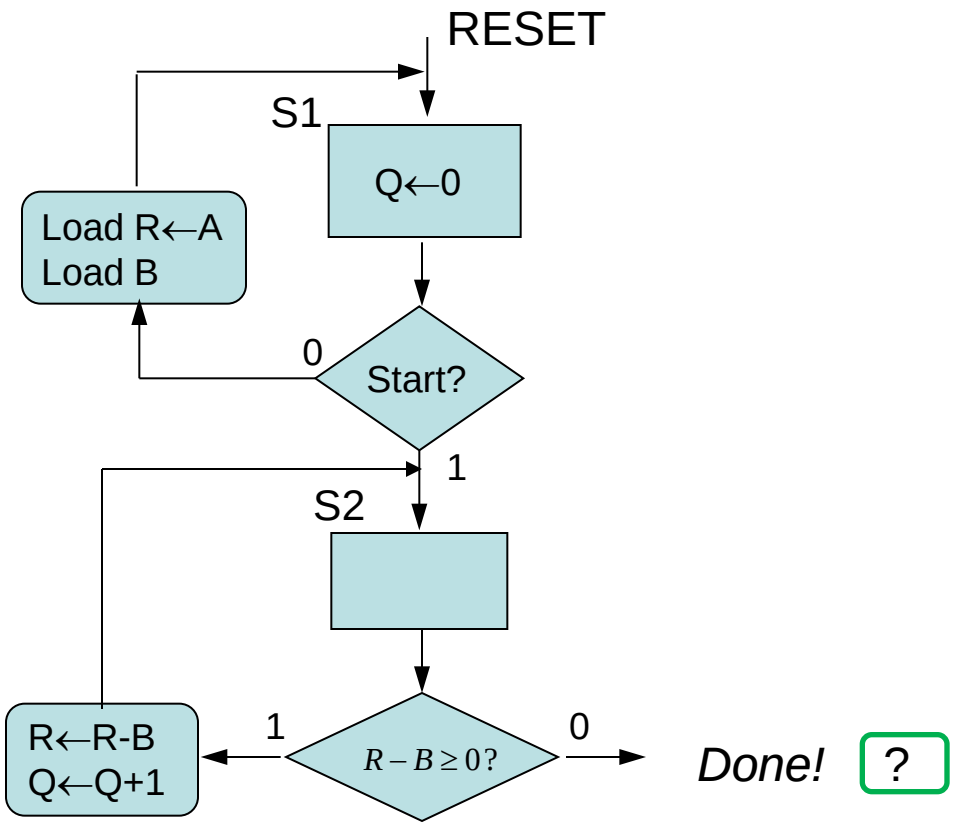
BV 10.5 ASM chart



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



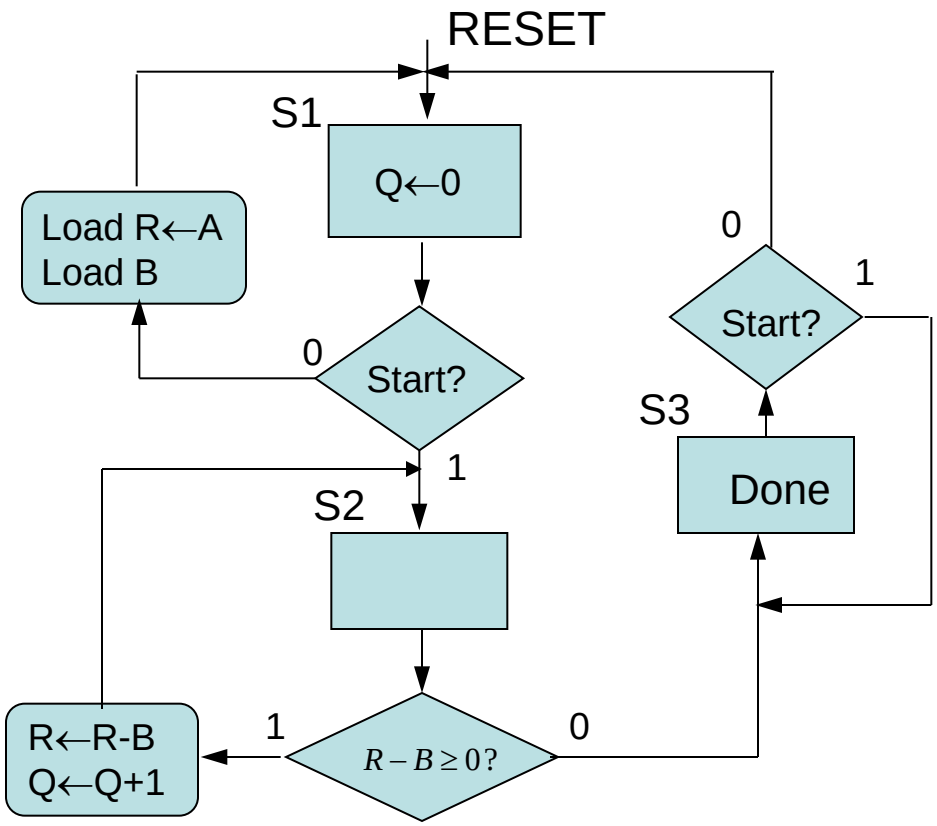
Q = 0;
R = A
While ((R – B) ≥ 0) do
R = R – B;
Q = Q + 1;
End while;

?

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;

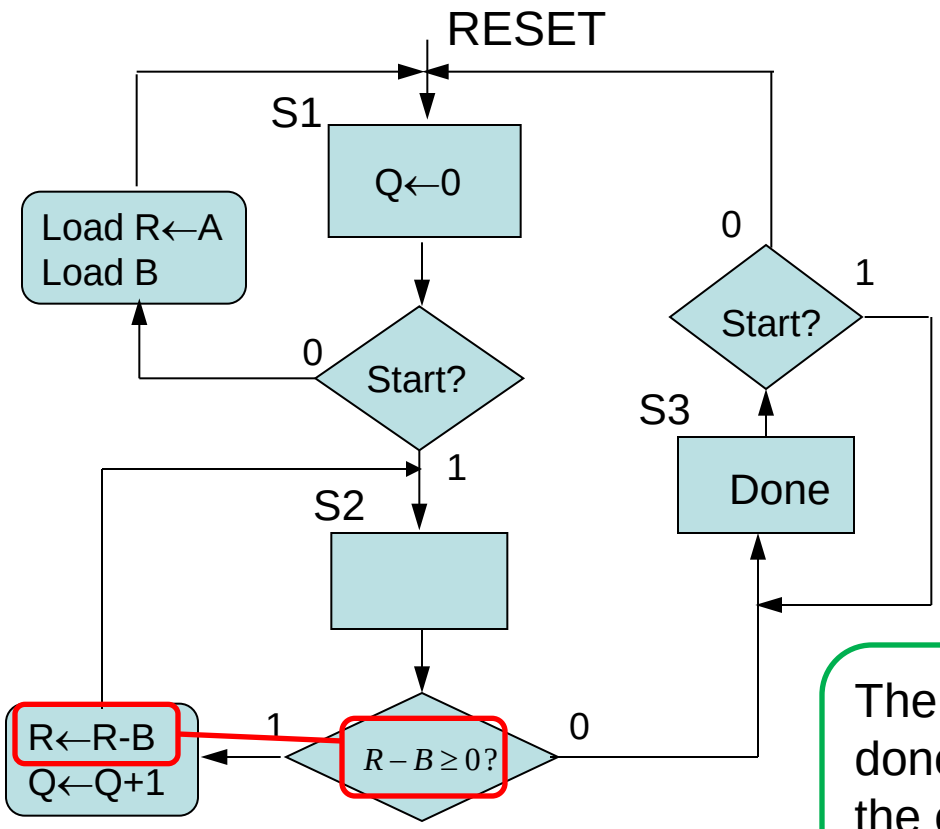


Wait for Start to be released (0), to restart.

William Sandqvist william@kth.se

How to test condition?

