automaten übung

david vajda

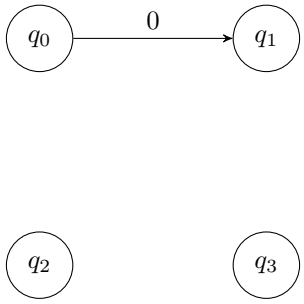
1. September 2026

```
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4   3   2
3   1   4
2   4   1
1   2   3
```

0.1 mealy automat

andere kodierung folgt

- anders nummeriert
- binär kodiert



ok, was wir hier feststellen, fehlt die ausgabe. auf y, sonst: würde mealy und moore überflüssig. dazu muss eine andere kodierung vorgenommen werden.

1 nächster automat

aus den geraden erwähnten grund hier ein neuer automat:

```
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 3 , 1 ) ( 2 , 0 )
3  ( 2 , 0 ) ( 1 , 0 )
2  ( 1 , 0 ) ( 4 , 1 )
1  ( 4 , 1 ) ( 3 , 1 )
```

diesen in einer schöneren tabelle eine möglichkeit der formulierung (meiner meinung nach):

z_n	x ??	$z_{n+1_{x=0}}$	$y_{x=0}$	$z_{n+1_{x=1}}$	$y_{x=1}$
4	$0/1$ (x)	3	1	2	0
3	x	2	0	1	0
2	x	1	0	4	1
1	x	4	1	3	1

1.1 die entstehung dieses automats

zunächst ein mal die frage: wie ist sind diese automaten zu stande gekommen: ich habe ein python3 skript zu grund gelegt. die erste version geht so: die erste version ist diese hier

```
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach

import random
random.seed ()

V=[1,2,3,4]
V1=[1,2,3,4]
V2=[1,2,3,4]

print ("# (c) david vajda")
print ("# 08/27/2026")
print ("# python3, zustaende, vier zustaende")
print ("# ... graph: schaltwerk, einfach")

i=3
while i >= 1:
 j1 = random.randint (0,i-1)
 j2 = random.randint (0,i-1)
 if i+1 in V1:
  yes1 = True
 else:
```

```
 yes1 = False
if i+1 in V2:
 yes2 = True
else:
 yes2 = False
if yes1:
 V1.remove (i+1)
if yes2:
 V2.remove (i+1)
v1 = V1 [j1]
v2 = V2 [j2]
V1.pop (j1);
V2.pop (j2);
if yes1:
 V1.append (i+1)
if yes2:
 V2.append (i+1)

print (i+1, " ", v1, " ", v2, " ")
i = i - 1

v1 = V1.pop (0);
v2 = V2.pop (0);
print (i+1, " ", v1, " ", v2, " ")
```

erzeugt unter anderem diese ausgabe:

```
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4   3   2
3   1   4
2   4   1
1   2   3
```

die vorgängerversion vom bisher letzten, der wohl offensichtlich seinen dienst tut ist die

```
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach

import random
random.seed ()
```

```
V=[1,2,3,4]
V1=[1,2,3,4]
V2=[1,2,3,4]

print ("# (c) david vajda")
print ("# 08/27/2026")
print ("# python3, zustaende, vier zustaende")
print ("# ... graph: schaltwerk, einfach")

i=3
while i >= 1:
 j1 = random.randint (0,i-1)
 j2 = random.randint (0,i-1)
 if i+1 in V1:
  yes1 = True
 else:
  yes1 = False
 if i+1 in V2:
  yes2 = True
 else:
  yes2 = False
 if yes1:
  V1.remove (i+1)
 if yes2:
  V2.remove (i+1)
 v1 = V1 [j1]
 if v1 in V2:
  yes11 = True
 else:
  yes11 = False
 if yes11:
  V2.remove (v1)
 v2 = V2 [j2]
 V1.pop (j1);
 V2.pop (j2);
 if (v1 != v2) and yes11:
  V2.append (v1)
 if yes1:
  V1.append (i+1)
 if yes2:
  V2.append (i+1)

 print (i+1, " ", v1, " ", v2, " ")
 i = i - 1

v1 = V1.pop (0);
```

```
v2 = V2.pop (0);
print (i+1, " ", v1, " ", v2, " ")
```

diese version enthält einen fehler und die ausgabe beim automaten wurden nicht berücksichtigt.

```
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach

import random
random.seed ()

V=[1,2,3,4]
V1=[1,2,3,4]
V2=[1,2,3,4]

print ("# (c) david vajda")
print ("# 08/27/2026")
print ("# python3, zustaende, vier zustaende")
print ("# ... graph: schaltwerk, einfach")

i=3
while i >= 1:
 j1 = random.randint (0,i-1)
 j2 = random.randint (0,i-1)
 if i+1 in V1:
  yes1 = True
 else:
  yes1 = False
 if i+1 in V2:
  yes2 = True
 else:
  yes2 = False
 if yes1:
  V1.remove (i+1)
 if yes2:
  V2.remove (i+1)
 v1 = V1 [j1]
 if v1 in V2:
  yes11 = True
 else:
  yes11 = False
 if yes11 and (i > 1):
  V2.remove (v1)
```

```
  j2 = random.randint (0,i-2)
 v2 = V2 [j2]
 V1.pop (j1);
 V2.pop (j2);
 if (v1 != v2) and yes11:
  V2.append (v1)
 if yes1:
  V1.append (i+1)
 if yes2:
  V2.append (i+1)

 print (i+1, " (", v1, ",", random.randint(0,1), ") (", v2, ",", random.randint(0,1), ")")
 i = i - 1

v1 = V1.pop (0);
v2 = V2.pop (0);
print (i+1, " (", v1, ",", random.randint(0,1), ") (", v2, ",", random.randint(0,1), ")")
```

dieser automat erzeugt zum beispiel folgende ausgabe:

```
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 2 , 0 ) ( 3 , 0 )
3  ( 1 , 1 ) ( 2 , 0 )
2  ( 4 , 0 ) ( 4 , 1 )
1  ( 3 , 0 ) ( 1 , 1 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 1 , 1 ) ( 3 , 1 )
3  ( 2 , 1 ) ( 1 , 1 )
2  ( 4 , 0 ) ( 4 , 0 )
1  ( 3 , 1 ) ( 2 , 1 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 2 , 1 ) ( 1 , 0 )
3  ( 4 , 0 ) ( 2 , 0 )
2  ( 1 , 0 ) ( 4 , 1 )
```

```
1  ( 3 , 1 ) ( 3 , 1 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 3 , 1 ) ( 1 , 0 )
3  ( 2 , 0 ) ( 4 , 0 )
2  ( 1 , 1 ) ( 3 , 0 )
1  ( 4 , 0 ) ( 2 , 0 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 3 , 0 ) ( 2 , 1 )
3  ( 1 , 1 ) ( 4 , 0 )
2  ( 4 , 0 ) ( 1 , 0 )
1  ( 2 , 0 ) ( 3 , 0 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 1 , 0 ) ( 3 , 0 )
3  ( 4 , 0 ) ( 2 , 1 )
2  ( 3 , 0 ) ( 1 , 0 )
1  ( 2 , 1 ) ( 4 , 0 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 1 , 0 ) ( 2 , 1 )
3  ( 4 , 0 ) ( 1 , 1 )
2  ( 3 , 1 ) ( 4 , 1 )
1  ( 2 , 1 ) ( 3 , 1 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 2 , 0 ) ( 3 , 1 )
3  ( 1 , 1 ) ( 2 , 1 )
2  ( 4 , 1 ) ( 4 , 0 )
1  ( 3 , 1 ) ( 1 , 1 )
```

```
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 1 , 1 ) ( 2 , 0 )
3  ( 4 , 0 ) ( 1 , 0 )
2  ( 3 , 0 ) ( 4 , 0 )
1  ( 2 , 0 ) ( 3 , 0 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 1 , 0 ) ( 2 , 0 )
3  ( 4 , 0 ) ( 1 , 1 )
2  ( 3 , 1 ) ( 4 , 1 )
1  ( 2 , 0 ) ( 3 , 1 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 1 , 1 ) ( 2 , 0 )
3  ( 4 , 1 ) ( 1 , 0 )
2  ( 3 , 1 ) ( 4 , 1 )
1  ( 2 , 1 ) ( 3 , 1 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 2 , 0 ) ( 1 , 0 )
3  ( 1 , 0 ) ( 4 , 0 )
2  ( 4 , 1 ) ( 3 , 1 )
1  ( 3 , 0 ) ( 2 , 1 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 1 , 0 ) ( 3 , 1 )
3  ( 2 , 0 ) ( 1 , 1 )
2  ( 4 , 1 ) ( 4 , 1 )
1  ( 3 , 0 ) ( 2 , 1 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
```

```
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 1 , 0 ) ( 3 , 1 )
3  ( 2 , 1 ) ( 1 , 0 )
2  ( 4 , 0 ) ( 4 , 0 )
1  ( 3 , 0 ) ( 2 , 1 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 2 , 0 ) ( 3 , 0 )
3  ( 1 , 0 ) ( 2 , 0 )
2  ( 4 , 0 ) ( 4 , 0 )
1  ( 3 , 0 ) ( 1 , 1 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 1 , 0 ) ( 2 , 0 )
3  ( 2 , 0 ) ( 1 , 0 )
2  ( 4 , 1 ) ( 4 , 0 )
1  ( 3 , 1 ) ( 3 , 0 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 2 , 0 ) ( 3 , 1 )
3  ( 1 , 0 ) ( 2 , 0 )
2  ( 4 , 1 ) ( 4 , 0 )
1  ( 3 , 1 ) ( 1 , 1 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 3 , 1 ) ( 1 , 1 )
3  ( 1 , 1 ) ( 4 , 1 )
2  ( 4 , 1 ) ( 3 , 0 )
1  ( 2 , 1 ) ( 2 , 0 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
```

```
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 3 , 1 ) ( 2 , 0 )
3  ( 2 , 0 ) ( 1 , 0 )
2  ( 1 , 0 ) ( 4 , 1 )
1  ( 4 , 1 ) ( 3 , 1 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 3 , 1 ) ( 2 , 1 )
3  ( 2 , 1 ) ( 1 , 1 )
2  ( 1 , 0 ) ( 4 , 1 )
1  ( 4 , 1 ) ( 3 , 1 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 2 , 0 ) ( 3 , 0 )
3  ( 1 , 0 ) ( 2 , 0 )
2  ( 4 , 0 ) ( 4 , 1 )
1  ( 3 , 1 ) ( 1 , 0 )
davidvajda@localhost ~ % python3 graphautomat08272026C.py
# (c) david vajda
# 08/27/2026
# python3, zustaende, vier zustaende
# ... graph: schaltwerk, einfach
4  ( 2 , 0 ) ( 1 , 1 )
3  ( 4 , 1 ) ( 2 , 0 )
2  ( 1 , 1 ) ( 4 , 1 )
1  ( 3 , 1 ) ( 3 , 1 )
davidvajda@localhost ~ %
```

was haben wir eigentlich getan. in der fernuni-hagen hatte ich bei

- ungerer
- keller
- schiffmann
- bähring

wohl noch weitere. für computersysteme 1.
und prüfung abgenommen, cornelia dallinger, computersysteme 1/2. hier werden, in computersysteme 1, die zustandsgraphen die automaten beschreiben zum thema (ausdruck). in algorithmischer mathematik wird die graphentheorie zum inhalt gemacht.

als ich auf die prüfung computersysteme 1/2 lernte, simulierte ich so mehr schlecht als recht automaten, um es zu üben, um so die prüfung mit einer 3.7 zu bestehen. ich weise sowohl auf die bestandene prüfung hin. zwar sind das benotete prüfungen was an der fernuni-hagen nicht die einzigen sind, sondern für den ba. sind sie mehr oder weniger pflicht und nur studenten mit abitur sind zu gelassen, ob sie sich nun den B.A. holen oder nicht. die unbenoteten prüfungen sind nicht immer einfach, nur: in diesem falle ist computersysteme 1/2 so wollte ich schon sagen, wohl das kalkül der technischen informatik überhaupt. die informatik besteht wohl aus drei säulen: technische, praktische und theoretische. ich denke aber die technische ist am ende die entscheidenste (a) - trotz allem - ist ansichtssache, aus meiner sicht. nicht nur: es gibt vielleicht verschiedene formen technische informatik zu prüfen, ich denke die einzig gültige ist am ende computersysteme 1/2. es ist eher das gesamte kalkül. wenn man an den automaten denkt, indem der computer mit mikroprogrammspeicherwerk, das programm in einer nicht veränderten version, deswegen doppelt so gross, wegen der verzweigung, wie in den eingescanten per hand geschriebenen dokumenten steht: dann ist technische informatik. . . . ein sehr starkes thema. ich möchte mich kurz fassen.

ich bin zu frieden. ich selber gedenke noch ein paar prüfungen zu tun, Bachelor hole ich mir eher nicht. etwas das ich versuchen werde ist wohl: algorithmische mathematik

und hier werden eben graphen beschrieben. das ist so oder so nicht irrelevant. aus folgendem grund:

in algorithmischer mathematik werden graphen vorgestellt. dabei

1. tiefensuche
2. breitensuche
3. eulertour

die eulertour ist sehr interessant, gucken wir in meine homepage. sie bezieht sich auf graphen, die keine bäume sind. tiefen und breitensuche finden einen baum. in jedem graphen, sie verbinden, besonders mit stapel und schlangen, algorithmisch alle möglichen knoten im graph.

über das üben dieser habe ich gelernt die eulertour in python3 um zu setzen. wie hier, mit einer art, schlange oder stapel, jedenfalls elemente herausnehmen,

erreichen wir bei einem graphen unkompliziert, dass mehr oder weniger: elemente mehr als nötig kanten zu nachbarn habe, dafür knoten ausgelassen werden, unzusammenhängende graphen, laut algorithmischer mathematik.

ok: ich möchte an dieser stelle auf unterschiede und gemeinsamkeiten zu meinen bisherigen übungen verweisen.

zunächst bevor ich diese graphentheorie kannte, hatte ich bei den nachbarn das übliche problem. am ende habe ich mehr oder weniger zufallszahlen genommen, durch umsortieren solange, bis es passt. das ist keine besonders gute lösung in der graphentheorie selber, aber nicht untersucht, (beispiel: permutation - umbennenung von knoten, wobei der name eines knotens generell keine rolle spielt - und ich weise darauf hin - da ich arabisch ebenso übe - und auch hieroglyphen hier auch frei schreiben will, aber beschlossen habe nicht mehr handschriftlich zu schreiben, werde ich vielleicht nicht unbedingt aber vielleicht auch diesen graphen hier mit arabischen ziffern ausstatten, z.B. Hieroglyphen, stuhl und tisch? oder eben binärkodiert).

ich möchte auf den unterschied der darstellung hinweisen, bezüglich des „Informatischen“inhalts. bzw. der darstellung in der technischen informatik.

hier sind automaten, das wurde bei meinen bisherigen übungen zunächst vernachlässigt, ich weise auf unterschiede und gemeinsamkeiten hin

1. unär
2. binär

kodiert. unär bedeutet, zunächst habe ich computersysteme 1/2 auswendig gelernt

```
von null beginnend, un"ar kodiert, konsekutiv nummeriert
```

wichtig ist, dass gegenbeispiel IP-Addresse
es taucht oft die verwirrung auf, jedenfalls bei mir damals am technischem gymnasium: gibt es, wenn man diese verwirrung zu wort bringt, IP-Addressen der form

```
1011 1101.0011 1010.0000 0000.0000 0000
```

nein es gibt nur:

```
...
1111 1111.1111 11111.1111 0000.0000 0000
...
1111 1111.0000 0000.0000 0000.0000 0000
1111 1110.0000 0000.0000 0000.0000 0000
```

```
1111 1100.0000 0000.0000 0000.0000 0000
..
0000 0000.0000 0000.0000 0000.0000 0000
```

das heisst, die msb 1er bit werden voll aufgefüllt. ein passendes beispiel von codes ist 2 aus 5. hier gibt es 2 einer bit in 5. dabei rueckt das lsb immer eines vor, bis die immer nach rechts (lsb) weiter vorrueckende grenze erreicht ist. inzwischen rueckt das msb 1er bit vor. darauf hin das lsb 1er bit zu der grenze die darauf nach rechts vorrueckt, waehrend das msb 1er bit wieder auf den anfang zurueck geht.

das entspricht so einem vorgang, als beispiel einer schiften, oder ip-addresse, aber eben zerfaellt die sache hier.

ein anderes beispiel ist die unäre kodierung. striche an die wand machen, beim aller einfachsten zaehlen, bedeutet ip-addresse. denken wir an: dekodierer, ich weise darauf hin, es gibt:

1. kodierer
2. dekodierer
3. multiplexer
4. demultiplexer

ein dekodierer liesse sich noch einigermassen mit einem demultiplexer erreichen, wuerde ich aus dem kopf sagen. ich weise aber auf eindeutigen unterschiede hin.

ein dekodierer ist ein kodierer.

ein multiplexer ist quasi ein element, was serialisiert - als multiplexed. (rs232 - v.24) - es legt ein 8 bit breites byte mit einem takt - zeitlich auf eine leitung. das gegenteil ist der demultiplexer. ein dekodierer ist immer noch ein kodierer (DIN 44300/183) (siehe auch: DIN 40700, DIN 40900) ist ein schaltnetz, das die zeichen eines zeichenvorrats in die zeichen eines andereren zeichenvorrats umsetzt (Computersysteme 1/2). ok: blos: ein zeichen in der informatik ist binär kodiert (siehe BCD: ausser am Sichtgeraet: siehe BCD und Abakus, warum: die perlen im Abakus, sind in einer zeile das was eine ziffer, arabisch und indisch ist (oder chinesisch), und die binärkodierung am LCD HD 44780, kann eben Chinesisch, arabisch, hebraeisch, inka, maja, azteken, brahmanische schriften, latein und so weiter sein). ok: BCD, entspricht trotzdem einer möglichen anderer kodierung der Arabischen bzw. indischen ziffern.

ok: ein dekodierer wiederum ist ein kodierer. das heisst, der zielzeichenvorrat ist von der wortbreite wohl meist ein weniger kleiner oder manchmal vielleicht ein weniger groesser, oder gleich, eher gleich oder kleiner, wie die in der quelle. ausgangsmenge. wie auch immer. der dekodierer ist ein kodierer. und zwar:

einer bei dem für jedes zeichen in der ausgangsmenge immer nur eine leitung gleichzeitig an ist.

ok: man kann also hier sehen: unär kodierung entspricht jetzt eher dem, das bit wandert, wie geschiftet.

typischer befehl: SLL, SLR - shift logical left/right. wenn ein bit im quelloperanden, 1 ist dann prima. ok:

andere unterschiede:

1. würde ich von der binären kodierung ausgehen, was laut computersysteme 1/2 möglich ist, dann: gibt es einen vorteil, die bitbreite ($lg(ceil(n))$ - oder floor. da ich hier romane schreibe wie also auf dem papier, sage ich hier ehrlich was sache ist. es gibt den 2er, 10er und e logarithmus, das erkläre ich hier nicht. der 10er logarithmus ist wohl ld und der 2er logarithmus wohl: lg wobei ich nachschauen könnte, jetzt aber verzichte, eine andere schreibweise und die ist immer möglich: (log_2), also die basis unten angeben, das ist das üblichste, ich glaube, das d steht fuer dezimal. und ich glaube, bei dem Dezibel wird $ld\left(\frac{P_{in}}{P_{out}}\right)$ benutzt, eingangs und ausgangsleistung, laesst sich wohl auch umdrehen, jedenfalls, dezibel, im gegensatz, zum bel und ich glaube da steht, ld. 10er. so: ansonsten basis unten angeben. wo ceil und wo floor liegt weiss ich nie.

   jedenfalls: man muss aufrunden und den 2er logarithmus auf die anzahl der knoten anlegen, und wenn ich das habe. dann erhaelte ich die bitbreite. ok:

   etwas, das die sache nun klassisch verändern würde ist folgendes: brauche ich beim binären einen addierer? habe ich damit alles falsch gemacht?

   (a) ich sage: ja und nein, nein deswegen weil es schaltet, trotzdem krotten falsch, warum?

   (b) weil wir in computersysteme 1 und 2 gelernt haben: übergangsschaltnetz und ausgangsschaltnetz.

   (c) die zustände selber sollten einfach fluppen, ok: hier ist es alles andere als krotten falsch, warum? weil ich mit addieren durch die graphentheorie nicht weit komme, eben dann sind graphen nicht einfach durch permutation eigentlich der gleiche, sondern wenn ihre kanten sich massgeblich unterscheiden. ok: addition hilft hier nichts

   (d) die aeussere darstellung ist aufgabe des uebergangsschaltnetz. und leider trotzdem falsch.

   (e) warum? eben deswegen nehme ich hier: unär und eben deswegen: mealy und moore und ich nehme unär

   (f) denn: ein unäres schaltwerk von den zuständen einfach durch schiften. über das übergangsschaltnetz was nun kein addierer ist, einfach dann

mit der eingabe die naechsten zustand erzeugen. falsch ist: mealy erfordert: bei der ausgabe wird die eingabe mitgenommen

(g) doch damit ist die sache nicht falsch. denn erstens ist der addierer ein schaltnetz und zweitens: auch beim unären entstehen hier variationen...

ok: wo weit so gut, nur:

es gibt noch hinsichtlich der tabelle alternative darstellungen. was in der mathematik üblich wäre.

das habe ich mir im smartphone auf facebook gemerkt:

```
a war so noch ein Fehler, der ist behoben das war klar, der ist so noch behoben plus, dass i

Das schreib ich dir noch rein in das PDF, was ich mache, wenn wir eine Null oder eins zu den

Daraus ein Baum machen, aber nur bedingt, weil das ist dann ein pattern  Matching

Was ich damit endlich beschreiben kann, was nichts mit KMP zu tun hat, sondern halten Automa

Das Problem bei dem Baum ist ja ich kann jetzt von einem Knoten aus dem Baum beschreiben für

Schreibe ich ins PDF rein wie gesagt aber auch noch ein Fehler

Den hab ich behoben bloß der Unterschied ist

Am was bei Algorithmen in C steht

Robert sedgewick
Heißt das der sozusagen der Compiler für die formale Sprache an der Stelle, die sich auch in
```

ich bin dahinter gekommen, dass: und das ist mir wichtig, dass eigentlich: ein mikroprogrammsteuerwerk genauso harter speicher wie bei der havard architektur sein könnte wie: diese selber, aber einen doppelten speicher braucht, wegen der möglichen verzweigung oder die addresse ist im wort (konstante, immidiate) im wort fuer das operationswerk vorhanden, oder andere version: weil es nicht immer nur bedingungen gibt, ein kleiner speicher hinten angehaengt.

ok. das ist die eine erkenntnis, wie ich sie in den papieren habe, die hier anhaenge. das andere ist aber: das bezog sich nicht darauf.

