# M U L T I P R O J E K T
# C H I P - G R U P P E

B A D E N - W Ü R T T E M B E R G

W O R K S H O P J U N I 1 9 9 0

O F F E N B U R G

H E R A U S G E B E R : F A C H H O C H S C H U L E U L M

Vorwort

Auch bei dem Workshop SS 1990 der Multiprojekt-Chip-Gruppe konnte die nun schon bewährte Tradition fortgesetzt werden, kompetente Vertreter der Industrie für Vorträge zu gewinnen.
Den Herren Knapp vom AEG Entwicklungszentrum in Ulm und Schmidtpott vom ITT Intermetall sei an dieser Stelle nochmals für ihre Berichte gedankt.
Nachdem bereits bei dem Workshop WS 1989/1990 die Problematik der Ausbildung beim IC-Entwurf behandelt wurde, gab Herr Schmidtpott mit seinem Beitrag
" Das Berufsbild des Ingenieurs in der Halbleiterentwicklung "
weitere wertvolle Hinweise für die künftige Ausbildung der Ingenieure an unseren Hochschulen.
Mein Dank gilt auch Frau König von Hewlett Packert GmbH, die uns als Nutzer der APOLLO-Workstation mit den Problemen der Workstation HP/APOLLO-Strategie vertraut machte.

Ulm, Juni 1990

Prof. Führer
Fachhochschule Ulm

*INHALTSVERZEICHNIS*

1.

# *Leiterplatten-Layout mit Mentor Graphics Werkzeugen*

*von*

*W. Ritzert*

Vortrag, gehalten am 15. Juni 1990 auf der Tagung der MPC Gruppe B/W in Offenburg

# 1. Einleitung

Der folgende, kurze Überblick über Leiterplatten-Layout kann kein Schnellkurs über dieses Thema sein. Ein brauchbarer Einstieg wird von Mentor Graphics mit dem Tutorial in "Getting Started with Board Station and Managing Your PCB System" angeboten.

Hier folgen nur einige Anmerkungen aus der Sicht eines Benutzers.

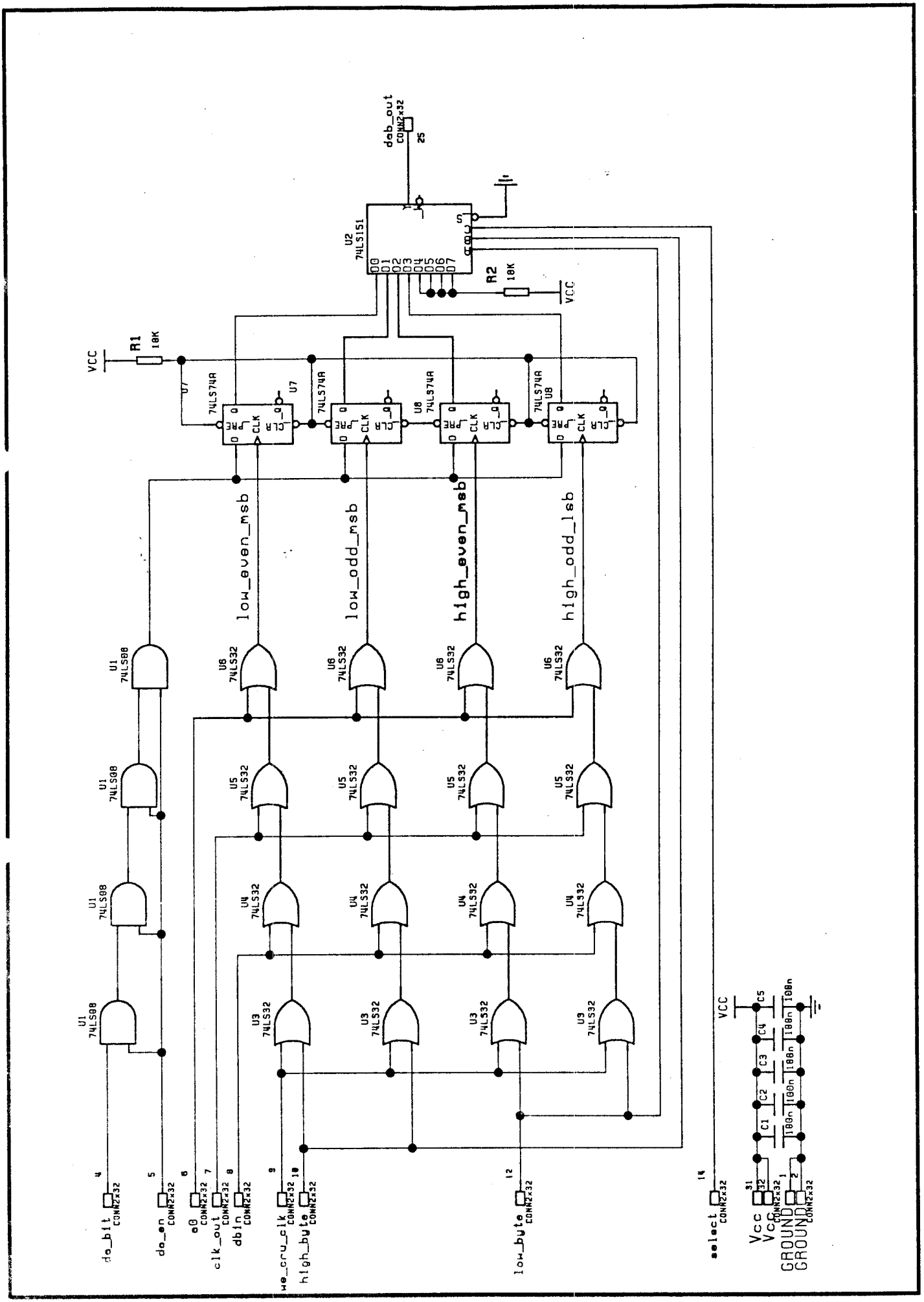
Bild 1 zeigt das Schaltbild "Mux-Board", das auch dem Tutorial des o.g. MG-Ordners (Nr. 20 966) zugrunde liegt.

Die Durchführung des Tutorials ist 'narrensicher', weil bereits alle Daten, die für das Leiterplatten-Layout benötigt werden im Directory /user/mentor/mux_board zusammengestellt sind. Alle Schwierigkeiten, die der Benutzer mit LIBRARIAN und PACKAGE bei der Bearbeitung eigener Entwürfe und bei der Verwendung eigener Bauelemente haben kann, sind damit umgangen.

Um das Problem etwas praxisgerechter zu machen, wurde deshalb nur die Struktur des Schaltbilds aus dem Tutorial verwendet. Die Bauelemente wurden durch solche aus der ls_lib und aus der accuparts_lib ersetzt. Dabei entstanden doch einige Schwierigkeiten, insbesondere mit den Portins und Portouts bzw. mit dem Anschlußstecker.

Zu sehen sind in einem Schaltbild wie z.B. in Bild 1 nur Kopien von Symbolen, die in NETED Instances genannt werden, und Symbolnamen (z.B. 74LS08). Aus dem Schaltbild ist nicht zu entnehmen, ob Dual-In-Line-Bauteile oder SMD-Bauteile benutzt werden sollen. Es ist auch nicht zu erkennen, daß evtl. mehrere Gatter in einem Bauteil zusammengefaßt werden können und wenn ja, wieviele. Den Kondensatoren, Widerständen und Steckern sieht man nicht an , welche Bauform für sie vorgesehen ist.

Die Liste ließe sich fortsetzen, es sollte damit nur gezeigt werden, daß die Informationen aus einem Schaltbild normalerweise nicht ausreichen, um eine Leiterplatte zu konstruieren.


dob_out
CONN2x32
25
U2
74LS151
R2
18K
VCC
R1
18K
VCC
U7
74LS74A
U8
74LS74A
low_even_msb
low_odd_msb
high_even_msb
high_odd_lsb
U1
74LS08
U6
74LS32
U5
74LS32
U4
74LS32
U3
74LS32
do_bit
do_en
e0
clk_out
dbin
we_cru_clk
high_byte
low_byte
select
Vcc
Vcc
GROUND
GROUND
C1
C2
C3
C4
C5
100n

# 2. Vorbereitungen

Um ein Leiterplatten-Layout durchführen zu können, müssen den Symbolen des Schaltbilds (also den Instances) bestimmte Properties zugewiesen werden (siehe unten).

Nach dem obligatorischen 'Check Sheet' wird die Schaltung mit 'expand_pcb <design-name> für die Anwendung der PCB-Tools vorbereitet.

## 2.1. Symbole und Symbolbibliotheken

Schaltplan-Symbole sind in den sogenannten Symbol-Bibliotheken enthalten (z.B. /user/ls_lib, /user/accuparts_lib usw.).

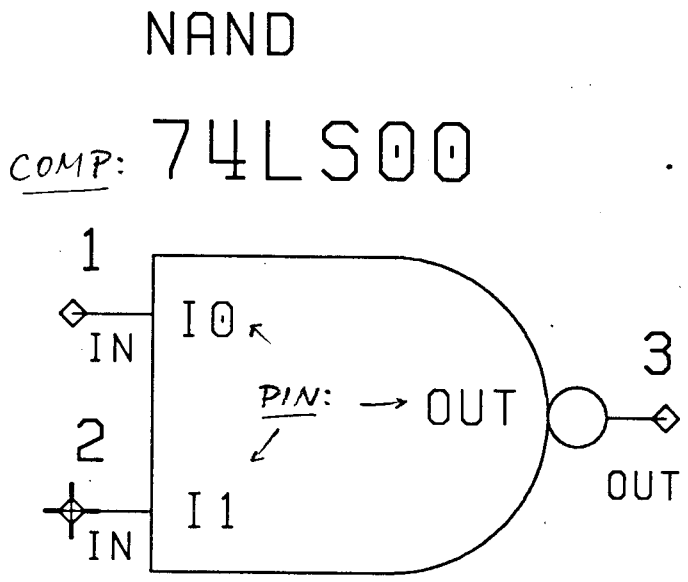
Im Gegensatz hierzu sind die eigentlichen Bauelemente oder Bauteile, die auf der Leiterplatte plaziert und verdrahtet werden, z.B. unter /user/pcb_parts zu finden.