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

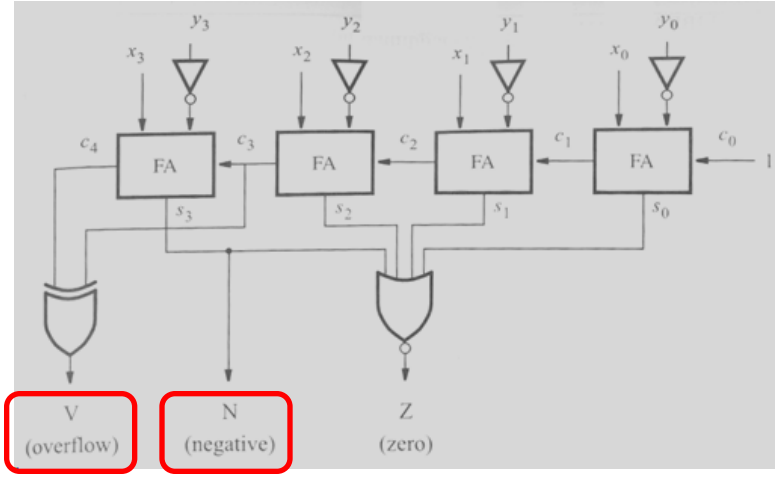


The test that *R-B* ≥ 0 is done simultaneously with the operation *R-B* by inspecting flags.

- How do we know when R-B<0 ?

Do you remember? $\geq$

Adder connected as comparator



$$X - Y$$

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

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

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

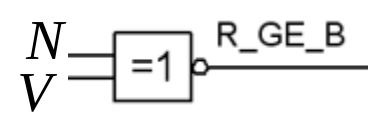
$$X < Y \quad \Rightarrow \quad N \oplus V$$

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

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

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

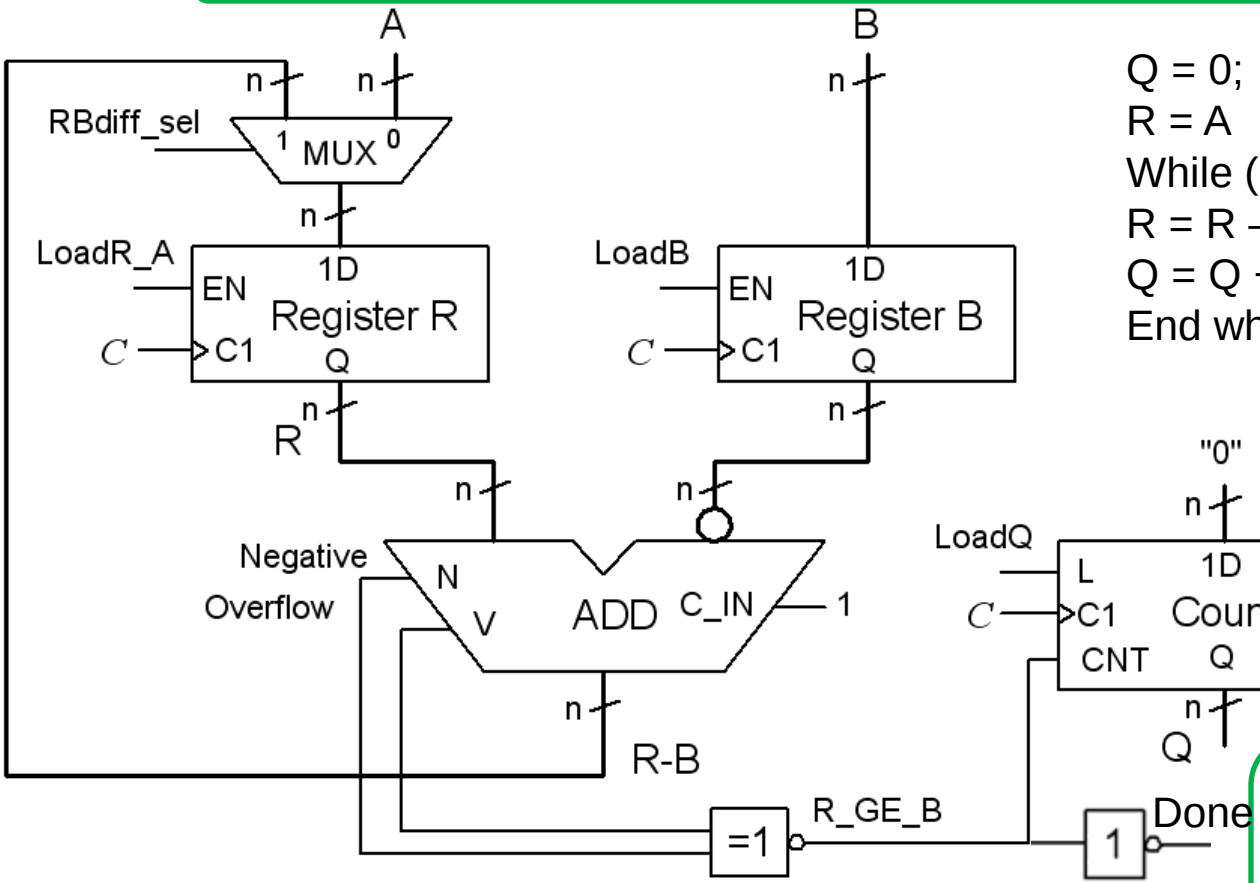
This is how a computer can do the most common comparisons …



William Sandqvist william@kth.se

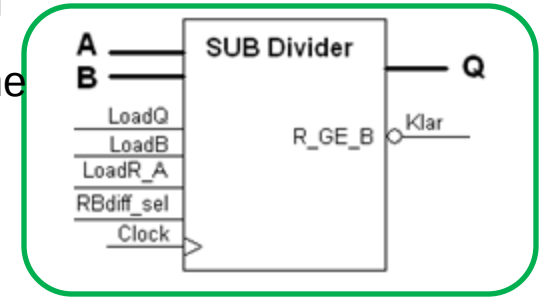
BV 10.5 **datapath** circuit

- ASM chart lays out the foundation for the **hardware**.



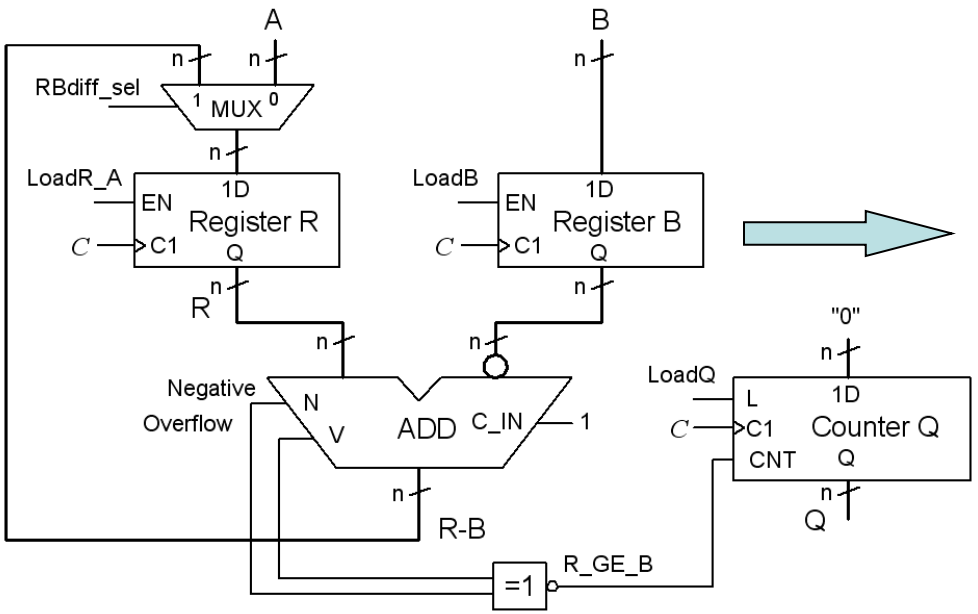
Q = 0;
R = A
While ((R – B) ≥ 0) do
R = R – B;
Q = Q + 1;
End while;