1.2 tabelle

z_n	x ??	$z_{n+1_{x=0}}$	$y_{x=0}$	$z_{n+1_{x=1}}$	$y_{x=1}$
4	0/1 (x)	3	1	2	0
3	x	2	0	1	0
2	x	1	0	4	1
1	x	4	1	3	1

nun zu erst, die alternative darstellung der tabelle

hier wie gerade eben.

ich weise noch darauf hin, noch mal, ..., ich mache um papier zu sparen, aber um mit TikZ von LaTeXimmer gleich schaltpläne ieee-91/1984, automatendiagramme, zu tun... immer gleich alles mit LaTeX. durchaus mit kaneda. wobei LaTeXund TikZ ist zwang. es geht darum, dass ich einen grossrechner baue, (a) der form uniac, eniac usw. dies müsste mit transistoren machbar sein. über legierung lassen sich vielleicht, ich sage vielleicht, in einem hochofen der marke vevor oder alternative von denen ich destilliergeräte habe, für einen relativ erschwinglichen preiss, etwa 700 Euro aufwärts vielleicht legierungen herstellen: verhältnis z.B. 80:20. dies liesse sich eher mit gallium, als germanium tun z.B. einige haben komplizierte kühlverfahren mit bor oder brohm, sagt sich schwierig, schon der begriff dunkelkammer ist in der chemie oft keine schwierigkeit, legierungen lassen sich, vermutlich relativ einfach machen, mein onkel mütterlicherseits war goldschmied, silber in den ringen kaufen 825. 925 z.B. und legierung beschrieben in elektrotechnik gelb, autor jetzt nicht auswendig im gegensatz zum transistor, sage ich ja: müssen wir machen und sind wie wir sind. ok: köennte machbar sein, dann mit P und N und halbleiter auf halbleiter wie elektrotechnik kontakt an kontakt, will bild ersten transistors auf wikipedia gesehen haben, aber: jetzt ALU bestellt, mükra, spott günstig, damit lässt sich mit speicher und regsiter einfach cpu machen, dann transistor, habe teile eines univacs auf ebay gefunden, platine mit transistor. man könnte einfach: dann modul für modul. dafür aufbau von logikgatter mit transistor üben. univac ähnliches ding von mir: gekaufter transistor, dann später oder modul für modul, oder teilweise, eigene transitoren, wäre auch eine möglichkeit: gekaufte plus module mit eigenen... ok: deswegen auch mit kaneda, schaltpläne mit transistor üben.

jetzt die tabelle anders gestellt. was sind noch aufgaben? die transformation von mealy und moore

ich weise noch darauf hin, dass ich nicht vorhabe das univac ähnliche ding, dann als endziel erreicht zu haben und weiter nichts zu tun, die übungen sind standard.

ok: und die transformation ist wichtig und dann noch folgendes: ich konstruiere das als schaltnetz und so weiter, als plan, aber wenn erreicht, noch: wie gesagt: und zwar unär dies mal, und dazu noch: dass man das wort des automaten nimmt, mit ausgabe $y = 0$ oder 1 und, dass man: dadurch auch Pattern Matching erreicht. ok.

wenn das so weit ist, muss ich noch vorstellen, was ich LaTeXüber TikZ weiss, das ist hier auch noch thema, damit die zeichnung so schön tut, das ist mit teil der übung

erst trotzdem die tabelle:

eine andere tabelle hat doppelt so viele zeilen: weil man ein mal von $x = 0$ und ein mal von $x = 1$ ausgeht.

z_n	x ??	$z_{n+1_{x=0}}$	$y_{x=0}$	$z_{n+1_{x=1}}$	$y_{x=1}$
4	0/1 (x)	3	1	2	0
3	x	2	0	1	0
2	x	1	0	4	1
1	x	4	1	3	1

hier:

z_n	x	z_{n+1}	y
4	0	3	1
4	1	2	0
3	0	2	0
3	1	1	0
2	0	1	0
2	1	4	1
1	0	4	1
1	1	3	1

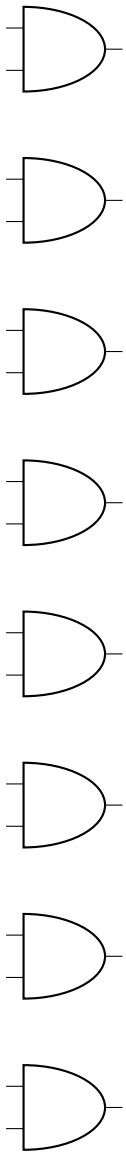
hier noch mal, weil es so schön ist, und papier mit schönheit zusammenhängt.

ok, dann geht es um die kodierung der zustände. während, $4, 3, 2, 1$ ja eine möglichkeit wäre als menge V schreiben, V Vertices (Knoten) und: müsste man hier bei 0 beginnen und die übersetzung wäre unär und binär kodiert:

das ist mealy:

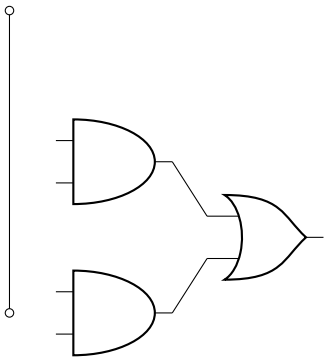
z_n	x	z_{n+1}	y
4	0	3	1
4	1	2	0
3	0	2	0
3	1	1	0
2	0	1	0
2	1	4	1
1	0	4	1
1	1	3	1

jetzt kommen wir zum thema: tikz. der unterschied, das hier ist ein übungsskript. das heisst, ich mache jetzt, das was ich sonst tue, ich lerne es nicht laut auswendig aber schreibe es wiederholt hin, warum? damit es im kopf sitzen bleibt. ich probiere zwar, lange rede kurzer sinn, hier die übung für automaten zu machen, die übung beinhaltet TikZ. und TikZ enthält Knoten für AND OR und Knoten. im Graphen. so, damit schreibe ich die wiederholt, weil das ist teil der übung. zunächst die TikZ darstellungen von denen ich ausgehe:



2 exersize

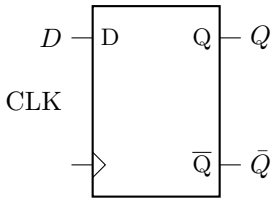
- Make a lot of and from top to bottom, make a DNF

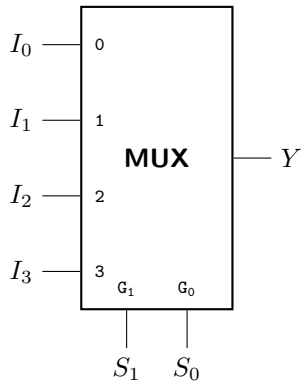


das sieht als code so aus.

```
\begin{circuitikz}
 \node [and port] (and00) at (0,0) {};
 \node [and port] (and01) at (0,2) {};
 \node [and port] (and02) at (0,4) {};
 \node [and port] (and03) at (0,6) {};
 \node [and port] (and04) at (0,8) {};
 \node [and port] (and05) at (0,10) {};
 \node [and port] (and06) at (0,12) {};
 \node [and port] (and07) at (0,14) {};
\end{circuitikz}
\section{exersize}
\begin{itemize}
\item Make a lot of and from top to bottom, make a DNF
\end{itemize}
\begin{circuitikz}
 \draw (0,0) to[short, o-o] (0,4);
 \node [and port] (and00) at (2,0) {};
 \node [and port] (and01) at (2,2) {};
 \node [or port] (or00) at (4,1) {};
 \draw (and00.out) -- (or00.in 2);
 \draw (and01.out) -- (or00.in 1);
\end{circuitikz}
```







das sieht als code so aus:

```
\begin{circuitikz}
% Configure global block options for a rectangular look
    \ctikzset{blocks/scale=1.5, blocks/fill=gray!10}

    % Draw the main multiplexer body as a rectangular block
    \node [draw, minimum width=2cm, minimum height=4cm, thick] (mux) at (0,0) {};

    % IEEE 91-1984 Center functional label
    \node [anchor=center, font=\sffamily\bfseries] at (mux.center) {MUX};

    % Inputs (Left Side) with IEEE dependency tagging (G for control, numbers for channels)
    \draw (mux.west) ++(0, 1.5)  coordinate (in0) -- ++(-0.5,0) node[left] {$I_0$};
    \draw (mux.west) ++(0, 0.5)  coordinate (in1) -- ++(-0.5,0) node[left] {$I_1$};
    \draw (mux.west) ++(0, -0.5) coordinate (in2) -- ++(-0.5,0) node[left] {$I_2$};
    \draw (mux.west) ++(0, -1.5) coordinate (in3) -- ++(-0.5,0) node[left] {$I_3$};

    % Interior labels for internal dependency paths
    \node [anchor=west, font=\scriptsize\ttfamily] at (mux.west) [yshift=1.5cm, xshift=2pt]
    \node [anchor=west, font=\scriptsize\ttfamily] at (mux.west) [yshift=0.5cm, xshift=2pt]
    \node [anchor=west, font=\scriptsize\ttfamily] at (mux.west) [yshift=-0.5cm, xshift=2pt]
    \node [anchor=west, font=\scriptsize\ttfamily] at (mux.west) [yshift=-1.5cm, xshift=2pt]

    % Control Select Lines (Bottom Side) acting as G-dependency inputs
    \draw (mux.south) ++(-0.4, 0) coordinate (s1) -- ++(0,-0.5) node[below] {$S_1$};
    \draw (mux.south) ++(0.4, 0)  coordinate (s0) -- ++(0,-0.5) node[below] {$S_0$};

    % Internal control mapping text (G stands for AND-dependency under IEEE-91)
    \node [anchor=south, font=\scriptsize\ttfamily] at (mux.south) [xshift=-0.4cm, yshift=2p
    \node [anchor=south, font=\scriptsize\ttfamily] at (mux.south) [xshift=0.4cm, yshift=2pt

    % Output (Right Side)
    \draw (mux.east) -- ++(0.5,0) node[right] {$Y$};
    \end{circuitikz}
```

und jetzt vieles üben wird durch das ständige hinschreiben von AND PORT usw. getan. und diese beispiele hier sind übrigens, mit google AI/KI entstanden. das heisst, es sind klassische beispiele für TikZ, aber CircuiTikZ. und: sie sind mit google ki entstanden, ich übe dann ein paar mal das aufschreiben von Port And und so weiter, dann kommt der automat, dann mealy wie hier, dann moore, dann als schaltplan. und dann:

die ausgabewörter. wenn es so weit kommt, sage ich mal: dann vielleicht noch mit gattern in der schaltung realisieren, das wort auf dem oszilloskop angucken.

halt falsch! mir kam dann wieder in den sinn es auch mit abel zu machen! ich sage falsch! falsch! falsch!

ich habe ja gesagt: ich wollte es nur noch schriftlich machen, ohne das bemühen der bauteile. weil die bauteile kosten geld, sollen da sein, sollen nicht so sehr benutzt werden.

dasselbe gilt für abel und den automat. ABEL - HDL - Hardware description language. genauso wie auswendig wissen, wie geht GAL - Generic Array Logic, kein Gate Array Logic. OK: trotzdem: das sollte auf dem papier geschehen. auf dem papier habe ich festgestellt, dass das papier knapp wird und dass ich eigentlich TikZ brauche, das heisst, ich nehme immer noch keine schaltung. ich habe festgestellt, dass ich codes genauso auf papier schreiben kann, was atmega8 betrifft, aber auch galasm, wie ABEL, ABEL und galasm wie palasm, aber bei linux galasm, selbe syntax: eine sprache HDL - jedenfalls: auch das kann ich auf papier machen, da ich auf papier verzichte am computer, aber das muss ja nicht in die hardware rein. schon hier stellt sich die frage nach einem simulator. ich nehme einen simulator. ich benutze auch hardware, nur jetzt nicht, wichtig ist, den quelltext der auch schon geübt wurde, über den keine unklarheit besteht, immer wieder aufschreiben, der simulator ist für mich keine pflicht, es stellt sich an der stelle die frage, ob man das benutzt.

also: auf jedenfall: keine hardware ständig, dann: papier, aber eben nicht papier, doch computer und vielleicht simulator.

ok: jetzt das üben des TikZ. und wenn ich das habe, mealy und moore. und wenn das, das wort und dann noch als schaltung in IEEE-91/1984. ok, mit gattern und so weiter.. und dann kommen hier noch die papiere vom komplexen schaltwerk rein, die einzige übung mit LaTeXist dann maximal das ASM-Diagramm, das muss geübt werden und: sonst papiere rein und: später noch: das Quine, dazu schaltplan.... und ein balkendiagramm, aber ohne: oszilloskop benutzen.

3 Erwähnung zwischendrin

Das schreibe ich nachher ins Skript rein. Ich geb dann noch mal zu. Ich hab von meiner Schwester malen Modellbaudetektor Empfänger bekommen.

Sag dann aber auch die ganze Wahrheit. Ein Klassenkamerad von mir. Damals hat über meinen Computer hergezogen das ist der IBM 5140 der jetzt zum Glück zu mir kommt vielleicht zurecht, weil die Firma sammelt die wohl alle ein und es kann sein, dass sie das dann bei mir ist ja gut aufgehoben.

Klassenkamerad fing damals mit 32 Bit an und 486 und 386 Intel daraufhin habe ich Stress geschoben und das schöne Ding kaputt gemacht. Jetzt kommt es zum Glück wieder. Bei mir ist es gut aufgehoben.

Dann gebe ich zu, dass ich einen Modellbaukasten von meiner Schwester mit einem Diodenempfänger hatte. Natürlich ist die Diode eine einfache AA Diode und gerade das macht auch der Schöne aus. Es war kein alter Drehkondensator. Ich schreib das hier nachher noch ins Skript rein. ich kann es ja genauso übernehmen.

Stellt sich die Frage, muss ich das noch mal schreiben? Nein ich hab vorher ja schon was rein kopiert

Auf die Frage, ob ich Steckbretter nehme ja ich löte nicht

Das ist ja sowieso nachher in das Script ver wurschtelt wird ist es egal

Ich löte nicht ich hab zwar ein Lötkolben ich hatte auch ein besseres ich kann als Wiedergutmachung für die zwei Transistoren bei meinem Kollegen sagen. Ich hab ja mein Zeug an die Straße gestellt es wurde benutzt, aber es hat 100 prozent funktioniert. Besonders das Oszilloskop.

Zwei stk500 jedenfalls mein Kollege, der Johannes fast hatte damals. Ich hoffe nicht, dass der dies ist, der mich hier auf Mikrocontroller. Netz angemacht hat. Es war trotzdem bisschen seltsam, und ich glaube, dass Frau Baerbock, die hat die Gruppe hier so im Griff, aber im Blick glaube ich, und es ist keine Durcheinander. Theorie von Psychologen, Herr Psychologie. Sie müssen sich da nicht vor allen Dingen sie runterschrauben und auch die Theorien von manchen

Leuten das technische informatiktechnische Informatik ist. Es stimmt so nicht.

Und wie gesagt, wenn es bis nach zu Frau Thunberg reicht wegen zwei Transistoren aus einem Modellbau, der Johannes fast und sag ich mal ich bin trotzdem auf das tg

Die hass Mentalität dieser Gruppe dieser Gruppen, teilweise ist bezeichnet sich auch in klügeren Schritt überlegen können wir kommen wir den Ant Transistor. Stattdessen kommt ein Akte. überheblichkeit wie auch Frau Baerbock, die immer irgendwas hin will, guckt er wohl hierhin ich hab ein Graffiti hier an der Wand gefunden. Da steht Baerbock drauf, das stand wohl schon länger drauf. Was auch immer ist ich weiß es nicht. Ich sag jetzt folgendes.

Also ob die damit weiß ich nicht aber diese ganzen Gruppierung gucken auf jedenfalls meine Schwester hatte mir ein Detektor Empfänger mit AA Diode gekauft. Glas

Und ich wusste, wie es geht. Leider hab ich den auch nicht mehr zum Thema Steckbrett, sag ich folgendes

Ich weiß nicht, was der Johannes meint und ob der das auf Mikrocontroller.net ist. Ich sag nur mal folgendes folgendes also tatsächlich hatte der damals auch einen Baukasten und dafür entschuldige ich mich. Ich habe dafür stk 500 an die Straße gestellt, die ich übrigens von dem Geld meiner Mutter gekauft hatte, die ich von dem Geld meiner Mutter gekauft hatte, als ich bei ihr gewohnt hat deshalb wahrscheinlich ein Nachbar an sich genommen und er hat es zurecht. Bitte nicht kaputt machen weiter benutzen das ist ihr Zeug. Ich verfüge darüber nicht. Wahrscheinlich waren sie das das ist ihr Zeug. Ich verfüge darüber nicht. Meine meisten Computer wurden als Geburtstagsgeschenke von meinen Eltern gemacht. Plus übrigens, diesen Dingern, die hab ich bei meiner Mutter gekauft und Oszilloskop damals günstiger an damit denke ich hoffe ich mal und das sage ich jetzt mal unkompliziert. Es gibt Leute, die machen immer komplizierter. Er ist wieder Gutmachung für die zwei Transistoren. Das war auch ein Baukasten. Dabei waren Plan. Wie funktioniert ein Transistor mit Schleuse das hatte der als Plan. Das hatte der als Plan und zwei Transistoren aus dem Baukasten. Diese zwei Transistoren hab ich wahrscheinlich. Entweder ich hab sie ihm sogar wiedergegeben. Den Plan hab ich ihm auf jeden Fall wiedergegeben. Transistoren konnte man schon damals viel kriegen, deswegen hab ich daran nicht gedacht. Tut mir trotzdem leid man sieht wie wichtig die einzelnen Bauteile sind. Man sieht, wie wichtig die einzelnen Bauteile sind und tut mir trotzdem leid. Im Plan hat er wieder gekriegt. Das andere Problem war dann trotzdem

Wie gesagt also jetzt zu den Steckbrettern zu den Steckbrettern sag ich folgendes wie man sieht kann man das ohne löten machen wenn ich jetzt zum Beispiel ein Univac

Oder wie das Ding damals ist es ist vielleicht auch noch das Auto da kannst du auch ne doppelte Verwechslung geben kann geben es gibt Unix Reifen in Ungarn gibt es Unix Reifen die gibt es hier als Betriebssystem gut ich weiß nicht ob die wirklich so Eniac ja, ich glaub schon, d.h. nicht, dass ich mich jetzt wieder von angeblich linken Mädchen Gruppierung oder wenn auch nicht anmachen lasse, dass sie mir sagen ich hätte keine Ahnung von dem Computer. Ich weiß, wie tut, man kann es mir Transistoren machen aber man sieht hier ohne den zu bauen bin jetzt zum Beispiel die ALU kommt

Ich glaube übrigens, dass dieser Johannes fast nicht ganz so böse ist, sondern dass das, was mir im Microcontroller Netz ich weiß es nicht. Ich entschuldige mich dafür deine Paste schon mal gut machen Sie es nicht komplizierter. Ich weiß, dass man ich war trotzdem am technischen hat damit aber nichts zu tun und es ist hier nicht immer Teil Streit für ein Thema des Streitthema. Was tun, wenn man jetzt zum Beispiel diese alten Computer baut die Groß Rechenmaschine

Ja, das sind maximal bei dem Ding zwei Transistoren abhandengekommen die konnte man allerdings auch so kaufen. Ist trotzdem nicht nett plus den Plan hat er wieder gekriegt wie Transistoren tun wenn man jetzt das Ding nimmt also den Computer als Großrechner baut, dann kann man Transistor nehmen Der erste Schritt wird es mit der ALU zu machen, so wie das jetzt ist.

So wie das jetzt ist, kann man das so machen aber man kann auch einzelne Transistor nehmen. Man kann die übereinander stapeln. Jetzt kann man natürlich da kann man immer noch Teile löten. Jetzt kann ich die schlecht auf Streifenplatinenlöten schlecht auf Streifen oder ich kann dir sogar noch ätzen, wenn ich das aber modular mache und die einzelnen einstecke. Und es ist ja üblich bei Computern das war auch bei dem IBM PC. Und ist sie noch immer so. Sie können die Teile zusammenstecken erst mal, Sie können das IC ein und aussteigen. Es gibt kein Streit, dass die Dinge, die ich an die Straße gestellt hab, dass ich die nicht neu habe. Ich bin zufrieden. Ich hab alles neu und das andere Programmiergerätes übrigens ein günstigeres ich hab alles neu da muss man sich keine Sorgen machen. So, jetzt kann man, wenn sie darauf achten, die die Platinen teilweise zusammenstecken. Daneben gibt es die üblichen buskabel . Man kann aber auch die IC einzeln einstecken. Gut wenn ich jetzt ne Maschine

Bau dann ist die kein wenig gegenüber im Nachteil wenn ich dann hinkriege, Transistoren zu machen der alten gegenüber. Wenn ich dann hinkriege, Transistoren zu machen, kann ich dir auch mit der Hand machen. Also mit dem Hochofen vielleicht wenn wir das gelingt und dann kann ich die einzelnen reinstecken, dann kann ich auch Streifen Raster Platin nehmen ätzen. Ich kann aber genauso gut in anderen Experimenten wenn ich zum Beispiel so eine nehmen also ne ALU dann kann ich dir ne Steckbrett reinstecken dann kann ich dir den Steckbrett reinstecken also in Steckbrett und wenn ich das habe, dann kann ich dir so lassen. Also anders als bei so einer Groß Rechenmaschine wo ich das so reinsetzen und aber sozusagen übereinander gestapelt und dann mit Kabeln also übereinander gestapelt und dann kann ich in dem Stapel einzelne Komponenten ersetzen. In der steckt Brett kann ich das aber sitzen lassen deswegen sind auch tatsächlich diese Experimentierekästen nicht schlecht und die kann man auch heil und ganz lassen übrigens und die AA ist auch interessant. Trotzdem das geht jetzt auch so sag ich mal und das heißt auch nicht. Also wie gesagt AA Diode das kriege ich noch hin die kann ich. Ich kann den auch noch mal kaufen trotzdem das ist jetzt soweit so und dann kann man die Steckbretter so lassen. Danke schön

4 graphen

die übung mit LaTeX. ich habe ja gesagt, dass das eine Übung ist, durchaus für die Darstellung. ich mache folgendes: ich übe - einfach die knoten, knoten für knoten, wie vorher gesagt.

```
\begin{center}
\begin{tikzpicture}[->, >=stealth', node distance=3cm, auto]
  % Zustände definieren
  \node[state] (q0) {$q_0$};
  \node[state, right of=q0] (q1) {$q_1$};
  \node[state, below of=q0] (q2) {$q_2$};
  \node[state, right of=q2] (q3) {$q_3$};

   \node [and port] (and04) at (0,8) {};
 \node [and port] (and05) at (0,10) {};
 \node [and port] (and06) at (0,12) {};
 \node [and port] (and07) at (0,14) {};

 \node
\end{tikzpicture}
\end{center}
```

übung

jetzt uebung:

```
\begin{verbatim}
node node node node node node node
state state state state state
and port and port and port and port
state and port state and port state and port

node state and port node state or port
node node node node node node
\node \node \node \node \node
and port
or port and port or port and port or port
state state state state

\node[state]
\node[and port]
\node[or port]

\node[state]
\node[and port]
\node[or port]

\node \node \node \node
or port and port or port and port or port
and port and port or port
\node \node \node \node
state state state state state

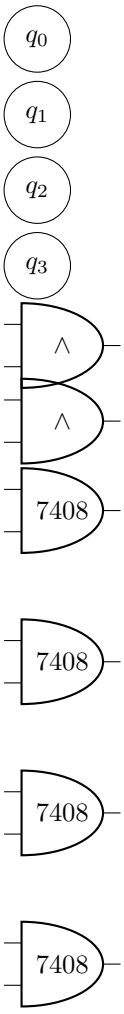
\node [state]
\node [state]
```

ok, mit namen

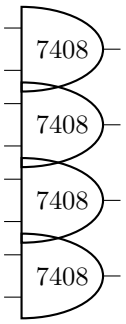
```
\node [state] (q0);
\node [state] (q1);
\node [state] (q2);
\node [state] (q3);
\node [state] (q4);
\node [state] (q5);

\node [or port] (or0);
\node [and port] (and0);
\node [or port] (or1);
\node [and port] (and1);
\node [or port] (or2);
\node [and port] (and2);
\node [or port] (or3);
\node [and port] (and3);
```

```
\node [state] (q0);
\node [state] (q1);
```



so, das sind und gatter:



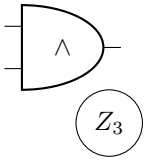
ok, das tut nicht so gut:

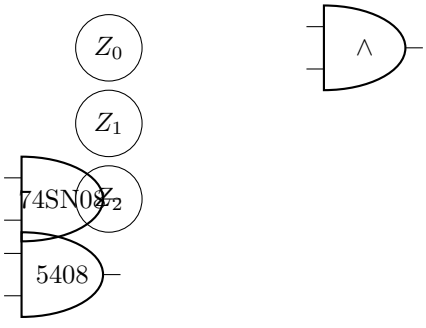
```
\node \node \node \node \node \node
and port or port and port or port and port or port
state state state state state
\node [state] (q0);
\node [state] (q1);
\node [state] (q2);

\node [and port] (and0);
\node [and port] (and1);

\node [and port] (and2) at (4,0) {$\wedge$};
\node [and port] (and3) at (4,4) {$\wedge$};

\node [state, below of=and2] (q3);
```





```
\node [state] (q0) {$Z_0$};
\node [state, below of=q0] (q1) {$Z_1$};
\node [state, below of=q1] (q2) {$Z_2$};

\node [and port, below of=q1] (and0) {74SN08};
\node [and port, below of=and0] (and1) {5408};

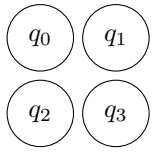
\node [and port] (and2) at (4,0) {$\wedge$};
\node [and port] (and3) at (4,4) {$\wedge$};

\node [state, below of=and3] (q3) {$Z_3$};
```

nun folgt der Automat, aus der letzteren Tabelle:

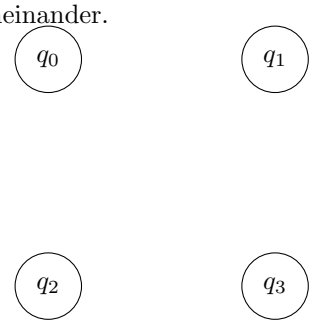
5 Automat

z_n	x	z_{n+1}	y
4	0	3	1
4	1	2	0
3	0	2	0
3	1	1	0
2	0	1	0
2	1	4	1
1	0	4	1
1	1	3	1

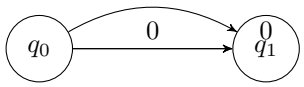



```
\begin{tikzpicture}[->, >=stealth', node distance=3cm, auto]
```

ok, es geht nicht anders als so: weil, man sieht, die knoten sind zu sehr aneinander.



z_n	x	z_{n+1}	y
4	0	3	1
4	1	2	0
3	0	2	0
3	1	1	0
2	0	1	0
2	1	4	1
1	0	4	1
1	1	3	1





diese formulierung ist, sagen wir auf das gerichtet, wo in der tabelle, die zustände nicht anders kodiert sind, als wie in der mathematik bisher üblich. wir nehmen in folge noch arabische zeichen. ok. doch wir brauchen eine binäre kodierung und: eine: mit unärer, am besten auch in der tabelle. vorher allerdings der graph so mit dem, was das betrifft.

ok, das

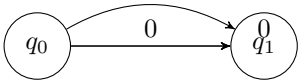
```
\begin{tikzpicture}[->, >=stealth', node distance=3cm, auto]
\node [state] (q0) {$q_0$};
\node [state, right of=q0] (q1) {$q_1$};
\node [state, below of=q0] (q2) {$q_2$};
\node [state, right of=q2] (q3) {$q_3$};

 \path (q0) edge node [midway, above] {0} (q1);

 \draw (q0) -- (q1) node [bend left, above] {0} ;

\draw (q0) to [bend left, above] (q1);
\end{tikzpicture}
```

ergab das hier:





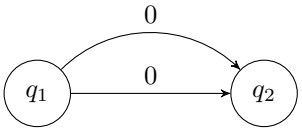
damit nehme ich:





ok, dann machen wir folgendes:

z_n	x	z_{n+1}	y	
4	0	3	1	
4	1	2	0	
3	0	2	0	
3	1	1	0	ok:
2	0	1	0	
2	1	4	1	
1	0	4	1	
1	1	3	1	





ok, habe ich damit erreicht:

```
\begin{tikzpicture}[->, >=stealth', node distance=3cm, auto]
\node [state] (q1) {$q_1$};
```

```
\node [state, right of=q1] (q2) {$q_2$};
\node [state, below of=q1] (q3) {$q_3$};
\node [state, right of=q3] (q4) {$q_4$};

\draw[->] (q1) -- node[bend left] {0} (q2);

\draw (q1) to[bend left=45] node[above] {0} (q2);
\end{tikzpicture}
```

die einzige bei mir bisher funktionierte version, ist somit die

```
\draw (q1) to[bend left=45] node[above] {0} (q2);
```

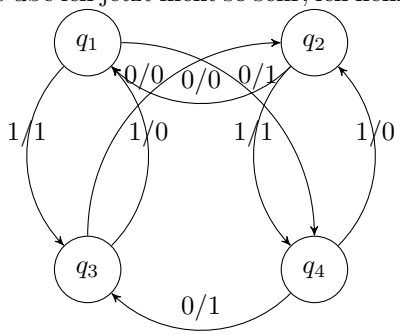
ok, ich sage gleich:

1. ich weiss backslsh node
2. ich weiss port and
3. ich weiss state
4. ... und zwar 100 pro. das heisst: ich kann
   - mit koordinaten
   - mit below und right und so weiter (of)
   - mit name in runden klammern und beschriftung in geschweiften...

ok, wenn wir das haben.

