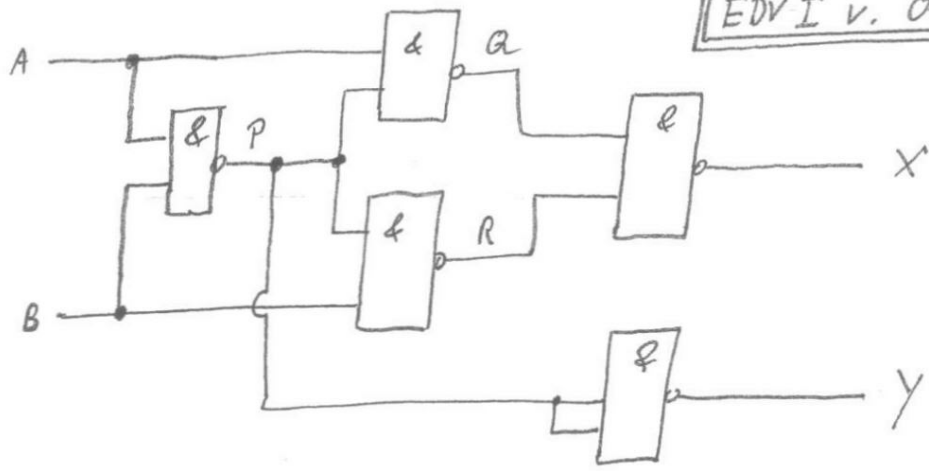
$a \cdot (b+c) = a \cdot b + a \cdot c$] $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$

EDV I v. 04.02.03

(A1)



14
6
14
10
14
14
10
82

$P = \overline{A \wedge B} = \bar{A} \vee \bar{B}$

$Q = \overline{A \wedge P} = \bar{A} \vee \bar{P}$

$= \bar{A} \vee (A \wedge B) = (\bar{A} \vee A) \wedge (\bar{A} \vee B) = \bar{A} \vee B$

$R = \overline{B \wedge P} = \bar{B} \vee \bar{P}$

$= \bar{B} \vee (A \wedge B) = (\bar{B} \vee A) \wedge (\bar{B} \vee B) = A \vee \bar{B}$

$X = \overline{Q \wedge R} = \bar{Q} \vee \bar{R}$

$= \overline{\bar{A} \vee B} \vee \overline{A \vee \bar{B}}$

$= (A \wedge \bar{B}) \vee (\bar{A} \wedge B) \Rightarrow$ Antivalenz

$Y = \overline{P \wedge P} = \bar{P} \vee \bar{P} = \bar{P}$

$= \overline{\bar{A} \vee \bar{B}}$

$= A \wedge B =$ AND

8

4

B	A	$\bar{B}$	$\bar{A}$	$A \wedge \bar{B}$	$\bar{A} \wedge B$	$(A \wedge \bar{B}) \vee (\bar{A} \wedge B)$	$A \wedge B$
0	0	1	1	0	0	0	0
0	1	1	0	1	0	1	0
1	0	0	1	0	1	1	0
1	1	0	0	0	0	0	1
						Antivalenz	UND