- Hardware

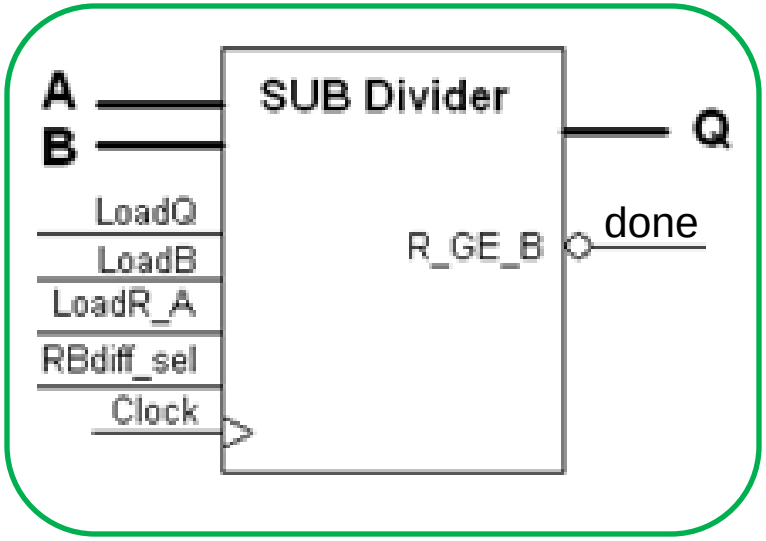


William Sandqvist william@kth.se

BV 10.5 **Hardware**



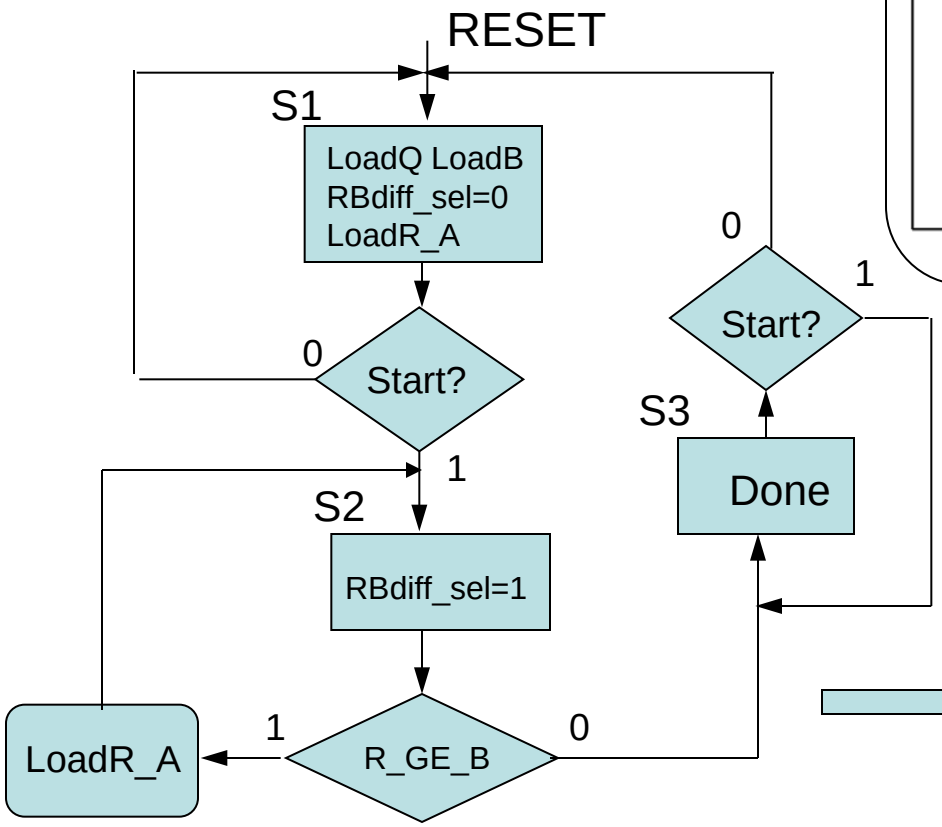
- Hardware block

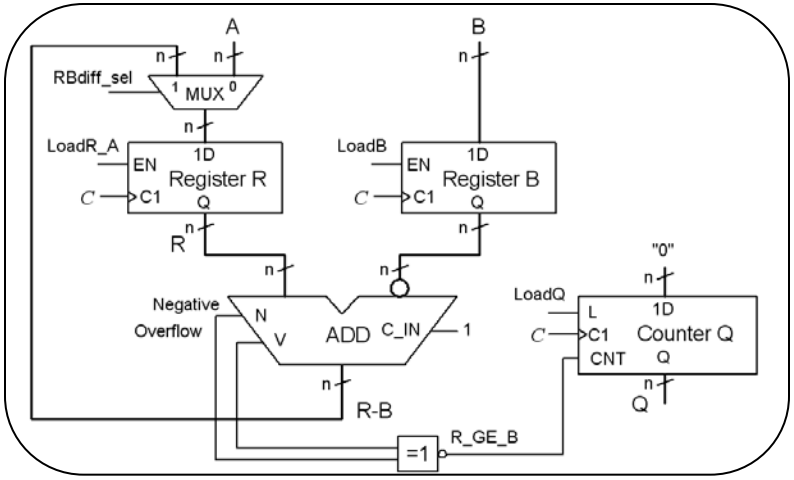


William Sandqvist william@kth.se

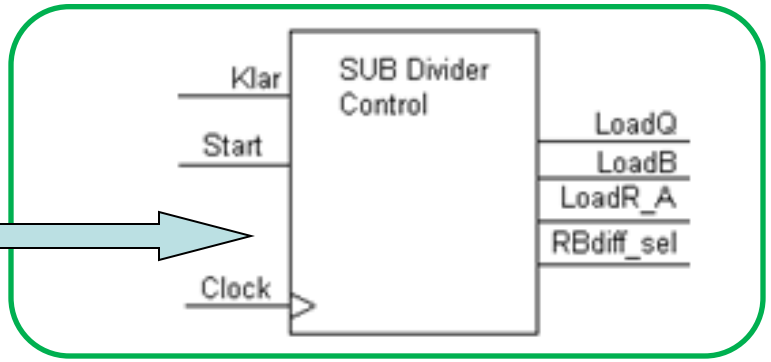
BV 10.5 ASM **control**

- ASM chart is also used as the Control circuit **State diagram**.