- path
- draw

das funktioniert ordentlich, das übe ich jetzt nicht so sehr, ich nehme, das gerade



beschriebene. ok, mit tabelle.

z_n	x	z_{n+1}	y
4	0	3	1
4	1	2	0
3	0	2	0
3	1	1	0
2	0	1	0
2	1	4	1
1	0	4	1
1	1	3	1

wir machen daraus mealy vorher eine andere kodierung.

z_n	x	z_{n+1}	y
3	0	2	1
3	1	1	0
2	0	1	0
2	1	0	0
1	0	0	0
1	1	3	1
0	0	3	1
0	1	2	1

eine andere kodierung, zunächst mit der richtigen nummerierung:

z_n	x	z_{n+1}	y
0b11	0	0b10	1
0b11	1	0b01	0
0b10	0	0b01	0
0b10	1	0b00	0
0b01	0	0b00	0
0b01	1	0b11	1
0b00	0	0b11	1
0b00	1	0b10	1

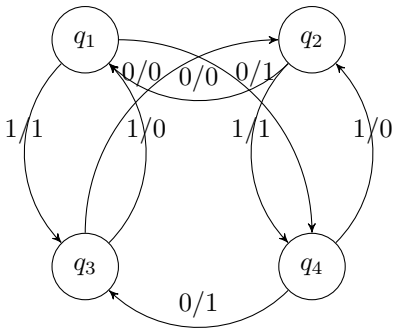
nachdem wir das nun erreicht haben, kommt die binärkodierung...

z_n	x	z^+_{n+1}	y
0b1000	0	0b0100	1
0b1000	1	0b0010	0
0b0100	0	0b0010	0
0b0100	1	0b0000	0
0b0010	0	0b0000	0
0b0010	1	0b1000	1
0b0001	0	0b1000	1
0b0001	1	0b0100	1

jetzt die unäre kodierung. jetzt stimmt die kodierung. als nächstes der moore automat. alle knoten müssen gespalten werden, die von der kante, als eingang, also nachbar, von dem es ausgeht. sowohl in der ausgabe, 1 haben, als auch 0. kurz sagen wir XOR

z_n	x	z_{n+1}^+	y	XOR
0b1000	0	0b0100	1	1
0b1000	1	0b0010	0	1
0b0100	0	0b0010	0	0
0b0100	1	0b0000	0	0
0b0010	0	0b0000	0	0
0b0010	1	0b1000	1	0
0b0001	0	0b1000	1	1
0b0001	1	0b0100	1	1

dazu gehören, ich mache da sternchen dran:



z_n	x	z_{n+1}^+	y	XOR
0b1000	0	0b0100	1	1
0b1000	1	0b0010	0	1
0b0100	0	0b0010	0	0
0b0100	1	0b0000	0	0
0b0010	0	0b0000	0	0
0b0010	1	0b1000	1	0
0b0001	0	0b1000	1	1
0b0001	1	0b0100	1	1

also muss aus knoten 0b0100, muss 0b0010

z_n	x	z_{n+1}^+	y	XOR
0b1000	0	0b0100	1	1
0b1000	1	0b0010	0	1
0b0100	0	0b0010	0	0
0b0100	1	0b0000	0	0
0b0010	0	0b0000	0	1
0b0010	1	0b1000	1	1
0b0001	0	0b1000	1	0
0b0001	1	0b0100	1	0

jetzt wird das verdoppelt: für moore:

z_n	x	z_{n+1}^+	y	X
0b1000/0/1	0	0b0100/1	1	
0b1000/0/1	1	0b0010/0	0	
0b0100/0	0	0b0010/0	0	
0b0100/0	1	0b0000/0	0	
0b0010/0/1	0	0b0000/0	0	
0b0010/0/1	1	0b1000/1	1	
0b0001/1	0	0b1000/1	1	
0b0001/1	1	0b0100/1	1	

jetzt wird das verdoppelt: für mealy. ok: jetzt machen wir folgendes:

wir haben jetzt einen moore automaten, und wenn wir den haben, ...

dann habe ich beim ausgang, ein mal mit /0 und ein mal /1

ok: und, er bekommt, so zu sagen, das doppelte an zeilen. nur, weil aber zwei zustaende von 4 verdoppelt werden, 8.

z_n	x	z_{n+1}^+	y	XOR
0b1000/0	0	0b0100/1	1	1
0b1000/1	0	0b0100/1	1	1
0b1000/0	1	0b0010/0	0	1
0b1000/1	1	0b0010/0	0	1
0b0100/0	0	0b0010/0	0	0
0b0100/0	1	0b0000/0	0	0
0b0010/0	0	0b0000/0	0	1
0b0010/1	0	0b0000/0	0	1
0b0010/0	1	0b1000/1	1	1
0b0010/1	1	0b1000/1	1	1
0b0001/1	0	0b1000/1	1	0
0b0001/1	1	0b0100/1	1	0

man kann das noch anders begrei-

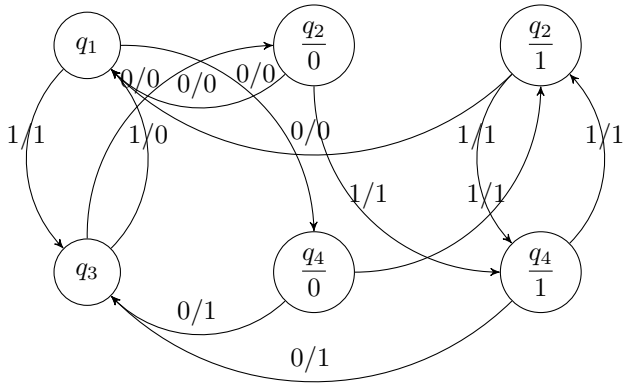
fen:

z_n	x	z_{n+1}^+	y	XOR
$\frac{0b1000}{0}$	0	$\frac{0b0100}{1}$	1	1
$\frac{0b1000}{1}$	0	$\frac{0b0100}{1}$	1	1
$\frac{0b1000}{0}$	1	$\frac{0b0010}{0}$	0	1
$\frac{0b1000}{1}$	1	$\frac{0b0010}{0}$	0	1
$\frac{0b0100}{0}$	0	$\frac{0b0010}{0}$	0	0
$\frac{0b0100}{0}$	1	$\frac{0b0000}{0}$	0	0
$\frac{0b0010}{0}$	0	$\frac{0b0000}{0}$	0	1
$\frac{0b0010}{1}$	0	$\frac{0b0000}{0}$	0	1
$\frac{0b0010}{0}$	1	$\frac{0b1000}{1}$	1	1
$\frac{0b0010}{1}$	1	$\frac{0b1000}{1}$	1	1
$\frac{0b0001}{1}$	0	$\frac{0b1000}{1}$	1	0
$\frac{0b0001}{1}$	1	$\frac{0b0100}{1}$	1	0

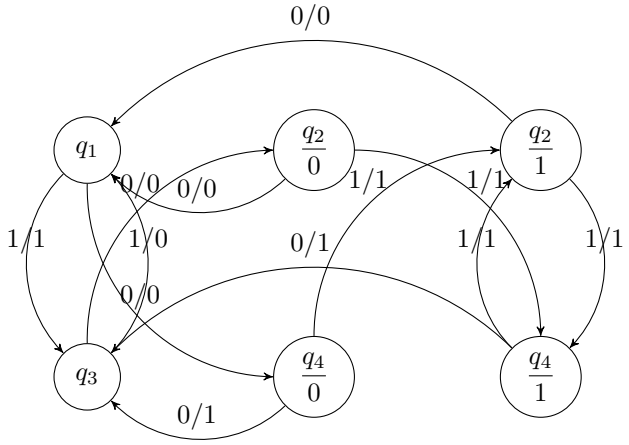
und jetzt der automat:

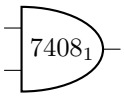
das mit dem verdoppeln war quatsch, das sehe ich gleich.

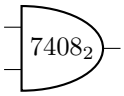
z_n	x	z_{n+1}	y
4	0	3	1
4	1	2	1
3	0	2	0
3	1	1	0
2	0	1	0
2	1	4	1
1	0	4	0
1	1	3	1

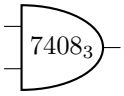


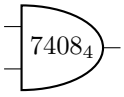
hier eine andere kanten:











da mache ich morgen weiter, das könnte jetzt zu quine mc cluskey führen, aber ich mache es nicht, das könnte auch hier rein. das ist das eine. und ich könnte das schaltwerk weiter machen, mache ich nicht, für heute ist pause. ich nehme jetzt noch das eingescante rein und dann kommt das mikroprogrammsteuerwerk rein und dann passt die sache. dann kommt auf die homepage. als pdf mit LaTeX. und mit latex2html

das mikroprogrammsteuerwerk:

Daniel Vagaa(x), 08/24/2026
Schaltwerke, Seite 3

Eine andere Idee ist daß ich die Schaltwerke auch mit Würfeln tue.
Also generell dienen die Würfel dazu
a) Wahrscheinlichkeitsrechnung
b) Die Reihen, seien es:
- √ Crash
- √ CRC
- √ Vierschlüsselung
- √ Fletcher's checksum

oder Codes:
- √ Airen
- √ Dreiexcess

c) aber auch die Umrechnung von Zahlensystemen.
Rechnen wir mit der Zuordnung
$\{ f(1)=0,\ f(x=2)=1,\ f(x=3)=2, \ldots \}$
haben wir es mit einem 6er System zu tun.
a) Das lässt sich ins Dreier umrechnen
b) bis 10er...

Jamil Vojcla (c)
08/24/2026
Schaltwerk mit Latex, Seite 1

also:

1.) Die erste Idee, die ich habe
reitet sich an dem papier-
Stapel, den ich habe

a) Da die Unruhe hier
merklich/spürbar (deutsch)
bin ich noch ein raus
gegangen vor die Stadt
und dort habe ich die
Idee verbessert

b) Es handelt mehr oder
weniger um zwei Ideen..

2.) Der Stapel papier ist
hinzuziehend bezeichnend.

3.) ich habe xx oder so, ja längst
ein anderes verfahren für mich
eingeführt. nämlich bei tabellen
und graphen würfel real zu
benutzen:

- bei der umrechnung vom
binärsystem kann der
nur nützlich sein.
- Die Zufallszahlen so wie
alle weiteren Reihen...

David Vajda (c)
08/24/2026
Schaltwerke, Seite 2

lassen sich besser auf papier oder
jetzt am Computer das ist jetzt
der neue teil der idee...
also wie früher alle
ein laut auswendig gelerntes.
a) Regn gehört minix
b) astm cortex
c) vieles weitere
am Computer.
Das ist durchaus den Code auswendig
schreiben, z. B von minix.
Binäres dividieren, oder über-
haupt schriftlich geht am
Besten mit Excel.
Ich benutze meinen großen pc meist
aber mein ibook. mit Linux
kann dieser auf einem externen
Datenträger auch benutzt. Erleichtert
Zahn
Netzwerk...
Das muß allerdings eingerichtet werden...
Der papier Stempel ist bezeichnend.
von daher jetzt alles am Computer

Daniel Vogtla (c) 08/24/2026
Schaltwerke, Seite 4

Was man damit auch machen kann sind graphen.

a) Ziel ist es alles zu verwalten

b) neben den graphen, das sind Zwei: Kommen Wahrheits-tabellen in frage.

nebenbei das Funktionale:

- y - Wort
- Byte
- Doppelwort

und so weiter

a) Das lässt sich kodieren

b) Den ganzen als Matrix berechnen...

c) oder: crc... anwenden.

Doch die graphen sind

a) ein feine graphen

b) asm - durchsuche

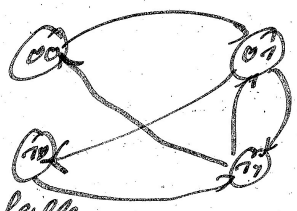
Die graphen dienen auch Schaltwerken. Das heißt Automaten.

für einen einfachen automaten:

soll auch ein Schaltplan daraus gleich mit tikz in Latex zu machen

(c) Jamil Vajda
08/24/2026
Schaltwerke, Seite 5

Wenn man nun den einfachen graphen
anguckt:
für "einfache" Automaten.
Denn ist das ein graph aus 4
Zuständen in dem fall:
bei dem (zwischen zwei Zuständen)
pro Zustand zwei Kanten existieren
so z.B.



in dem falle
weniger Kanten...
Das lässt sich einfach mit
Würfeln realisieren. Schon
auf eine andere art und
weise.
Man würfelt nun Kante für
Kante sondern z.B ein langes wort:
5463126653...
Daraus lassen sich die entsprechenden...
entnehmen...
Bei diesem einfachen graph für
ein Schaltwerk..

Daniel Vajda(c)
08/24/2026
Seite 6, Schaltwerk..

fehlt noch das übergangsschaltnetz
und das ausgangsschaltnetz...
Das ist entsprechend als Schaltplan
zu machen.
auch am Computer.
mit (E-) Kamera...
allerdings lässt sich das
alles mit Tikz in Latex
machen.
Ein wenig schwierig.
hier brauche ich 4 zustände, muss
die entsprechend verbinden.
Tikz bietet viel
Vielleicht muss die Maße
für alles mit Worten einzeln:
wenn man es oft macht
geht es anders.
Die zweiten graphen betreffen komplexe
automaten.
Dabei müssen ganz konkret unterscheiden.
a) der begriff Hardware-
Algorithmus

Daud Vajda (c) 08/24/2026
Schaltwerke, Seite 11

1.) man muß den Schaltplan des Steuerwerks machen
2.) den Schaltplan mit allgemeinen Registern des Operationswerks
3.) man muß den Hardware algo über würfeln:
    a) erzeugen
    b) aufzeichnen.

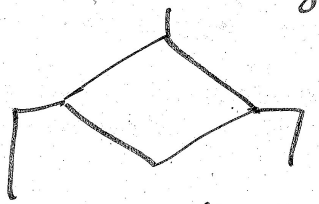
Nach ihm treten sich
    1.) Steuerwerk
    2.) operationswerk.
was im asm Diagramm neben Zustände was Knoten sind entwickelt ist (Knoten mit 3 und 2 Wahlbaum und Kanten)...
    ist Ahl notation.
Diese definiert operanden. 6 Register z.B.
    r6, r5, r4, r3, r2, r1
    ... bzw r6, ..., r0
so wie operationen: 6
    add (+) r7 ← r5 + r4
    sub (-) r5 ← r5 - r4
aber auch vergleiche. normal gibt es drei >, =, < trichonomerte malle ...

Samvid Vyoma (c) 08/24/2026
Schaltwerke, Seite 10

Es lässt sich meinerseits über das
Wort Modus bis Streiten. Aber

Zustand ⇔ Modus

erhält sein Steuerwort von dem
operationswerk über verbinder.



gleichzeitig ist Steuerwerk = automat.
um das verhalten zu beschreiben.
braucht es einen/ asm Diagramm
Das ist wiederum ein graph
mit Knoten mit
zwei Kanten:
Herein und Heraus
vierech waagerecht

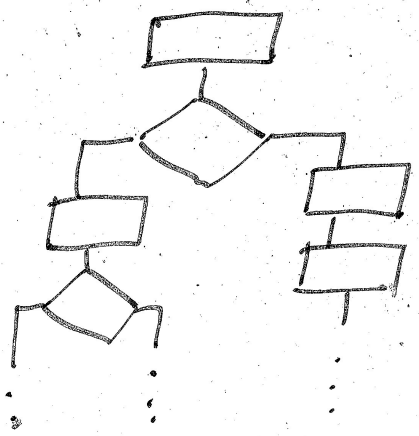
drei + Kanten: Raute..
Es steuert das operationswerk
besteht aus befehlen. Zu tun ist
aufgabe

Jannick Vojdan (c)
08/24/2026
Schaltwerke, Seite 8

Doch sind auch in ihrer fülle
mathematisch Menge * oft
unterschiedlich und verschieden zahlreich
Calteströmlich deutsche Begriffe (etwas
wie Comic halt beschriebenes Begriff -
interpretierbar? aber so nicht gemeint))
Also:
Entscheidend neben den Operationen
sind aber auch Regisks.
Die Operanden sind
a) Quelloperand(en) ten?
b) Zieloperant...
Diese Regisks können z.B.
r5.. r0
sein.
Die operationen wiederum *
1.) Das operationswerk
a) führt operationen aus
b) hat operanten (Regisks)
Jedweiteren über das Steuerwerk
gesteuert.
2.) Das verhalten des Steuerwerkes -
ein eigentliches - das eigentliche
Unterwort - der sein Steuerwort
man kann es auch Steuermodus
nennen.

(c) David Vyčla
08/24/2026
Schaltwerke, Seite 7

Hardware algorithmen werden
generell mit asm Diagrammen
beschrieben:



Dabei müssen wir aber Streng-
unterscheiden:

1.) zwischen Operationswerk
2.) Dem Steuerwerk.

Das Operationswerk ist in der Lage
a) vergleiche auszuführen
b) rechenoperationen zu:
grundrechenarten:
vor allem: addition + subtraktion
inkrement/dekrement auszuführen
c) so wie links und Rechtsschieben.

Daud Vejda (c)
08/24/2026
Schaltwerke, Seite 9

Zu unterscheiden sind Zustand (a) und (b) modus
in einem Schaltnetz hängt der Zustand wohl nur vom Modus, der wahl das Eingabewort ist.
Das Eingabewort... regelt den Modus

In einem Schaltwerk... wir benutzen einen Graphen in der Mathematik so genannt...
Dieser stellt im Automat mit den Knoten zustände dar.
Hier: Das wort einerseits eingabewort aussen + den Modus den ein Zustand selbst erzeugt entstehen der Modus für den Folgezustand

Anmerkung:
Es ist bei den Übungen für "simple" Automaten auch für Übungen für umwandlung mealy → moore zu sorgen.

So: Das Steuerwerk wiederum erhält unabhängig - Länge rede kurzer Sinn gerade eben- ...

Jurid Vayda (c)
08/24/2026
Graph, Seite 12 (Schaltwerk)

Diese + der 6 Register lassen sich kodieren wie beim Computer üblich

add: 0
sub: 1
shl: 2
shr: 3
inc: 4
dec: 5 ...

so wie die operanden:

r5: 5
r4: 4
r3: 3
⋮
r0: 0

aber auch die vergleiche:

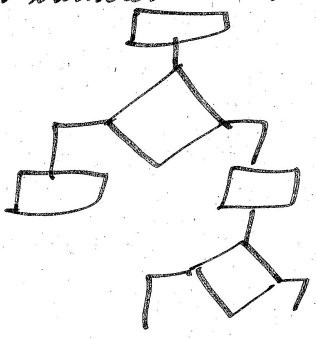
ne: 0
eq: 1
le: 2 ...
ge: 3
lt: 4
gt: 5

So: Dann haben wir zustände die sind nummeriert:

z0
z1
z2
⋮
zn

David Vajda (c) 08/24/2026
Schaltwerk, Seite 13

Dabei handelt es sich um graphen



und die nachbarn (knoten lassen)
sich auswürfeln.
bei einem gesamten würfelt
man ein großes wort
6534122...
und zerlegt es entsprechend.
auch das läßt sich mit tilz
vollziehen...
Doch: ich möchte
dies mal ausprobieren
und deswegen heute nach
normal mit papier...
avg. würfel...

David Vajda (c) 08/25/2026
Seite 1 asm
So: Diagramme...

also)
man nimmt 11 Zustände
oh: 12 oder 13
oder 16. Sonst hat man Mikro-
programmsteuerwerk mit 8
Leitungen...
Ein komplexes Schaltwerk
lässt sich nicht mit einem
Übergangsschaltnetz lösen.
Das... einfache ein
überg... Schaltnetz ist das auch...
ansschaltnetz
uv: Das muß ein Komplexes
mit Eprom oder fuses sein...
und das hat mit 8 zu wenig...
also 16 und 4 bit Eingang...
So: Dann nehmen wir
für das asm Diagramm:
16 Zustände maximal
Da wir Ranten haben:
wir brauchen 3 Leitungen...
mindestens: ...
also: Dann würfeln wir:
Halt! 1. Mal!

Daniel Vajda (c)
08/25/2026
Diagramm, Seite 2

also: die erste Bedingung ist:
Das: man einfach würfelt:
also man würfelt eine ganz
lange (fast) Zahl:

3452162216...

sehr lange:
und mit umrechnung!
z.B über umrechnung
transformation und so
weiter:
a) kann man nach
binär
b) zehnersystem
c) <- grob c: aber vor allem!
über den Rest - Rest!
über den Rest können wir
jeweils für die entsprechenden
Aufgaben:

1.) Zustände
2.) Verlinkte Zustände
3.) Nummern der Befehle..
4.) Register und operanten..

die für die entsprechende Sache
für das entsprechende 11...16 Zustände
2 Register die Zahlen ausrechnen

Jamd Vayda (c)
08/25/2026
Seite 3, asm Diagramme.

So: wir würfeln eine entsprechend
lange Zahl...
- Zustände (11... 16)
- Register
- Operation
wird im entsprechenden einzeln aus
der Lade gezogen...
ganz einfach...
nun müssen wir erst mal:
16 Zustände festlegen.
Sagen wir: wir haben das...
und weil wir das haben:
würfeln wir nun ob
Zustand Z0, Z1, Z2, ... Z15
Die Zustände sind durchnummeriert..
müssen wir nun auswürfeln:
wenn auch bei system..
Dann können wir von 65 z.B.
Rest 16 rechnen ohne besondere
Vorstellung.
3 Zustände werden dann
benötigt.
ok: wenn das (gespeichert ist:) falsch.
bin gerade wo anders im Zimmer hingelaufen
1.) das ganze findet dies mal

Daniel Vogda (c)
08/25/2026
Seite 4, axm Diagramme

mit:
papier statt...
Ich sage ja: es muß in Zukunft
automatisch mit Latex gehen!
automatisch mit tikz...
automatisch!
Das man das auch so hinschreibt!
Trotzdem!
Es ist jetzt erst aufgabe:
Wir sollen herausfinden!
+ axm Diagramm
+ Hardware Algorithmus
+ Operationswerk
+...

Dawid Vajda (c)
08126)5122026
Seite 5, vom Diagramme..

abo: habe ich gesagt



daß man jetzt nur würfelt?
und es auf papier macht?
Neinnnn!
ich habe gesagt!!
a) man würfelt ein mal
ena macht papier
b) habe ich gesagt!!!
ich habe gesagt dann
auf papier
und das dürft nicht!!!
Dass man es dann nicht
auch noch in Latex
einsetzt!!
auch nach als zweiten Schritt!
Anfang! für alle opôkcu!

Dumm Vagaa (c)
08/25/2026
Seite 6, asm Diagramm!

Kaffee (1.)
Dumm anfangen!
Dumm anfangen!

30, 31, ... und los!

So. 1015 mesz east
Tübingen (don't know
what: coordinates
longitude, latitude?
know: long: west/east
latitude: north south
Druck ...
würfeln? dying..
L?
temperatur
zu viel!?

fertig mit den neuen..

$\Omega = (6454262216556115431 5)$

ok: jetzt nehmen wir
1.) 1cm rede zu viel...
(Ereign) Ereignis
weise ni .. - bla bla to
much?
Ereignis nur ergebnis
Elementarmenge $\Sigma = \{1, 2, \ldots, 63$
ist die "mögliche" menge
1 bleibt als Ergebnis? ! Es bleibt das
Ob sind das Elementarereignisse!

Daniel Vajda (c)
08/25/2026
Seite 7, asm Diagrammen!

Elementarereignisse...
Zweitens: sorry: zu viel da-
Rede!
nur
6464...
ist eine lange Zahl lässt sich
aber als
1.) elementarereignisse adressieren
2.) tupel: paare: trippel
(a, b): paar
(a, b, c): tripel
$(a_n, \ldots, a_0)$: n-tupel
und:
wenn man:
blau und gelben
würfel hat:
(blau, gelb): paar
aber (6,4, 3,4, ...'
gemultiplexed! ok...
^sry... nicht
zu laut?
(:)??
Ja ob wg. ??!
ok: jetzt haben wir die Zahl:
Ω08262026saravaj0001

Dawid Vajda (c)
08/25/2026
Seite 8, zum Diagramm...

Dann nehmen wir die Zustände

$Z = \{z_0, z_1, \dots, z_{15}\}$

ok und schneiden von der Zahl:

0x08252026dawidvajda0001

or what ever →

Stücke ab...

ab ersko drei für 3 der 16 Zustände
bei 16 Zuständen löst man:

16

mal 3. aber wir nehmen

den anfang

$64 \,\%\, 16 = x \stackrel{?}{=} ??$

irgendwas

wir könnten das andere liegen lassen. bei modulo zwangsweise nötig?

auch nicht!

wir addieren es zum nächsten-

kurz weiß verarbeitung

bis so etwas große Null noch?

damit nicht nächste

Zahl: jetzt. vorher:

raucherpause..

