# Memory technologies

<table>
<thead>
<tr>
<th>Technologi</th>
<th>Access time</th>
<th>Cost $/GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>1 ns</td>
<td>1000</td>
</tr>
<tr>
<td>DRAM</td>
<td>50 ns</td>
<td>100</td>
</tr>
<tr>
<td>HDD</td>
<td>10 ms</td>
<td>1</td>
</tr>
</tbody>
</table>

Fast memory is expensive and inexpensive memories are slow!

Principal figures.

William Sandqvist  william@kth.se
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

**Memory chip:**

- $p$ words, width $q$ bits
- Number of rows $r \leq N/p$
- Number of columns $k \geq M/q$
- Number of chips $K = r \times k$

$$K = r \times k$$
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.
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

William Sandqvist  william@kth.se

32M \cdot 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.

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

Except from the memory cells the chip also contains a lot of other digital circuits.
Column address counter can quickly address of the "neighboring memory cells" - the memory can more quickly deliver a burst with several bytes in sequence, than an totally random access.
Burst provides faster average access

- To access 1 “random” word in the memory takes three buscycles $3T_{Bus}/word$ (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 access 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.

William Sandqvist  william@kth.se
Ex 12.1 Dynamic Memory

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 $K = r \times k = 8 \times 8 = 64$. 
b) How many chips are needed for 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  \( k = \frac{M}{q} = \frac{72}{8} = 9 \).

Number of rows  \( r = \frac{N}{p} = \frac{512M}{32M} = 16 \).

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

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).
b) How many chips are needed for $512\text{M} \times 72$?

**Memory** $N = 512\text{M}$ $M = 72$ bits. **Chip** $p = 32\text{M}$ $q = 8$ bits.

Number of columns $k = \frac{M}{q} = \frac{72}{8} = 9$.

Number of rows $r = \frac{N}{p} = \frac{512\text{M}}{32\text{M}} = 16$.

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

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.
Suppose that the ROM and the SRAM is to be connected to a 16-bit microprocessor having 24 bit addressing.
How big is the figure SRAM, and which is the address area expressed in hexadecimal numbers.
SRAM size?

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

• **Chip:**
  \[ p = 512k \quad q = 8 \text{ bits} \]

• **Memory:**
  \[ r = 3 \quad k = 2 \quad K = 2 \times 3 = 6 \]
  \[ M = k \times q = 2 \times 8 = 16 \text{ bits} \]
  \[ N = p \times r = 512k \times 3 = 1,5M \]

William Sandqvist  william@kth.se
SRAM Control?

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

- **Chip:**
  \[ p = 512k \quad q = 8 \text{ bits} \]

- **Memory:**
  \[ r = 3 \quad k = 2 \quad K = 2 \times 3 = 6 \]

\[ M = k \times q = 2 \times 8 = 16 \text{ bits} \]

\[ N = p \times r = 512k \times 3 = 1,5M \]

\[ \overline{RD} = ? \]

\[ \overline{WR} = ? \]
How big is the figure SRAM, and which is the address area expressed in hexadecimal numbers

- **Chip:**
  \( p = 512k \quad q = 8 \text{ bits} \)

- **Memory:**
  \( r = 3 \quad k = 2 \quad K = 2 \times 3 = 6 \)

\[
M = k \times q = 2 \times 8 = 16 \text{ bits}
\]

\[
N = p \times r = 512k \times 3 = 1.5M
\]

\[
\overline{RD} = \overline{RD}
\]

\[
\overline{WR} = \overline{WR}
\]
SRAM address range?

William Sandqvist  william@kth.se
SRAM address range?

<table>
<thead>
<tr>
<th>Computer:</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decoder:</td>
<td>1</td>
<td>0</td>
<td>0 ... 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mem start:</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>• Begin hex</td>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Mem end:</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>• End hex</td>
<td>B</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
SRAM address range?

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|

William Sandqvist  william@kth.se
Change the address range! ?

Change the address range to 980000 – AFFFFF ?

980000
10|01|1000|0000|0000|0000|0000|

AFFFFF
10|10|1111|1111|1111|1111|1111|1111|

William Sandqvist  william@kth.se
Change the address range! ?

Change the address range to 980000 – AFFFFF ?