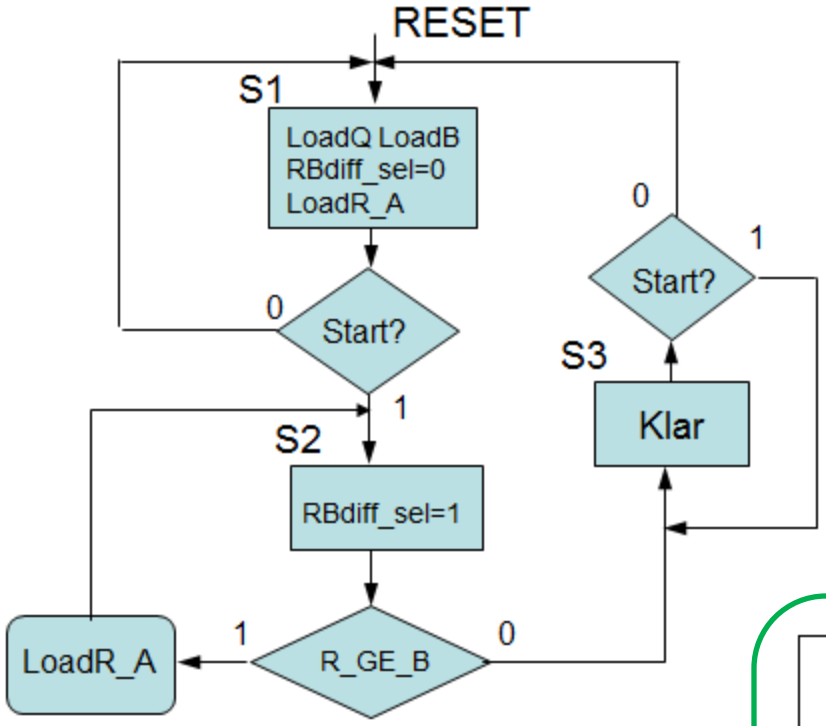


- Control

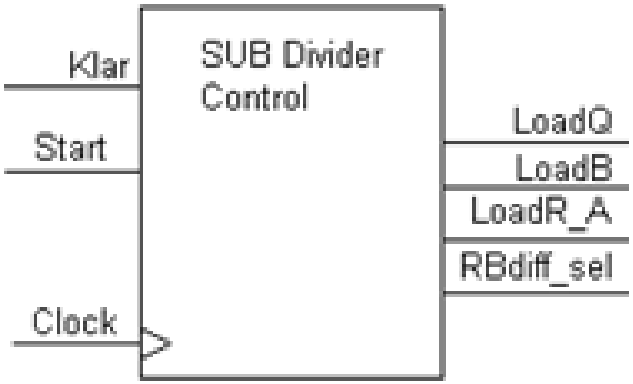


William Sandqvist william@kth.se

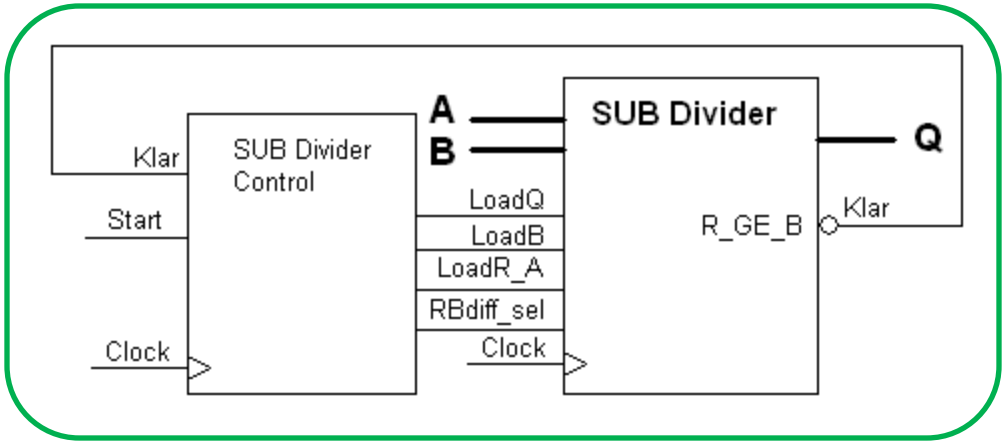
BV 10.5 **complete system**



- Control



- Complete system



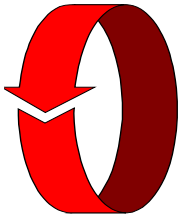
”SUB Divider Control”
can be constructed as a Moore machine with three states from the state diagram (ASM).

William Sandqvist william@kth.se

William Sandqvist william@kth.se

sin + cos values?

Another bigger Digital system …



$$x = x + y/2$$

$$y = y - x/2$$

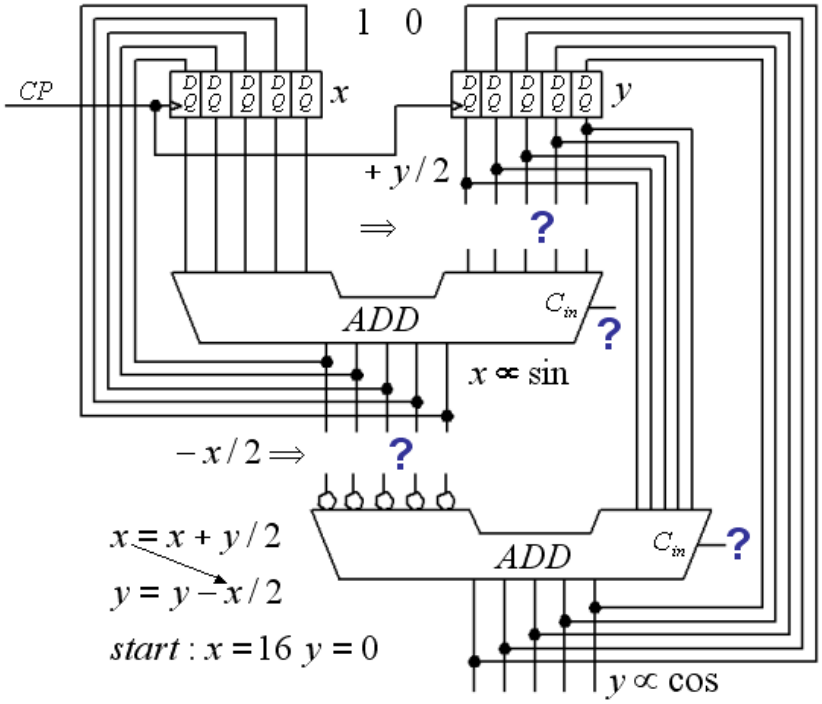
$$start: x = 16 \quad y = 0$$

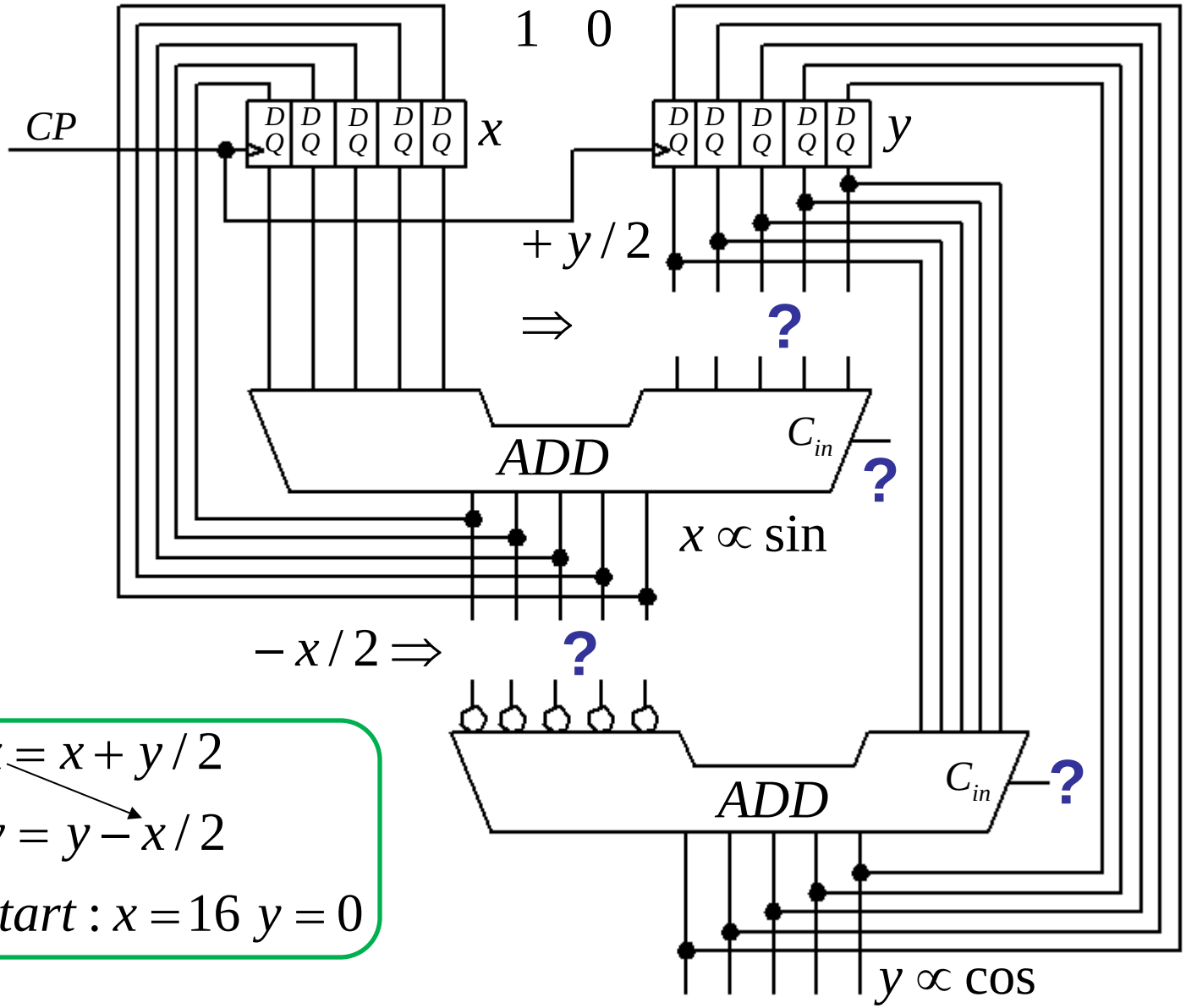
William Sandqvist william@kth.se

Osquar and Osqulda are implementing a digital design algorithm, to their thesis work. The algorithm calculates **sine** and **cosine** values.

$x = x + y/2$; $y = y - x/2$; (start values $x = 0$, $y = 16$).

Help them with how $+y/2$ and $-x/2$ can be implemented in the figure (see four places with question marks). Constants with values 1 and 0 are available at need.