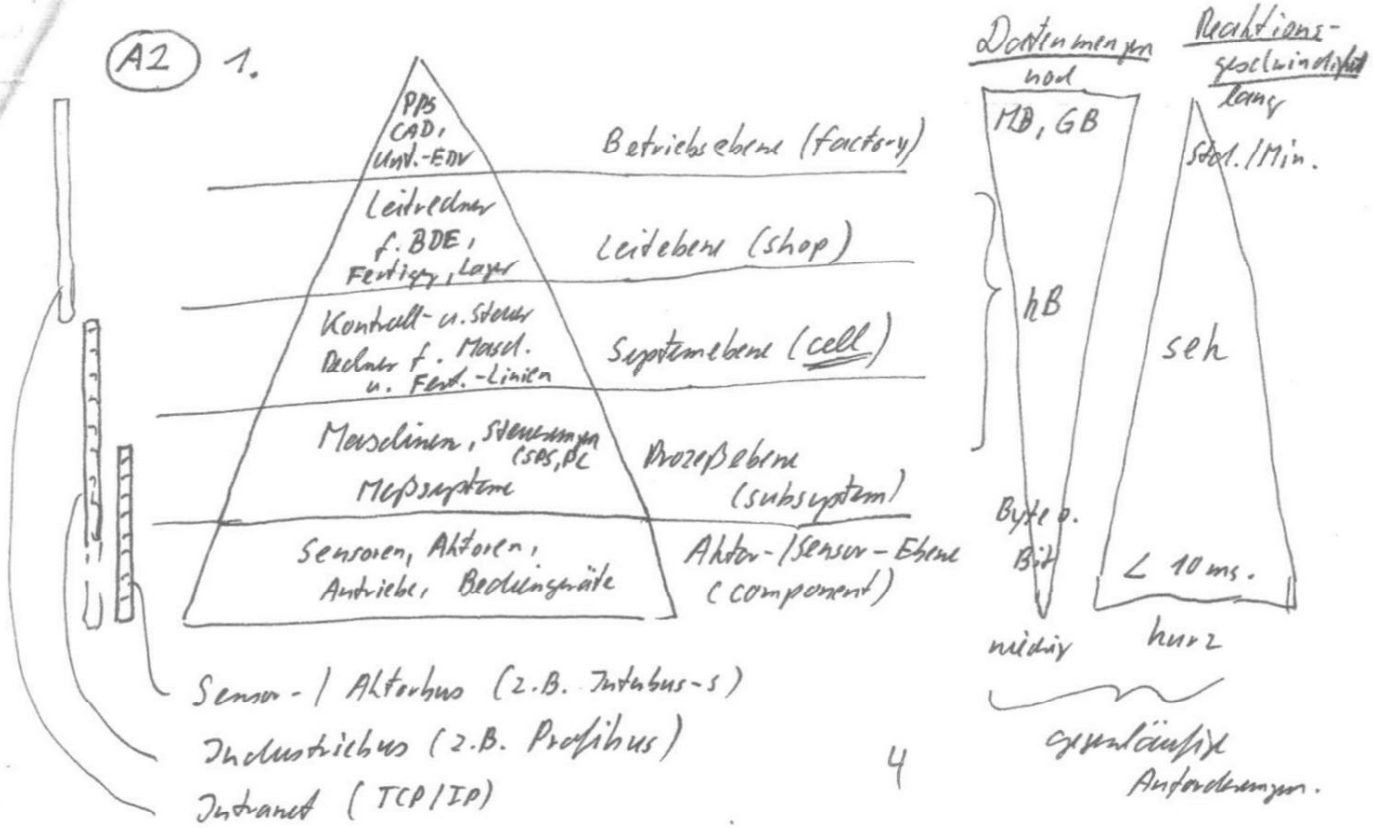
=> Halbaddierer
Z = X (Ziffer)
Ü = Y (Übertrag)

2

Es wird nur 1 Gattertyp verwendet, da dies die technologisch günstigste Fertigungsmöglichkeit bietet.

14

A2 1.



4

2. $Z = X \wedge (\bar{X} \vee S')$

$= (X \wedge \bar{X}) \vee (X \wedge S')$

$= 0 \vee (X \wedge S')$

$= \underline{\underline{X \wedge S'}}$ 2

6

(A3) 1. $n = 1$ stellig $\quad \left.\begin{matrix}0\\1\end{matrix}\right\}$ $J = ld\frac{1}{p}$, $p = 50\% \Rightarrow$ Wahrsch. ↓

$J = ld\,2 = 3{,}322\,lg\,2 = 3{,}322 \cdot 0{,}30103$

$= 1$ bit

$n = 2$ stellig $\quad \left.\begin{matrix}00\\01\\10\\11\end{matrix}\right\}$ $p = 25\%$

$J = ld\,4 = 3{,}322 \cdot lg\,4 = 3{,}322 \cdot 0{,}60206$

$= 2$ bit

$n = 3$ stellig $\quad \left\{\begin{matrix}000\\001\\010\\011\\100\\101\\110\\111\end{matrix}\right.$ $p = 12{,}5\%$

$J = ld\,8 = 3{,}322 \cdot lg\,8 = 3{,}322 \cdot 0{,}90309$

$= 3$ bit

Die einfache Berechnung ergibt sich aus dem Ansatz, den Informationsgehalt als ~~Kehrwert~~ Zweierlogarithmus des Kehrwerts der Wahrscheinlichkeit des Auftretens einer Bitkombination zu definieren.