## 2.2. Properties im Zusammenhang mit dem Leiterplatten-Layout

Bild 2 zeigt Properties, die einem Symbol mit dem Namen $74LS00 aus der Mentor-LS-Bibliothek zugewiesen sind. Davon sind nur die folgenden Properties für das Leiterplatten-Layout erforderlich:

```
PIN (name)      'I0'      'I1'      'OUT'
```

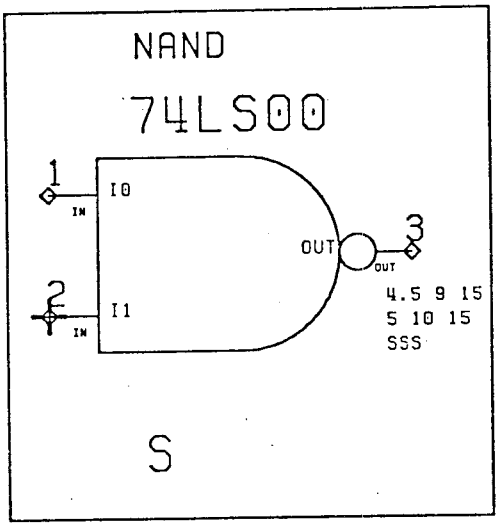
und `COMP  '74LS00'`



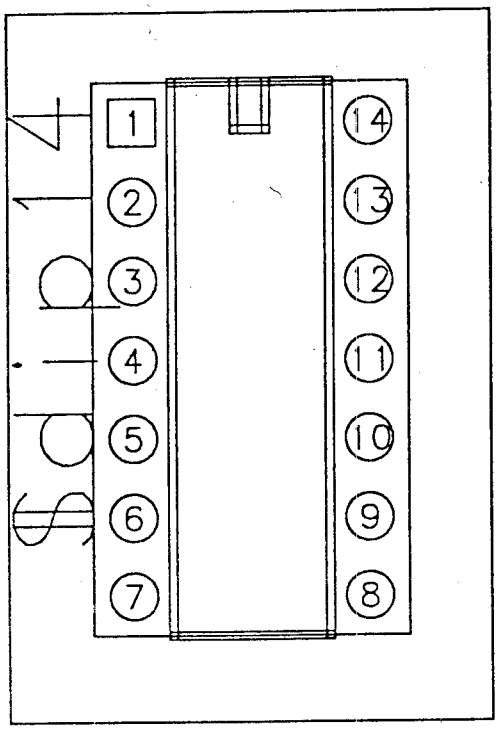

```
4.5 9 15
5 10 15

SSS
```

Bild 2

## SYMBOL:




## GEOMETRY:




## MAPPING:

```
# mapping file for $74ls00
symbol 1  1
 pin i0 1  1
 pin i1 2  1
 pin out 3  0
symbol 2  1
 pin i0 4  1
 pin i1 5  1
 pin out 6  0
symbol 3  1
 pin i0 9  1
 pin i1 10  1
 pin out 8  0
symbol 4  1
 pin i0 12  1
 pin i1 13  1
 pin out 11  0
power ground 7
power vcc 14
```

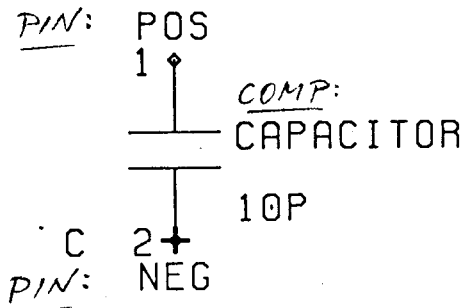
## CATALOG:

```
# PART NUMBER CATALOG FOR /user/ls_lib
#
PART_NO                    COMP_PROPERTY  GEOMETRY       MAPPING_FILE     SYMB_CNT

pn-74ls00                  74ls00         $dip14         $74ls00.map          4

prop_file_1=ps_material  \
index_1=plastic prop_file_2=ps_plastic index_2=$dip14 junction_max_t=90 pow_derating=a pins_out=4  \
pow_min=11.4 pow_max=39.1 pow_typ=20 pow_del_max=4 pow_del_typ=2 mfg=t.i.
```



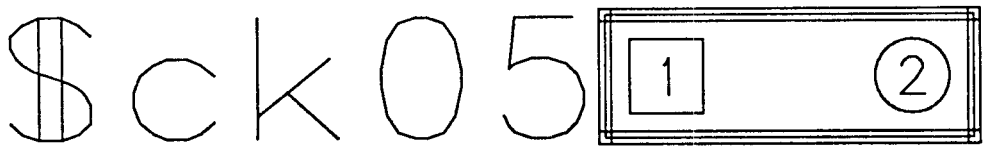
Bild 3

SYMBOL:

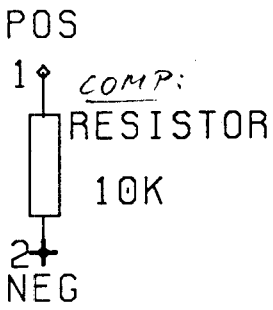



MAP:

```
#
# MAP FILE: cap.map
#
SYMBOL 1 0 "CAPACITOR" "capacitor"
 PIN "NEG" "2" 1
 PIN "POS" "1" 1
```

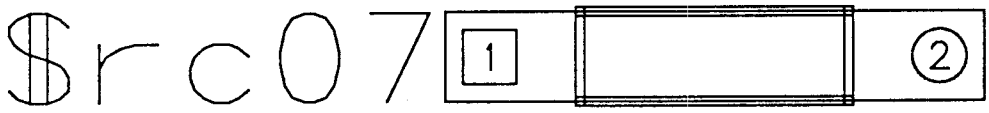



SYMBOL:




MAP:

```
#
# MAP FILE: res.map
#
SYMBOL 1 0 "RESISTOR" "resistor"
 PIN "POS" "1" 0
 PIN "NEG" "2" 0
```




CATALOG:

```
#
# PART NUMBER CATALOG FOR design.catalog
#
PART_NO                  COMP_PROPERTY  GEOMETRY      MAPPING_FILE    SYMB_CNT

 "pn-dec_cap"              "CAPACITOR"     "$ck05"         "cap.map"          1
 "pn-pullup_wid"           "RESISTOR"      "$rc07"         "res.map"          1
```

# 3. Verknüpfungen zwischen Symbolen und physikalischen Bauelementen (Parts)

## 3.1. Mapping- und Catalog-Files

Die Zuordnung zwischen Symbolen und Bauelementen wird über Teile-Nummern in einem sogenannten **Catalog-File** vorgenommen.

Die Zuordnung zwischen logischen Pin-Namen und physikalischen Anschluß-Nummern wird in einem **Mapping-File** vorgenommen.

Die Bilder 3 und 3a zeigen die Symbole, Mapping-Files und Katalog-Einträge für die Teile-Nummern pn-74ls00, pn-dec_cap und pn-pullup_wid.

### Symbol-Swapping

Zum Zweck einer einfacheren Entflechtung können Gatter verschiedener Bauteile getauscht werden.

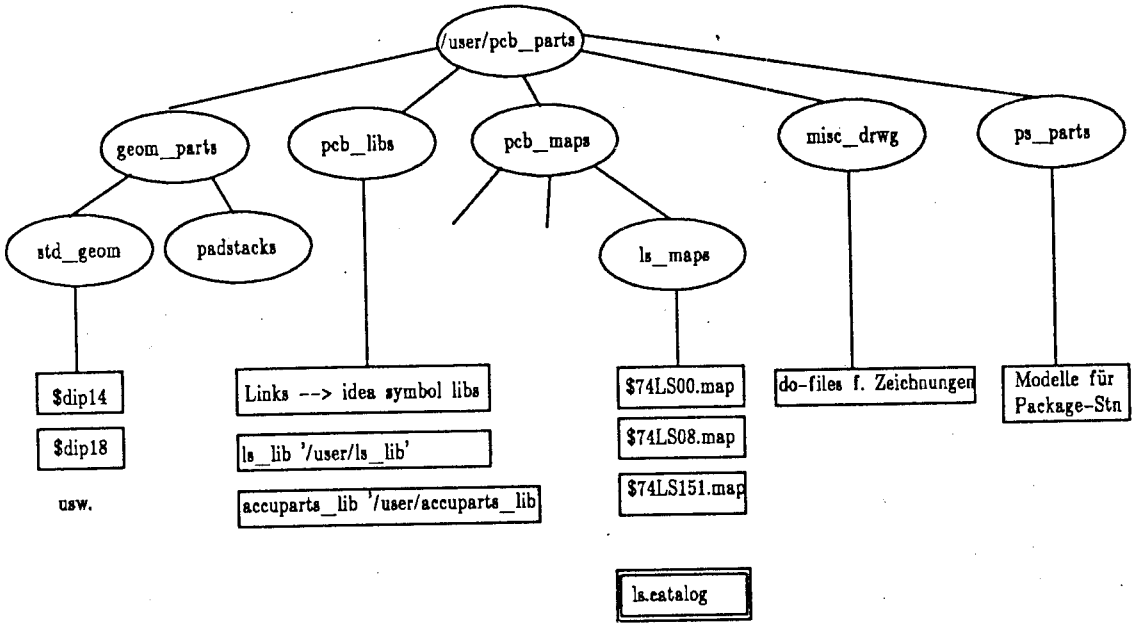
### Pin-Swapping

Ebenfalls zum Zweck einer einfacheren Entflechtung können Pins miteinander vertauscht werden.

Der Swap-Code '0' sperrt das Swapping. Gleiche Zahlen > 0 markieren Gatter bzw. Pins, die vertauschbar sind. (z.B. die Pins I0 und I1 beim 7400, oder auch alle 4 Gatter des 7400)

# 4. Bauelemente-Bibliotheken (pcb_parts)

Bild 4 zeigt die Struktur der pcb_parts - library.




Struktur der PCB_PARTS Library

Bild 4

## 4.1. Symbole (pcb_libs)

pcb_libs enthält Links (Zeiger) auf die normalen Symbol-Bibliotheken (idea symbol libs).

## 4.2. Geometrien (geom_parts)

geom_parts enthält die Geometriedaten der Bauteile, wie z.B. für $dip14:

```
lock window on -all
create component $dip14
page 8.5 5.5 0.03 inches 0.0 0.0 0.0 0.0 co$$dip14
point mode vertex
attr 'component_pin_definition'  '14'   -scale 0.3 0.0
attr 'component_pin_definition'  '13'   -scale 0.3 -0.1
attr 'component_pin_definition'  '12'   -scale 0.3 -0.2
attr 'component_pin_definition'  '11'   -scale 0.3 -0.3
attr 'component_pin_definition'  '10'   -scale 0.3 -0.4
attr 'component_pin_definition'  '9'   -scale 0.3 -0.5
attr 'component_pin_definition'  '8'   -scale 0.3 -0.6
attr 'component_pin_definition'  '7'   -scale 0.0 -0.6
attr 'component_pin_definition'  '6'   -scale 0.0 -0.5
attr 'component_pin_definition'  '5'   -scale 0.0 -0.4
attr 'component_pin_definition'  '4'   -scale 0.0 -0.3
attr 'component_pin_definition'  '3'   -scale 0.0 -0.2
attr 'component_pin_definition'  '2'   -scale 0.0 -0.1
attr 'component_pin_definition'  '1'   -scale 0.0 0.0
path silkscreen 0.01 0.05 0.05 0.05 -0.65 0.25 -0.65 0.25 0.05 0.05 0.05
path silkscreen 0.01 0.17 0.05 0.17 -0.01 0.13 -0.01 0.13 0.05
text silkscreen "^$ref" -0.05 0.05 0.1  br  -vert
attr 'component_placement_outline' '' -scale 0.35 0.05 -0.05 0.05 -0.05 -0.65
0.35 -0.65
```

Mit ADD ATTRIBUTES in LIBRARIAN können weitere Attribute hinzugefügt werden. Für einen Stecker, der über den Leiterplattenrand hinausragen soll, müßte man beispielsweise das Attribut component_outline_overhang hinzufügen.