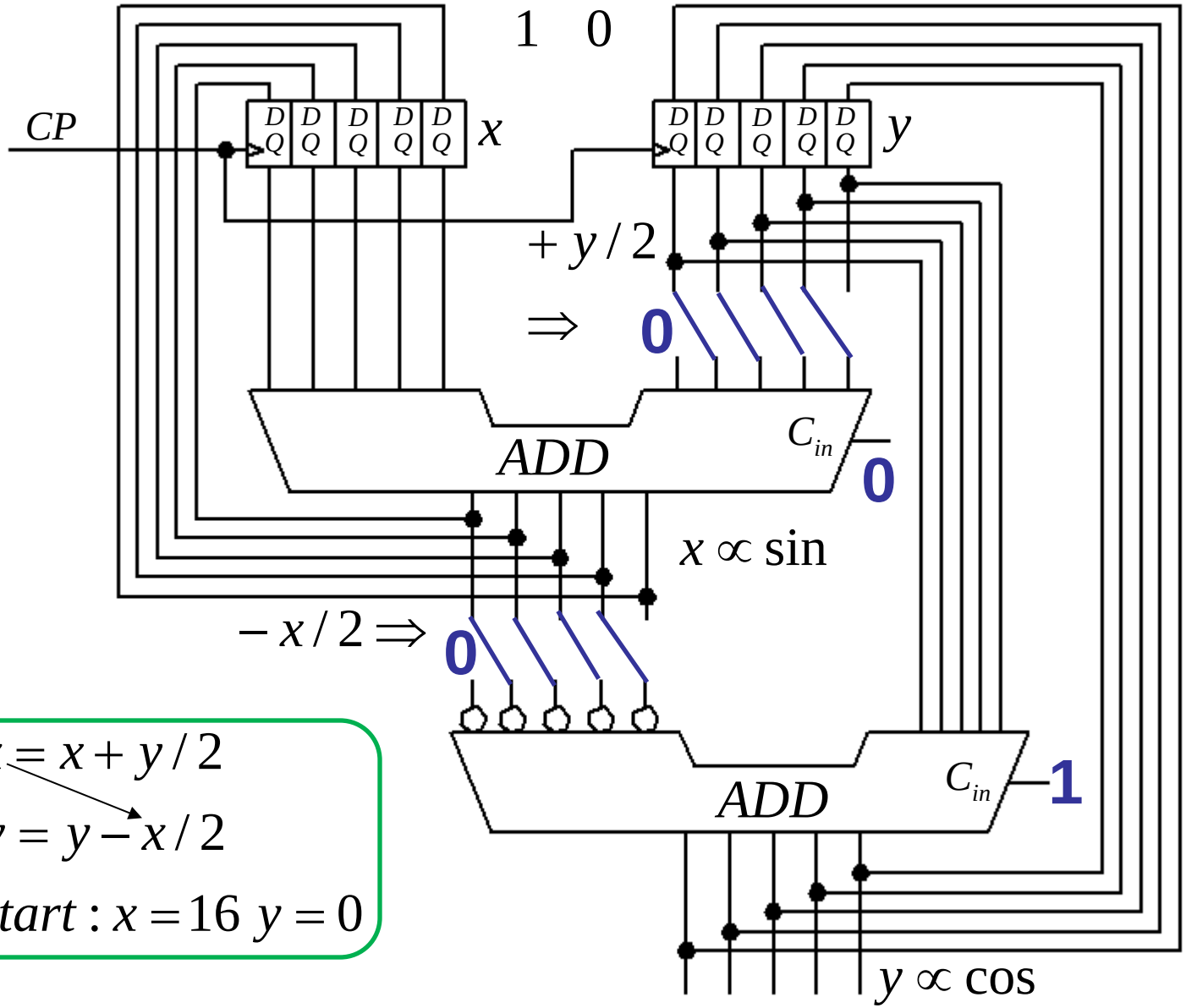




$$x = x + y/2$$

$$y = y - x/2$$

$$start : x = 16 \quad y = 0$$

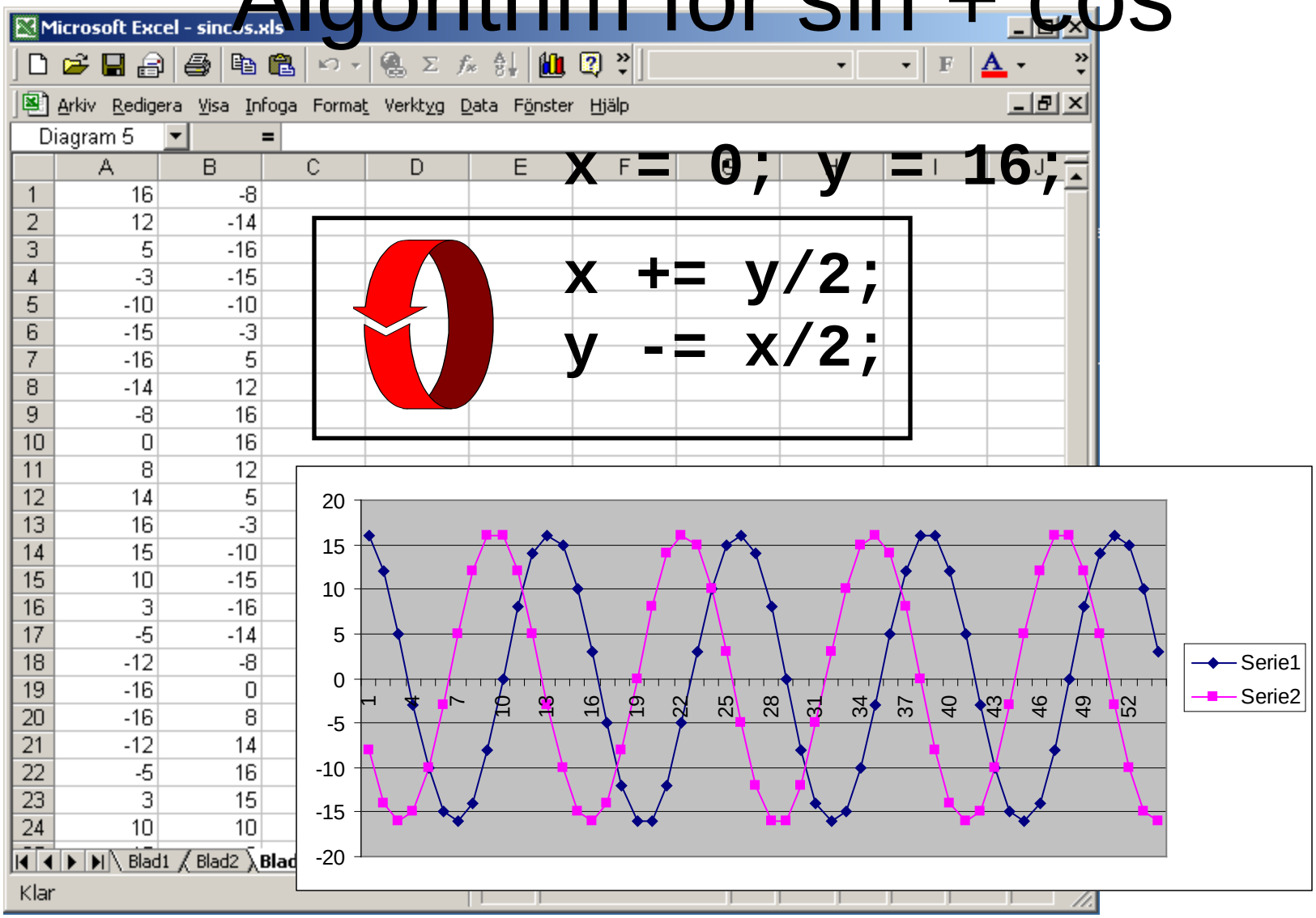


$$x = x + y/2$$

$$y = y - x/2$$

$$start : x = 16 \quad y = 0$$

Algorithm for sin + cos



William Sandqvist william@kth.se

Rehearsal before the exam

Ex 6.10 Combinatorial circuit 5 variables

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

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

	x_4	x_3	x_2	x_1	x_0	f		x_4	x_3	x_2	x_1	x_0	f
0	0	0	0	0	0	0	16	1	0	0	0	0	1
1	0	0	0	0	1	0	17	1	0	0	0	1	0
2	0	0	0	1	0	0	18	1	0	0	1	0	1
3	0	0	0	1	1	0	19	1	0	0	1	1	0
4	0	0	1	0	0	0	20	1	0	1	0	0	0
5	0	0	1	0	1	0	21	1	0	1	0	1	0
6	0	0	1	1	0	0	22	1	0	1	1	0	0
7	0	0	1	1	1	0	23	1	0	1	1	1	0
8	0	1	0	0	0	0	24	1	1	0	0	0	1
9	0	1	0	0	1	1	25	1	1	0	0	1	1
10	0	1	0	1	0	0	26	1	1	0	1	0	1
11	0	1	0	1	1	1	27	1	1	0	1	1	1
12	0	1	1	0	0	1	28	1	1	1	0	0	0
13	0	1	1	0	1	1	29	1	1	1	0	1	0
14	0	1	1	1	0	1	30	1	1	1	1	0	0
15	0	1	1	1	1	1	31	1	1	1	1	1	0