$\Rightarrow P_x = \frac{1}{2^n} \rightarrow J = ld\frac{1}{\frac{1}{2^n}} = ld\,2^n = n$ (nur bei Gleichverteilung).

2. 6 Kärtchen lassen 6 Fragen, d.h. 6 J/N-Antworten, zu. $\Rightarrow$ 6 bit

Damit können 64 unterschiedliche Werte ($2^6 = 64$) bestimmt werden.

6 Bit: $\quad \sqcup^{2^5}\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup^{2^0}$

a) Die Karte mit "Fett 1" enthält alle Zahlen (Dual-) mit dem niederwertigsten Bit (2^0) = 1

b) Die Karte mit "Fett 2" enthält alle Dualzahlen mit $2^1 = 1$, z.B. $0+2$, $0+2+1$, $0+4+2$, $0+4+2+1$, usw.

⋮

f) Die Karte mit "Fett 32" enthält alle Dualzahlen mit dem höchstwertigsten Bit (2^5) = 1, also 32 .... 63.

Die Zahl 43 ist z.B. zu finden auf den Karten

$32 = 1 \cdot 2^5$
$8 = 1 \cdot 2^3$
$2 = 1 \cdot 2^1$
$1 = 1 \cdot 2^0$

$101011|_2 = 43|_{10}$

↑ Es müssen also die fettgedruckten Zahlen addiert werden (auf denen die 43 zu finden ist).

9

14

(A4)

```
1
2
3
4
5   #include<iostream.h>
6   #include<iomanip.h>
7
8   int main()
9   {
10  int cm, ft, zeile=0;
11  float restcm, in;
12
13  cout.setf(ios::fixed | ios::right);
14  cout.precision(1);
15  cout << "\n";
16
17  for(cm=160; cm<=198; cm=cm+2)
18  {
19     ft     = cm/30.48;
20     restcm = cm - ft*30.48;
21     in     = restcm/2.54;
22     cout  << setw(5) << cm << " cm   = "
23           << setw(4) << ft << " ft. "
24           << setw(4) << in << " in. " << "\n";
25     zeile++; if(zeile==5) {cout << "\n"; zeile=0;};
26  }
27  return (0);
28  }
```

2

2

10

alt:

```
int ft, cm;
float in;
  :
  ft = cm/30.48;
  in = cm/2.54 - (12*ft);
  :
```

A5

```
1   // Boolsche Variablen
2   // Wahrheitstabelle Volladdierer
3
4
5   #include <iostream.h>
6   #include <iomanip.h>
7   int main()
8   {
9   bool A, B, C, X, UE;
10  cout << "\n"
11       << "  C  A  B   |  UE   X\n"
12       << " ---------------------\n";
13
14  for(C=0; C<=1; C++)
15  if(C==0)
16  {
17     for(B=0; B<=1; B++)
18     {
19        for(A=0; A<=1; A++)
20        {
21           X  = (A&&!B)||(!A&&B);
22           UE = A&&B;
23           cout << "  " << C
24                << "  " << A
25                << "  " << B
26                << "   |   " << UE
27                << "   " << X
28                << "\n";
29        }
30     }
31  }
32  else
33  {
34  cout << " ---------------------\n";
35     for(B=0; B<=1; B++)
36     {
37        for(A=0; A<=1; A++)
38        {
39           X  = (A&&B)||(!A&&!B);
40           UE = A||B;
41           cout << "  " << C
42                << "  " << A
43                << "  " << B
44                << "   |   " << UE
45                << "   " << X
46                << "\n";
47        }
48     }
49  }
50  return (0);
51  }
52
53
54
55
```