## 4.3. Mapping- und Catalog-Files (pcp_maps)

pcb_maps enthält die Mapping- und Catalog-Files, wie z.B. für die ls_lib:
siehe hierzu die Bilder 3 und 3a.

Mit jeder Teile-Nummer (pn-...) wird im Catalog-File eine Verknüpfung von Symbol (COMP_PROPERTY), Geometrie (GEOMETRY), und MAPPING_FILE festgelegt und die Anzahl der Symbole pro Bauelement (SYMB_CNT) angegeben. (Das Catalog-File muß im selben Directory stehen, wie die Mapping-Files, auf die Bezug genommen wird.)

Zu beachten ist, daß das (Schaltplan-) Symbol mit seiner COMP-Property und nicht mit seinem Symbolnamen, unter dem es in der Symbol-Library steht, referenziert wird !

# 5. Standard-Suchpfade

Die Programme LIBRARIAN, PACKAGE, LAYOUT und FABLINK suchen nach Bauteilen in bestimmten Standard-Verzeichnissen:

### Mentor directory

Die Struktur des Directories pcb_parts wurde schon mit Bild 4 vorgestellt. Alle Standard-Bauelemente wird man zunächst hier suchen.

### Company directory

Hier kann man sich ein Verzeichnis fh_teile mit den Unterverzeichnissen fh_geom, fh_lib und fh_maps vorstellen, in denen alle fh-spezifischen Teile wie z.B. Stecker, Buchsen usw. untergebracht sind.

### Project directory

Wenn mehrere Bearbeiter am selben Projekt (z.B. mit dem Namen P1) arbeiten, kann man ein Verzeichnis mit dem Namen P1 und den 3 Unterverzeichnissen project_geom, project_lib und project_maps einrichten.

### User directory

Direkt im Home-Directory eines Benutzers sollte man ein Verzeichnis mit dem Namen pcb_parts einrichten. Dieses Verzeichnis enthält nicht nur die 3 Unterverzeichnisse mit den Namen user_geom, user_lib und user_maps, zur Aufnahme benutzer-spezifischer Teile, sondern auch die Links, die auf die o.g. Directories zeigen.

Hier ein Beispiel für den Inhalt von pcb_parts im Home-Directory eines Benutzers:

```
link    company_geom    "//ws4/local_user/fh_teile/fh_geom"
link    company_lib     "//ws4/local_user/fh_teile/fh_lib"
link    company_maps    "//ws4/local_user/fh_teile/fh_maps"
link    project_geom    "//ws4/local_user/P1_project/project_geom"
link    project_lib     "//ws4/local_user/P1_project/project_lib"
link    project_maps    "//ws4/local_user/P1_project/project_maps"
dir     user_geom
dir     user_lib
dir     user_maps
```

### Design directory

Innerhalb des Design Directories befinden sich wiederum 3 Unterverzeichnisse mit den namen design_geom, design_lib und design_maps. Sie sind für die design-spezifischen Teile gedacht, die zu einer bestimmten Leiterplatte gehören. Der Anfänger wird hier jedoch zunächst alle seine Teile unterbringen und evtl. sogar Teile aus der Mentor Bibliothek hierher kopieren.
Dies hat den Vorteil, daß 'alles beisammen ist'. Es hat auch den Vorteil, daß bei einem Backup nur ein Tree, nämlich das Design-Directory zu sichern ist.

Figure 1–2 shows how PACKAGE fits into the rest of the PCB System.

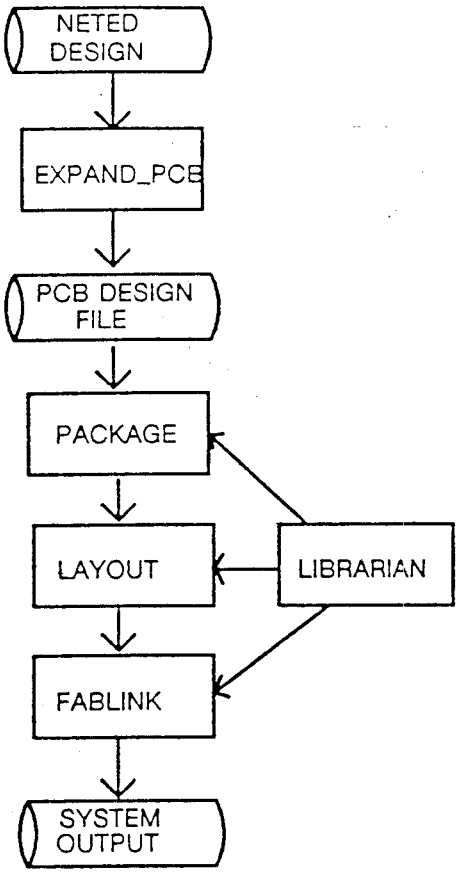



**Figure 1–2 PACKAGE Overview Diagram**



Bild 5

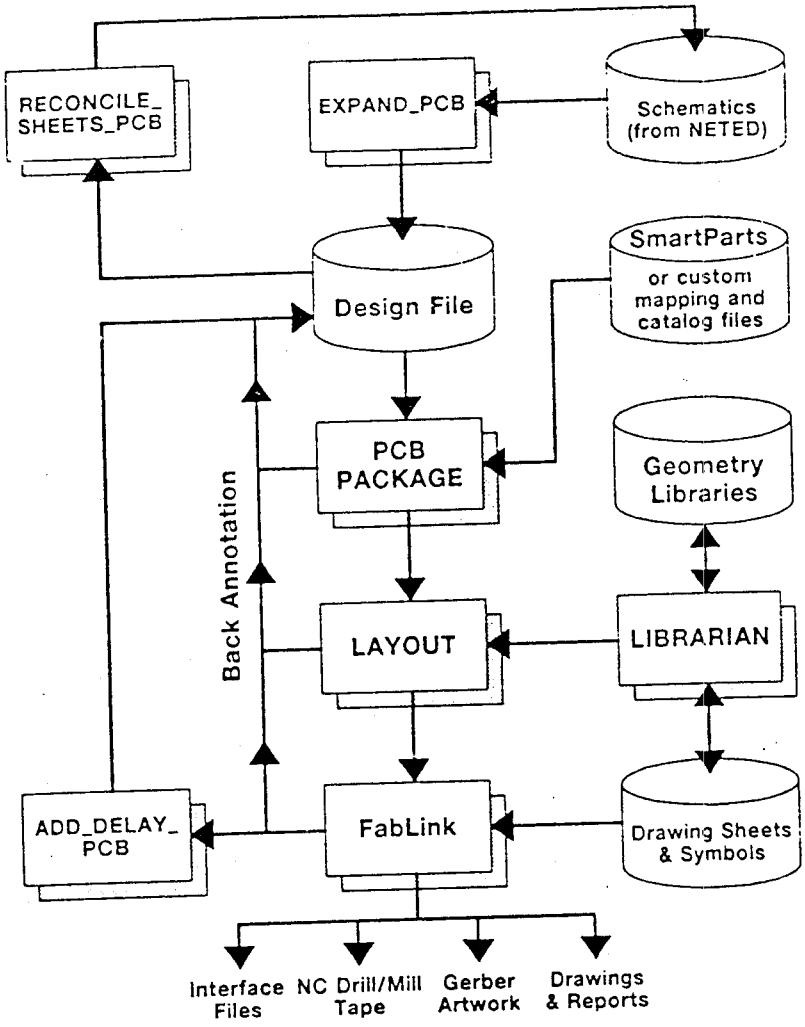



Figure 2–2 Board Station Diagram



Bild 5a

# 6. Ablauf einer Entflechtung

PCB-Design mit Mentor Graphics Tools ist sehr flexibel und deshalb für den Anfänger auch schwer zu übersehen. Allein die Bedienungsanleitungen von "Getting Started ..." , PACKAGE, LIBRARIAN, LAYOUT und FABLINK umfassen zusammen ca. 1500 Seiten. Der Verfasser hat bisher noch keinen vollen Überblick über das System.

Im Folgenden wird versucht, rezeptförmig eine Methode anzugeben, die nicht unbedingt die eleganteste und speichersparendste sein muß, die aber relativ sicher zu einem befriedigenden Ergebnis führt.

## 6.1. Aufbereitung der Daten

Zunächst verschaffe man sich anhand des Schaltbilds und ergänzender Unterlagen einen Überblick über die einzusetzenden Bauelemente incl. aller Stecker der Abmessungen der Leiterplatte, der Aussparungen auf der Leiterplatte, innerhalb derer nicht geroutet bzw. innerhalb derer nicht plaziert werden darf usw.
Auch die Breite der Leiterbahnen, die Schutzabstände zwischen den Leiterbahnen, die Durchmesser der Lötaugen, der Bohrlöcher, der Lötstopmaske usw. müssen festgelegt werden.

Diese sog. Design-Rules können abschließend erst dann festgelegt werden, wenn die zur Verfügung stehenden Blenden des Fotoplotters bekannt sind. Außerdem sei hier auch auf Betrachtungen im "LAYOUT User's Manual" ab Seite 4-68 hingewiesen !

Im Design-Directory, das i.A. den Namen der unter NETED gezeichneten Schaltung haben wird (z.B. 'mx'), legt man 3 Unterverzeichnisse an:

design_geom, design_libs, und design_maps

Unter design_maps wird ein File mit dem Namen **design.catalog** angelegt.

Im Design Directory sollte sich auch bereits das Package-Config-File pkg.config befinden, das etwa folgenden Inhalt haben kann:

```
#REL 7.0 config file version 1.0                Beispiel !!!
INST spicepar spicepartol
COMP component_placement_overhang
CONNECTOR edge
REFERENCE R RES RESISTOR WIDERSTAND
REFERENCE C CAPACITOR CAP ELKO KONDENSATOR
REFERENCE J JUMPER CONN
REFERENCE Q NPN PNP
```

pkg.config steuert u.A.:

die Art der Zuweisung von Reference Property Werten zu bestimmten Symbolen und z.B. das Erkennen von Bauelementen als Stecker.

(siehe PCB PACKAGE User's Manual, 3-30, Creating the Pkg.config File.)

## 6.2. Zuweisung von Symbolen zu Bauelementen mit PACKAGE

Schon vor dem Aufruf von PACKAGE sieht man in den Mentor- und in anderen Bauteile-Bibliotheken nach, welche Bauelemente dort als **Geometrie-Files**, als **Mapping-Files** und als **Katalog-Einträge** mit Teilenummern vorhanden sind und notiert sich deren Pfadnamen.

Nach dem Aufruf von PACKAGE mit '**package <design-name>**' gibt man mit

**Setup > Catalog Directory > Specify**

die Suchpfade für Kataloge ein, in denen nach Bauelementen gesucht werden soll z.B.

/user/pcb_parts/pcb_maps/ls_maps
/user/username/design_name (z.B. mx)/design_maps

Dann wählt man die zu ladenden Kataloge mit:

**Setup > Catalog Library > Load Catalog Library**

wobei die gewünschten Kataloge in einem Formular anzuklicken sind.