6.10 Combinatorial circuit 5 variables

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

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

$\overline{x}_4$

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

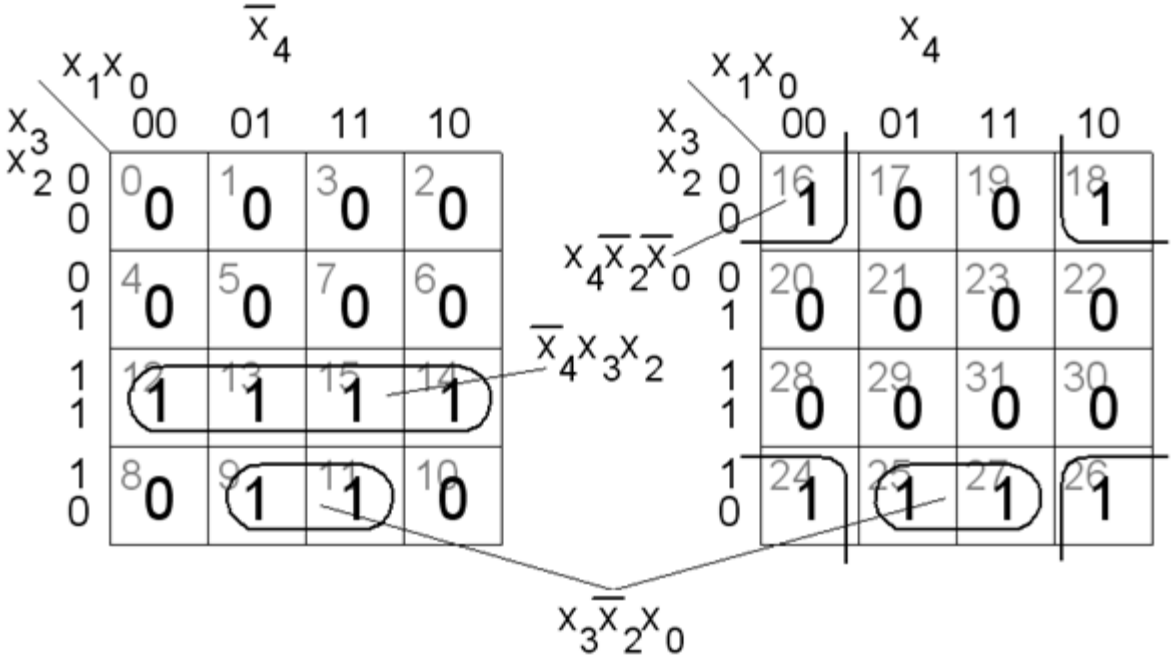
x_4

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

6.10 Combinatorial circuit 5 variables

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

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

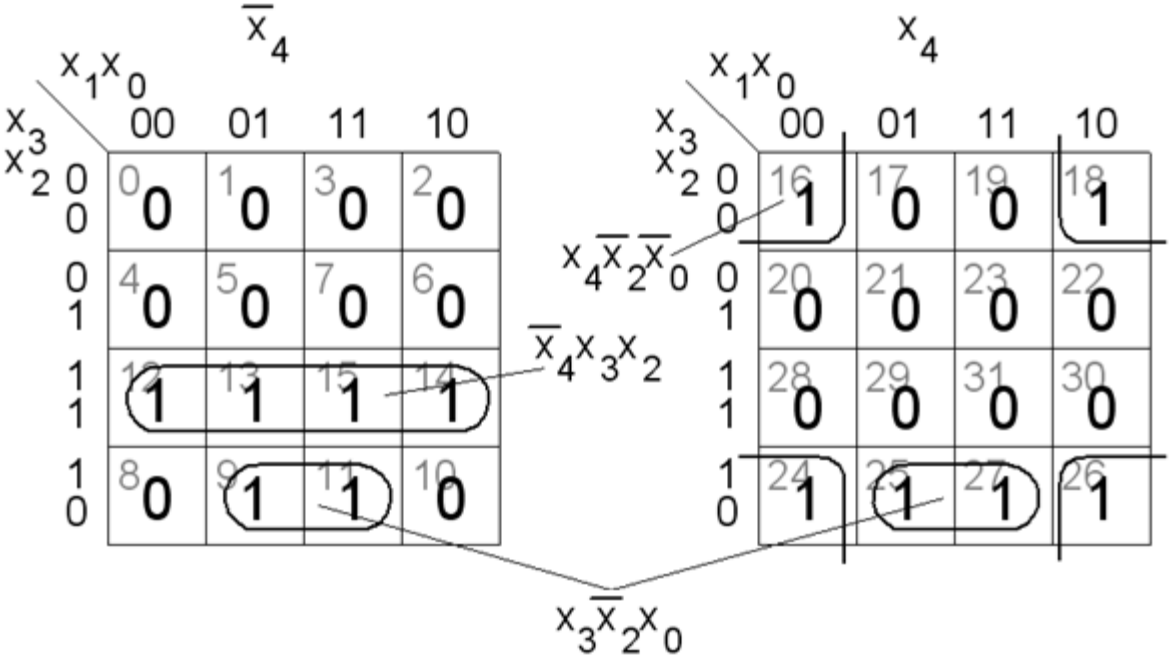


William Sandqvist william@kth.se

6.10 Combinatorial circuit 5 variables

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

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



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

William Sandqvist william@kth.se

6.10 Combinatorial circuit 5 variables

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

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

$\overline{x}_4$

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

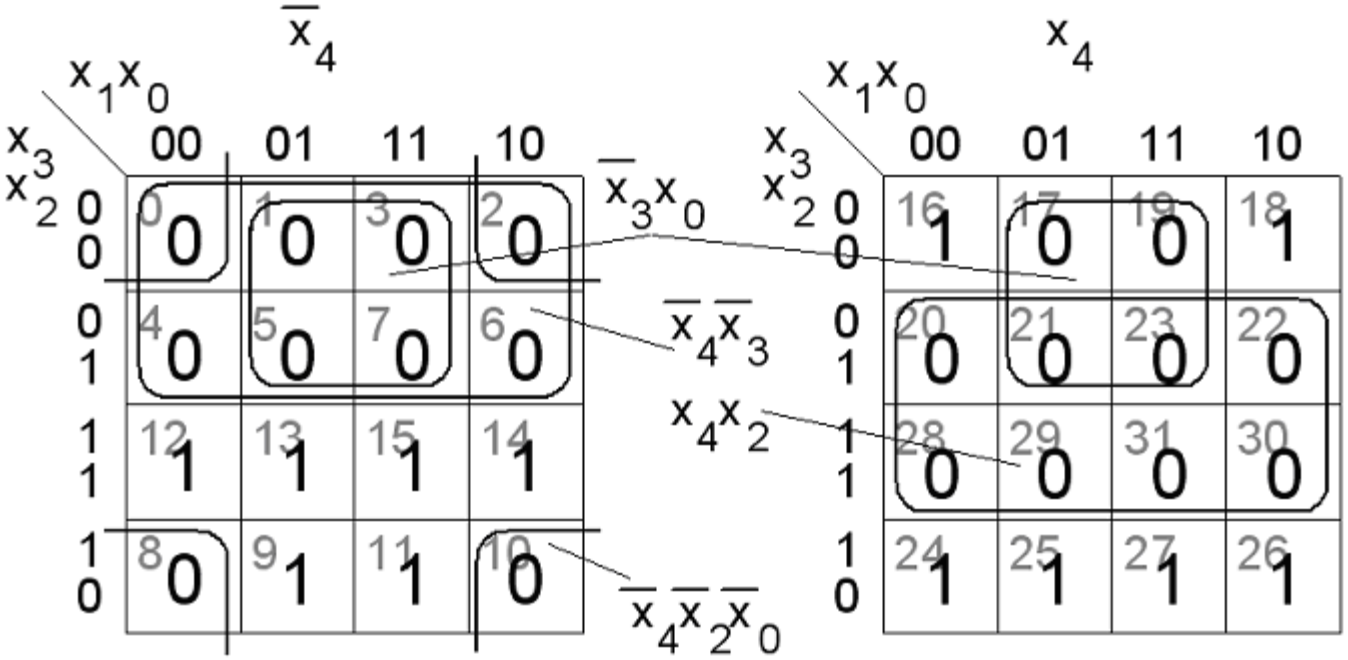
x_4

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

6.10 Combinatorial circuit 5 variables

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

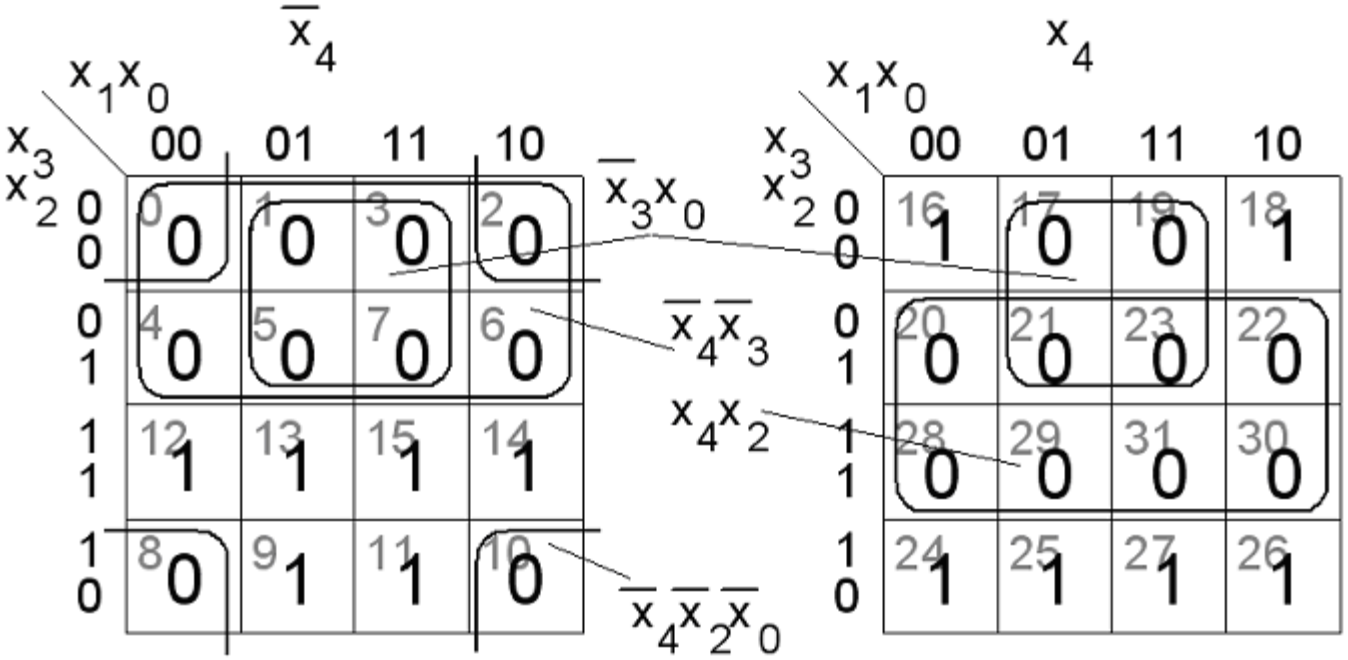