980000
10|01|1000|0000|0000|0000|0000|

AFFFFF
10|10|1111|1111|1111|1111|1111|

"10|011" → "3"
"10|101" → "5"
Change the address range!?

Change the address range to 980000 – AFFFFF?

980000
100110000000000000000000

AFFFFF
10101111111111111111111

"10|011" → "3"
"10|101" → "5"
Change the address range! ?

Change the address range to 480000 – 5FFFFFF ?

William Sandqvist  william@kth.se
Change the address range! ?

Change the address range to 480000 – 5FFFFFF ?

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

5FFFFFF
0101|1111|1111|1111|1111|1111|1111|1111|
Change the address range! ?

Change the address range to 480000 – 5FFFFFF ?

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

5FFFFFF
01|01|1111|1111|1111|1111|1111|1111|

"01|001" → "1"
"01|011" → "3"
Change the address range! ?

Change the address range to 480000 – 5FFFFFF ?

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

5FFFFFF  
01|01|1111|1111|1111|1111|1111|

"01|001" → "1"  
"01|011" → "3"
Change the address range! ?

Change the address range to 480000 – 5FFFFFF ?

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

5FFFFFF
01|01|1111|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 \times 16 \text{ bitar address range} \ 000000 \ldots \text{ and forward. ROM Chip } 512k\times8.

- 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.
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 bit 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.

**Memory:**
\[ N = 2 \text{ M} \times (4 \cdot 512k) \text{ word is } M = 16 \text{ bitar} \]

**Memory chip:**
\[ p = 512 \text{ k word is } q = 8 \text{ bitar} \]

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

William Sandqvist  william@kth.se
ROM Control connections?

$RD = ?$
ROM Control connections?

\[ \overline{RD} = \overline{OE} \]
Decoder ROM connection?

Microprocessor
A0-A23

RD
WR

V D0-D15

BIN/OCT
1 2 4
A19
A20
A21
G1
G2A
G2B
EN

A0 ROM
-A18
CS 00
OE -07

A0 ROM
-A18
CS 00
OE -07

A0 ROM
-A18
CS 00
OE -07

A0 ROM
-A18
CS 00
OE -07

D0-D15

D0-D7

William Sandqvist  william@kth.se
Decoder connection?

William Sandqvist  william@kth.se
Decoder ROM addresses?

Decoded inside memory chips

Four rows with memory chips

William Sandqvist  william@kth.se
Decoder ROM addresses?

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

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

0000|0000|0|0|0|0 - 0000|0111|F|F|F|F 000000-07FFFF
0000|1000|0|0|0|0 - 0000|1111|F|F|F|F 080000-0FFFFF
0001|0000|0|0|0|0 - 0001|0111|F|F|F|F 100000-17FFFF
0001|1000|0|0|0|0 - 0001|1111|F|F|F|F 180000-1FFFFF

Totaly ROM 000000 – 1FFFFF

William Sandqvist william@kth.se
Decoder SRAM+I/O addresses?

00ab|c|mmmm|mmmm|mmmm|mmmm|mmmm

William Sandqvist william@kth.se
Decoder SRAM+I/O addresses?

00ab|cmmm|mmmm|mmmm|mmmm|mmmm
0010|0000|0|0|0|0 - 0010|0111|F|F|F|F 200000-27FFFF
Decoder SRAM+I/O adresses?

SRAM as before

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

0010|1000|0|0|0|0 - 0010|1111|F|F|F|F  280000-2FFFFFF
0011|0000|0|0|0|0 - 0011|0111|F|F|F|F  300000-37FFFFF
0011|1000|0|0|0|0 - 0011|1111|F|F|F|F  380000-3FFFFFF

William Sandqvist william@kth.se
Decoder SRAM+I/O adresser?

- Possible SRAM+I/O adresser 200000 – 3FFFFFF
Memory map

The memory map for this example.
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.
I/O addresses, at the decoder output "4", 200000 – 27FFFF according to the earlier task.