Jamil Vajda (c)
08/25/2026
Seite 9, vom Diagramme...

ok: jetzt bleibt ein problem:
1.) die würfelfolge besteht aus
$S = [1,2,3,4,5,6]$
in das: 6er system umgesetzt
bedeutet das
$[(x()]\ f(x=1) = 0$
$f(x=2) = 1$ ...

ok: bei 16 zuständen ist das
umgeschickt. das ist zwar hex
system! 16
0,1,2,.. f (mit 0x)!
trotzdem 16 existiert im dezimal-
system!
und es gilt zwei möglichkeiten
von bin → dec
a) wir rechnen die potenzen
zusammen

$0b\overset{3}{1}00\overset{0}{1}.0011$
$_{b\,3}\ \ _{s\,2\,1\,0}$

$2^3 + 2^4 + 2^1 + 2^0$

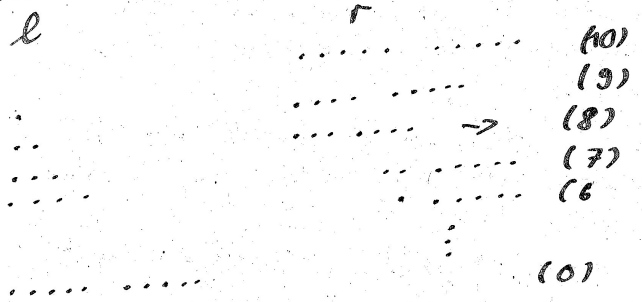
oder wir machen das was wir
verwirrender weise "anders herum"
machen würden
wir teilen durch $10_{dec} =$
1010_{bin} aber im bin system. nehmen
den Rest..

Daniel Voyda (c)
08/25/2026
arm Programme, Seite 10

Vorwort!!
Jetzt lange rede!

a) Das ist was man üblicher bei bcd tut...
Zahlenumrechnung für darstellung am Bildschirm
↑ Dies ist aber geringe argumentation!

1.) wenn man abakus anguckt hat man



Das sind aber die indischen Ziffern:
0, 1, 2, ... , 10
und nicht einfach bis ebenso wie arabische



und maya 

dabei:  = 7 dies ist eine Ziffer:



David Vojda (c)
08/25/2026
Vom Diagramm: Seite II

Zwar!
ist bcd ein typisches Fall
von Darstellung auch ohne
jedes teilen!

0b0000 : 0d ... 0b1010 : $\not{\#}^{10}$
fehlt

Trotzdem entsteht es bei division
durch 10_{dec}
ist aber binär 1010_{bin}!

und : ob zwar :
0b0000 : binär $\hat{=}$ 0_{dec} : dezimal ?!
ist in bcd trotzdem:
0b1000 : als dezimalziffer 8
zu sehen!

ok: bcd:
nicht nur unser klassisches
phänomen zur darstellung
am Bildschirm!

man rechnet um:
$dec(0b1001.0011) = 2^7 + 2^4 + 2^1 + 2^0$
$= 128 + 16 + 2 + 1$

aber auch:
II.) $0b1001.0011 / 1010$
$\swarrow 0b$

Daniel Vojdac)
08/25/2026
Seite 12, assm diagramm
Dieses zweite verfahren ist
wirrig!
und!
wenn wir mit würfel
1...6 (0...5)
würfeln aber 16 zustände haben.
müssen wir vorher in dec umrechnen.
Sonst:
Zufallszahl gestört!
ok:
Das könnten wir so machen.
teilen durch
10 im 6er system!
aber halt!
an der stelle tut sich ein
problem auf!
warum ist es eigentlich einfach!
im binärsystem zu teilen?!
warum geht integer division!?
ganz einfach!
1.) schriftliches Dividieren!!!
2.) aber: noch mehr:
wenn im schriftlich dividiere!!!

Daniel Vojda (c)
08/25/2026
asm diagramm! Seite 13

Das macht alles anderes!
verfehrisch! aber zu rein
linär dividieren ist einfach!

weil schriftlich dividieren einfach
ist!
aber auch linär!
warum

1010 kann:
beim schriftlichen dividieren
subtrahieren nicht verwechseln

1111 ← hier weil das da
− 1010 steht brauche ich:
11111 nicht
aber in
~~41 ok~~ 10000
− 1010 } 1 Stelle
drüber

maximal 1 oder 0!
nur bei 0 wg. Subtraktion
415 Stellen
verlor: beweis!
Beweis 1010 nein
+ 1010 geteilt
2 durch!

Jannik Vogel (c)
08/25/2026
aus Diagramm, Seite 14

$$\begin{array}{r} 1111 \\ +\,1111 \\ \hline 11110 \end{array}$$

also:

1111 gelt in 11110
neu
wird aber neu
zu
10 0000
↑
6 Stellen!
und wenn dann gelt auch!
10000
- 1111
oder 4 stellen!
1111
- 1111
also: binär: Das beinhaltet
binär: gelt ~~2~~ 0 · 10.2
mal ist möglich!
Subtrahtion! Schriftliche
Division!
Dann: einfach: blop
bei System?

David Vogda (c)
08/25/2026
asm Diagramm: Seite 15

also:
es gibt ja zwei möglichkeiten:
1.) teilen!
2.) wie vorher

dec(1010)$_b$ = $2^3 + 2^1 = 10$
teilt? ↑ ok $2^1 = 2$
eigentlich 10

ok: wenn ich im bin system teile:
nichts blicken!
blop

521
$= 5 \cdot 6^2 + 2 \cdot 6^1 + 1 \cdot 6^0$
$= 5 \cdot 36 + 2 \cdot 6 + 1$
im dec system!
ist:
150 + 30 + 12 + 1
193 im dec system!
und jetzt mod 16
warum eigentlich?
also computer zufallszahlen!
wenn:
dann teile (wie hash??)
mod? ja und am pc
ja: blop. hier python3 + d
hier pyd? / zufall
16 21 oder

David Leyda (c)
08/25/2020
aom.... Seite 16

also:
Zufall:
2 rest
in prog!
wg. max: wer
die Reihe
jetzt: Diagramm
+ Darbe Computerspiele I/II
Bildung, Schiffman,....
!! kommt liegt
Idiige, Baum?
↑ suchen!!!

Jawid Vajda (1)
08/25/2026
Seite 17
asm diagramm:

also die Zahl:

64 im Dezimalsystem ist

$6 \cdot 6^1 + 4 \cdot 6^0$

$= 36 + 4 = 40$

oh 40 mod 16 = (32, 2, 8)

↑ hä?

mod paar(

(quotient, rest)

aust rest

(32, 2, 8) @ mod 3 ...

also 8

38 ist Doule

Lecht.

pause machen weiter...

David Vajda (c)
08/25/2026
asm Diagramm, Seite 17

Q = 64542522155561154315

gut:
64, 54, 26, 22

16 Zustände $Z = \{z_0, z_1, \dots z_{15}\}$

welche 3 Zustände sind ◇ ?

welche Nachbarzustände ? (Kanten?)

$dec(64_{6er}) = 6 \cdot 6^1 + 4 \cdot 6^0$
$= 36 + 4 = 40$

und 40 mod (rest) 16 = 8

ok... bleibt 32 als: quotient · divisor
2 · 16

32 + 54 = 86,

$dec(86_{6er}) = 8 \cdot 6^1 + 6 \cdot 6^0$
$= 46 + 6 = 52$

52 rest 16 = (32 + 16 + 272) rest 16
= (48 + 4) rest 16 = 4

$z_{m_2} = 7$

ok: $z_{m_0} = z_4$
$z_{m_1} = z_9$

(25 + 48) rest 16
= 73 rest 16 =
= (64 + 7) rest 16 = 7

Jamd Vajda, Seite 18
08/25/2026
zum Diagramme...

nun kommen die
kante$_{2\,out}$

vertices:

3 vertices: 3 nachbarn:
wie bei normalen...
Knoten für anweisungen

1 in
1 out
2 nachbarn zwei kanten
bei Kante beziehung:
3
1 in, 2 out...
3 kanten, 3 Nachbarn...

64+22 = 86

86 rest 16 sind:
64 + 16 = 80 ✗ falsch
5·(8)16 = 50+30 ✓ ok

also rest 6
$v_{m_1} = (4, 6)$
$v_{m_2} = (15 \overset{8}{\leftarrow}) = (8, 15)$

80 + 15 = 95
95 rest 16 = 15
$v_{m_3} = (7, 6)$...

15 + 55 = 70
rest 16
= 6

David Vajda (c)
08/25/2026
Seite 20
Asm Diagramm...
nun folgen

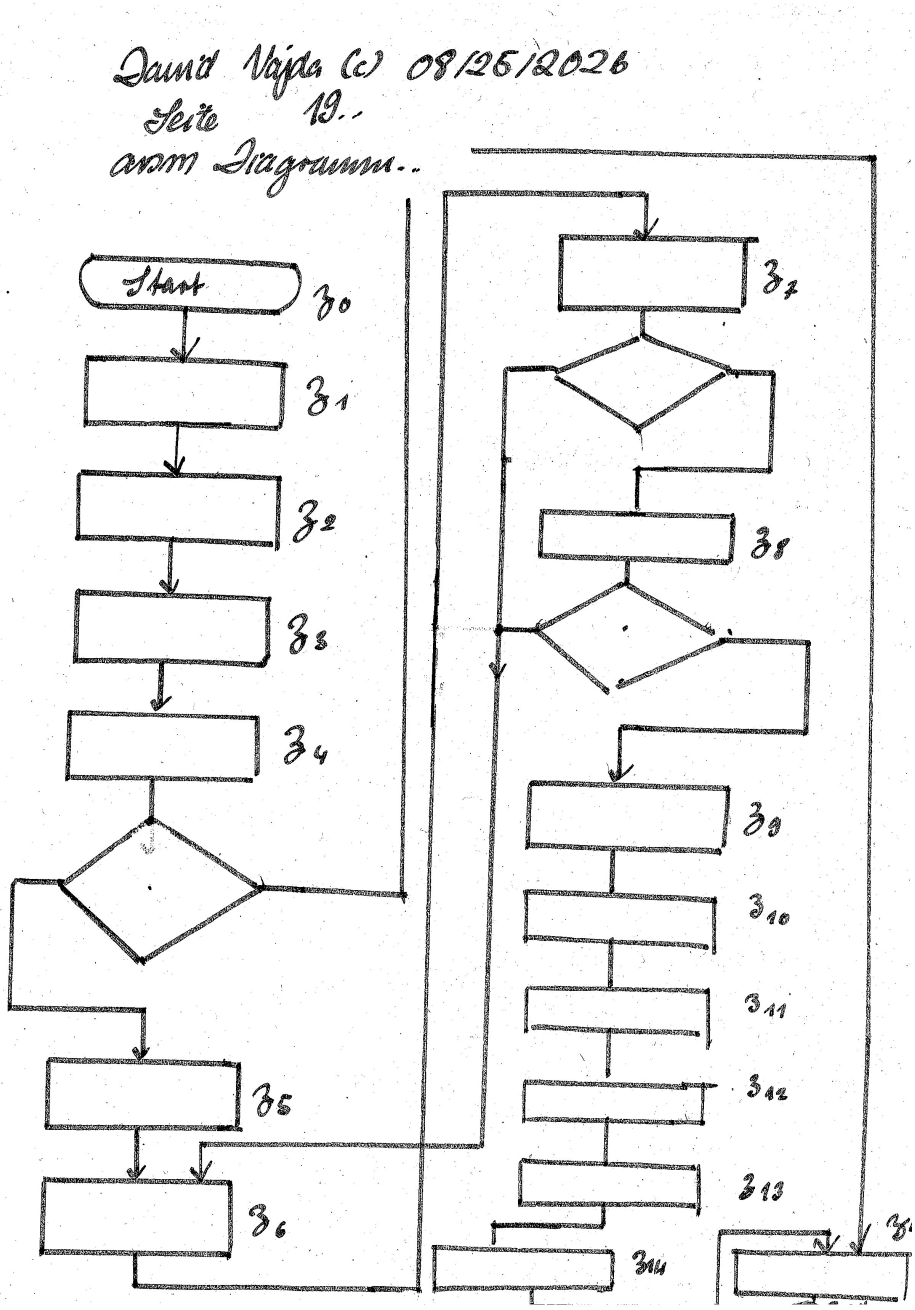
Register
operanden + operatoren...

r0: 1 x=1, 0
r1: 2 x=2, 1
r2: 3 x=3, 2
r3: 4 x=4, 3
r4: 5 x=6, 4
5
r5: 6 x=6, 5

oh: 3 operatoren für anweisungen
kein inkrement
dekrement
r1 ← r1 + r2
wobei vorher
r2 ← ?

oh: hier gehts so nicht
braucht man aber nicht..

add:	+	0 (1)	> :	0 (1)
sub:	-	1 (2)	= :	1 (2)
mul:	*	2 (3)	< :	2 (3)
div:		3 (4)	!= :	3 (4)


David Vajda (c) 08/26/2026
Seite 19..
vom Diagramm..
Start
Z0
Z1
Z2
Z3
Z4
Z5
Z6
Z7
Z8
Z9
Z10
Z11
Z12
Z13
Z14

Daniel Vajda (c) 08/25/2026
Seite 21
vom Diagramm...

wir haben vergessen den 6er
falsch 6er System dann zu
machen. Mach nichts.
Dann war es halt in dem
Falle so.
dazu: wir müssen
(f(x=1)=0, f(x=2)=1, ...,
f(x=6))
und dann kommen Register +
Befehle +... (+) indem wir erst
↑ Fehlerhaft
so :)
↑ merkt ob?
eine Folge von Zahlen erzeugen...
6er Folge r6, ..., r0
4er Folge: für opc:
add/sub/mul/div +, -, *, /
>, =, <, !=
0, 1, 2, 3...

David Vajda (c)
08/25/2026
Seite 22, aom diagramm...

64 54 25 22 15 55 61 15

$6 \cdot 6^1 + 4 \cdot 6^0 = 36 + 4 = 40$
$5 \cdot 6^1 + 4 \cdot 6^0 = 30 + 4 = 34$
$2 \cdot 6^1 + 5 \cdot 6^0 = 12 + 5 = 17$
$2 \cdot 6^1 + 2 \cdot 6^0 = 12 + 2 = 14$
$1 \cdot 6^1 + 5 \cdot 6^0 = 6 + 5 = 11$
$5 \cdot 6^1 + 5 \cdot 6^0 = 30 + 5 = 35$
$6 \cdot 6^1 + 1 \cdot 6^0 = 36 + 1 = 37$
$1 \cdot 6^1 + 5 \cdot 6^0 = 6 + 5 = 11$

ok nun: wir brauchen:

16 Zustände: ____ Lesbare notation
= + 3 + 13 (2+3) keine distributivität...

3·16 Register à 6 Werten
(0..5)

40 mod 6 =

falsch wir hätten eben
0..5 drauf machen
sollen...

David Vajda (c) 08/25/2026

Seite 23, zum diagrammieren:

64 54 23 22 15 55 61 15

$\sim$

53 43 14 11 04 44 50 05

ok: so besser...

$5 \cdot 6^1$... ok wir nehmen wir den fall den taschenrechner zur hilfe

$a_3 \cdot 6^3 + a_2 \cdot 6^2 + a_1 \cdot 6^1 + a_0 \cdot 6^0$

$+ \dots$

$6^{10} =$.
$6^9 =$:
$6^8 =$
$6^7 =$
6^6 :
:

besser anders herum

$(a_n) = 6^n$

$(a_n) = (a_{n-1}) \cdot 6$

$(a_0) = 1$

ok: wir (a_n) machen anstelle (b_n)... basis jedenfalls

$(a_0) = 1$
$(a_1) = (a_0) \cdot 6 = 1 \cdot 6 = 6$
$(a_2) = (a_1) \cdot 6 = 6 \cdot 6 =$
$(a_3) = (a_2) \cdot 6 =$

David Vajda (c)
08/25/2026
Seite 22, asm diagramm...

64 54 25 22 15 55 61 15

$6 \cdot 6^1 + 4 \cdot 6^0 = 36 + 4 = 40$
$5 \cdot 6^1 + 4 \cdot 6^0 = 30 + 4 = 34$
$2 \cdot 6^1 + 5 \cdot 6^0 = 12 + 5 = 17$
$2 \cdot 6^1 + 2 \cdot 6^0 = 12 + 2 = 14$
$1 \cdot 6^1 + 5 \cdot 6^0 = 6 + 5 = 11$
$5 \cdot 6^1 + 5 \cdot 6^0 = 30 + 5 = 35$
$6 \cdot 6^1 + 1 \cdot 6^0 = 36 + 1 = 37$
$1 \cdot 6^1 + 5 \cdot 6^0 = 6 + 5 = 11$

ok nun: wir brauchen:

16 Zustände: ____ Lebode notation
= + 3 + 13 (2+3) keine distributivität...

3·16 Register à 6 Werten
(0..5)

40 mod 6 =

Falsch wir hätten eben
0..5 daraus machen
sollen...

Daniel Vajda (a)

08/25/2026

Seite 23, zum Diagramm..

40341714 nehmen wir jetzt
als Zahl und den Taschenrechner
zur Hilfe: Das heißt:
wir teilen durch 16
wenn wir das tun der Taschenrechner..
teilt und macht nicht wie
intel div befehl
div bx
wobei dx:ax
geteilt wird
oder
div bh ? bl geht dl ?
je zwar wäre operand nicht
möglich
div dl
dann ax...
ok: dann würde der Rest...
dx würde der Rest
ax enthält den ganzzahl
quotient..
wenn ich taschenrechner nehme,
bleibt eine Nachkommazahl...
wie den Rest herausfinden...
ich ziehe das vor dem Komma
ab und multipliziere mit dem Divisor?

David Vejda (c) 08/25/2026
Seite 24, asm diagramm...

ok: Dann habe ich den Rest.
Den teil vorne merken. also los
gehts es...
(Rest (zu) mal 16x durch)
Das ist richtig
16x durch 4 teilen
Rest : operation
16x3 x durch 6 teilen
operanden
ist asm Diagramm..

David Vejda (c) 08/25/2026
Seite 24, asm diagramm...

ok: dann haben wir den Rest.
den teil vorne merken. also los
gehts es...
(Rest (w) mal 16x durch)
das ist wichtig
16x durch 4 teilen
Rest: operation
16x3 x durch 6 teilen
operanden
wie asm Diagramm..

gut halt: nun haben wir
eine andere Zahl.
wir haben:

$6^1 \cdot a_1 + 6^0 \cdot a_0$

ok. doch wir müßten haben:

$6^n \cdot a_n + 6^{n-1} \cdot a_{n-1} + \dots + 6^0 a_0$

ok...
und doch ist das dasselbe: Man
denke an binär:

09 1001. 1100

dann haben wir

$0_2 \overset{3}{1}\overset{2}{0}\overset{1}{0}\overset{0}{1}. \overset{3}{1}\overset{2}{1}\overset{1}{0}\overset{0}{0}$ $\quad$ 3210 3210
1001 . 1100
wirkt dasselbe?! Derlei ausprobieren

David Vojda CO 08/25/2026

Seite 26, asm diagramm...

also:

$(a_0) = 1$
$(a_1) = (a_0) \cdot 6 = 1 \cdot 6 = 6$
$(a_2) = (a_1) \cdot 6 = 6 \cdot 6 = 36$
$(a_3) = (a_2) \cdot 6 = 36 \cdot 6 = 216$
$(a_4) = (a_3) \cdot 6 = 216 \cdot 6 = 1296$
$(a_5) = (a_4) \cdot 6 = 6 \cdot 1296 = 7776$
$(a_6) = (a_5) \cdot 6 = 6 \cdot 7776 = 46656$
$(a_7) = (a_6) \cdot 6 = 6 \cdot 46656 = 279936$
$(a_8) = (a_7) \cdot 6 = 6 \cdot 279936 \cdot 6 = 1.679.616$
$(a_9) = (a_8) \cdot 6 = 6 \cdot 1679616 = 10.077.696$

jetzt mit taschenrechner:

$15 \cdot 1 + 61 \cdot 36 + 55 \cdot 216 + 15 \cdot 1296 +$
$22 \cdot 7776 + 25 \cdot 46656 + 54 \cdot 279936 +$
$64 \cdot 1.679.616$

$=$

$2211 + 31320 + 1337472 +$
122611968

$=$

122611968
$+ \quad 1337472$
$+ \quad 31320$
$+ \quad 2211$

1

ah: taschenrechner:
123982971

ah
das ist die Zahl sieht echt aus!
sieht immer mit rest teilen...

Daniel Vajda (c) 08/25/2026

Seite 27 arm diagramm...

Das sind ~~regiole~~ operationen

123982971 / 4 = 30995742.75 Rest 3 Zeile : op3

30995742/4 = 7748935.5 Rest 0 × 4 = 0.5 = 2

7748935/4 = 1937233.75 Rest: 0.75 · 4 = 3

1937233.0/4 = 484308.25 Rest: 4 · 0.25 = 1

484308/4 = 121077 Rest 0. 0 · 4 = 0

121077/4 = 30269.25 ⇒ 30269 Rest 1

30269/4 = 7567.25 ⇒ 7567 Rest 1

7567/4 = 1891 Rest 3

1891/4 = 472 Rest 3

472/4 = 118 Rest 0

118/4 = 29 Rest 2

29/4 = 7 Rest 1

7/4 = 1 Rest 3

3/4 = 0 Rest 3

ok: (3, 2, 3, 1, 0, 1, 1, 3, 3, 0, 2, 1, 3, 3)
13 12 11 10 9 8 7 6 5 4 3 2 1 0

Das sind ⌊ ok fehlen 2

!!

machen wir noch die Zahl 4×4 3×16 /6 wg. reg oder 4 reg besser zu benutzen 14 dann ⇒ bedeutet ???

Daud Vajda (c) 08/25/2026
Seite 28 aom diagramm...

nun wandeln wir die inzwischen
existierende Zahlenfolge entsprechend
der Knoten (im graph)
bedingter
"unbedingter" Zustand...
ausdruck!

in Befehle um:

3: z_1: div /
2: z_2: mul *
3: z_4: ne not equal !=
1: z_5: sub -
0: z_6: add +
1: z_7: equal eq ==
1: z_8: eq equal ==
3: z_8: div /
3: z_{10}: div /
0: z_{11}: add +
2: z_{12}: mul *
1: z_{13}: sub -
3: z_{14}: mul *
3: z_{15}: mul

kommodore weiss hat es hier gesehen
sonst hätte ich mir bei 18 einen
schelz erlaubt von vorne anzufangen

diese kombi benutze ich jetzt auch für
zielregister (operant)...
und als ersten Quelloperant 0 0 41 1
d o t dest source t:02

Daud Vajda (c) 08/25/2026
Seite 28 aom diagramm...

um wandeln wir die inzwischen
existierende Zahlenfolge entsprechend
der Knoten (im Graph)
bedingter
"unbedingter" Zustand...
ausdruck!

in Befehle um:

3: z_1 : div /
2: z_2 : mul *
3: z_4 : ne not equal !=
1: z_5 : sub -
0: z_6 : add +
1: z_7 : equal eq ==
1: z_8 : eq equal ==
3: z_9 : div /
3: z_{10} : div /
0: z_{11} : add +
2: z_{12} : mul *
1: z_{13} : sub -
3: z_{14} : mul *
3: z_{15} : mul

komischer weise hat es hier gereun
sonst hätte ich mir bei 18 einen
scherz erlaubt von vorne anzufangen
Diese Kombi benutze ich jetzt auch für
Zielregister (operant)...
und als ersten Quelloperant * o 41 1
Not dort nun t:source t:02

Jamil Vajda (c) 09/25/2026

Seite 29 asm diagramm...

um den zweiten Quelloperanden
A zu nehmen nehme ich das
ganze noch mal anders herum.

Damit ich natürlich fest einen
Computer unter der Voraussetzung:

1.) arbeitsdatenspeicher
zu Register
2.) die befehle wirklich
nachgeladen ← pc/ip
3.) das nachladen aus ...
dem programmspeicher
ermöglicht spring ...

$r_3 \leftarrow r_3 / r_3$
$z_2 \leftarrow z_2 \cdot z_3$
$z_3 \leftarrow$

auch z_3 vergessen...

pause!!!