$$f(x_4, x_3, x_2, x_1, x_0) \quad \overline{f} = ?$$



6.10 Combinatorial circuit 5 variables

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

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

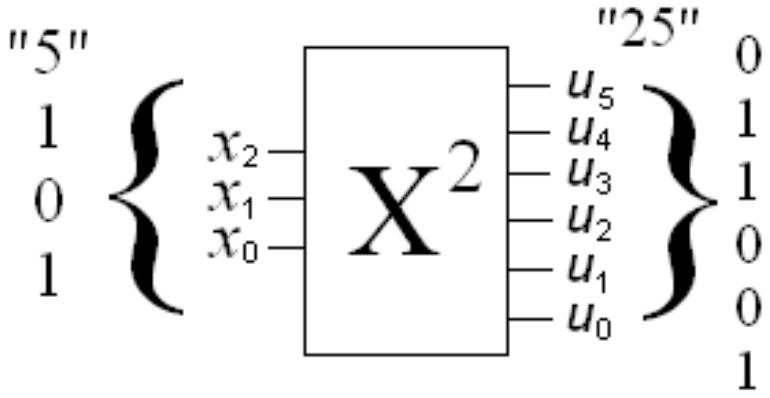


$$\overline{f} = \overline{x_4}\,\overline{x_3} + \overline{x_3}\,x_0 + x_4\,x_2 + \overline{x_4}\,\overline{x_2}\,\overline{x_0}$$

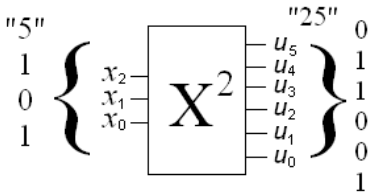
William Sandqvist william@kth.se

Ex 8.1 Binary squarer

Bring out the Boolean equations for a network at minimal SP-form which transforms a three-bit binary coded number X (x_2, x_1, x_0) to a binary coded six bit number U $(u_5, u_4, u_3, u_2, u_1, u_0)$ which is equal to the square of the number $U = X^2$.

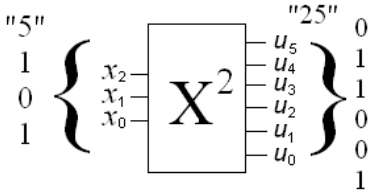


8.1 Truth table



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

8.1 Karnaugh map



Of truth table it shows that u_1 always is equal to 0. u_1 uotput could therefore be connected to 0V (ground) so it will get the constant 0. One can further see that u_0 always is the same as x_0. u_0 output can therfore be connected directly to x_0 input.

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

$$u_2 = x_1 \bar{x}_0$$

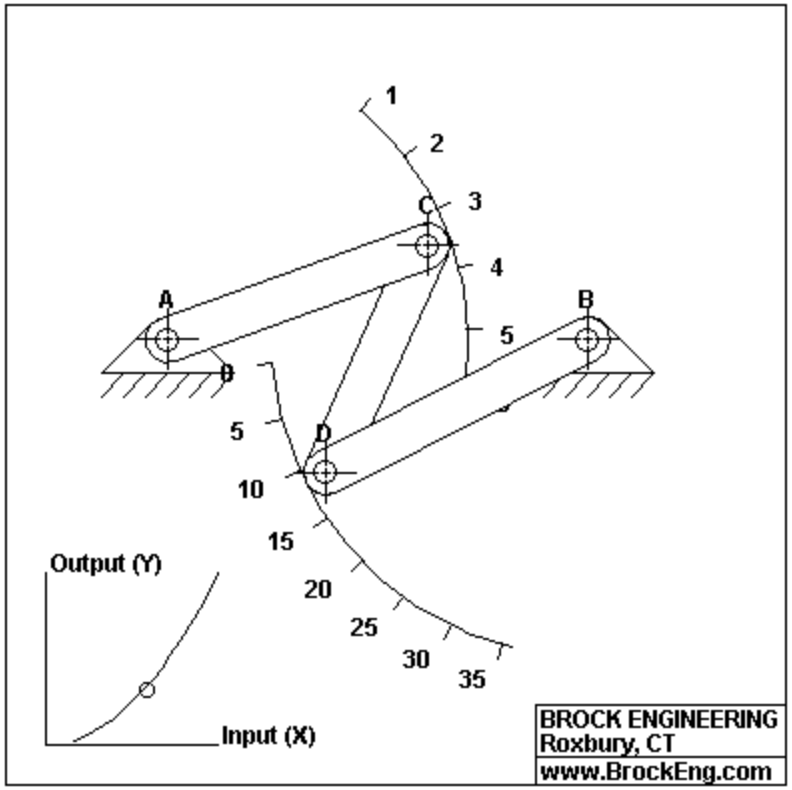
x_2 \ x_1x_0	00	01	11	10
0	0	0	0	1
1	0	0	0	1