Versucht man jetzt mit

**Build > Default**

die Zuweisung von Schaltplan-Symbolen (Instances) zu physikalischen Bauelementen durchzuführen, so werden alle Bauelemente, bei denen dies geglückt ist, mit einem 'A' markiert.

Alle anderen Einträge werden mit '*E*' markiert.

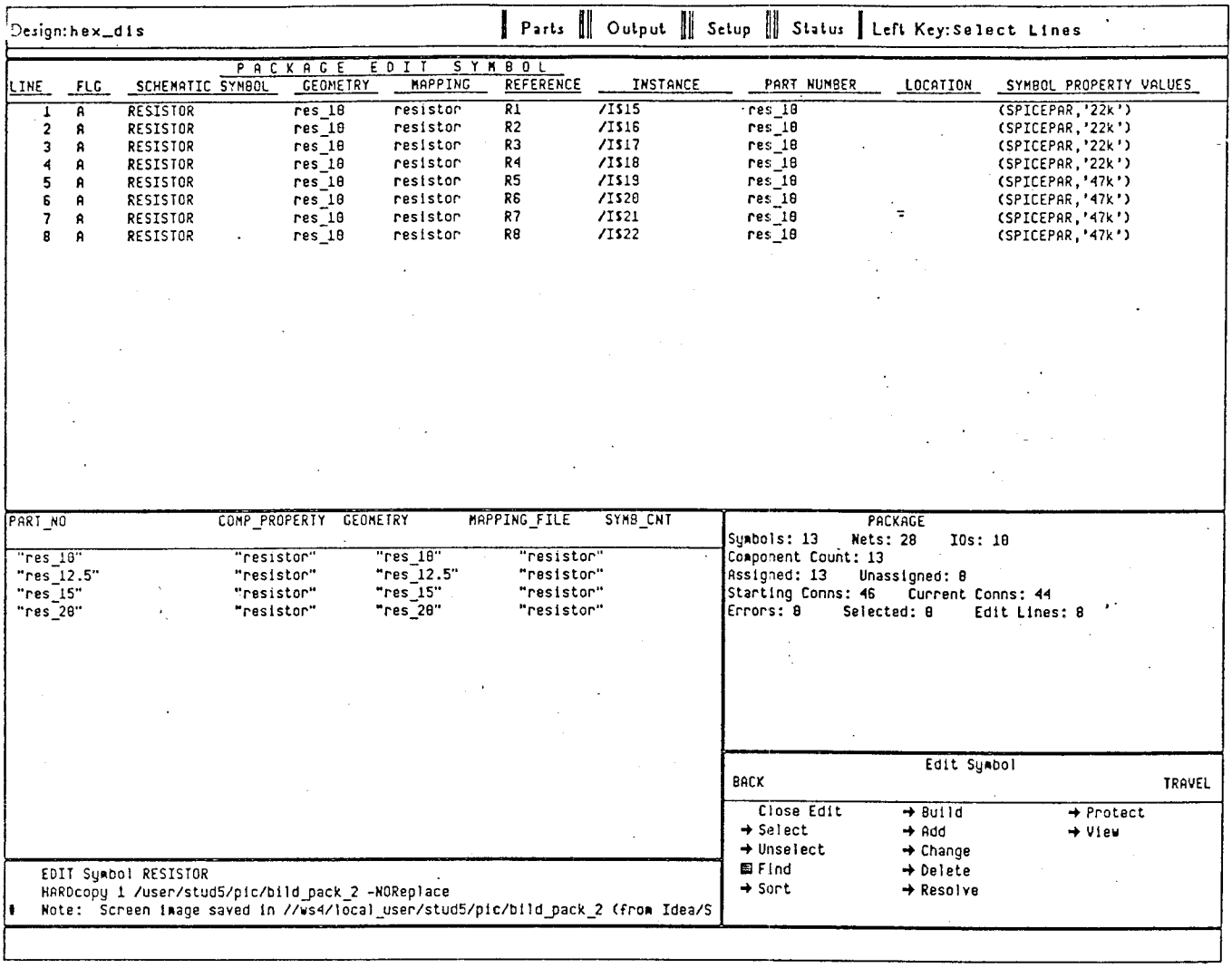
Ggf. müssen weitere Kataloge geladen werden. Die **fehlenden Teile** müssen mit LIBRARIAN erstellt werden.

Nach einem fehlerfreien BUILD-Durchgang können die automatisch erfolgten Zuweisungen eines Symbols zum erstbesten, passenden Bauteil manuell geändert werden. Ein Beispiel wäre ein Widerstand aus dem Schaltbild, dem ein Bauteil $rc05 mit einem Rastermaß von 10mm zugewiesen wurde, der als 1 Watt Widerstand aber mit einem entsprechend größeren Bauteil gekoppelt werden muß. Nach einer solchen Operation muß wieder ein BUILD-Lauf durchgeführt werden.

Vor dem Verlassen von PACKAGE erstellt man das für die weitere Arbeit notwendige Parts-File durch:

**Output > Save All > Save Parts**

(siehe auch PCB PACKAGE User's Manual, 4-23, Creating a Parts File in PACKAGE)


Design:hex_dis
Parts
Output
Setup
Status
Left Key:Select Lines
PACKAGE EDIT SYMBOL
LINE FLG SCHEMATIC SYMBOL GEOMETRY MAPPING REFERENCE INSTANCE PART NUMBER LOCATION SYMBOL PROPERTY VALUES
1 A RESISTOR res_10 resistor R1 /I$15 res_10 (SPICEPAR,'22k')
2 A RESISTOR res_10 resistor R2 /I$16 res_10 (SPICEPAR,'22k')
3 A RESISTOR res_10 resistor R3 /I$17 res_10 (SPICEPAR,'22k')
4 A RESISTOR res_10 resistor R4 /I$18 res_10 (SPICEPAR,'22k')
5 A RESISTOR res_10 resistor R5 /I$19 res_10 (SPICEPAR,'47k')
6 A RESISTOR res_10 resistor R6 /I$20 res_10 (SPICEPAR,'47k')
7 A RESISTOR res_10 resistor R7 /I$21 res_10 (SPICEPAR,'47k')
8 A RESISTOR res_10 resistor R8 /I$22 res_10 (SPICEPAR,'47k')
PART_NO COMP_PROPERTY GEOMETRY MAPPING_FILE SYM8_CNT
"res_10" "resistor" "res_10" "resistor"
"res_12.5" "resistor" "res_12.5" "resistor"
"res_15" "resistor" "res_15" "resistor"
"res_20" "resistor" "res_20" "resistor"
PACKAGE
Symbols: 13 Nets: 28 IOs: 10
Component Count: 13
Assigned: 13 Unassigned: 0
Starting Conns: 46 Current Conns: 44
Errors: 0 Selected: 0 Edit Lines: 0
Edit Symbol
BACK
TRAVEL
Close Edit
→ Select
→ Unselect
Find
→ Sort
→ Build
→ Add
→ Change
→ Delete
→ Resolve
→ Protect
→ View
EDIT Symbol RESISTOR
HARDcopy 1 /user/stud5/pic/bild_pack_2 -NOReplace
Note: Screen image saved in //ws4/local_user/stud5/pic/bild_pack_2 (from Idea/S

## 6.3. Erstellung fehlender Bauteile mit LIBRARIAN

Alle noch fehlenden Bauteile, einschließlich der Leiterplatte, der Pads (Lötaugen), der Vias (Durchkontaktierungen) usw., werden mit LIBRARIAN erstellt.

Da unter PACKAGE bereits ein Parts-File erzeugt wurde, kann man die vorhandenen Teile mit

**Libraries > Draw Parts**

und **Libraries > View Part Windows > View All**

grafisch darstellen lassen.

Mit **Status > Check Parts > All Parts** läßt man sich anzeigen, welche Teile noch fehlen und beginnt mit deren Erstellung.

Fast in jedem Fall gehören zu den fehlenden Teilen die Padstacks und Vias, das Board selbst und ggf. Stecker und Buchsen.

(Bauteilen können mit Attributen bestimmte Pads zugewiesen werden. Fehlt das entsprechende Attribut bei einem Bauteil, wird der bei der Board-Erstellung angegebene Default-Padstack verwendet.)

Prozeduren zur Erstellung der genannten Teile sind z.B. in "Getting Started with Board Station ..." 2-27 bis 2-33 beschrieben. (Kompliziertere Boards können auch mit Hilfe von 3D-DESIGN konstruiert werden.)

Die Ergebnisse dieses Teils der LIBRARIAN-Sitzung werden mit

**Output > Save > Parts > Save Parts for Design**

geschrieben und gesichert.

## 6.4. Erstellung von Support-Daten mit LIBRARIAN

Neben der Aufgabe als Grafik-Editor hat LIBRARIAN noch eine wichtige Aufgabe bei der Erstellung und Ergänzung der Mapping- und Katalog-Files.
(siehe Preparing the Support Data in "Getting Started with Board Station ... ", 2-39 ff)

Man kann sich den Inhalt eines beliebigen Files mit **Status > View File > Any File** anzeigen lassen.

In dem File

**design_maps/design_parts.catalog**

müssen Einträge für diejenigen Teile vorgenommen werden, für die sich auch in den anderen Katalogen noch keine Teile-Nummern (part_numbers) befinden. Dabei geht man zweckmäßig wie folgt vor:

1. Einlesen des (Schaltplan-)Symbols mit:

   **Libraries > View Symbol Libraries > ...**

2. Prüfung, ob die Bauelemente-Geometrie vorhanden ist, mit:

   **Libraries > Draw Parts ...**

   (Bildschirm-Ausgabe ähnlich Bild 6)