William Sandqvist  william@kth.se
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"

Remains to decode:
\[ \overline{A_{18}} \cdot \overline{A_{17}} \cdot \overline{A_{16}} \cdot \overline{A_{15}} \cdot \overline{A_{14}} \cdot \overline{A_{13}} \cdot \overline{A_{12}} \cdot \overline{A_{11}} \cdot \overline{A_{10}} \cdot \overline{A_{9}} \cdot \overline{A_{8}} \cdot \overline{A_{7}} \cdot \overline{A_{6}} \cdot \overline{A_{5}} \cdot \overline{A_{4}} \cdot \overline{A_{3}} \]

William Sandqvist  william@kth.se
Connections
Connections

A bit too many inputs?
incomplete decoding?

Addressing becomes ambiguous!
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.

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 addresses as **volatile**.

This, you will meet in Computer Engineering course.
How to construct a bigger Digital system?
One approach for implementing integer division is to perform repeated subtraction as indicated in pseudo-code.

\[
Q = 0; \\
R = A \\
\text{While } ((R - B) \geq 0) \text{ do} \\
R = R - B; \\
Q = Q + 1; \\
\text{End while;}
\]

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.

William Sandqvist william@kth.se
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;
Q = 0;
R = A
While ((R − B) ≥ 0) do
R = R − B;
Q = Q + 1;
End while;

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

Q ← 0
Start?
0
0

Load R← A
Load B
S1

R ← R - B
Q ← Q + 1
S2

Subtract B! Increment Q!
Q = 0;
R = A
While ((R - B) ≥ 0) do
    R = R - B;
    Q = Q + 1;
End while;

RESET

Q ← 0

Start?

Load R ← A
Load B

0

S1

R ← R - B
Q ← Q + 1

1

S2

R - B ≥ 0?

1

0

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

RESET

S1

Load R ← A
Load B

Q ← 0

Start?

S2

R ← R – B
Q ← Q + 1

R – B ≥ 0?

Done!

William Sandqvist william@kth.se
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.
How to test condition?

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

How do we know when R-B<0 ?

The test that \( R-B \geq 0 \) is done simultaneously with the operation \( R-B \) by inspecting flags.
Do you remember? \( \geq \)

Adder connected as comparator

\[
\begin{align*}
X - Y & \Rightarrow Z = 1 \\
X < Y & \Rightarrow N \oplus V \\
X \leq Y & \Rightarrow Z + N \oplus V \\
X > Y & \Rightarrow \overline{Z} + N \oplus V = \overline{Z} \cdot (N \oplus V) \\
X \geq Y & \Rightarrow \overline{N} \oplus V
\end{align*}
\]

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;
BV 10.5 Hardware

- Hardware block
BV 10.5 ASM control

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

- Control
BV 10.5 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
sin + cos values?

Another bigger Digital system ...

\[ x = x + y / 2 \]
\[ y = y - x / 2 \]

start : \( x = 16 \), \( y = 0 \)
Osquar and Osquulda are implementing a digital design algorithm, to their thesis work. The algorithm calculates sine and cosine values.

\[ x = x + y/2; \quad y = y - x/2; \quad \text{(start values } x = 0, y = 16). \]

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_ : \( x = 16 \), \( y = 0 \)
\[ x = x + y / 2 \]
\[ y = y - x / 2 \]

*start: \( x = 16 \), \( y = 0 \)*
Algorithm for sin + cos

\[ x = 0; \ y = 16; \]

\[ x += y/2; \ y -= x/2; \]
Rehearsal before the exam
Ex 6.10 Combinatorial circuit 5 variables