14

A6

```
1
2
3
4   // DGL 2. O.; Determinanten-Lösung
5
6   #include<iostream.h>
7
8   // Klassendeklaration ***************************
9   class det2o   // Determinante 2. Ordnung
10  {
11  public:
12  int   b01, a11, a12,
13        b02, a21, a22;
14  float D1,  D2,  D0,  x, y;
15  berechnen();
16  };
17
18
19  // Methodendefinition (Elementfunktion) **********
20  det2o::berechnen()
21  {
22  D1 = b01*a22 - b02*a12;
23  D2 = b02*a11 - b01*a21;
24  D0 = a11*a22 - a21*a12;
25  if(D0==0) goto ende;
26  y = D1/D0;
27  x = D2/D0;
28  ende:
29  return(0);              // sollte vorhanden sein!!!
30  }
31
32
33  // Hauptprogramm ********************************
34  int main()
35  {
36  det2o beispiel;
37
38  beispiel.b01=4, beispiel.a11=1, beispiel.a12=-3,
39  beispiel.b02=1, beispiel.a21=1, beispiel.a22=-2;
40
41  beispiel.berechnen();
42
43  if(beispiel.D0==0) {cout << "Keine Lösung!"; goto ende;};
44  cout << "x = " << beispiel.x << "\n"
45       << "y = " << beispiel.y << "\n";
46  ende:
47  return(0);
48  }
49
50  // Falls beispiel.a12=-2, keine Lösung, da Geraden parallel
51  /* Geraden sind: y = 3x + 4
52                   y = 2x + 1      Lsg.:  x = -3,  y = -5  */
```

14

(A7) 1. Eine ALU besitzt mind. folgende Eigenschaften:

- erweitertes 4-Bit-Addier-Subtrahier-Werk
- UND-Verknüpfung von zwei 4-Bit Worten
- ODER- " " " " "
- XOR- " " " " "

2

Damit liegen alle arithmetischen und logischen Grundfunktionen vor

2. Die vorliegende Schaltung stellt einen sog. Akkumulator dar.

Es ist ein Register der CPU, das im Hinblick auf die Ausführung von Befehlen optimiert ist.
Früher war es oft das einzige Register, das als Ziel eines Befehls, also für die Bereitstellung des Operationsergebnisses, benutzt werden kann.
Heute stehen viele Register in den µPs zur Verfügung. Jedoch sind einige nur für den Akku geschwindigkeitsoptimiert.

Akkumulator = Daten addieren ("akkumulieren").

Heute z.B. EAX-Register (32 Bit breit) für
- Multiplikation
- Division
- i/o
- schnelle Verschiebung

Die Dateneingabe erfolgt nur über die B-Eingänge.
Die Bearbeitung ist taktgesteuert.

- Ein Wort B gelangt über die ALU ins Register, wo es gespeichert wird.
- Nach einem Takt steht es als A=Z am Eingang A der ALU z. Verfüg.
- Wird jetzt das nächste Wort auf B gegeben, kann über den Befehlseingang $U_3 \dots U_0$ eine Operation in der ALU durchgeführt werden.
- Mit dem nächsten Takt übernimmt das Register das Rechenergebnis, wobei ein eventueller Übertrag ebenfalls taktgesteuert übernommen wird.
- Damit steht das Ergebnis als Z und Ü zur Verfügung

Also: 1. Schritt: Registerinhalt := A
2. " : " := Registerinhalt + B
↳ z.B. bei Addition.

6

Akku: Mit nur 1 Eingang kann ein Akku aufgrund seiner Übertragsspeicher arithmetische u. logische Operationen durchführen.

u_3	u_2	u_1	u_0	Befehls-name	Beschreibung
0	0	0	0	NOP	no operation (Registerinhalt bleibt erhalten)
0	0	0	1	SP1	store plus 1 (Schreibe "1" ins Register)
0	0	1	0	CMA	complement accu (negiere Akkuinhalt)
0	0	1	1	LDA	load accu (Lade B ins Register)
0	1	0	0	CLA	clear accu (Inhalt auf null setzen)
0	1	0	1	INC	increment (Inhalt um 1 erhöhen)
0	1	1	0	DEC	decrement (" " " erniedrigen)
0	1	1	1	ADD	add (addiere B zum Inhalt)
1	0	0	0	SUB	subtract (subtrahiere B vom ")
1	0	0	1	AND	UND-Verkn. (von Akku A)
1	0	1	0	IOR	ODER- " (" " ")
1	0	1	1	XOR	Exkl.-ODER-Verkn. (" " ")
1	1	0	0	SM1	store minus 1 (Schreibe -1 ins Register)

2

Befehlsfolge für eine Subtraktion:

LDA
SUB

10