19:04

mükra electronic

SN74LS181 782820

Marke: No

€1,53

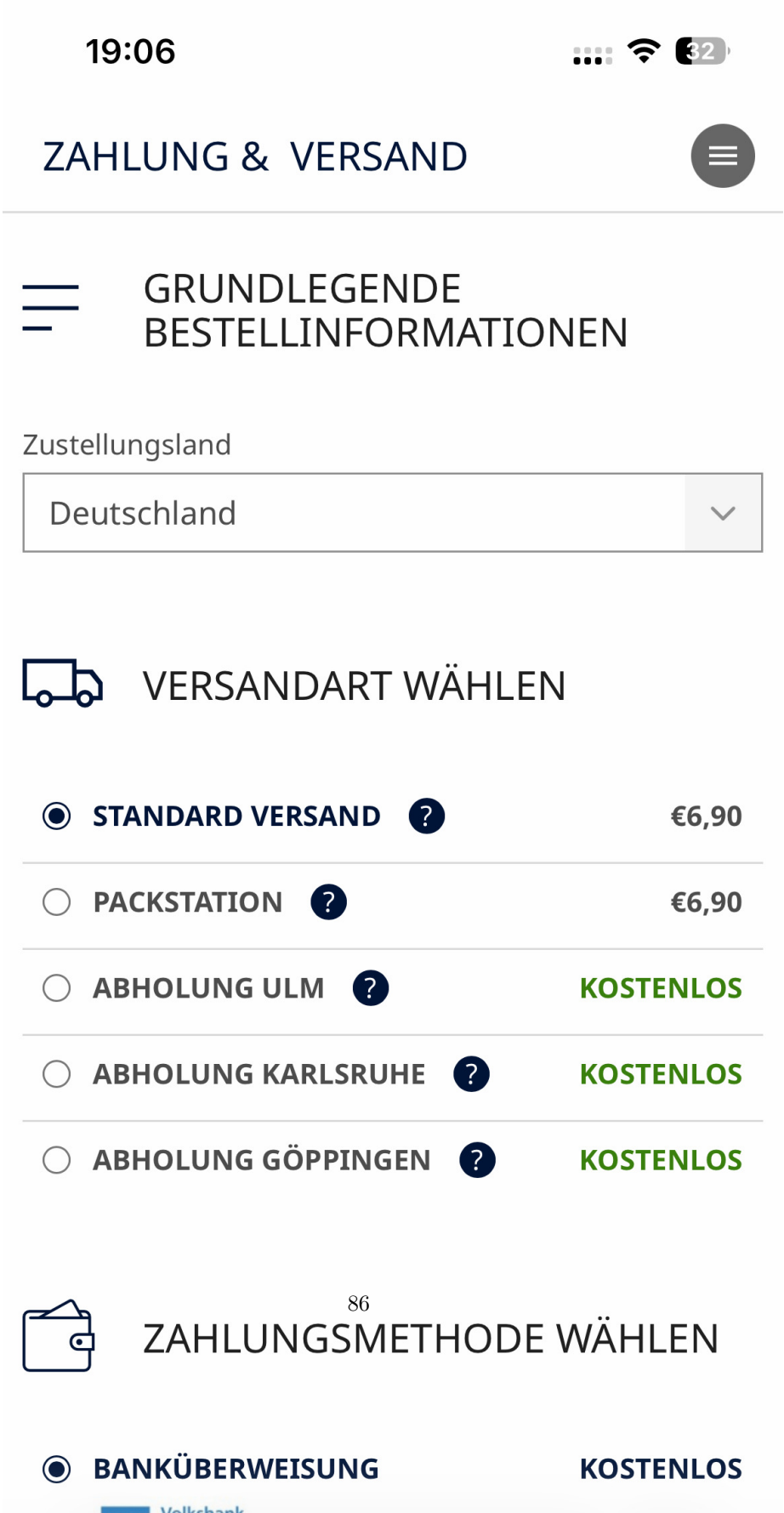
Lieferung bis: 31.08.2026

1

In den Warenkorb

TTL-Schaltung, Gehäuse: DIL24 4-Bit arithmetische / logische Einheit

Detaillierte Informationen


19:06
ZAHLUNG & VERSAND
GRUNDLEGENDE BESTELLINFORMATIONEN
Zustellungsland
Deutschland
VERSANDART WÄHLEN
STANDARD VERSAND €6,90
PACKSTATION €6,90
ABHOLUNG ULM KOSTENLOS
ABHOLUNG KARLSRUHE KOSTENLOS
ABHOLUNG GÖPPINGEN KOSTENLOS
ZAHLUNGSMETHODE WÄHLEN
BANKÜBERWEISUNG KOSTENLOS

19:25

mükra electronic

SN74LS181 782820

Marke: No

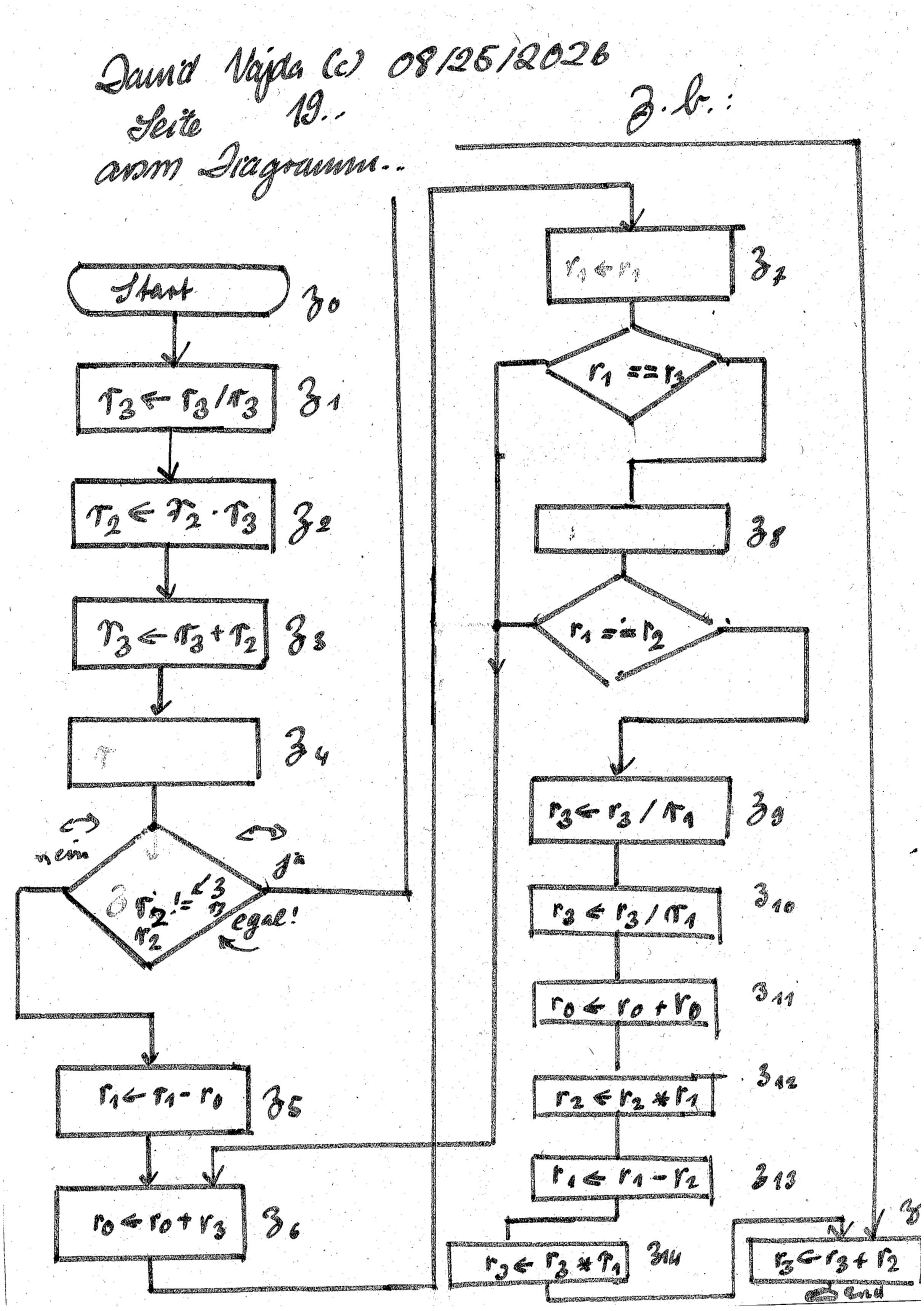
€1,53

Lieferung bis: 31.08.2026

1 In den Warenkorb

TTL-Schaltung, Gehäuse: DIL24 4-Bit arithmetische / logische Einheit

Detaillierte Informationen


David Vajda (c) 08/25/2026
Seite 19..
zum Diagramm..
Z.b.:
Start
Z0
r3 ← r3 / r3
Z1
r2 ← r2 · r3
Z2
r3 ← r3 + r2
Z3
Z4
nein
ja
r2 != r3
r2
equal!
r1 ← r1 - r0
Z5
r0 ← r0 + r3
Z6
r1 ← r1
Z7
r1 == r3
Z8
r1 == r2
r3 ← r3 / r1
Z9
r3 ← r3 / r1
Z10
r0 ← r0 + r0
Z11
r2 ← r2 * r1
Z12
r1 ← r1 - r2
Z13
r3 ← r3 * r1
Z14
r3 ← r3 + r2
Ende

Daniel Vajda (c) 08/25/2026
Seite 29 asm diagramm..
um den zweiten Quelloperanden
A zu nehmen nehmae ich das
ganze noch mal anders herum.

Damit ich natürlich fest einen
Kompute unter der Voraussetzung:

1.) arbeitsdatenoperator
zu Register
2.) die befehle wirklich
nachgeladen ← pc/ip
3.) das nachladen aus ...
dem programmspeicher
ermöglicht spring ...

$r_3 \leftarrow r_3 / r_3$
$z_2 \leftarrow z_2 \cdot z_3$
$z_3 \leftarrow$

auch z_3 vergessen... Das darf nicht bei z_3 sondern r!

pause!!!

$z_3 \leftarrow z_3 \cdot z_3$ ← tlz?
entscheidung! rtl
in asm Diagramm rtl wie
fernsehen
abreibweise

bzw. $R \subseteq rtl$
R untermenge..

6 und nun. . . jetzt kommt das schaltnetz

z_n	x	z_{n+1}^+	y	XOR
$\frac{0b1000}{0}$	0	$\frac{0b0100}{1}$	1	1
$\frac{0b1000}{1}$	0	$\frac{0b0100}{1}$	1	1
$\frac{0b1000}{0}$	1	$\frac{0b0010}{0}$	0	1
$\frac{0b1000}{1}$	1	$\frac{0b0010}{0}$	0	1
$\frac{0b0100}{0}$	0	$\frac{0b0010}{0}$	0	0
$\frac{0b0100}{0}$	1	$\frac{0b0000}{0}$	0	0
$\frac{0b0010}{0}$	0	$\frac{0b0000}{0}$	0	1
$\frac{0b0010}{1}$	0	$\frac{0b0000}{0}$	0	1
$\frac{0b0010}{0}$	1	$\frac{0b1000}{1}$	1	1
$\frac{0b0010}{1}$	1	$\frac{0b1000}{1}$	1	1
$\frac{0b0001}{1}$	0	$\frac{0b1000}{1}$	1	0
$\frac{0b0001}{1}$	1	$\frac{0b0100}{1}$	1	0

```
z3+ := (z0 and not x) or (z1 and x)
z2+ := (z3 and not x) or (z0 and x)
z1+ := (z3 and x) or (z2 and not x)
z0+ := (z2 and x) or (z1 and not x)

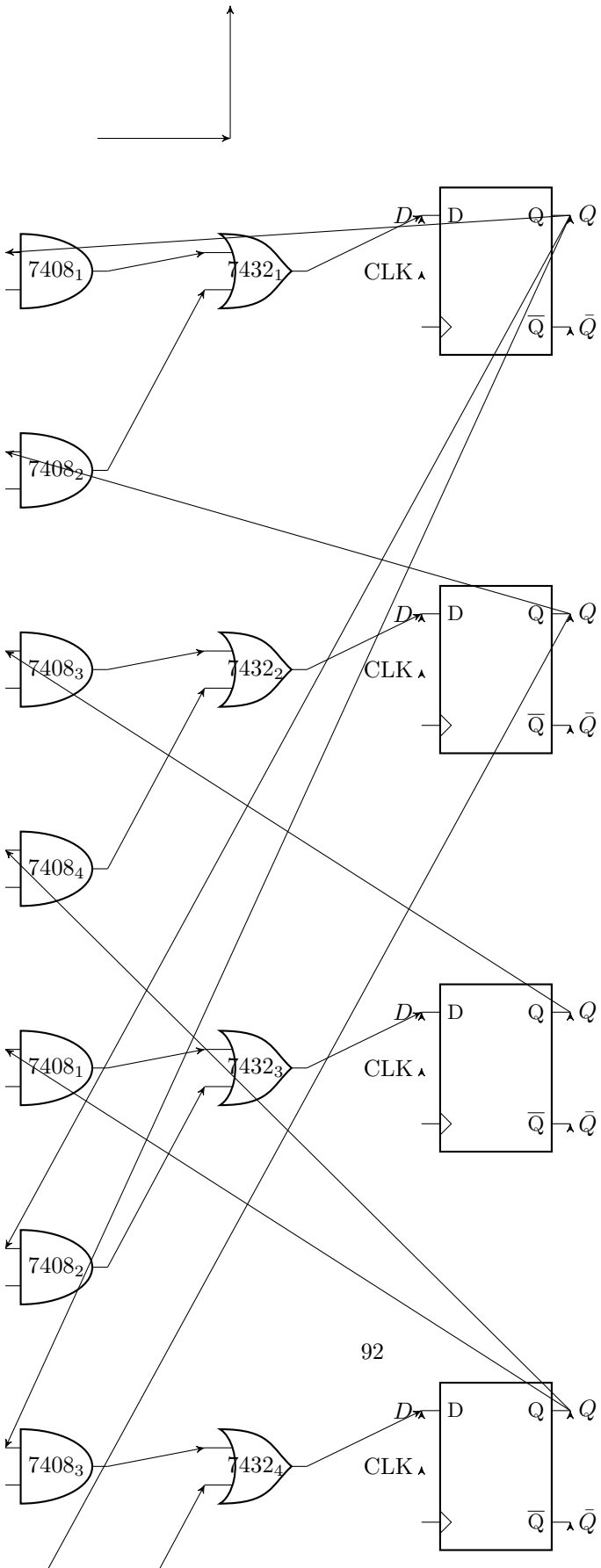
y := (z3 and not x) or (z1 and not x) or \\
 (z0 and not x) or (z0 and x)
```

in der tabelle war ein fehler: der zustand 1 muss nicht als 0 kodiert sein. ist es auch, blos leider nur im zielzustand, im ausgangszustand, ist es eins. im

z_n	x	z_{n+1}^+	y
0b1000	0	0b0100	1
0b1000	1	0b0010	0
0b0100	0	0b0010	0
0b0100	1	0b0001	0
0b0010	0	0b0001	0
0b0010	1	0b1000	1
0b0001	0	0b1000	1
0b0001	1	0b0100	1

zielzustand war es null

jetzt haben wir das schaltnetz, dann gucke ich, dass ich das als schaltnetz tue, und zwar: soll ich mit kaneda, ja oder nein? ich denke so oder so, ja, aber ich mache es trotzdem mit LaTeX. und zwar indem ich erst mal 8 UND Gatter hinmache 7408, oder wie, also nach IEEE-91/1984. ok: und: jetzt kommt: vier ODER Gatter, wir verbinden das, auch, dann kommt noch Kaneda. und dann 4 FF. wie von vorher übernommen.


7408$_1$
7432$_1$
7408$_2$
7408$_3$
7432$_2$
7408$_4$
7408$_1$
7432$_3$
7408$_2$
7408$_3$
7432$_4$
D
Q
CLK
$\overline{\text{Q}}$
D
Q
$\bar{Q}$

kurze zeichenübung - ist teil der sache:

```
\begin{tikzpicture}[->, >=stealth', node distance=3cm, auto]
\node [and port] (and00) {$7408_1$};
\node [and port, below of=and00] (and01) {$7408_2$};
\node [and port, below of=and01] (and02) {$7408_3$};
\node [and port, below of=and02] (and03) {$7408_4$};
\node [and port, below of=and03] (and04) {$7408_1$};
\node [and port, below of=and04] (and05) {$7408_2$};
\node [and port, below of=and05] (and06) {$7408_3$};
\node [and port, below of=and06] (and07) {$7408_4$};

\node [or port, right of=and00] (or00) {$7432_1$};
\node [or port, right of=and02] (or01) {$7432_2$};
\node [or port, right of=and04] (or02) {$7432_3$};
\node [or port, right of=and06] (or03) {$7432_4$};

\draw (and00.out) -- (or00.in 1);
\draw (and01.out) -- (or00.in 2);
\draw (and02.out) -- (or01.in 1);
\draw (and03.out) -- (or01.in 2);
\draw (and04.out) -- (or02.in 1);
\draw (and05.out) -- (or02.in 2);
\draw (and06.out) -- (or03.in 1);
\draw (and07.out) -- (or03.in 2);


    \node[flipflop D, right of=or00] (D00) {};
    \draw (D00.pin 1) node[left] {$D$} ;
    \draw (D00.pin 2) node[left] {CLK};
    \draw (D00.pin 6) node[right] {$Q$};
    \draw (D00.pin 4) node[right] {$\bar{Q}$};


        \node[flipflop D, right of=or01] (D01) {};
    \draw (D01.pin 1) node[left] {$D$} ;
    \draw (D01.pin 2) node[left] {CLK};
    \draw (D01.pin 6) node[right] {$Q$};
    \draw (D01.pin 4) node[right] {$\bar{Q}$} {};


        \node[flipflop D, right of=or02] (D02) {};
    \draw (D02.pin 1) node[left] {$D$} ;
    \draw (D02.pin 2) node[left] {CLK};
    \draw (D02.pin 6) node[right] {$Q$};
    \draw (D02.pin 4) node[right] {$\bar{Q}$};
```

```
        \node[flipflop D, right of=or03] (D03) {};
    \draw (D03.pin 1) node[left] {$D$} ;
    \draw (D03.pin 2) node[left] {CLK};
    \draw (D03.pin 6) node[right] {$Q$};
    \draw (D03.pin 4) node[right] {$\bar{Q}$};

\draw (or00.out) -- (D00.pin 1);
\draw (or01.out) -- (D01.pin 1);
\draw (or02.out) -- (D02.pin 1);
\draw (or03.out) -- (D03.pin 1);


\draw (D00.pin 6) -- (and06.in 1);
\draw (D01.pin 6) -- (and07.in 1);

\draw (D03.pin 6) -- (and04.in 1);
\draw (D00.pin 6) -- (and05.in 1);

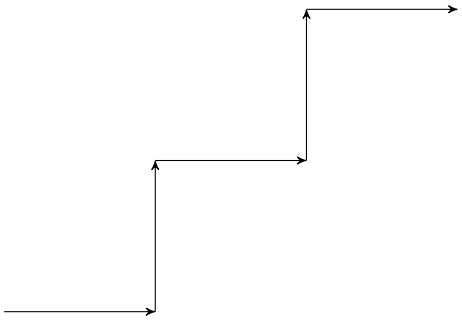
\draw (D03.pin 6) -- (and03.in 1);
\draw (D02.pin 6) -- (and02.in 1);

\draw (D01.pin 6) -- (and01.in 1);
\draw (D00.pin 6) -- (and00.in 1);

\draw (0,2) -- (2,2);
\draw (2,2) -- (2,4);

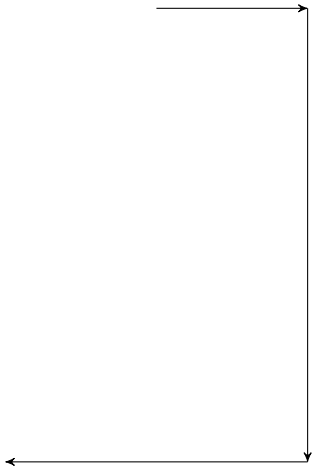


\end{tikzpicture}
```



Schnecke:

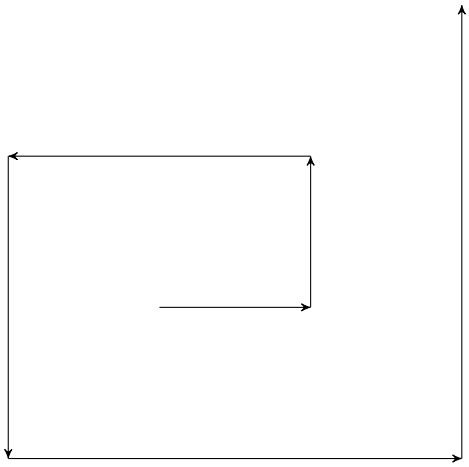
$$((n,m),(n+2,m),(n+2,m-4),(n-4,m-4),(n-4,m+8),\dots))$$



nein:

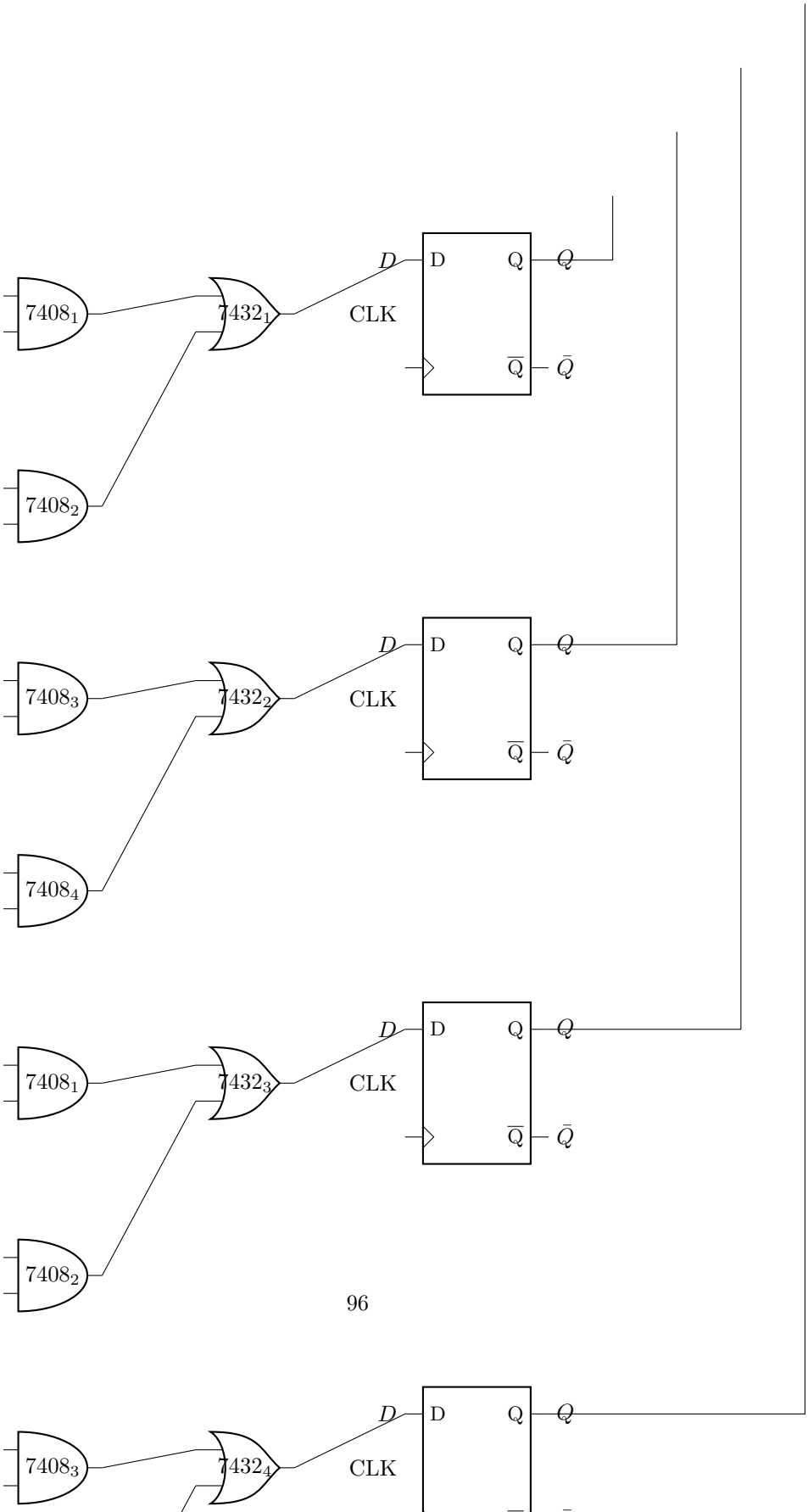
Schnecke:

$$((n,m),(n+2,m),(n+2,m+2),(n-4,m+2),\dots))$$



Es wäre interessant eine (vielleicht)