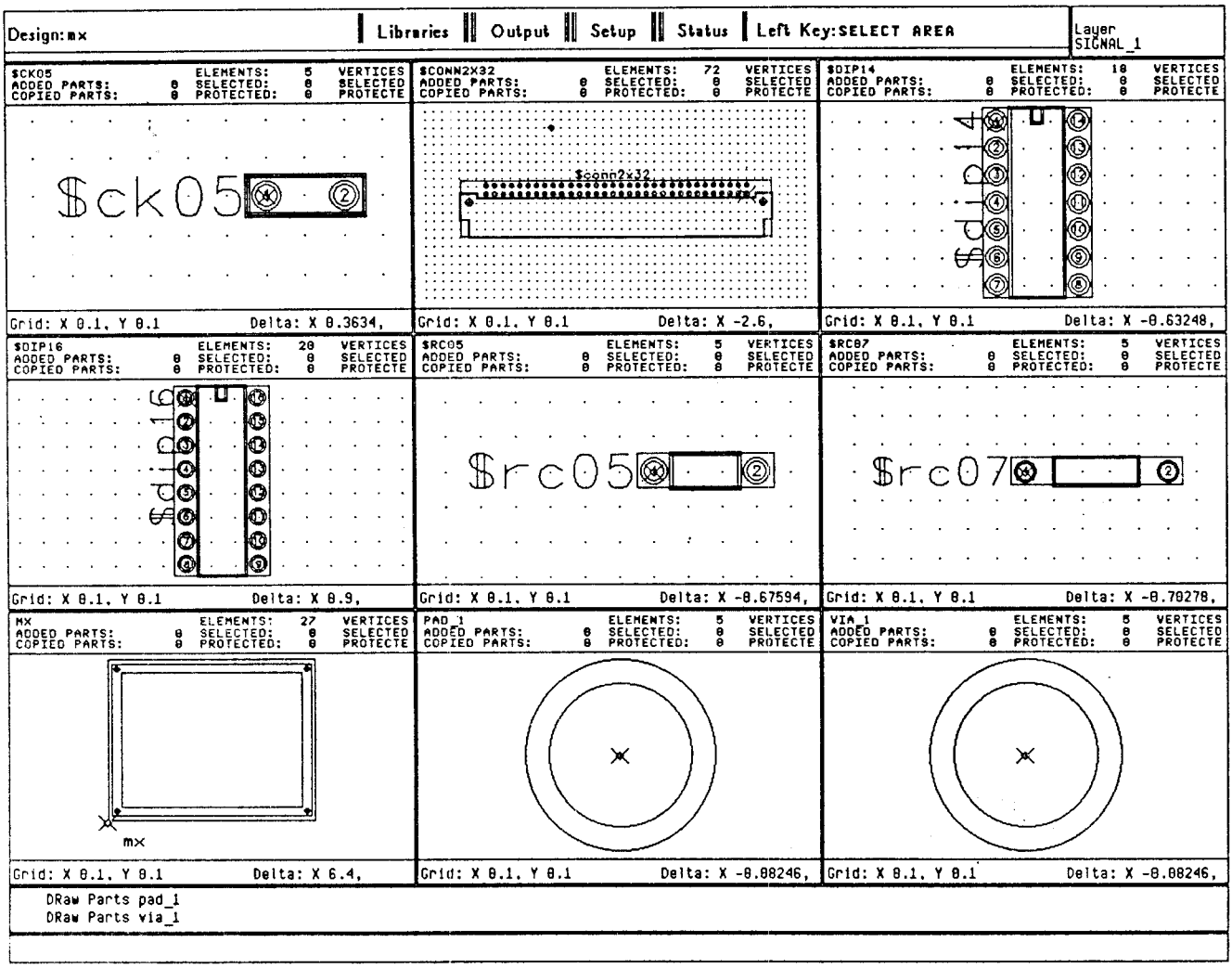



Bild 6

3. Lesen des (zum Design-Directory gehörenden) Katalogs mit:

   **Libraries > View Catalogs > Design** (click 'VIEW')

4. Erstellung des Part-Number-Eintrags im Katalog-File und des Mapping-Files

   **Libraries > Create Part Number**

   Bild 7 Zeigt das auszufüllende Formular, Bild 8 das daraus folgende Schirmbild.

Damit wird sowohl ein Mapping-File vorbereitet, als auch ein Katalog-Eintrag.
Die neu entstandenen Daten sind damit aber noch nicht in Dateien gesichert ! (s.u.)

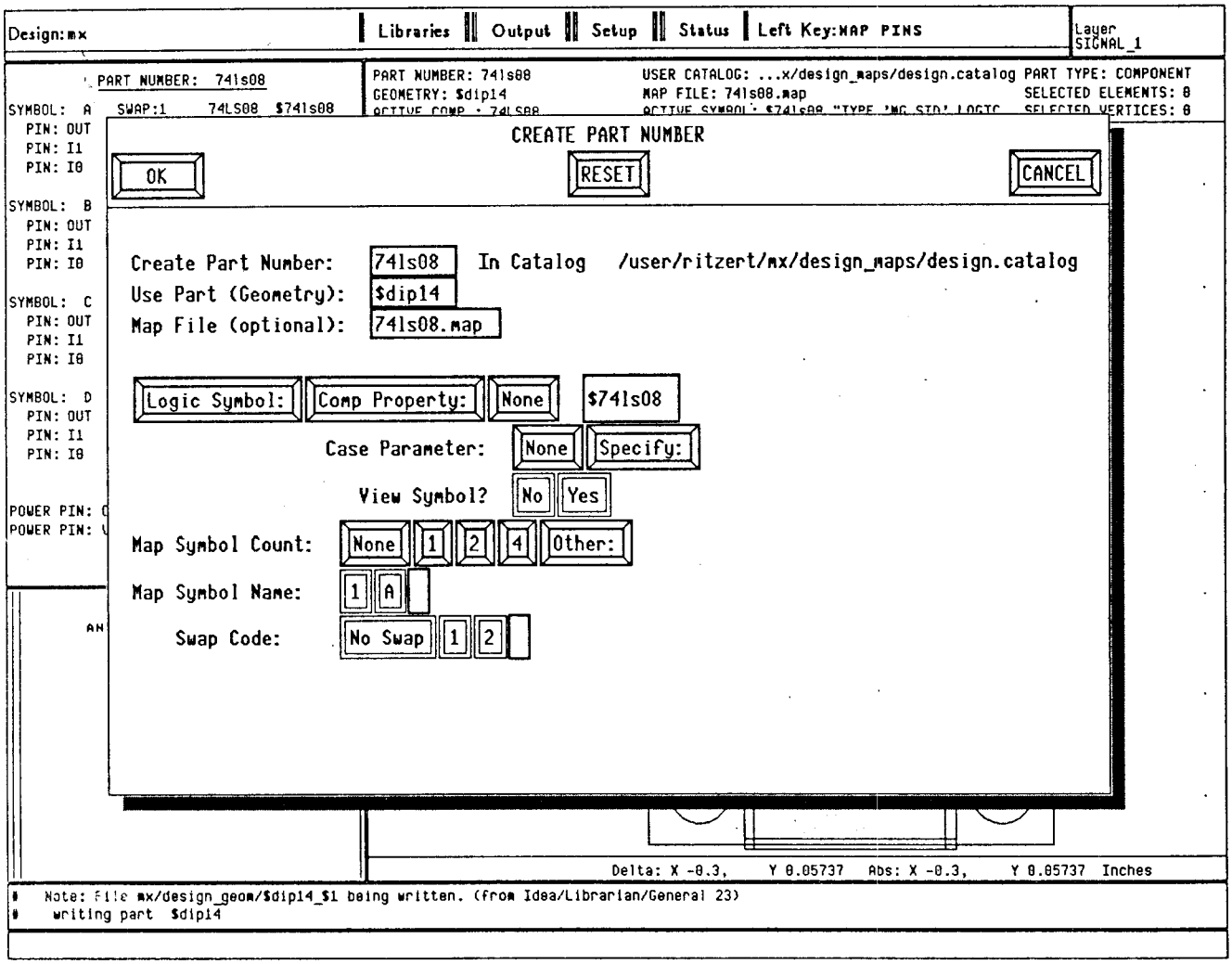



Bild 7

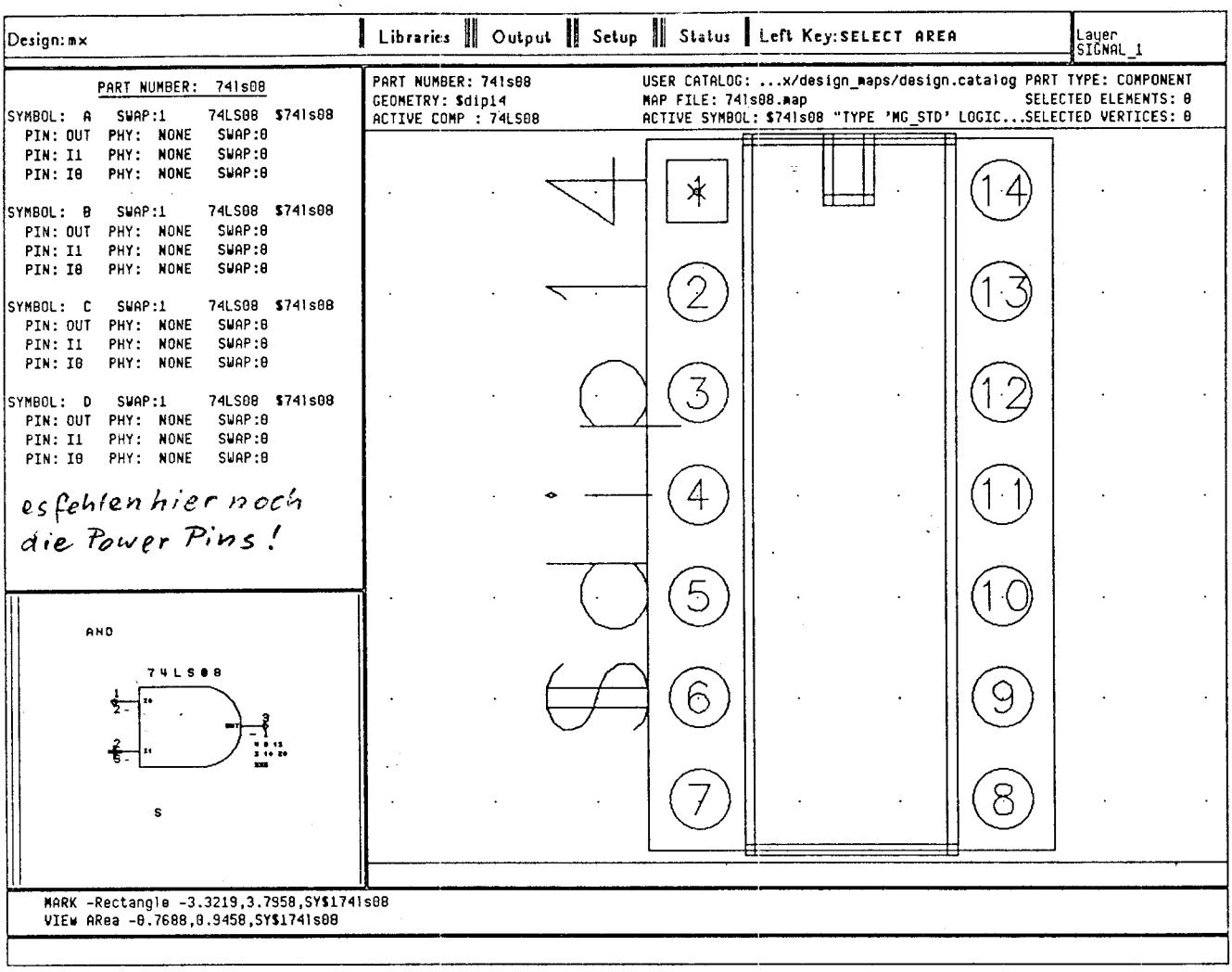



Bild 8

5. Prüfung von Part-Number und Mapping-File:

   **Status > Check Part Number > Active Part Number**

6. Sicherstellung, daß noch der richtige Katalog aktiv ist:

   **Libraries > View Catalogs > Design (click 'VIEW')**

und **WRITE im ausgegebenen Formular anklicken.**

Nachdem bereits der erste Teil der Sitzung mit **Output > Save > Parts > Save Parts for Design** gesichert worden war, kann LIBRARIAN mit

**Output > Quit**

verlassen werden.
Im Anschluß daran ist ein neuer PACKAGE-Lauf durchzuführen !

## 6.5. Plazieren und Routen mit LAYOUT

LAYOUT besteht aus den Programmteilen Placement und Routing.