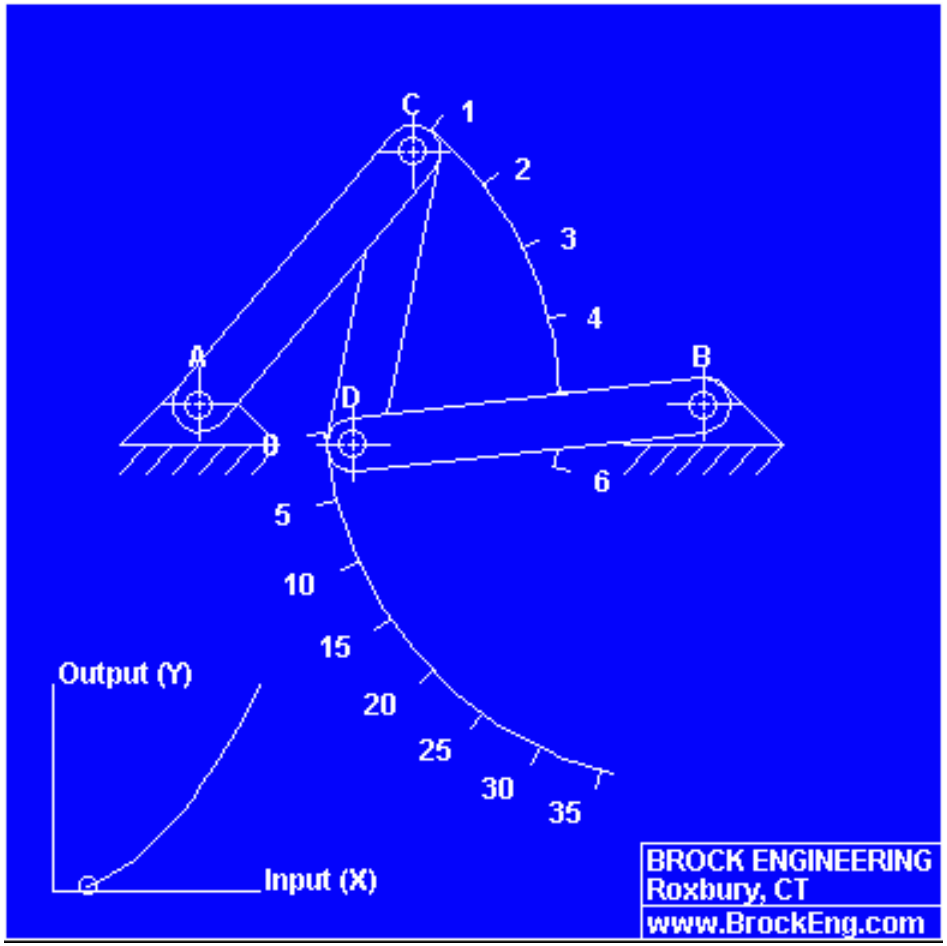
$$u_1 = 0$$

$$u_0 = x_0$$

William Sandqvist william@kth.se

Mechanical "squarer"





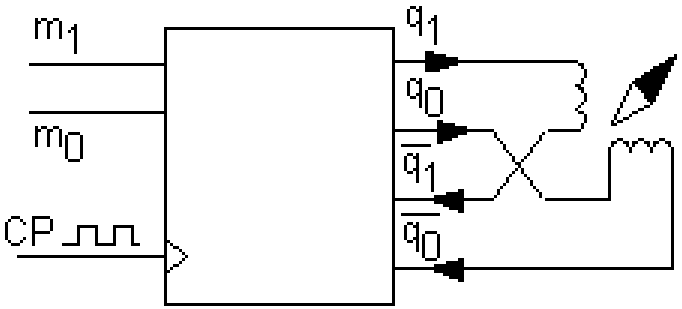
Brock institute for advaced studies function generator

Ex. 10.9 Stepper motor controller

A stepper motor is a digital component that is driven by pulses.

Stepper motors are usually connected to a counter counting Gray code.

Figure calculator also has a mode-input, m_1m_0.

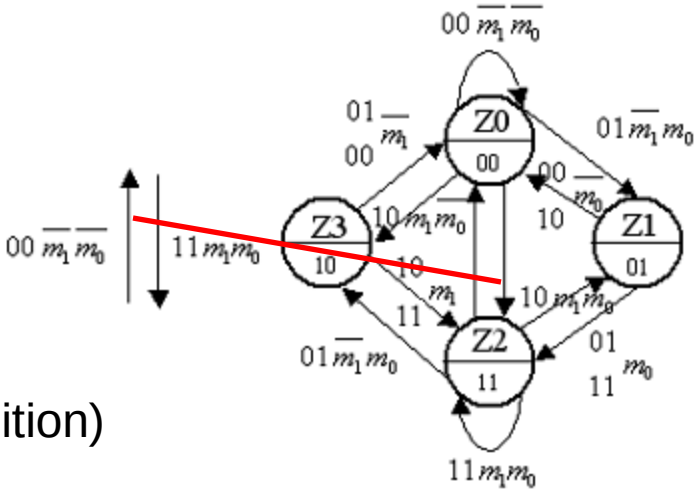


$m_1m_0 = 00 \rightarrow$ Reset (fixed position)

$m_1m_0 = 01 \rightarrow$ count up (cw)

$m_1m_0 = 10 \rightarrow$ count down (ccw)

$m_1m_0 = 11 \rightarrow$ Preset (another fixed position)



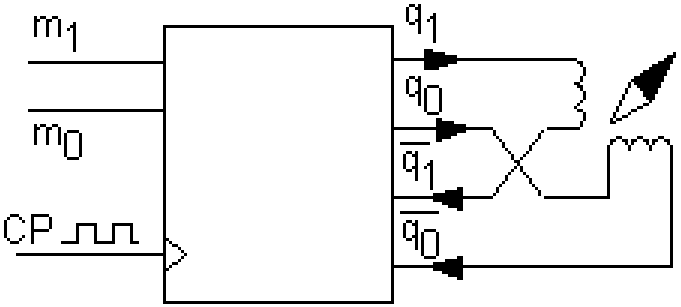
10.9 State diagram

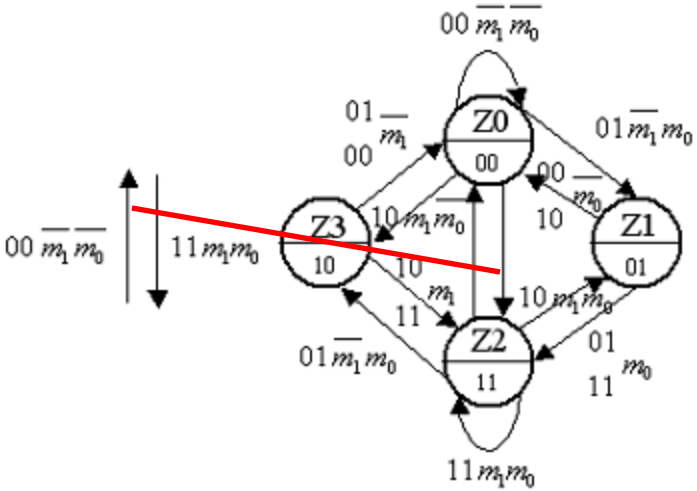
$m_1m_0 = 00 \rightarrow$ Reset (fixed position)

$m_1m_0 = 01 \rightarrow$ count up (cw)

$m_1m_0 = 10 \rightarrow$ count down (ccw)

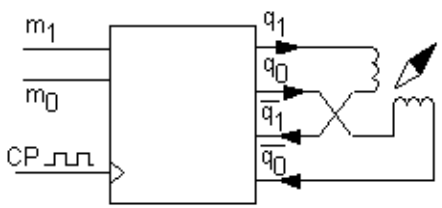
$m_1m_0 = 11 \rightarrow$ Preset (another fixed position)

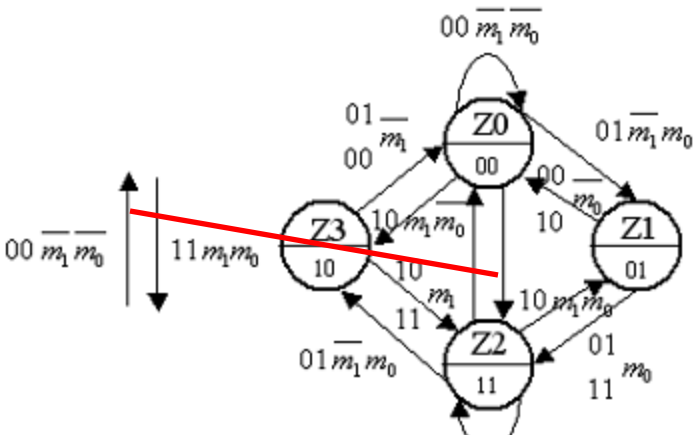




Sometimes you write boolean conditions instead of just the numbers at the arrows. In the figure, both the condition and numbers are used.

10.9 State table and next state decoder





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

$q_1 q_0$ \ $m_1 m_0$	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

William Sandqvist william@kth.se