1. Schnecke mit metrisch (in anführungsstrichen) gleich bleibenden längen
2. grösser werden


7408$_1$
7408$_2$
7432$_1$
D
CLK
D
Q
Q
$\overline{\text{Q}}$
$\bar{Q}$
7408$_3$
7408$_4$
7432$_2$
D
CLK
D
Q
Q
$\overline{\text{Q}}$
$\bar{Q}$
7408$_1$
7408$_2$
7432$_3$
D
CLK
D
Q
Q
$\overline{\text{Q}}$
$\bar{Q}$
7408$_3$
7408$_4$
7432$_4$
D
CLK
D
Q
Q
$\overline{\text{Q}}$
$\bar{Q}$

```
\begin{tikzpicture}[node distance=3cm, auto]
\node [and port] (and00) {$7408_1$};
\node [and port, below of=and00] (and01) {$7408_2$};
\node [and port, below of=and01] (and02) {$7408_3$};
\node [and port, below of=and02] (and03) {$7408_4$};
\node [and port, below of=and03] (and04) {$7408_1$};
\node [and port, below of=and04] (and05) {$7408_2$};
\node [and port, below of=and05] (and06) {$7408_3$};
\node [and port, below of=and06] (and07) {$7408_4$};

\node [or port, right of=and00] (or00) {$7432_1$};
\node [or port, right of=and02] (or01) {$7432_2$};
\node [or port, right of=and04] (or02) {$7432_3$};
\node [or port, right of=and06] (or03) {$7432_4$};

\draw (and00.out) -- (or00.in 1);
\draw (and01.out) -- (or00.in 2);
\draw (and02.out) -- (or01.in 1);
\draw (and03.out) -- (or01.in 2);
\draw (and04.out) -- (or02.in 1);
\draw (and05.out) -- (or02.in 2);
\draw (and06.out) -- (or03.in 1);
\draw (and07.out) -- (or03.in 2);


    \node[flipflop D, right of=or00] (D00) {};
    \draw (D00.pin 1) node[left] {$D$} ;
    \draw (D00.pin 2) node[left] {CLK};
    \draw (D00.pin 6) node[right] {$Q$};
    \draw (D00.pin 4) node[right] {$\bar{Q}$};


        \node[flipflop D, right of=or01] (D01) {};
    \draw (D01.pin 1) node[left] {$D$} ;
    \draw (D01.pin 2) node[left] {CLK};
    \draw (D01.pin 6) node[right] {$Q$};
    \draw (D01.pin 4) node[right] {$\bar{Q}$} {};

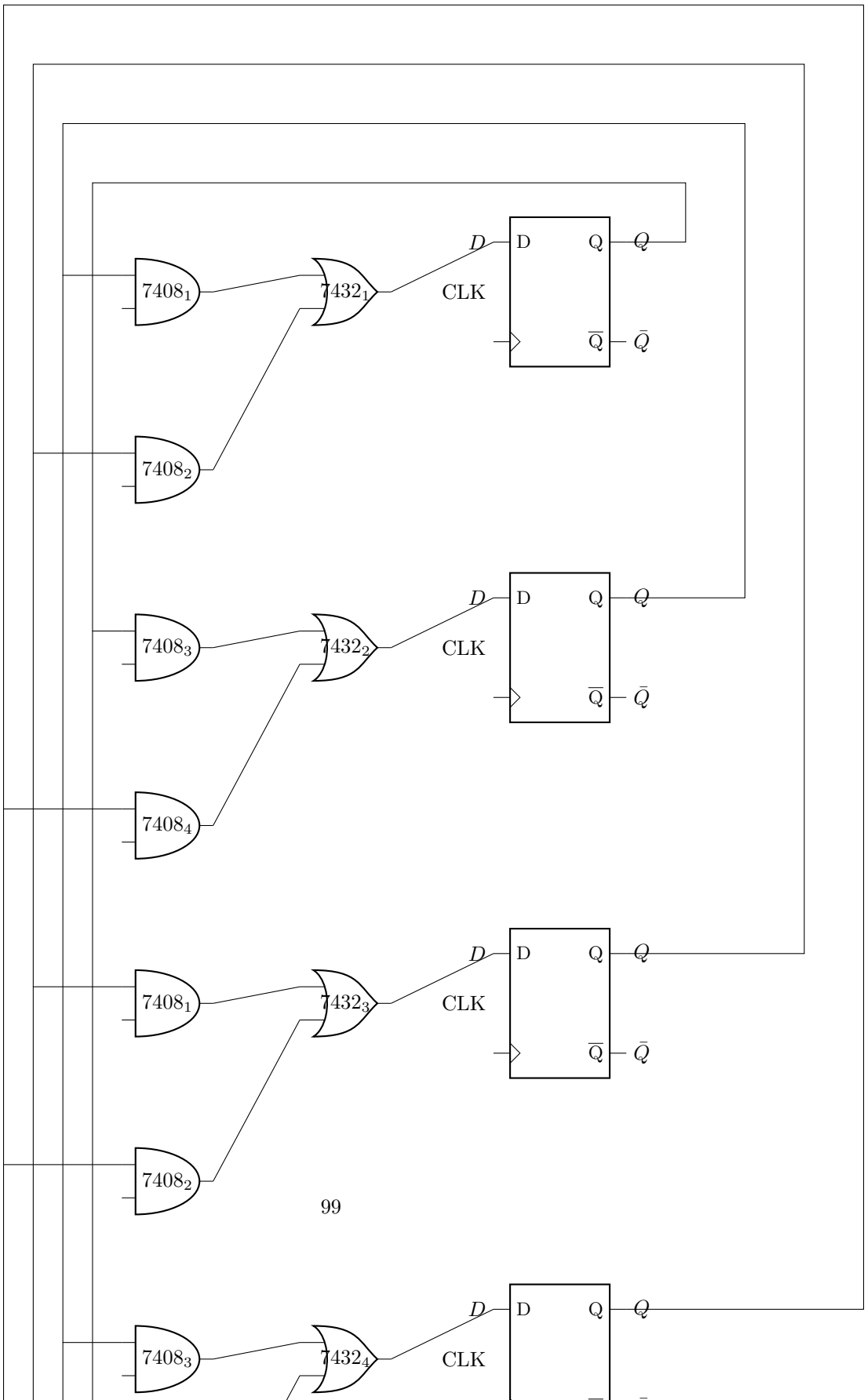

        \node[flipflop D, right of=or02] (D02) {};
    \draw (D02.pin 1) node[left] {$D$} ;
    \draw (D02.pin 2) node[left] {CLK};
    \draw (D02.pin 6) node[right] {$Q$};
    \draw (D02.pin 4) node[right] {$\bar{Q}$};
```

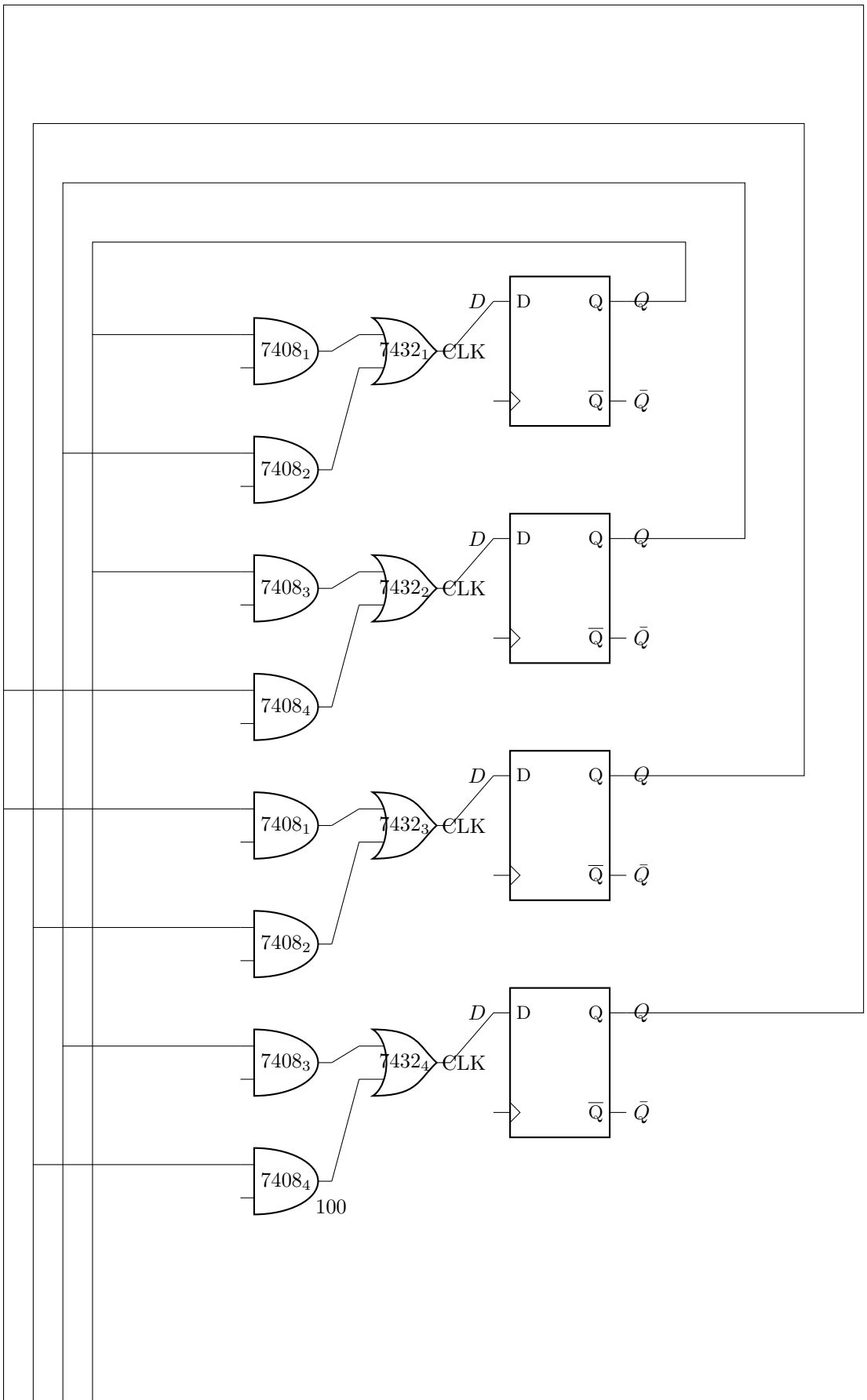
```
        \node[flipflop D, right of=or03] (D03) {};
    \draw (D03.pin 1) node[left] {$D$} ;
    \draw (D03.pin 2) node[left] {CLK};
    \draw (D03.pin 6) node[right] {$Q$};
    \draw (D03.pin 4) node[right] {$\bar{Q}$};

\draw (or00.out) -- (D00.pin 1);
\draw (or01.out) -- (D01.pin 1);
\draw (or02.out) -- (D02.pin 1);
\draw (or03.out) -- (D03.pin 1);

\draw (D00.pin 6) -- +(1,0) -- +(1,1);
\draw (D01.pin 6) -- +(2,0) -- +(2,8);
\draw (D02.pin 6) -- +(3,0) -- +(3,15);
\draw (D03.pin 6) -- +(4,0) -- +(4,22);

\end{tikzpicture}
```


7408$_1$
7432$_1$
7408$_2$
7408$_3$
7432$_2$
7408$_4$
7408$_1$
7432$_3$
7408$_2$
7408$_3$
7432$_4$
D
CLK
Q
$\overline{\text{Q}}$
Q
$\bar{Q}$


7408_1
7432_1
CLK
D
Q
Q
$\overline{\text{Q}}$
$\bar{Q}$
7408_2
7408_3
7432_2
7408_4
7408_1
7432_3
7408_2
7408_3
7432_4
7408_4
100

```
z3+ := (z0 and not x) or (z1 and x)
z2+ := (z3 and not x) or (z0 and x)
z1+ := (z3 and x) or (z2 and not x)
z0+ := (z2 and x) or (z1 and not x)

y := (z3 and not x) or (z1 and not x) or \\
 (z0 and not x) or (z0 and x)

\begin{tikzpicture}[node distance=2cm, auto]
\node [and port] (and00) {$7408_1$};
\node [and port, below of=and00] (and01) {$7408_2$};
\node [and port, below of=and01] (and02) {$7408_3$};
\node [and port, below of=and02] (and03) {$7408_4$};
\node [and port, below of=and03] (and04) {$7408_1$};
\node [and port, below of=and04] (and05) {$7408_2$};
\node [and port, below of=and05] (and06) {$7408_3$};
\node [and port, below of=and06] (and07) {$7408_4$};

\node [or port, right of=and00] (or00) {$7432_1$};
\node [or port, right of=and02] (or01) {$7432_2$};
\node [or port, right of=and04] (or02) {$7432_3$};
\node [or port, right of=and06] (or03) {$7432_4$};

\draw (and00.out) -- (or00.in 1);
\draw (and01.out) -- (or00.in 2);
\draw (and02.out) -- (or01.in 1);
\draw (and03.out) -- (or01.in 2);
\draw (and04.out) -- (or02.in 1);
\draw (and05.out) -- (or02.in 2);
\draw (and06.out) -- (or03.in 1);
\draw (and07.out) -- (or03.in 2);


    \node[flipflop D, right of=or00] (D00) {};
    \draw (D00.pin 1) node[left] {$D$} ;
    \draw (D00.pin 2) node[left] {CLK};
    \draw (D00.pin 6) node[right] {$Q$};
    \draw (D00.pin 4) node[right] {$\bar{Q}$};


        \node[flipflop D, right of=or01] (D01) {};
    \draw (D01.pin 1) node[left] {$D$} ;
    \draw (D01.pin 2) node[left] {CLK};
    \draw (D01.pin 6) node[right] {$Q$};
    \draw (D01.pin 4) node[right] {$\bar{Q}$} {};
```

```
        \node[flipflop D, right of=or02] (D02) {};
    \draw (D02.pin 1) node[left] {$D$} ;
    \draw (D02.pin 2) node[left] {CLK};
    \draw (D02.pin 6) node[right] {$Q$};
    \draw (D02.pin 4) node[right] {$\bar{Q}$};


        \node[flipflop D, right of=or03] (D03) {};
    \draw (D03.pin 1) node[left] {$D$} ;
    \draw (D03.pin 2) node[left] {CLK};
    \draw (D03.pin 6) node[right] {$Q$};
    \draw (D03.pin 4) node[right] {$\bar{Q}$};

\draw (or00.out) -- (D00.pin 1);
\draw (or01.out) -- (D01.pin 1);
\draw (or02.out) -- (D02.pin 1);
\draw (or03.out) -- (D03.pin 1);

\draw (D00.pin 6) -- +(1,0) -- +(1,1) -- +(-9,1) -- +(-9,-22);
\draw (D01.pin 6) -- +(2,0) -- +(2,6) -- +(-9.5,6) -- +(-9.5,-20);
\draw (D02.pin 6) -- +(3,0) -- +(3,11) -- +(-10,11) -- +(-10,-18);
\draw (D03.pin 6) -- +(4,0) -- +(4,17) -- +(-10.5,17) -- +(-10.5,-17);

\draw (and07.in 1) -- +(-3.5,0);
\draw (and06.in 1) -- +(-3,0);

\draw (and05.in 1) -- +(-3.5,0);
\draw (and04.in 1) -- +(-4,0);

\draw (and03.in 1) -- +(-4,0);
\draw (and02.in 1) -- +(-2.5,0);

\draw (and01.in 1) -- +(-3.0,0);
\draw (and00.in 1) -- +(-2.5,0);
\end{tikzpicture}
```

7 quine mc cluskey

```
(C) David Vajda
Sun Aug 30 19:25:20 2026
4 Network - TTL - Disjunktive Normalform

  0 0 0 0 0    0
  1 0 0 0 1    1
```

```
 2 0 0 1 0    0
 3 0 0 1 1    1
 4 0 1 0 0    0
 5 0 1 0 1    0
 6 0 1 1 0    0
 7 0 1 1 1    1
 8 1 0 0 0    1
 9 1 0 0 1    1
10 1 0 1 0    1
11 1 0 1 1    1
12 1 1 0 0    1
13 1 1 0 1    1
14 1 1 1 0    0
15 1 1 1 1    0
```

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

```
(C) David Vajda
Sun Aug 30 19:25:20 2026
4 Network - TTL - Disjunktive Normalform

 1 0 0 0 1    1
 3 0 0 1 1    1
 7 0 1 1 1    1
 8 1 0 0 0    1
 9 1 0 0 1    1
10 1 0 1 0    1
11 1 0 1 1    1
12 1 1 0 0    1
13 1 1 0 1    1
```

```
Gruppe 1:
  1 0 0 0 1    1
  8 1 0 0 0    1
Gruppe 2:
  3 0 0 1 1    1
  9 1 0 0 1    1
10 1 0 1 0    1
12 1 1 0 0    1
Gruppe 3:
  7 0 1 1 1    1
11 1 0 1 1    1
13 1 1 0 1    1

1:3  0 0 - 1
1:8  - 0 0 1
8:9  1 0 0 -
8:10  1 0 - 0
8:12  1 - 0 0

3:7  0 - 1 1
3:11  - 0 1 1
9:11  1 0 - 1
9:13  1 - 0 1
10:7  1 0 1 -
12:13 1 1 0 -


1:3  0 0 x 1
1:8  x 0 0 1
8:9  1 0 0 x
8:10  1 0 x 0
8:12  1 x 0 0

3:7  0 x 1 1
3:11  x 0 1 1
9:11  1 0 x 1
9:13  1 x 0 1
10:7  1 0 1 x
12:13 1 1 0 x



1:8  x 0 0 1
3:11  x 0 1 1
8:12  1 x 0 0
3:7  0 x 1 1
```

```
9:13  1 x 0 1
9:11  1 0 x 1
8:10  1 0 x 0
1:3  0 0 x 1
10:7  1 0 1 x
12:13 1 1 0 x
8:9  1 0 0 x



1:8  x 0 0 1
3:11  x 0 1 1

8:12  1 x 0 0
3:7  0 x 1 1
9:13  1 x 0 1

9:11  1 0 x 1
8:10  1 0 x 0
1:3  0 0 x 1

10:7  1 0 1 x
12:13 1 1 0 x
8:9  1 0 0 x



1:8  x 0 0 1
3:11  x 0 1 1
1:8:3:11  x 0 x 1

Gruppe 1:
8:12  1 x 0 0
Gruppe 2:
9:13  1 x 0 1
3:7  0 x 1 1

8:12   1 x 1 1
8:12:9:13  1 x 0 x


9:11  1 0 x 1
8:10  1 0 x 0
1:3  0 0 x 1

9:11:8:10  1 0 x x
```

```
9:13:1:3  - 0 x 1

8:9  1 0 0 x
10:7  1 0 1 x
12:13 1 1 0 x

8:9:10:7  1 0 x x
8:9:12:13  1 x 0 x



1:8:3:11  x 0 x 1
8:12   1 x 1 1
8:12:9:13  1 x 0 x
9:11:8:10  1 0 x x
9:13:1:3  x 0 x 1??
8:9:10:7  1 0 x x
8:9:12:13  1 x 0 x
```

gut ein wenig weiter, standard übung

```
punkt, linie, zoll = finger, handbreit, fuss
punkt, linie, zoll = finger, handbreit, fuss
punkt, linie, zoll = finger, handbreit, fuss
punkt, linie, zoll = finger, handbreit, fuss
elle, klafter, lachter, meile
elle, klafter, lachter, meile
elle, klafter, lachter, meile
elle, klafter, lachter, meile

punkt = 0.35mm
punkt = 0.35mm
punkt = 0.35mm

1pt = 1 dot = 0.35mm
1pt = 1 dot (??) = 0.35mm

# sorry nur ein wenig uebung

1 pt = 0.35mm
1 pt = 0.35mm

punkt, linie, zoll = finger, handbreit, fuss
```

```
1 zoll = 1 inch, ein zeigefinger spitze = 2.5cm bei david vajda (c)
1 zoll = 1 inch

punkt, linie, zoll = finger, handbreit, fuss
1 fuss = 3 zoll
1 fuss = 3 zoll
1 elle = 2 fuss
1 elle = 2 fuss
... hier herrscht unklarheit, gleich lernen.

cent, gran, karat, skrupel, heller, pfennig, mark, pfund, unze, centner
cent, gran, karat, skrupel, heller, pfennig, mark, pfund, unze, centner
cent, gran, karat, skrupel, heller, pfennig, mark, pfund, unze, centner

1 pfund = 2 mark
1 pfund = 2 mark

1pfd = 1 pfund
stearling silber: 785, 885, ...

cent, gran, karat, skrupel, heller, pfennig, mark, pfund, unze, centner
```

8 das wurde geübt

nächster teil. es ging um die zufallszahlen in der y-Reihe. in der Senkrechten.
bisher habe ich das wuerde sich von der wahrscheinlichkeit eines zufallsexperiments, genau gleich verhalten, wobei hier kommen vielleicht besonderheiten der Zufallszahlen zustande.

so: ich habe jede 0 und 1 in der senkrechten y einzeln zufällig erzeugt.

trotzdem kann man ja 16 Bit werte nehmen, worte. so, das kann man erzeugen, und als y nehmen.

wenn wir das haben, dann: haben wir die senkrechte als wert. aber, auch wir können vielleicht matrizen,
als: mit der Waagerechten und Senkrechten machen, aber verschiedene.
ok: deswegen lohnt sich umrechnung, hier auch die codes.

trotzdem bevorzuge ich: jetzt erst quine mc cluskey standard übung zu machen, dann auch

```
abel ... galasm
```

und dann geht es drum speicherbausteine usw. zu machen. so jetzt weiter. trotzdem erst diese zufallszahlen und das ganze ding.

```
(C) David Vajda
Sun Aug 30 19:25:20 2026
4 Network - TTL - Disjunktive Normalform

  1 0 0 0 1    1
  3 0 0 1 1    1
  7 0 1 1 1    1
  8 1 0 0 0    1
  9 1 0 0 1    1
10 1 0 1 0    1
11 1 0 1 1    1
12 1 1 0 0    1
13 1 1 0 1    1

Gruppe 1:
  1 0 0 0 1    1
  8 1 0 0 0    1
Gruppe 2:
  3 0 0 1 1    1
  9 1 0 0 1    1
10 1 0 1 0    1
12 1 1 0 0    1
Gruppe 3:
  7 0 1 1 1    1
11 1 0 1 1    1
13 1 1 0 1    1

1:3  0 0 - 1
1:8  - 0 0 1
8:9  1 0 0 -
8:10  1 0 - 0
8:12  1 - 0 0

3:7  0 - 1 1
3:11  - 0 1 1
9:11  1 0 - 1
9:13  1 - 0 1
10:11 1 0 1 -
12:13 1 1 0 -

1:3  0 0 x 1
1:8  x 0 0 1
8:9  1 0 0 x
8:10  1 0 x 0
8:12  1 x 0 0
```

```
3:7  0 x 1 1
3:11  x 0 1 1
9:11  1 0 x 1
9:13  1 x 0 1
10:11 1 0 1 x
12:13 1 1 0 x



1:8  x 0 0 1
3:11  x 0 1 1
8:12  1 x 0 0
3:7  0 x 1 1
9:13  1 x 0 1
9:11  1 0 x 1
8:10  1 0 x 0
1:3  0 0 x 1
10:11 1 0 1 x
12:13 1 1 0 x
8:9  1 0 0 x



1:8  x 0 0 1
3:11  x 0 1 1

8:12  1 x 0 0
3:7  0 x 1 1
9:13  1 x 0 1

9:11  1 0 x 1
8:10  1 0 x 0
1:3  0 0 x 1

10:7  1 0 1 x
12:13 1 1 0 x
8:9  1 0 0 x



1:8  x 0 0 1
3:11  x 0 1 1
1:8:3:11  x 0 x 1

Gruppe 1:
8:12  1 x 0 0
```

```
Gruppe 2:
9:13  1 x 0 1
3:7  0 x 1 1

8:12   1 x 1 1
8:12:9:13  1 x 0 x


9:11  1 0 x 1
8:10  1 0 x 0
1:3  0 0 x 1

9:11:8:10  1 0 x x
9:13:1:3  - 0 x 1

8:9  1 0 0 x
10:7  1 0 1 x
12:13 1 1 0 x

8:9:10:7  1 0 x x
8:9:12:13  1 x 0 x



1:8:3:11  x 0 x 1
8:12   1 x 1 1
8:12:9:13  1 x 0 x
9:11:8:10  1 0 x x
9:13:1:3  x 0 x 1
8:9:10:7  1 0 x x
8:9:12:13  1 x 0 x
```

zwischenübung: minix

```
kernel, mm, fs, (h)

./kernel/proc.c
./kernel/main.c
./kernel/mpx88.s
./kernel/klib88.s
./kernel/memory.c
./kernel/stty.c
./kernel/floppy.c
./kernel/clock.c
./kernel/disk ?? wini...c
```

```
./h/glo.h
./h/const.h
./h/type.h

.. (proc.??nein)

ok:

virtual
physisch/physical/physikalisch

clicks
bytes

vir
phys
clicks
bytes

vir, phys, clicks, bytes

map
vir
phys
len

vir phys len

mem_map
mem_vir
mem_phys
mem_len

struct mem_map {
 vir_clicks mem_vir;
 phys_clicks mem_phys;
 vir_clicks mem_len;
};

NR_REGS
NR_SEGS

11 REGS: ax bx dx cx sp bp si di cs ds ss

T: Text (Code): cs
```

```
D: Data: ds
S: Stack: ss

PIC 8259A: Programmable Interrupt Controller 8259A

Port: 0x20, 0x21, 0xA0, 0xA1 (0xa0, 0xa1):

0x20: Port
0x21: Mask
0xA0: Port
0xA1: Mask

X: Gift: 13: alt: wg. untere Nummer
X passt zu 13
XT: alter IBM PC XT

AT: neuer
AT: advanced...

#define XT_WINI_INT ?? 13
CLOCK: 8
KEYBOARD: 9

SYS_INT, ja

XT_WINI_INT
KEYBOARD_INT
CLOCK_INT
TTY_INT??
FLOPPY_INT...

trap divide interrupt

divide: abbildung auf den 0 vektor, eintrag in der interrupt vektortabelle, mit index 0 in d
trap: alle interrupts > 15, bedeutet: alle: die nicht am PIC 8259A sind
interrupt <= 15

suprise, trap, divide:
suprise: interrupt nicht erwartet

ok: botschaften:

sys_call: die einzigen system_calls die ausgetauscht werden sind:
SENDEN und EMPFANGEN, nachher in der
./kernel/proc.c?? nachher nachdenken
```

```
ok:
mini_send und mini_rec (interrupt und: sys_call)

in der
kernel.h?? oder proc.h <- zweiteres ist die

proc structure:

struct proc {

} procs [NR_TASKS+NR_PROCS];


darein kommt:

int p_regs [NR_REGS];
extern struct mem_map p_map [NR_SEGS];

usertime und systime
realtime time clocks ticks

jedenfalls:
usertime, system time vom prozess
vom aufgerufenen kind

stack pointer, anfang, grenze
int pid

process id;

..

also:

extern struct proc {
 extern int p_regs [NR_REGS];
 extern struct mem_map [NR_SEGS];
 int pid;
 // stackpointer, times...
} procs [NR_TASKS+NR_PROCS];

gut:
./kernel/main.c

grosse routine: main:
MM, FS, INIT
```

Der Kernel selber wird durch INIT eher repraesentiert, was weitere prozesse startet

TASKS: Betriebssystem
PROCS: User

schleife:

und: einsprungsaddressen:

keyboard_int, clock_int, floppy_int, disk_int
sys_call divide trap ....

erst: in der schleife die segmente:
T, D, S: anfang und laenge
und weitere...

dann interrupts initialisieren

set_vec (

set_vec (keyboard_int, ...)

ok:

mpx88.s:
klib88.s

mpx88.s

MINIX:
TTY_INT:
CLOCK_INT:
...

klib88.s:
hilfsroutinen:

dma_read
dma_write
lock
unlock
getchrome
video buffer copy...

proc.c:

mini_rec, mini_send, sys_call, interrupt

bei dem einen, werden processe gleich mit aufgebaut
bei dem anderen, nur die botschaften gesendet

diese sind caller, empfaenger und botschaft

dann kommen:
ready, unready, schedule...

ok... ich lerne heute mal auswendig, dann lerne ich das
spaeter weiter...
egal, wie vollstaendig, noch der ARM Cortex

ARM: Advanced Risc Machine

Advanced Risc Machine ldt.

ldt.: gmbh:

Linien:
A: Operating System, ... mit allem
R: Embedded System
M: Microcontroller

ARM Cortex-M7
ARM Cortex-A7
ARM Cortex-R7

ok: Dann Versionen:

zunaechst: ARM7TDMI ... what ever:

ARM: Architecture Application Level Reference Manual
TRM: technical Reference Manual

ARM, ARM, ARM

ISA: Immer: Instruction Set Architecture

ARM: Advanced Risc Machine
ARM: Architecture Application Level Reference Manual
ARM: Operating System, Embbeded, Microcontroller

Das sind Profiles

ok: Waynes?

ok:
Apple Computer, Arcorn und: VLSI Technology

VLSI: Very Large Scale Integration

VUSE: Very Ultra Super Extra

Small Medium Large

Memory Modell: x86: Tiny, Small, Medium, Compact, Large, Huge, Flat ...