Placement und Routing kann man sowohl automatisch als auch interaktiv durchführen. Man kann zwischen Placement und Routing beliebig hin- und herschalten.

Nach dem Aufruf von LAYOUT durch **layout <design-name>** legt man mit Hilfe des Setup-Menüs verschiedene Einstellparameter fest, wie z.B.:

```
Cursor Snap
Display Grid
Routing Grid
Draw Style
usw.
```

Nur wenn Auto Checking auf ON gesetzt ist, werden die Design-Rules laufend überprüft. Sonst erst nach Check Placement bzw. Check Wires.

Zu beachten ist, daß es ein Placement Grid (Display Grid) und ein Routing Grid gibt. Um optimale Routing-Möglichkeiten zu gewährleisten, sollte das der Rasterabstand für das Placement Grid ein Vielfaches dessen für das Routing Grid sein. (Vorschlag: Placement Grid = 0.1 inch.)

### Placement

In den meisten Fällen wird man mit Placement im interaktiven Modus beginnen, um Bauteile mit festen Plätzen, wie z.B. Stecker, LED's usw., an den dafür vorgesehenen Stellen zu plazieren. (Auch kann einem Bauteil schon in LIBRARIAN das Attribut 'Fixed' zugewiesen worden sein, womit es seinen festen Platz hat.)

Mit

**PLACEMENT MENU > Select > Protect > Protect Selected Components**

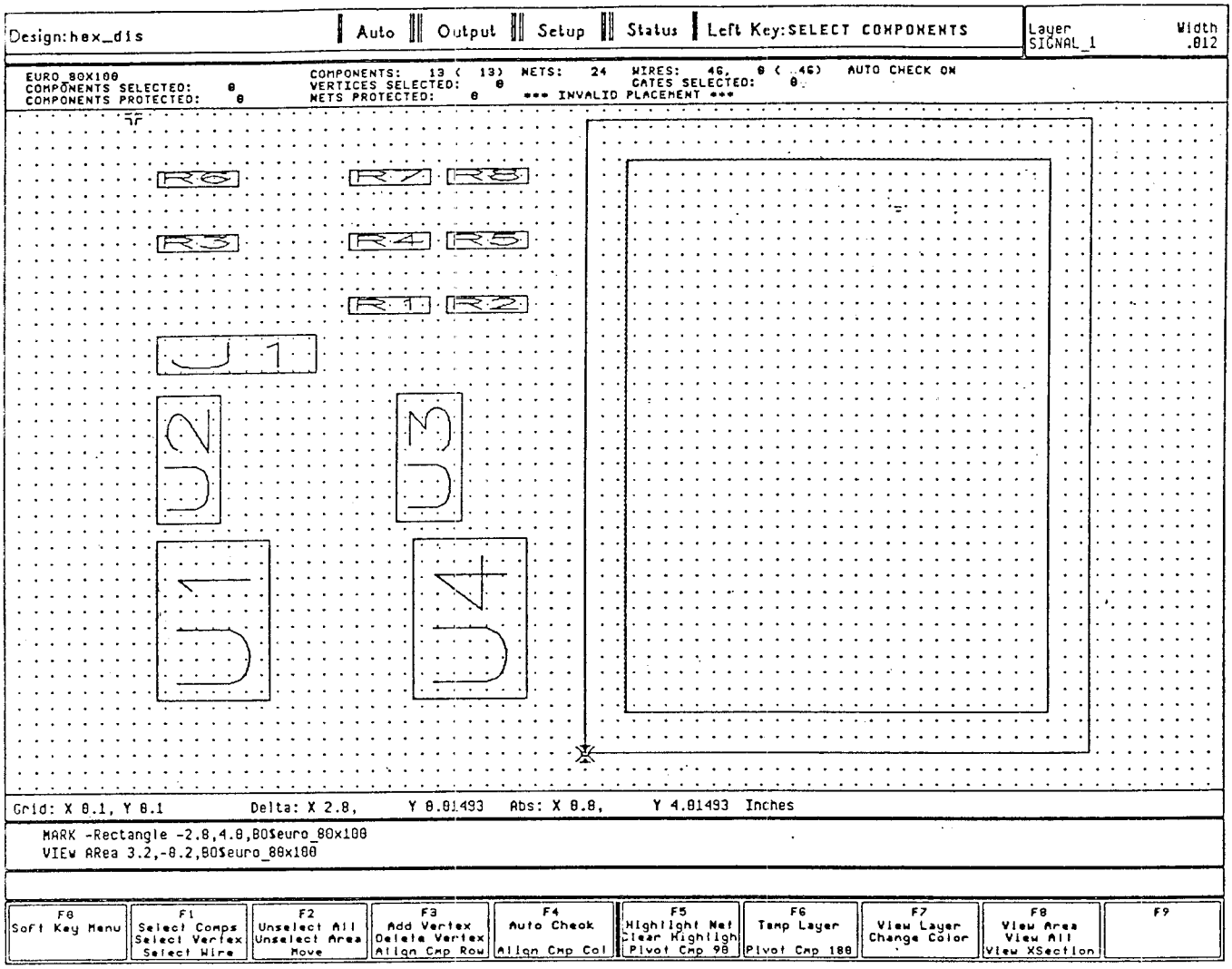
kann man plazierte Bauelemente gegen unbeabsichtigte Verschiebung sperren.

Die noch nicht plazierten Bauteile können weiterhin interaktiv nach verschiedenen Verfahren plaziert werden:

Placing Components by Connectivity
Placing Components from a Logic Sheet
Placing Components from a List
Placing a Component Group (z.B. für Bus-Strukturen)

Hierbei sind die zuschaltbaren Guide Wires ("Gummi-Fäden") sehr hilfreich.

Bauteile, deren Position vom Anwender nicht festgelegt werden sollen, können anschließend automatisch plaziert werden (**Constructive Placement**).


Design:hex_dis
Auto
Output
Setup
Status
Left Key:SELECT COMPONENTS
Layer
SIGNAL_1
Width
.012
EURO_80X100
COMPONENTS SELECTED: 0
COMPONENTS PROTECTED: 0
COMPONENTS: 13 ( 13) NETS: 24 WIRES: 46, 0 ( 46) AUTO CHECK ON
VERTICES SELECTED: 0
NETS PROTECTED: 0
GATES SELECTED: 0
*** INVALID PLACEMENT ***
R6
R7
R8
R3
R4
R5
R1
R2
J1
J2
J3
J1
J4
Grid: X 0.1, Y 0.1
Delta: X 2.8, Y 0.01493
Abs: X 0.8, Y 4.01493 Inches
MARK -Rectangle -2.8,4.8,80$euro_80x100
VIEw ARea 3.2,-0.2,80$euro_80x100
F0 Soft Key Menu
F1 Select Comps Select Vertex Select Wire
F2 Unselect All Unselect Area Move
F3 Add Vertex Delete Vertex Align Cmp Row
F4 Auto Check Align Cmp Col
F5 Highlight Net Clear Highligh Pivot Cmp 90
F6 Temp Layer Pivot Cmp 180
F7 View Layer Change Color
F8 View Area View All View XSection
F9

Bei der automatischen Plazierung wird u.A. nach folgenden Kriterien plaziert:

Minimierung von Leiterbahnlängen
Ausrichtung der Pads so, daß gerade Leiterbahnen entstehen

Weitere Hilfsmittel bei der Plazierung sind Histogramme über die zu erwartende Leiterbahndichte und ein Routing Predictor.

Der jeweilige Stand Plazierung kann mit

**Output > Save > Components > To Design**

abgespeichert werden.

## Routing

Routing erzeugt Verbindungsleitungen zwischen Bauteilen. Diese Leitungen werden 'wires' genannt. Vor dem Beginn des Routings müssen mit **Auto > Setup > Router** die Design Rules festgelegt werden:

Wire Width
Track Clearance
Pad Clearance
Via Usage
TJunctions Usage
Diagonals Usage

Erläuterungen finden sich im "LAYOUT User's Manual" ab Seite 4-27.

Geroutet wird auf der Leiterplatte, die als Stapel von Verdrahtungsebenen (stack of physical layers) angesehen wird.

LIBRARIAN speichert bei der Erstellung des Boards eine Datei mit dem Namen *tech_file* ab, das die Ebenen enthält:

Beispiel: *tech_file* für ein 2-lagiges Board

```
#  Physical Layers
#
DEFine PHYsical Layer  1 "PHYSICAL_1"  "PAD_1"  "SIGNAL_1"
DEFine PHYsical Layer  2 "PHYSICAL_2"  "PAD_2"  "SIGNAL_2"
```

Beispiel: *tech_file* für ein 4-lagiges Board

```
#  Phyical Layers
#
DEFine PHYsical Layer  1 "PHYSICAL_1"  "PAD_1"  "SIGNAL_1"
DEFine PHYsical Layer  2 "PHYSICAL_2"  "POWER_1"
DEFine PHYsical Layer  3 "PHYSICAL_3"  "POWER_2"
DEFine PHYsical Layer  4 "PHYSICAL_4"  "PAD_2"  "SIGNAL_2"
```

Das Routing selbst kann interaktiv oder automatisch durchgeführt werden.

Das interaktive Routing ist im "LAYOUT User's Manual" ab Seite 4-23 sehr gut beschrieben. Wichtig ist wieder, daß die gerouteten Verbindungen geprüft und abgespeichert werden mit:

**Status > Check Wires** und

**Output > Save > Wires > to Design**

Im Allgemeinen wird man nur einen Teil der Verbindungen interaktiv routen und den Rest dem Autorouter überlassen. Sicherheitshalber kann man die bisher existierenden Verbindungen durch

**ROUTING MENU > Select > Protect > Protect Selected Wires**

gegen Veränderungen schützen.

Das Autorouting kann im Hinblick auf verschiedene Optimierungskriterien parametriert werden. Das gesamte Konzept wird ab Seite 4-56 im "LAYOUT User's Manual" auf über 80 Seiten erläutert.

Auch hier werden die gerouteten Verbindungen wieder im *wires_file* abgespeichert mit:

**Output > Save > Wires > to Design**

## 6.6. Generierung von Fabrikationsunterlagen mit FABLINK

Der Name FABLINK läßt sich als "Bindeglied zur Fabrikation" deuten. FABLINK erzeug folgende Daten bzw. Unterlagen:

```
Artwork
Drill Data
Fabrication Drawing
Bill of Materials
```

**Artwork**

Unter Artwork versteht man eine geplottete Druckvorlage auf Film. Die ASCII-Files artwork_1 bis artwork_n enthalten Steuerdaten im Gerber-Format.

Für eine 4-lagige Leiterplatte müssen Files für z.B. folgende Druckvorlagen erstellt werden:

| Artwork Layer | Artwork File | Logical Layers |
|---|---|---|
| 1 | artwork_1 | PAD_1, SIGNAL_1 |
| 2 | artwork_2 | POWER_1 |
| 3 | artwork_3 | POWER_2 |
| 4 | artwork_4 | PAD_2, SIGNAL_2 |
| 5 | artwork_5 | SILKSCREEN |
| 6 | artwork_6 | SOLDER_MASK_1 |

**Drill Data**

Dieses ASCII-File enthält die Durchmesser und die Koordinaten für alle Bohrungen der Leiterplatte.