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

\[
\begin{array}{cccccc|c}
   x_4 & x_3 & x_2 & x_1 & x_0 & f \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 \\
2 & 0 & 0 & 0 & 1 & 0 & 0 \\
3 & 0 & 0 & 0 & 1 & 1 & 0 \\
4 & 0 & 0 & 1 & 0 & 0 & 0 \\
5 & 0 & 0 & 1 & 0 & 1 & 0 \\
6 & 0 & 0 & 1 & 1 & 0 & 0 \\
7 & 0 & 0 & 1 & 1 & 1 & 0 \\
8 & 0 & 1 & 0 & 0 & 0 & 0 \\
9 & 0 & 1 & 0 & 0 & 1 & 1 \\
10 & 0 & 1 & 0 & 1 & 0 & 0 \\
11 & 0 & 1 & 0 & 1 & 1 & 1 \\
12 & 0 & 1 & 1 & 0 & 0 & 1 \\
13 & 0 & 1 & 1 & 0 & 1 & 0 \\
14 & 0 & 1 & 1 & 1 & 0 & 0 \\
15 & 0 & 1 & 1 & 1 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccc|c}
   x_4 & x_3 & x_2 & x_1 & x_0 & f \\
16 & 1 & 0 & 0 & 0 & 0 & 1 \\
17 & 1 & 0 & 0 & 0 & 1 & 0 \\
18 & 1 & 0 & 0 & 1 & 0 & 1 \\
19 & 1 & 0 & 0 & 1 & 1 & 0 \\
20 & 1 & 0 & 1 & 0 & 0 & 0 \\
21 & 1 & 0 & 1 & 0 & 1 & 0 \\
22 & 1 & 0 & 1 & 1 & 0 & 0 \\
23 & 1 & 0 & 1 & 1 & 1 & 0 \\
24 & 1 & 1 & 0 & 0 & 0 & 1 \\
25 & 1 & 1 & 0 & 0 & 1 & 1 \\
26 & 1 & 1 & 0 & 1 & 0 & 1 \\
27 & 1 & 1 & 0 & 1 & 1 & 1 \\
28 & 1 & 1 & 1 & 0 & 0 & 0 \\
29 & 1 & 1 & 1 & 1 & 0 & 1 \\
30 & 1 & 1 & 1 & 1 & 0 & 0 \\
31 & 1 & 1 & 1 & 1 & 1 & 0 \\
\end{array}
\]
6.10 Combinatorial circuit 5 variables

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

\[ f(x_4, x_3, x_2, x_1, x_0) \quad f = ? \]
6.10 Combinatorial circuit 5 variables

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

\[ f(x_4, x_3, x_2, x_1, x_0) \quad f = ? \]
6.10 Combinatorial circuit 5 variables

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

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

\[ f = \overline{x_4} x_3 x_2 + x_3 \overline{x_2} x_0 + x_4 \overline{x_2} x_0 \]

William Sandqvist  william@kth.se
6.10 Combinatorial circuit 5 variables

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

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

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

\[ f(x_4, x_3, x_2, x_1, x_0) = \bar{f} = ? \]

\[ \bar{f} = x_4 x_3 + x_3 x_0 + x_4 x_2 + x_4 x_2 x_0 \]

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 \( \mathcal{U} (u_5, u_4, u_3, u_2, u_1, u_0) \) which is equal to the square of the number \( \mathcal{U} = X^2 \).
# 8.1 Truth Table

\[
\begin{align*}
\text{Truth Table for } & X^2 \\
\text{Input } X & x_2 & x_1 & x_0 & U = X^2 & u_5 & u_4 & u_3 & u_2 & u_1 & u_0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
2 & 0 & 1 & 0 & 4 & 0 & 0 & 0 & 1 & 0 & 0 \\
3 & 0 & 1 & 1 & 9 & 0 & 0 & 1 & 0 & 0 & 1 \\
4 & 1 & 0 & 0 & 16 & 0 & 1 & 0 & 0 & 0 & 0 \\
5 & 1 & 0 & 1 & 25 & 0 & 1 & 1 & 0 & 0 & 1 \\
6 & 1 & 1 & 0 & 36 & 1 & 0 & 0 & 1 & 0 & 0 \\
7 & 1 & 1 & 1 & 49 & 1 & 1 & 0 & 0 & 0 & 1
\end{align*}
\]
8.1 Karnaugh map

Of truth table it shows that $u_1$ always is equal to 0. $u_1$ output 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 therefore be connected directly to $x_0$ input.
Mechanical "squerer"
Brock institute for advanced 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

$m_1m_0 = 00 \rightarrow \text{Reset (fixed position)}$

$m_1m_0 = 01 \rightarrow \text{count up (cw)}$

$m_1m_0 = 10 \rightarrow \text{count down (ccw)}$

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

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

\[ q_1^+ q_0^+ (q_1 q_0 m_1 m_0) \]

\[ m_1 m_0 \]

\[ q_1^+ \]

\[ q_0^+ \]

\[ q_1^+ = q_0 m_0 + \overline{q}_0 m_1 \]

\[ q_0^+ = q_1 m_1 + \overline{q}_1 m_0 \]