Frequency: Low, Medium, High Frequency
Very High, Ultra High, Super High, Extra High

Very Large, Ultra Large...

ok: Very Large Scal Intergration Technology

Apple Computer, Arcorn und: VLSI Technology: ARM

STMicroelectronics: Atmel (heute)

dazu: PHILLIPS (PX...?)
OKI, ...

ARM Cortex-M7
ok: v4, v5, v6, v7

Thumb: daumen

bei dem prozessor:
Thumb-2: Erweiterter Befehlssatz: 32 Bit
Thumb-Instructions: 16 Bit...

ok:

Graph:
(auf Gut Deutsch: User und System)

User-Thread, Kernel...
ok: wie rein und raus:... gerade nicht praesent...

ok:
16 Register

13 GPR: General Purpose Register

r13 ... r0
Stack: Main Stack Pointer, Process Stack Pointer
Link Register: Ruecksprungaddresse
... noch was...

es gibt befehle, wie
ADD, ADC,
B: Branch, BC?? wie carry

flags sind generell: wie x86
sign, carry, overflow... (aehnlich)

atmega8 nebenbei gibt es ja noch neben
SIGN auch NEGATIVE ...
ok:

dann gibt es:
3 Formen der... Interrupts, oder wie:

jedenfalls generell:

NMI: Non Maskable Interrupts
Siehe: PIC 8259A, 0x21, 0xa1

Nested Vectored Interrupt

Verschachtetele Interrupt Routinen, oder der Vektor??

dann gibt es:

ABI: Application Binary Interface
EABI: Enhanced ABI

Advanced Periphal Bus
Advanced High Perphormance Bus
Advanced

Port, Bus:

Serial Wire: Seriell (V.24, RS232... z.B.)

JTAG: Joint Test Action Group

Debug Schnittstelle: JTAG...

```
Debug Port
JTAG Debug Port...



Es gibt mehrere Einheiten:

MMU: Memory Management Unit
MPU: Memory Protection Unit
LSU: Load and Store Unit
ALU: Arithmetic and Logic Unit
Power Management Unit

Neben, Neumann/harward...:
I/O, CPU und Mem:

Peripherals, auf dem IC, Debug Interface, what ever...

ok: pause, ich mache aufgaben weiter...
```

aufgaben:

- was muss man noch lernen: z.B. die TTL Gatter, mit Kaneda, am Besten, das standard so ungefaehr drin sitzt
- dann jetzt codes auch: CRC und so weiter... und: vorher, das fertig.
- ja: und binaeres dividieren...: das von neulich aus excel auf den anderen pc, neue aufgabe, 16 Bit Wort ausgeben und ausrechnen, erneut

```
(C) David Vajda
Sun Aug 30 19:25:20 2026
4 Network - TTL - Disjunktive Normalform

 0 0 0 0 0    0
 1 0 0 0 1    1
 2 0 0 1 0    0
 3 0 0 1 1    1
 4 0 1 0 0    0
 5 0 1 0 1    0
 6 0 1 1 0    0
 7 0 1 1 1    1
 8 1 0 0 0    1
 9 1 0 0 1    1
10 1 0 1 0    1
```

```
11 1 0 1 1    1
12 1 1 0 0    1
13 1 1 0 1    1
14 1 1 1 0    0
15 1 1 1 1    0

(C) David Vajda
Sun Aug 30 19:25:20 2026
4 Network - TTL - Disjunktive Normalform

  1 0 0 0 1    1
  3 0 0 1 1    1
  7 0 1 1 1    1
  8 1 0 0 0    1
  9 1 0 0 1    1
10 1 0 1 0    1
11 1 0 1 1    1
12 1 1 0 0    1
13 1 1 0 1    1

Gruppe 1:
  1 0 0 0 1    1
  8 1 0 0 0    1
Gruppe 2:
  3 0 0 1 1    1
  9 1 0 0 1    1
10 1 0 1 0    1
12 1 1 0 0    1
Gruppe 3:
  7 0 1 1 1    1
11 1 0 1 1    1
13 1 1 0 1    1

1:3  0 0 - 1
1:8  - 0 0 1
8:9  1 0 0 -
8:10  1 0 - 0
8:12  1 - 0 0

3:7  0 - 1 1
3:11  - 0 1 1
9:11  1 0 - 1
9:13  1 - 0 1
10:11 1 0 1 -
12:13 1 1 0 -
```

```
1:3  0 0 x 1
1:8  x 0 0 1
8:9  1 0 0 x
8:10  1 0 x 0
8:12  1 x 0 0

3:7  0 x 1 1
3:11  x 0 1 1
9:11  1 0 x 1
9:13  1 x 0 1
10:11 1 0 1 x
12:13 1 1 0 x



1:8  x 0 0 1
3:11  x 0 1 1
8:12  1 x 0 0
3:7  0 x 1 1
9:13  1 x 0 1
9:11  1 0 x 1
8:10  1 0 x 0
1:3  0 0 x 1
10:11 1 0 1 x
12:13 1 1 0 x
8:9  1 0 0 x



1:8  x 0 0 1
3:11  x 0 1 1

8:12  1 x 0 0
3:7  0 x 1 1
9:13  1 x 0 1

9:11  1 0 x 1
8:10  1 0 x 0
1:3  0 0 x 1

10:7  1 0 1 x
12:13 1 1 0 x
8:9  1 0 0 x
```

```
1:8  x 0 0 1
3:11  x 0 1 1
1:8:3:11  x 0 x 1

Gruppe 1:
8:12  1 x 0 0
Gruppe 2:
9:13  1 x 0 1
3:7  0 x 1 1

8:12   1 x 1 1
8:12:9:13  1 x 0 x


9:11  1 0 x 1
8:10  1 0 x 0
1:3  0 0 x 1

9:11:8:10  1 0 x x
9:13:1:3  - 0 x 1

8:9  1 0 0 x
10:7  1 0 1 x
12:13 1 1 0 x

8:9:10:7  1 0 x x
8:9:12:13  1 x 0 x




1:8:3:11  x 0 x 1 (hier ist was komisch)
8:12   1 x 1 1
8:12:9:13  1 x 0 x
9:11:8:10  1 0 x x
9:13:1:3  x 0 x 1 (hier ist was komisch), ok, es war eher das:
8:9:10:7  1 0 x x
8:9:12:13  1 x 0 x

9:13  1 x 0 1, ok, war rechtschreibfehler, gruppe (2 und 1)
9:13 und: 8:12, dann: selber ausdruck


1:8:3:11  x 0 x 1
8:12   1 x 1 1
8:12:9:13  1 x 0 x
```

```
9:11:8:10  1 0 x x
8:9:10:7  1 0 x x
8:9:12:13  1 x 0 x


  1 0 0 0 1    1
  3 0 0 1 1    1
  7 0 1 1 1    1
  8 1 0 0 0    1
  9 1 0 0 1    1
10 1 0 1 0    1
11 1 0 1 1    1
12 1 1 0 0    1
13 1 1 0 1    1

   1 3 7 8 9 10 11 12 13
1:8:3:11  1 1  1   1
8:12      1    1
8:12:9:13     1 1   1 1
9:11:8:10     1 1 1 1
8:9:10:7    1 1 1 1
8:9:12:13     1 1   1 1
```

ok, damit sind so oder so alle terme bedient und primimplikantentafel:

```
   1 3 7 8 9 10 11 12 13
1:8:3:11  1 1  1   1    p
8:12      1    1
8:12:9:13     1 1   1 1
9:11:8:10     1 1 1 1
8:9:10:7    1 1 1 1     p
8:9:12:13     1 1   1 1


   1 3 7 8 9 10 11 12 13
1:8:3:11  1 1  1   1    p
8:9:10:7    1 1 1 1     p
8:9:12:13     1 1   1 1
```

ok:

```
1:8:3:11  x 0 x 1
8:9:10:7  1 0 x x
8:9:12:13  1 x 0 x
```

```
 y = not x2 and not x0 or
   x3 and not x2 or
   x3 and not x1;


in galasm aus zu druecken, uebung:


device description module
device description module
device description module
device description module

GAL16V8
Module GAL16v808302026

... pins - gleich uebung

y = /x2 * /x0
  x3 * /x2
  x3 * /x1

Descriptions

oder vergessen...

ok: pins realitaet:

VCC, GND, /OE, CLK

Auf der einen Seite Input
Auf der anderen Seite Output

Im Westen: Input
Im Osten: Output

GND: Sued-West, wie immer
VCC: Pin 20: Nord-Ost

GND: Seite: Input
VCC Seite Output

CLK: ist auf Seite: Input
/OE: auf Seite Output
```

```
als

CLK X3 X2 X1 X0 NC NC NC NC GND
/OE Y NC NC NC NC NC NC NC VCC

so zum beispiel und ich glaube das ist richtig...


GAL16V8
Module GAL16v808302026

CLK X3 X2 X1 X0 NC NC NC NC GND
/OE Y NC NC NC NC NC NC NC VCC

y = /x2 * /x0
  x3 * /x2
  x3 * /x1

Description

kann sein # ist oder oder so:

GAL16V8
Module GAL16v808302026

CLK X3 X2 X1 X0 NC NC NC NC GND
/OE Y NC NC NC NC NC NC NC VCC

y = /x2 * /x0 # x3 * /x2 # x3 * /x1

Description

ok, wenn wir das haben, geht es weiter...

wir testen zunaechst mit eigenen mitteln, ist der ausdruck richtig, wir machen die tabelle i
auch ein timing diagramm.
```

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

so,

```
 y = not x2 and not x0 or
   x3 and not x2 or
   x3 and not x1;

testen auf not x2 and not x0...

  0 0 0 0 0 0
  1 0 0 0 1 1
  2 0 0 1 0 0
  3 0 0 1 1 1
  4 0 1 0 0 0
  5 0 1 0 1 0
  6 0 1 1 0 0
  7 0 1 1 1 1
  8 1 0 0 0 1
  9 1 0 0 1 1
10 1 0 1 0 1
11 1 0 1 1 1
12 1 1 0 0 1
13 1 1 0 1 1
14 1 1 1 0 1
15 1 1 1 1  1

ok:

not x2 and not x0
```

```
  0 0 0 0 0 0 1 1 1
  1 0 0 0 1 1 1
  2 0 0 1 0 0 1 1 1
  3 0 0 1 1 1 1
  4 0 1 0 0 0  1
  5 0 1 0 1 0
  6 0 1 1 0 0  1
  7 0 1 1 1 1
  8 1 0 0 0 1 1 1 1
  9 1 0 0 1 1 1
10 1 0 1 0 1 1 1 1
11 1 0 1 1 1 1
12 1 1 0 0 1  1
13 1 1 0 1 1
14 1 1 1 0 1  1
15 1 1 1 1  1


  0 0 0 0 0 0 1 1 1
  2 0 0 1 0 0 1 1 1
  8 1 0 0 0 1 1 1 1
10 1 0 1 0 1 1 1 1

x3 and not x2 or

  0 0 0 0 0 0  1
  1 0 0 0 1 1  1
  2 0 0 1 0 0  1
  3 0 0 1 1 1  1
  4 0 1 0 0 0
  5 0 1 0 1 0
  6 0 1 1 0 0
  7 0 1 1 1 1
  8 1 0 0 0 1 1 1
  9 1 0 0 1 1 1 1
10 1 0 1 0 1 1 1
11 1 0 1 1 1 1 1
12 1 1 0 0 1 1
13 1 1 0 1 1 1
14 1 1 1 0 1 1
15 1 1 1 1  1 1


  0 0 0 0 0 0  1
  1 0 0 0 1 1  1
  2 0 0 1 0 0  1
```

```
  3 0 0 1 1 1  1
  4 0 1 0 0 0
  5 0 1 0 1 0
  6 0 1 1 0 0
  7 0 1 1 1 1
  8 1 0 0 0 1 1 1
  9 1 0 0 1 1 1 1
10 1 0 1 0 1 1 1
11 1 0 1 1 1 1 1
12 1 1 0 0 1 1
13 1 1 0 1 1 1
14 1 1 1 0 1 1
15 1 1 1 1  1 1
```

x3 and not x1:

```
  8 1 0 0 0 1 1 1
  9 1 0 0 1 1 1 1
12 1 1 0 0 1 1 1
13 1 1 0 1 1 1 1
```

9 minix

SIGHUP, SIGINT, and SIGKILL are standard POSIX signals used in Unix-like operating systems
Here is the direct breakdown of their differences, behaviors, and standard use cases:

Overview Comparison

Signal Name Signal Number Trigger Method Catchable / Ignorable? Primary Modern Purpose
SIGHUP(Hangup) 1 Closing terminal, kill -1 Yes Reload configuration files or notify terminal
SIGINT(Interrupt) 2 Pressing Ctrl+C, kill -2 Yes Interrupt foreground processes gracefully.
SIGKILL (Kill) 9 kill -9 No (Handled by OS kernel) Forcefully and instantly destroy an unres

eigenbemerkung:

1. Errors
2. Call Nr
3. Signals

Daneben gibt es

1. Errors

2. Call Nr
3. Signals
4. Interrupts (Interrupt Vektor)

typische Fehler

```
34 Fehler? NERROR 34
#define NERROR 34
#define OK 0
#define ERROR -1

// Fehler Allgemein: ERROR (-1) immer, NERROR, 34 Betriebssystem Minix
// OK: 0

/*
ENOMEM
EIO
ENOENT (Oft...)

E NO ENVIRONMENT (No Environment - keine Umgebung)
ENOMEM (Kein Speicher)
EIO (IO Fehler)
EDIV (Device Fehler)

Geraet, IO, Memory

...

SIGNALS sind klassischer weise:
SIGHUB
SIGKILL
SIGINT
SIGILL
SIG...

alle lernen

/h/signal.h
/h/callnr.h
/h/error.h

SIGHUB?? wusste ich nicht

Closing Terminal
```

Also: Konsole: Bash Schliessen
Ctrl+C (das beruehmte, womit man immer sein programm beendet ist:
SIGINT

also, das Ctrl+C, was man immer drueckt, ist SIGINT

Die Konsole selber schliessen ist SIGHUB

Dieses Ctrl+C ist -2,
und dann sagt man in Zukunft mehr (ich - sie vielleicht schon - zum lernen)
Ctrl+C sondern SIGINT

und beim Beenden der Konsole SIGHUB ...

ok: SIGKILL
ist KILL prozess beenden und zwar von aussen... also wenn ich

pkill Processname/Prog Name..

oder

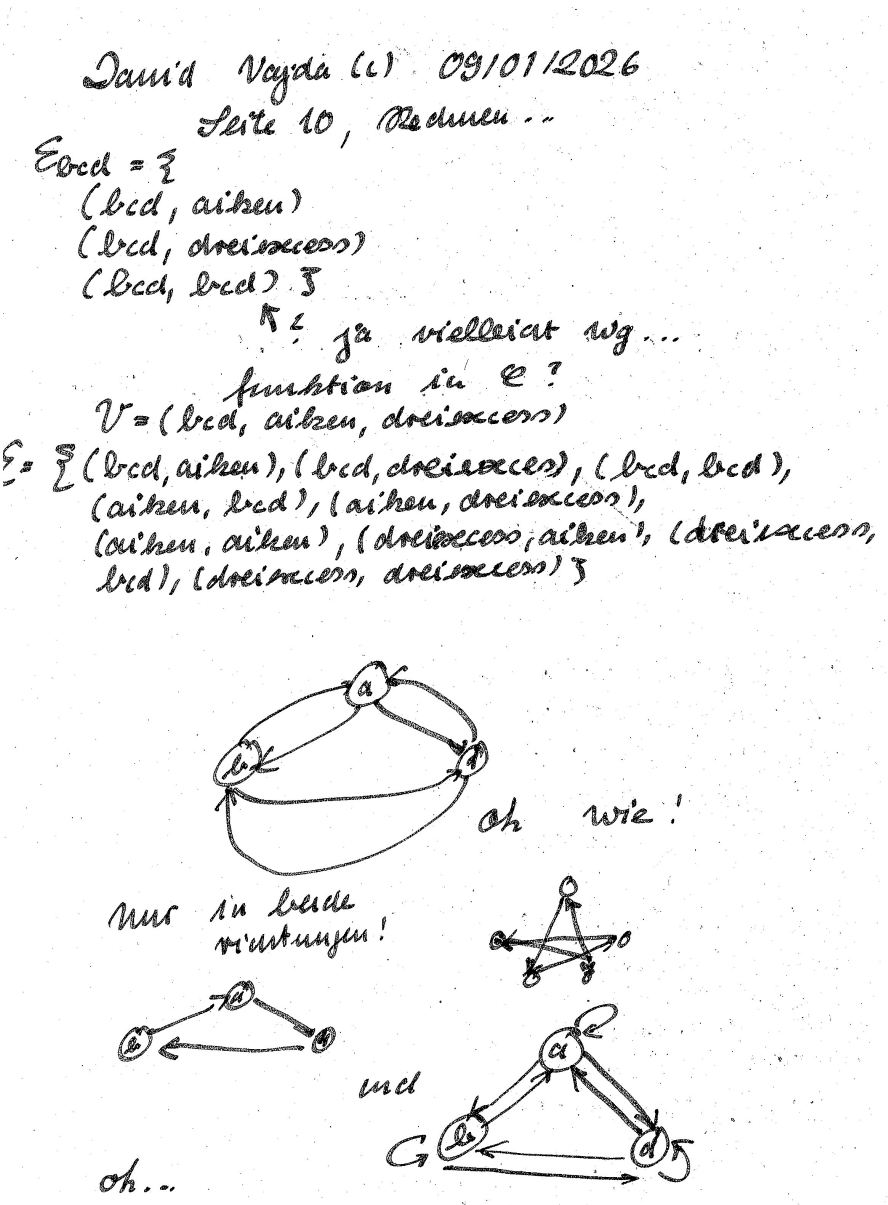
kill -KILL ProcessNr

Dann ist das

SIGKILL

Sighub - HangUp

10 hash table, codes, ...



David Vajda (c) 09/01/2026
Seite 8, Rechnen ...

1. bcd ziffer: 0b0010
2. bcd ziffer: 0b1000
3. bcd ziffer: 0b1001
4. bcd ziffer: 0b0101
5. bcd ziffer: 0b1001

oh

binär	bcd	arbeu_dreiexcess	
0b0000	0b0000(0)	0b0000 (0)	pt
0b0001	0b0001(1)	0b0001 (1)	pt
0b0010	0b0010(2)	0b0010 (2)	pt
0b0011	0b0011(3)	0b0011 (3)	0b0011 (0)
0b0100	0b0100(4)	0b0100 (4)	0b0100 (1)
0b0101	0b0101(5)	pt	0b0101 (2)
0b0110	0b0110(6)	pt	0b0110 (3)
0b0111	0b0111(7)	pt	0b0111 (4)
0b1000	0b1000(8)	pt	0b1000 (5)
0b1001	0b1001(9)	pt	0b1001 (6)
0b1010	pt	pt	0b1010 (7)
0b1011	pt	0b1011 (5)	0b1011 (8)
0b1100	pt	0b1100 (6)	0b1100 (9)
0b1101	pt	0b1101 (7)	pt
0b1110	pt	0b1110 (8)	pt
0b1111	pt	0b1111 (9)	pt

(c) Jamid Vojta 09/01/2026

Seite 3, Rechnen...

probe:

$bin(28959_{dec}) \overset{?}{=} 1110001000111111_b$

ok von links.

... .1111b

0111.0001.0001.1111b

28959 ist < 64 ok?

blop: ... ? 1... ← msb

probe!

$0111^{14}.0001^{12}.0001^{10}.1111_b$

15 13 11 9 8 7 6 5 4 3 2 1 0

$2^{10} = 1024$

... 2^{12}

$2^{12} = 4096$

$2^{13} = 8192$

$2^{14} = 8192 \cdot 2 =$

16384

16384
+ 8192
+ 4096
+ 256
+ 16
+ 8
+ 4
+ 1 3_1^2 ← ok

28959 !

...*

4+2=6

6+6=12

6+8=14

4+2+1=7 ✓

6+14=20

12+7=19 + 39 nicht gut

ok vorangehen ungerade ? ? ? aber das ist das erste ungerade in jedem fall also..*

8+9=17 ✓ 21

9+5=14 22+14=35

1+4=5, 3 ok

glück!!! ok!

(c) David Vajda 09/01/2026

Seite 5, Rechnen...

noch mal hinschreiben...

0b-0111.0001.0001.1111/1010b = ... 010010...

(c) a b g h i j k

...111

00110
- 0000
1100(c)
- 1010
00100
- 0000(f)
1001
- 0000 (g)
10011 (h)
- 1010
1001
- 0000
10010 (i)
- 1010
1000 (j)
10001
- 1010
001111(k)
- 1010
101
- 0000
...1

← 2 rückw. (Chaos! panique? panic)

← vom Eindruck zu schnell man schießt über das Ziel hinaus...

0b 01... von hinten!

0b 0001.0110.1001.0111 Rest

probe! 0b0101

0b dann ist stop!

← vielst gut aus 101 ist 5!

(c) Jamid Vayda
(08) 09/01/2026
Seite 2, Rechnen ...

28959
ich habe eine Lösung für das Problem gefunden:

echo $((RANDOM % (8192 - 4)))
↑ grep

28959 zunächst in eine Binärzahl umrechnen: 1110001000111111

28959/2 = 14479 Rest 1
14479/2 = 07239 Rest 1
7239/2 = 3619 Rest 1
3619/2 = 1809 Rest 1
1809/2 = 904 Rest 1
904/2 = 452 Rest 0
452/2 = 226 Rest 0
226/2 = 113 Rest 0
113/2 = 56 Rest 1
56/2 = 28 Rest 0
28/2 = 14 Rest 0
14/2 = 7 Rest 0
7/2 = 3 Rest 1
3/2 = 1 Rest 1
1/2 = 0 Rest 1

(c) David Vajda
(08/) 09/01/2026
Seite 1 Rechenübungen
+ Codes...

Was machen wir heute?
folgendes:

1.) aiken
2.) dreiexcess
3.) gray o. hamming?

... oh: wie auch immer...

Dazu kommt:

1.) Zufallszahlen
2.) fletchers Checksum
(wie adler-32)
3.) hash table...
4.) vor allem: crc...

oh: fangen wir an:

* dazu matrizenmultiplikation indem wir das senkrecht y Wort mit der $x_3\ x_2\ x_1\ x_0$ Matrix der input variablen multiplizieren...

oh:

wir fangen an: Eine Zufallszahl im pc:
echo $((RANDOM % MAX_INT))

(c) Daniel Vajda 09/01/2026

Seite 6, Rechnen...

$0b\,0001.0110.1001.0111$ Rest $0b\,0101$

$12 \quad 10$

$11 \; 9 \; 7 \; 6 \; 5 \; 4 \; 3 \; 2 \; 1 \; 0$

probe:

$$
\begin{array}{r}
4096 \\
+\,1024 \\
+\,512 \\
+\,128 \\
+\,16 \\
+\,4 \\
+\,2 \\
+\,1\,3\,1 \\
\hline
5783
\end{array}
$$

$6+4 = 10$

$2+8 = 10$

$6+4 = 10$

$2+1 = 3$

33

$9+1 = 10$

$2+2+1 = 5$

$\ldots + 3 = 18$

$5783 \cdot 10 + 5$

↳ Da existiert fehler!

weil:

28959 ich dachte 5 am Ende...

ok: projekt ablreden hier an der Stelle: nächstes ...

pause!

(c) Jamil Vayda 09/01/2026
Seite 4, Rechnen...

im nächsten Schritt teilen wir binär durch 10_{dec} (dec)? bin(10_{dec}) = 1010b

ohne ins - zu fallen * Ob 1010

Ob 0111.0001.0001.1111 / 1010b = 01011

- 1010 if... siehe * ← ausdruck?

(0000) falsch

0111

0111.0

- 1010

01000 (a)

- 0000 ← angst fehler hat sich eingeschlichen (klaube nicht) wird sich zeigen!

10000 (b)

- 1010

001101 (c)

- 1010

0011 ... ok nächste Seite ↙ Zeilen vorschub <cr><lf>?

* soll kreisen was heißt Seiten vorschub?

if ((0111 - 1010) ≥ 0) in Latex: vfill pagebreak

then ...

else

return 0111

pagebreak u. carriage return ↙ = parse error

? klingt eingehend!

(c) Daniel Vejda 09/01/2026

Seite 7, Rechnen...

als nächstes rechnen wir die Zahl trotzdem als Dezimal Zahl einfach in bcd...

und in die anderen Codes...

$bcd(28959_{dec})$ = 0b0010.1000.1001.0101.1001

↑ fehlt!