**Fabrication Drawing**

Erzeugt eine vermaßte Zeichnung der Leiterplatte.

**Bill of Materials**

Die Bill of Materials enthält zwei Stücklisten, eine geordnet nach Teile-Nummern, die andere geordnet nach der Ref.-Property aus dem Schaltplan. Sie unterscheidet sich im Aufbau von der Bill of Materials, die man unter PACKAGE erzeugen kann.

Mit Setup für FABLINK mußten vorher die 'Photoplot Data' (Filmgröße, Offset, Output Format, Job Header String usw.) und die Daten für das 'Aperture Wheel' (Blendenteller) eingegeben werden.

Vor dem Verlassen von Fablink (ggf. auch vor dem Verlassen von LAYOUT) ist mit

**Output > Back Annotation**

das File *pcb_design.erel* in folgenden Punkten zu ergänzen und auf den neuesten Stand zu bringen:

```
-component reference designator for each symbol instance
-board location for each symbol instance
-part number for each symbol instance
-component pin number for each logic pin
```

Erst danach kann man FABLINK mit

**output > Save & Quit** verlassen.

Um die Back Annotations aus dem *pcb_design.erel* File in das Schaltbild zu übertragen, ist in von der Command Line in AEGIS der Befehl

**$ reconcile_sheets_pcb <design-name>**

einzugeben. Nach dem 'Updaten' wird eine neue Version des Schaltbilds abgespeichert. Deshalb ist vor einem neuen Aufruf der PCB Layout Tools (PACKAGE, LIBRARIAN, LAYOUT und FABLINK) wieder ein Expand-Lauf mit **expand_pcb <design-name>** durchzuführen !

# 7. Schlußbemerkung

Mit diesem Überblick sollten einige Erläuterungen zum Leiterplatten-Layout mit Mentor Graphics Tools gegeben werden.

Was den Anfänger zunächst erschlägt, die Vielfalt der neuen Begriffe und Befehle. Man muß zunächst ein Gefühl dafür bekommen, wo man nach was suchen soll und welche Vielfalt das System bietet. Dazu soll der vorliegende Beitrag eine kleine Hilfe sein.

Es wurden absichtlich einige Kommandos, insbesondere für die Abspeicherung von Daten, die in der Hitze des Gefechts leicht übersehen werden wörtlich wiedergegeben, um den Anfänger vor dem Verlust von Daten zu bewahren.

W. Ritzert , im Juni 1990

# 2. Messungen an MOSFET's auf dem Wafer

Roland Friedrich
Labor für Prozeßcharakterisierung der Mikroelektronik
Fachhochschule für Technik Esslingen
Flandernstr.101
7300 Esslingen

## 1. Einleitung

Bei der Herstellung einer integrierten Schaltung entsteht die Notwendigkeit, die einzelnen Prozeßschritte möglichst genau zu überwachen und ihre Spezifikation einzuhalten.
Dieses Vorgehen ist zwingend, da z.B. die Herstellung eines 1MBit-DRAM's ca. 450 Prozeßschritte umfaßt und einen zeitlichen Rahmen von ca. 6 Wochen vom Wafer bis zum fertigen, elektrisch testbaren Chip beansprucht. Es ist leicht einzusehen, daß der Hersteller sich nicht auf die Funktionstests am fertigen Produkt beschränken kann.
Da elektrische Messungen am strukturierten "Device-Wafer" erst nach Aufbringen einer Metallisierungsebene möglich sind, müssen durch physikalische - überwiegend optische - Messungen die einzelnen Schritte vor dieser Metallisierung erfaßt und verifiziert werden, diese Aufgabe fällt der Prozeßcharakterisierung zu. Eine Übersicht über die Aufgaben der Charakterisierung von Halbleitern bei der Fertigung gibt das folgende Bild 1. [1]

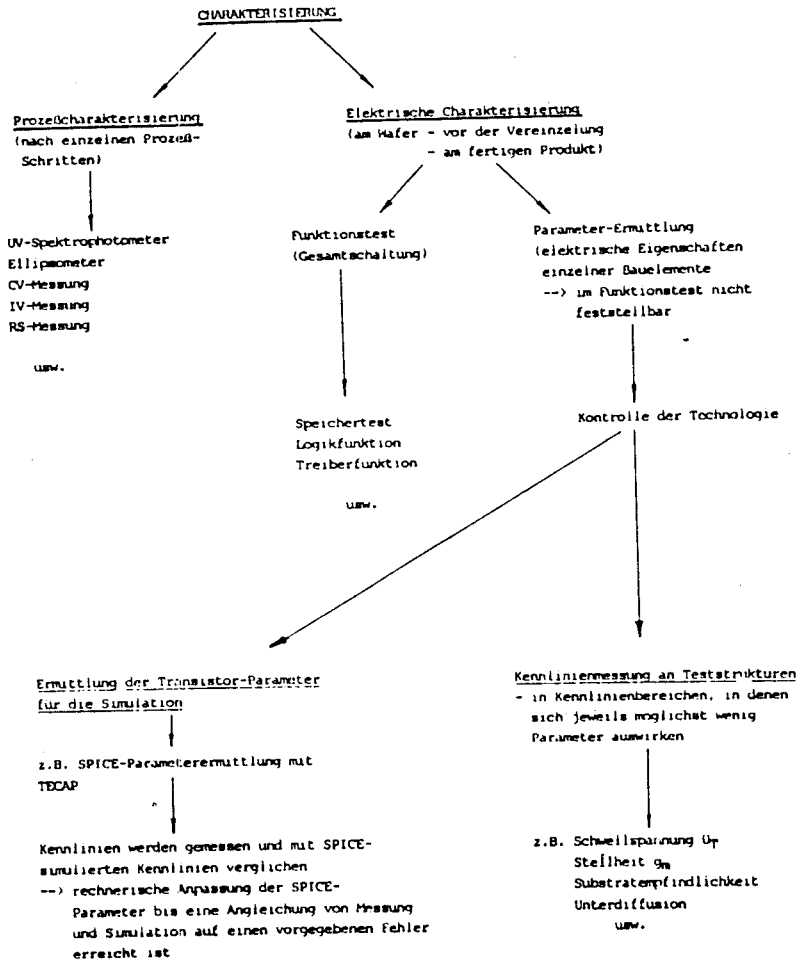



Bild 1: Übersicht über die Aufgaben der Halbleitercharakterisierung

Diese Arbeit befasst sich mit der elektrischen Charakterisierung einzelner Bauelemente auf dem Chip.

## 2. Teststrukturen auf Wafern

Der Hersteller einer integrierten Schaltung muß für den Entwurf und zur Simulation seiner Schaltungen Parameter für die einzelnen Bauelemente verwenden, die ihm z.B. von früheren Fertigungsprozessen bekannt sind oder die er durch Fertigung einer Testschaltung ermittelt hat. In der Verifizierungsphase eines Produktes muß dann nicht nur der Schaltungsentwurf verifiziert werden, sondern gleichzeitig auch der Fertigungsprozeß, der für die jeweilige Schaltung angewendet wird. Wurden für einen Schaltungsentwurf Parameter eines Vorgängerprozesses verwendet, so kann beispielsweise das Dotierungsprofil im Kanal eines FET's beim neuen Prozeß trotz gleicher Implantationsdosen usw. einfach durch das Hinzufügen eines weiteren Temperaturprozesses verändert sein - die erhöhte Temperatur dieses zusätzlichen Schrittes bewirkt ein tieferes Eindringen der Dotierstoffatome ins Substrat und damit ein verändertes Dotierungsprofil wodurch wiederum das Verhalten des Transistors verändert ist.
Zu Beginn der Fertigungseinführung eines neuen Produkts werden daher oftmals "Device-Chips" auf dem Wafer durch Testchips ersetzt, die statistisch verteilt über die Waferfläche eingesetzt werden - siehe Bild 2. Diese Testchips beinhalten z.B. kritische Schaltungsauszüge der Gesamtschaltung sowie einzelne Bauelemente für Meßzwecke. Wird der Fertigungsprozeß als stabil erachtet, können diese Testchips wegfallen und damit Platz für zusätzliche Produktchips machen. In diesem Stadium der Fertigung kann zwar der Prozeß als stabil erachtet werden, jedoch sind aktuelle Zustände der einzelnen Geräte und Anlagen laufend weiter zu überwachen. Für die Zwecke der elektrischen Charakterisierung werden einzelne Bauelemente zusammen mit Anschlußpads in die Schnittfuge zwischen die Chips gelegt, damit sind bereits vor dem Einbau der Schaltung ins Gehäuse elektrische Tests und Parametermessungen möglich.
Bild 3 zeigt die Anordnung solcher Teststrukturen in den Schnittfugen des Wafers, diese Teststrukturen werden beim Trennen der Chips zerstört.

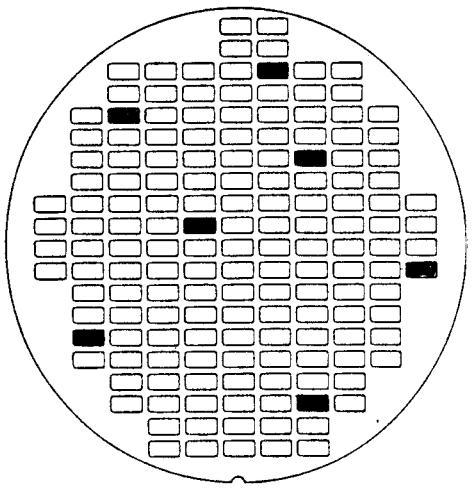


Bild 2: Produktchips werden durch Testchips ersetzt

Nachdem die erste Verdrahtungsebene aufgebracht ist, werden diese Bauelemente gemessen und mit ihren Sollwerten verglichen. Sollten hierbei starke Abweichungen festzustellen sein, wird der Wafer sofort verschrottet - Nacharbeit ist nicht möglich - bevor die weiteren Verdrahtungsebenen aufgebracht, der Wafer zersägt, eingebaut, gebondet und getestet wird.

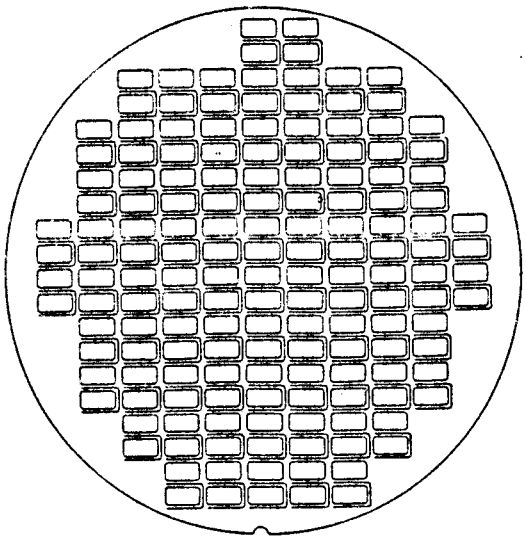


Bild 3: Lage der Teststrukturen zwischen den Chips

In Bild 4 ist am Beispiel eines 1MBit-Wafers der Firma IBM zu sehen, welche Bauelemente in einer solchen Schnitfuge enthalten sein können.