0b0000.0000.0000.0010.1000.1001.0101.1001

gepackt: 5 bcd ziffern in einem:

32 bit doppelwort!

ungepackt wäre es:

3 x 0b0000 bei 5 heißt 2 byte extra notwendig

0b0000.0010
0b0000.1000
0b0000.1001
0b0000.0101
0b0000.1001

Das sind 5 byte davon 4 byte!

ah: noch:

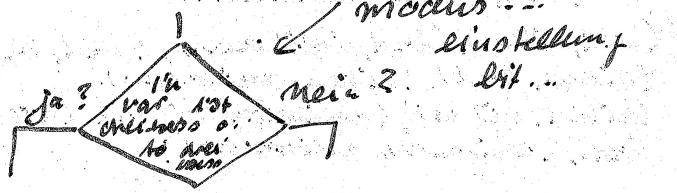
zahlen + dreierrest!

Jamid Vajda (c) 09/01/2026
Seite 9, Rechnen...

ok:

1.) 0b0010 ist ein aiken/
dreiexcess... code

2.) 0b 0010 in bcd in aiken/
dreiexcess ... code ausdrücken



aiken(0bcd 0010) =
aiken(0bcd 1000) =
aiken(0bcd 1001) =
aiken(0bcd 0101) =
aiken(0bcd 1001) =

dreiexcess(0aik 0010) =
dreiex(0aik 1000) =
dreiex(1001 aik) =
dreiex(0101 aik) =
dreiex(0b 1001 aik) =

dreiexcess(0bcd 0010) =

möglichkeiten
paare im graph!
?!? ok

Damit Vajda (c)
09/01/2026, Seite 11, Rechnen -

dann am besten 3 tabellen!

bcd Ausgangsmenge -
← hat als int. spez. zu werden..

n	bcd	aiken	excess
1	0bcd 0010 (2)	$0b\,0010_{aiken}$ (2)	pt
2	0bcd 1000 (8)	pt	$0b\,1000_{exc}$ (5)
3	0bcd 1001 (9)	pt	$0b\,1001_{exc}$ (6)
4	0bcd 0101 (5)	pt	$0b\,0101_{exc}$ (2)
5	0bcd 1001 (9)	pt	$0b\,1001_{exc}$ (6)

↑ wenn mehr fehlen..

n	aiken	bcd	excess
1	$0b\,0010_{aiken}$: pt	x	a
2	$0b\,1000_{aik}$: 5	0bcd 0101	$0b\,1011_{exc}$
3	$0b\,1001_{aik}$: 6	0bcd 0110	0b 10
4	↖ Lücke!	falsch!!:	
5			

ok abbrechen: beim
nächsten mal wieder?
nein: aber ect fletchers checksum!
brauch und what we are
dec + bcd

David Vajda (c) 09/01/2026
Seite 12, Rechnen--

Später:

- Matrizenmul!
- Ableiten, streichen, bzw führt zu verschiedenen matrizen + ausdrücken? (innerhalb?)
- crc! nicht vergessen
- Integral
- Logarithmus
  und!

Was mir fehlt ist:

fft: fast fourier transformation?
vielleicht!

lieber: polynom multiplikation!
gleich!

Daniel Vajda (c) 09/01/2026

Seite 13, Rechnen...

1.) fletcher's checksum auf:

$\mathcal{X} = (2, 8, 9, 5, 9)$

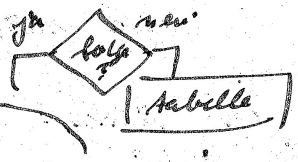
ergänzen mit Flecken!

```
sum1 := 0
sum2 := 0
for i := 0 to length(X) do
        sum1 := (sum1 + X[i]) modulo 7
        sum2 := (sum2 + sum1) modulo 7
endfor
checksum := (sum1, sum2)
```

als Folge ausdrücken?

1 Vorschrift

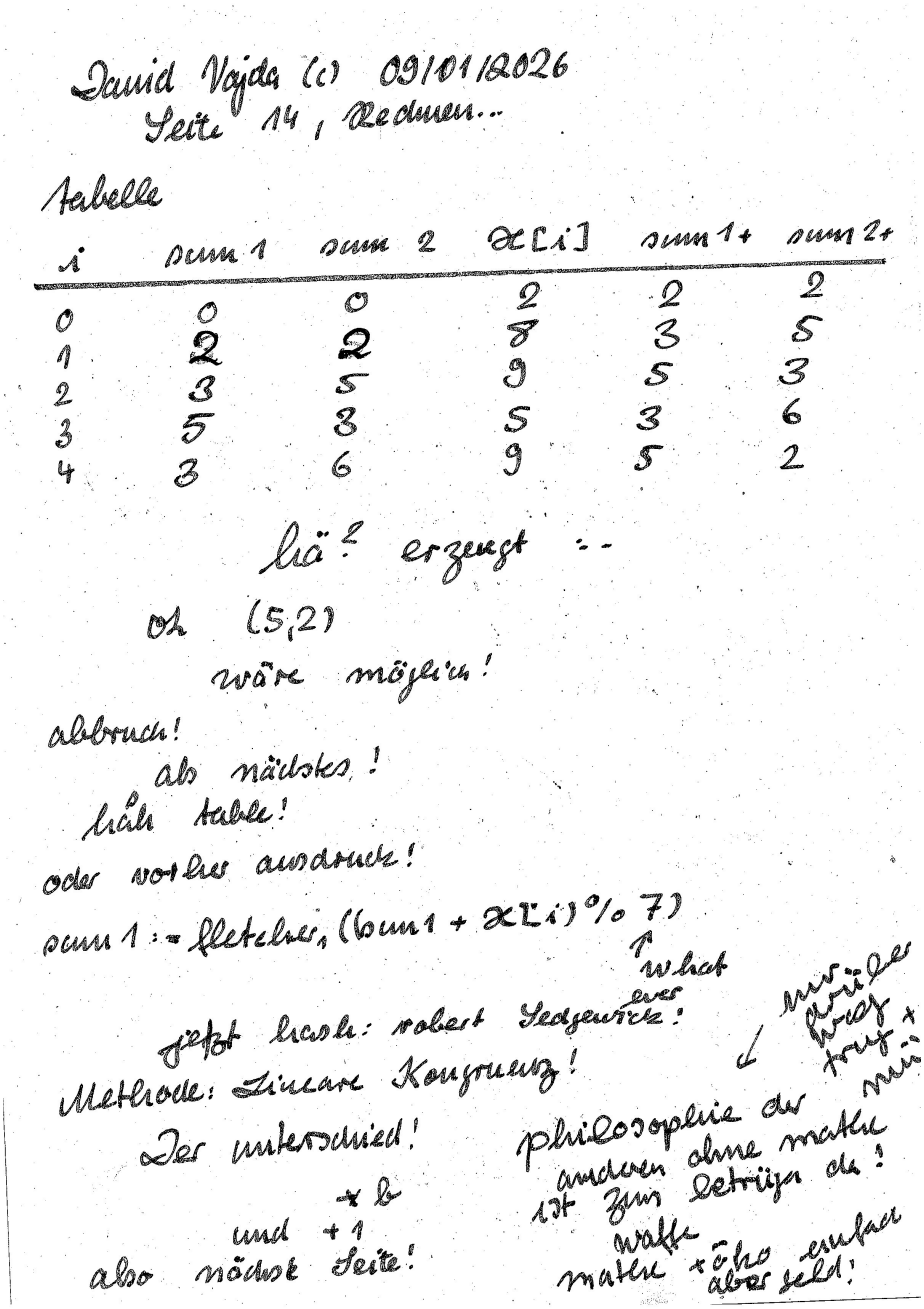
-später...



Tabelle

i	sum1	sum2	sum1+	sum2+	X[i]
0	0	0	2	2	2
1	2	2	(8+2) mit 7 = 3	5	8
2	3	5	0	5	9
3	0	5	5	3	5
4	5	3	0		

Verwendung Spalten oder Zeilen organisiert!



im nächsten schritt folgt die hash-table, die zufallszahlen, die fletchers checksum mit latex

```
X = (2,8,9,5,9)
```

```
fletcher's checksum
i sum1 sum2 X [i] sum1+ sum2+
0 0  0  2 2  2
1 2  2  8 3  5
2 3  5  9 5  3
3 5  3  5 3  6
4 3  6  9 5  4

sum1_(i+1) := fl1 (sum1_i + X [i]) % 7
sum2_(i+1) := fl2 (sum1_i + sum2) % 7

eigentlich eher:  sum1_(i) := fl1 (sum1_(i-1) + X [i]) % 7
ja!!! nein!, weil negatives i-1, fuer i=0!
oder mathematik folge faengt mit 1 an zu zaehlen, waehrend
sonst 0. ???
halt wiederlegt. y_(i+1) ist fuer i = 0, 1. damit stimmt die sache!

ernsthaft ist die aussage:
sum1_(i+1) := fl1 (sum1_i + X [i]) % 7
sum2_(i+1) := fl2 (sum1_i + sum2) % 7
y_(i+1) = hash (12*y_(i)+X[i]) % 7
weil: i = 0 ist i+1 = 1 und damit fangen in mathe folgen an, es gibt keinen
negativen index

(5,4)

moeglich:


 i X [i] sum1 sum2 sum1+ sum2+
0 2 0  0  2  2
1 8 2  2  3  5
2 9 3  5  5  3
3 5 5  3  3  6
4 9 3  6  5  4

hash-table

y_i = hash (12*y_(i-1)+X[i]) % 7
y_0 = 0

oder?? auch zu oben!!!

y_(i+1) = hash (12*y_(i)+X[i]) % 7
y = 0
```

```
i X [i] y_(i) y_(i+1)
0 2 0 2
1 8 2 (12*2 + 8) % 7 = (24+8) % 7= 32 % 7 = 4
2 9 4 (12*4 + 9) % 7 = (48+9) % 7 = 1
3 5 1 (12*1 + 5) % 7 = 17%7 = 3
4 9 2 (12*3 + 9) % 7 = (36+9) % 7 = (45 % 7) = 3

 ok:

 i X [i] y_(i) y_(i+1)
0 2 0 2
1 8 2 4
2 9 4 1
3 5 1 3
4 9 2 3

 random (lineare Kongruenz)

 a [i] = (a [i-1] * b + 1) % m

eine idee, die ich jetzt haette, waere b entsprechend der folge
X = (2,8,9,5,9) zu waehlen.
einfach als beispiel noch.
b = 12
m = 7
seed = a [0] = 2

i a [i]
0 2
1 (12*2+1) % 7 = 25%7 = (21+4)%7 = 4
2 (12*4+1) % 7 = 49%7 = 0
3 (12*0+1) % 7 = 1
4 (12*1+1) % 7 = (13 % 7) = 5
5 (12*5+1) % 7 = (60+1) % 7 = (56 + 5) % 7 = 5

 zwei forderungen:
 1.) nicht zwei zahlen duerfen zur selben zufallszahl fuehren
 2.) m und b muessen so gewaehlt sein, dass sie bei der zufallzsahl nicht auf dieselbe abbil

 kann man die gleichung loesen, ist die frage:

 y = (12*y+1) % 7

 ok, aber das spaeter.

 nun, erst in LaTeX Tabellen und dann: CRC Zyklische Redundanzpruefung.
```

i	X_i	$sum1$	$sum2$	$sum1+$	$sum2+$
0	2	0	0	2	2
1	8	2	2	3	5
2	9	3	5	5	3
3	5	5	3	3	6
4	9	3	6	5	4

ok: CRC-Check

```
0b0111.0001.0001.1111b

CRC mit 0b1010 und vorher mit... 11001

Rahmen  0b0111.0001.0001.1111
Pruefwert  0b1010
   ---------
      1101
      11010001

was habe ich vergessen?
1.) es wird ein pruefwert bestimmt
2.) die laenge - oft wohl 5 bit (also n bit)
3.) der Rahmen (nicht wie beim Paging) wird um n-1 0en bit geclearte bit erweitert.

Pruefwert: 1101b
Rahmen: 0b0111.0001.0001.1111.000

1101b

polynomdivision: ok
polynommultiplikation wollten wir machen, mit...
dann kommt langrage ... das brauchen wir ja bei FFT...


Rahmen:    0b011.1000.1000.1111.1000
Pruefwert:   0b101.1
Division:     / (bzw. XOR)
Zwischenergebnis: 0b110.0 ...
die frage ist, soll ich das hintere erhalten...

jetzt geht es tatsaechlich um: relevante stellen!

das wort wurde bisher nicht geklaert!
relevante stellen, naemlich die 1 im msb - most significant bit.

ok:
```

```
1.) frage: zwischenergebnis
2.) frage: von anfang relevante stellen

a) theoretisch: entspricht das der division die ich bisher binaer ausgefuehrt habe
b) theoretisch: ist die subtraktion dabei immer ein XOR
c) theoretisch: die relevanz von stellen.


Rahmen:    0b011.1000.1000.1111.1000
Pruefwert:   0b101.1
Division:     / (bzw. XOR)
Zwischenergebnis: 0b110.0 ...
Zwischenergebnis: 0b110.0000.1000.1111.1000

ok: wir brechen das ab und nehmen gleich von anfang an relevante stellen:

Rahmen:   0b11.1000.1000.1111.1000
Pruefwert:  0b1011

ok. wir verschieben den punkt

Rahmen:    0b1110.0010.0011.1110.00
Pruefwert:   0b1011
Division:   / (XOR)
Zwischenergebnis: 0b0110.0010.0011.1110.00

relevante stellen:

Rahmen:   0b1100.0100.0111.1100.0
Pruefwert:   0b1011
Divsion:   / (XOR)
Zwischenergebnis: 0b0111.0100.0111.1100.0

relevante stellen:   0b1110.1000.1111.1000

Rahmen:   0b1110.1000.1111.1000
Pruefwert:   0b1011
Division:   / (XOR)
Zwischenergebnis: 0b0101.1000.1111.1000

relevante stellen: 0b101.1000.1111.1000
Rahmen:   0b1011.0001.1111.000
Pruefwert:   0b1011
Division   / (XOR)
Zwischenergebnis: 0b0000.0001.1111.000
```

```
relevante stellen: 0b1111.1000
Rahmen:   0b1111.1000
Pruefwert:   0b1011
Divsion:   / (XOR)
Zwischenergebnis: 0b0100.1000

Relevante stellen:
Rahmen:   0b1001.000
Pruefwert:   0b1011
Division:   / (XOR)
Zwischenergebnis: 0b0010.000

ok, ich muss jetzt abbrechen, es waere moeglich da sind fehler drin
```

jetzt machen wir langrage!

also, wir haben die punkte: y-Werte.

$$Y = (2, 8, 9, 5, 9)$$

und das auf den x-Stellen, Abszissen-Werten.

$$X = (0, 1, 2, 3, 4)$$

und nun kommt langrage.
also:

1. ordinatenwerte: $Y = (2, 8, 9, 5, 9)$
2. abszissenwerte: $X = (0, 1, 2, 3, 4)$

langrage:

$$l_i(x) = \prod_{\substack{i=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

ich stelle wie ich das gesetzt habe, in LaTeXvor

```
$$l_i(x) = \displaystyle\prod_{\stackrel{i=0}{j\neq i}}^n\dfrac{x-x_j}{x_i-x_j}$$
```

$$l_i(x) = \prod_{\substack{i=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} = \frac{x - x_0}{x_i - x_0} \cdots \frac{x - x_{i-1}}{x_i - x_{i-1}} \cdot \frac{x - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{x - x_n}{x_i - x_n}$$

so, die formel die Formel in LaTeX

```
$$l_i(x) = \displaystyle\prod_{\stackrel{i=0}{j\neq i}}^n\dfrac{x-x_j}{x_i-x_j} = \dfrac{x-x
```

nun die formel, fuer $l_i(x)$ jeweils.
damit sind die absizssenwerte nicht optimal, aber so:
$X = (1, 2, 3, 4, 5)$

```
i x[i] x[j=0] x[j=1] x[j=2] x[j=3] x[j=4] (x-x[j=0])/(x[0]-x[j=0])  (x-x[j=1])/(x[0]-x[j=1])
0 2 x  8  9  5  9  x     (x-8)/(2-8)
1 8 2  x  9  5  9
2 9 2  8  x  5  9
3 5 2  8  9  x  9
4 9 2  8  9  5  x
```

ich mache pause

ok, ich probiere jetzt atmega8 programme inzwischen zu schreiben

```
;; (c) david vajda
;; 09/01/2026
;; m8/led count/exersize

.include "m8def.inc"

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRB, r16

ldi r16, 0xff
l1:
out PORTB, r16
rcall r1
dec r16
rjmp l1

r1:
push r16
push r17
push r18
ldi r16, 0xff
r1l1:
ldi r17, 0xff
r1l2:
```

```
ldi r18, 0x04
r1l3:
dec r18
brne r1l3
dec r17
brne r1l2
dec r16
brne rl1
pop r18
pop r17
pop r16
ret
```

ok, das dürfte funktionieren

```
;; (c) david vajda
;; 09/01/2026
;; m8/INT0/INT1

.include "m8def.inc"

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, (1 << INT0) | (1 << INT1)
out MCUCR, r16
ldi r16,
out GICR, r16
```

so, ich fasse die Beiträge zusammen, ich mache noch langrage vielleicht interpolation von Polynomen???

ich brauche dafuer eine richtige tabelle, vielleicht, fuer die einzelne, wie muss die aussehen auf papier???

spaeter ein wenig pause, trotzdem gut!

```
i x[i] x[j=0] x[j=1] x[j=2] x[j=3] x[j=4] (x-x[j=0])/(x[0]-x[j=0])  (x-x[j=1])/(x[0]-x[j=1])
0 2 x  8  9  5  9  x     (x-8)/(2-8)
1 8 2  x  9  5  9
2 9 2  8  x  5  9
3 5 2  8  9  x  9
4 9 2  8  9  5  x
```

dass diese tabelle drauf passt? nur wie?

etwa eine 3-D so zu sagen in zwei D. ok, machen wir mal, ist ja nicht ungewoehnlich

```
mit Robert Sedgewick oder Wikipedia.

vielleicht diese version ohne vollstaendiges ausfuellen.
i x[i] j x[j] (x-x[j])/(x[i]-x[j])
0 2 0 2 ---
  1 8 (x-8)/(2-8)
  2 9 (x-9)/(2-9)
  3 5 (x-5)/(2-5)
  4 9 (x-9)/(2-9)
1 8 0 2 (x-2)/(8-2)
  1 8 ---
  2 9 (x-9)/(8-9)
  3 5 (x-5)/(8-5)
  4 9 (x-9)/(8-9)
2 9 0 2 (x-2)/(9)
  1 8 (x-8)/()
  2 9 ---
  3 5 (x-5)/
  4 9 (x-9)/
3 5 0 2
  1 8
  2 9
  3 5 x
  4 9
4 9 0 2
  1 8
  2 9
  3 5
  4 9 x

ok: hier stellt sich was interessantes heraus:

wenn: bijektivitaet: surjektivitaet, injektivitaet...
also:

achso!

ich weiss, was der fehler war! ich habe die falschen werte genommen
ich habe y genommen nicht x. das ist das problem.
y und nicht x!


i x[i] j x[j] (x-x[j])/(x[i]-x[j])
0 0 0 0 ---
  1 1 (x-1)/(0-1)
  2 2 (x-2)/(0-2)
```

```
  3 3 (x-3)/(0-3)
  4 4 (x-4)/(0-4)
1 1 0 0 (x-0)/(1-0)
  1 1 ---
  2 2 (x-2)/(1-2)
  3 3 (x-3)/(1-3)
  4 4 (x-4)/(1-4)
2 2 0 0 (x-0)/(2-0)
  1 1 (x-1)/(2-1)
  2 2 ---
  3 3 (x-3)/(2-3)
  4 4 (x-4)/(2-4)
3 3 0 0 (x-0)/(3-0)
  1 1 (x-1)/(3-1)
  2 2 (x-2)/(3-2)
  3 3 ---
  4 4 (x-4)/(3-4)
4 4 0 0 (x-0)/(4-0)
  1 1 (x-1)/(4-1)
  2 2 (x-2)/(4-2)
  3 3 (x-3)/(4-3)
  4 4 ---
```

ok. das wäre besser, als schöne tabelle:

i	x_i	j	x_j	$\frac{x - x_j}{x_i - x_j}$
0	0	0	0	EDIVBYZERO
		1	1	$\frac{x-1}{0-1}$
		2	2	$\frac{x-2}{0-2}$
		3	3	$\frac{x-3}{0-3}$
		4	4	$\frac{x-4}{0-4}$
1	1	0	0	$\frac{x-0}{1-0}$
		1	1	EDIVBYZERO
		2	2	$\frac{x-2}{1-2}$
		3	3	$\frac{x-3}{1-3}$
		4	4	$\frac{x-4}{1-4}$
2	2	0	0	$\frac{x-0}{2-0}$
		1	1	$\frac{x-1}{2-1}$
		2	2	EDIVBYZERO
		3	3	$\frac{x-3}{2-3}$
		4	4	$\frac{x-4}{2-4}$
3	3	0	0	$\frac{x-0}{3-0}$
		1	1	$\frac{x-1}{3-1}$
		2	2	$\frac{x-2}{3-2}$
		3	3	EDIVBYZERO
		4	4	$\frac{x-4}{3-4}$

4	4	0	0	$\frac{x-0}{4-0}$	
		1	1	$\frac{x-1}{4-1}$	
		2	2	$\frac{x-2}{4-2}$	im nächsten schritt rechnet man die aus,
		3	3	$\frac{x-3}{4-3}$	
		4	4	EDIVBYZERO	

und dann addiert man sie. einzeln. und dann kommt das einsetzen, danach das integral.

```
(x-1)/(-1)*(x-2)/(-2)*(x-3)/(-3)*(x-4)/(-4) = ((x-1)*(x-2)*(x-3)*(x-4))/((-1)*(-2)*(-3)*(-4)
((x-1)*(x-2)*(x-3)*(x-4))/((-1)*(-2)*(-3)*(-4)) = [(x-1)*(x-2)]*[(x-3)*(x-4)]/(24) =
([x^2-2*x-x+2]*[x^2-3*x-4*x+12])/24 =
(x^2-3*x+2)*(x^2-7*x+12)/24 =

ok, was wir jetzt brauchen ist polynommultiplikation.
wir haben polynomdivision und die multiplikation

ein ansatz waere - also, natuerlich ist es an und fuer sich einfach, aber wir brauchen ein
verfahren.

und zwar koennten wir eine tabelle nehmen.
wir nehmen eine tabelle und nehmen bei beiden die potenzen, die koeffizienten...

eine quadratische? probieren wir es aus:
  b_2 b_1 b_0
a_2
a_1
a_0

oder wir druecken es aus, als:
  a_{22} a_{21} a_{20}
a_{12}
a_(11}
a_{10}

dabei entspricht das den potenzen
(x^2-3*x+2)*(x^2-7*x+12)

  (2,1)  (1,-3) (0,2)
(2,1)
```

```
(1,-7)
(0

ok, noch mal:

  a_{22} a_{21} a_{20}
a_{12}
a_(11}
a_{10}

vielleicht als paare, weil: wir muessen ja auch den exponenten multiplizieren

dabei entspricht das den potenzen
(x^2-3*x+2)*(x^2-7*x+12)

  (2,1)  (1,-3) (0,2)
(2,1)  (4,1)  (3,-3) (2,1)
(1,-7) (3,-7) (2,21) (1,-7)
(0,12) (2,12) (1,-36) (0,24)

ok und jetzt alle paare addieren, die gleiches (a,...) oder (a_1,...) haben

  (2,1)  (1,-3) (0,2)
(2,1)  (4,1)  (3,-3) (2,1)
(1,-7) (3,-7) (2,21) (1,-7)
(0,12) (2,12) (1,-36) (0,24)

(4,1)
(3,-3), (3,-7)
(2,1), (2,21,) (2,12)
(1,-7), (1,-36)
(0,24)

x^4 -10 * x^3 + 34 *x^2 + -43*x+24

waere moeglich, womoeglich fehler, also noch mal mathematisch.
```

i	x_i	j	x_j	$\frac{x - x_j}{x_i - x_j}$
0	0	0	0	EDIVBYZERO
		1	1	$\frac{x-1}{0-1}$
		2	2	$\frac{x-2}{0-2}$
		3	3	$\frac{x-3}{0-3}$
		4	4	$\frac{x-4}{0-4}$
1	1	0	0	$\frac{x-0}{1-0}$
		1	1	EDIVBYZERO
		2	2	$\frac{x-2}{1-2}$
		3	3	$\frac{x-3}{1-3}$
		4	4	$\frac{x-4}{1-4}$
2	2	0	0	$\frac{x-0}{2-0}$
		1	1	$\frac{x-1}{2-1}$
		2	2	EDIVBYZERO
		3	3	$\frac{x-3}{2-3}$
		4	4	$\frac{x-4}{2-4}$
3	3	0	0	$\frac{x-0}{3-0}$
		1	1	$\frac{x-1}{3-1}$
		2	2	$\frac{x-2}{3-2}$
		3	3	EDIVBYZERO
		4	4	$\frac{x-4}{3-4}$

4	4	0	0	$\frac{x-0}{4-0}$
		1	1	$\frac{x-1}{4-1}$
		2	2	$\frac{x-2}{4-2}$
		3	3	$\frac{x-3}{4-3}$
		4	4	EDIVBYZERO

	(2,1)	(1,-3)	(0,2)
(2,1)	(4,1)	(3,-3)	(2,1)
(1,-7)	(3,-7)	(2,21)	(1,-7)
(0,12)	(2,12)	(1,-36)	(0,24)

$$f(x) = x^4 - 10 \cdot x^3 + 34 \cdot x^2 + -43 \cdot x + 24$$

$(x-1)/(-1)\cdot(x-2)/(-2)\cdot(x-3)/(-3)\cdot(x-4)/(-4) = ((x-1)\cdot(x-2)\cdot(x-3)\cdot(x-4))/((-1)\cdot(-2)\cdot(-3)\cdot(-4)) = ((x$