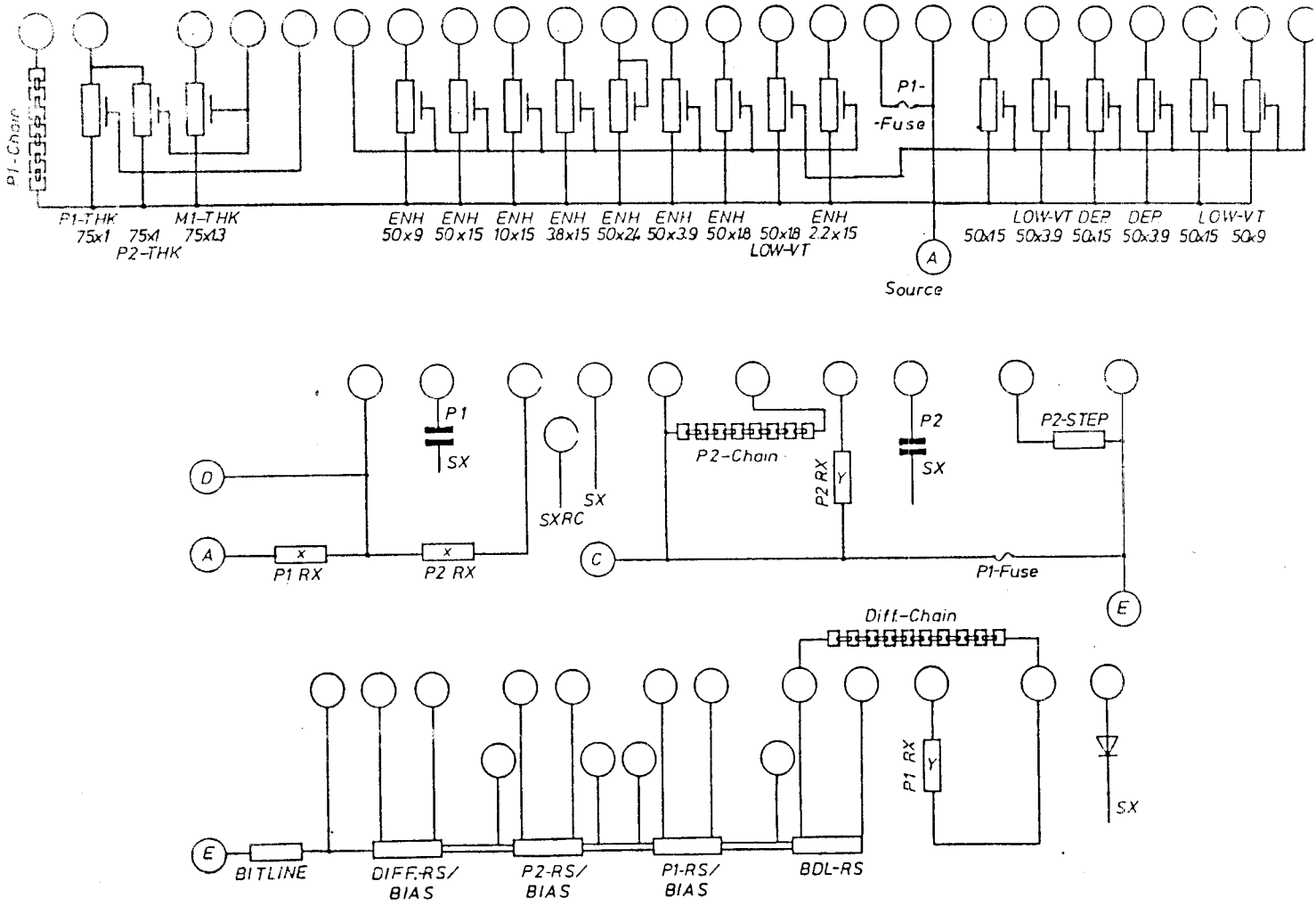



Bild 4: Bauelemente in einer Schnittfuge

## 3. Messungen an Bauelementen in den Schnittfugen

Im Labor für Prozeßcharakterisierung der Mikroelektronik an der Fachhochschule Esslingen wurde durch eine Diplomarbeit im WS 89/90 eine Messung dieser Bauelemente und eine Auswertung der gemessenen Daten ermöglicht [2] . Die Messung und Auswertung erfolgt dabei rechnergestützt von einem PC aus mit Hilfe des Programmpakets ASYSTANT GPIB, das eine einfache Programmierung von Meßroutinen IEC-Bus-steuerbarer Meßgeräte und eine Fülle von mathematischen und graphischen Auswerte- bzw. Ausgabemöglichkeiten beinhaltet - Bild 5 zeigt den Aufbau des Meßsystems.

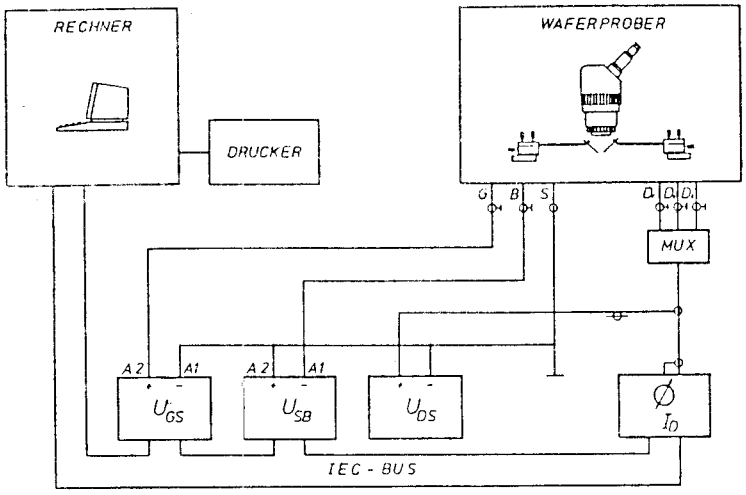



Bild 5: Meßplatz zur Kennlinienmessung auf dem Wafer

Exemplarisch wurden Messungen an 3 MOSFET's vom Anreicherungstyp durchgeführt und werden hier dargestellt.
Es handelt sich um Transistoren mit einem W/L - Verhältnis von 50/15 µm; 50/3,9 µm bzw. 10/15 µm, an denen jeweils die Steuerkennlinie, also der Drainstrom in Abhängigkeit von der Gate-Source-Spannung im Triodenbereich gemessen werden. Aus den Meßwerten dieser Messungen, deren graphische Darstellung Bild 6 ist, können die Parameter Einsatzspannung $U_T$, Leitfähigkeitskonstante $B_0$ , Unterdiffusion DELTA L und Substrateffektkonstante berechnet werden.

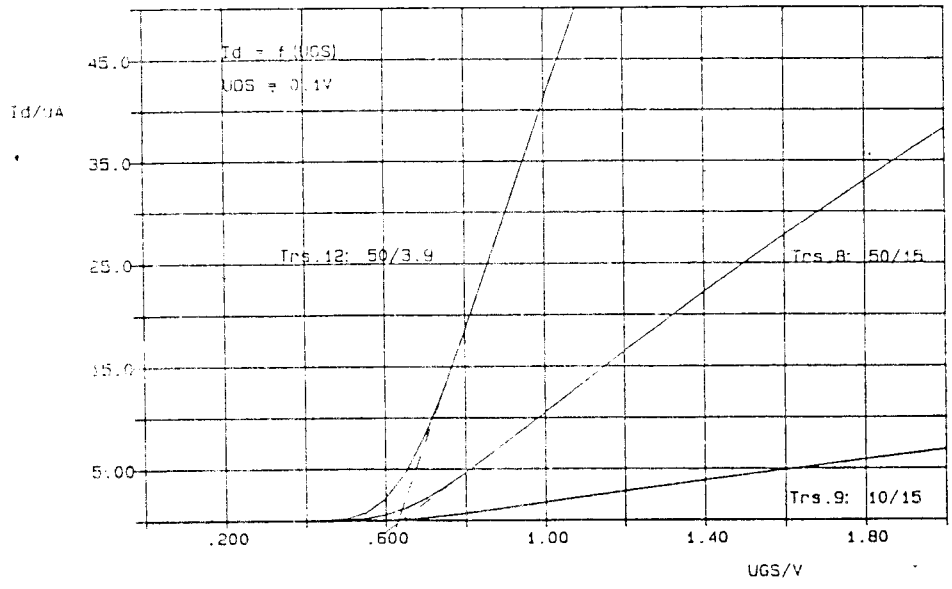



Bild 6: Steuerkennlinien dreier MOSFET's mit unterschiedlichem W/L - Verhältnis.

## 4. Ergebnisse der Messungen und Berechnungen

Die Einsatzspannung wird durch eine Zweipunktmessung mit anschließender Berechnung einer durch diese Punkte gelegten Geraden ermittelt, dabei ist der Schnittpunkt dieser Geraden mit der Abszisse gerade die Einsatzspannung $U_T$ des betreffenden Transistors - siehe gestrichelt eingezeichnete Geraden in Bild 6. Rechnerisch wurden die Werte

$U_T$ = 0,598 V (W/L = 50/3,9 µm)
$U_T$ = 0,61 V (W/L = 50/15 µm)
$U_T$ = 0,62 V (W/L = 10/15 µm)

ermittelt.
Aus der Steigung dieser Geraden läßt sich unter Berücksichtigung des entsprechenden W/L - Verhältnisses und der verwendeten Drain-Source-Spannung nach folgender Gleichung die Leitfähigkeitskonstante $B_0$ errechnen. [3]

$$B_0 = \frac{I_{D2} - I_{D1}}{(U_{GS2} - U_{GS1}) * U_{DS} * W/L} \qquad (1)$$

Die ermittelte Leitfähigkeitskonstante $B_0$ betrug im Mittel 89 µA/V$^2$, gemessen an einem MOSFET der Größe 50 * 15 µm.
Es ist dabei zu beachten, daß die Messung im geraden Teil der Steuerkennlinie erfolgt, da hier die Beweglichkeitsreduktion noch keine Rolle spielt. Diese Reduktion der Ladungsträger-Beweglichkeit ist an einem Abknicken der Steuerkennlinie bei höheren Gate-Source-Spannungen festzustellen.
Die Bestimmung der Unterdiffusion DELTA L erfolgt aus dem Vergleich des Widerstandes $R_{DSon}$ an zwei Transistoren mit unterschiedlicher Gatelänge aber gleicher Gatebreite, verwendet wurden $(W/L)_1$ = 50/15 µm und $(W/L)_2$ = 50/3,9 µm.
Aus

$$R_{DSon} = \frac{U_{DS}}{I_D} = \frac{L}{B_0 * (U_{GS} - U_T) * W} \qquad (2)$$

wird für diese Transistoren der Wert von $R_{DSon}$ bei jeweils gleicher Gate-Source-Spannung ermittelt und in einem Diagramm über $L_{nom}$ aufgetragen, siehe Bild 7. [4]

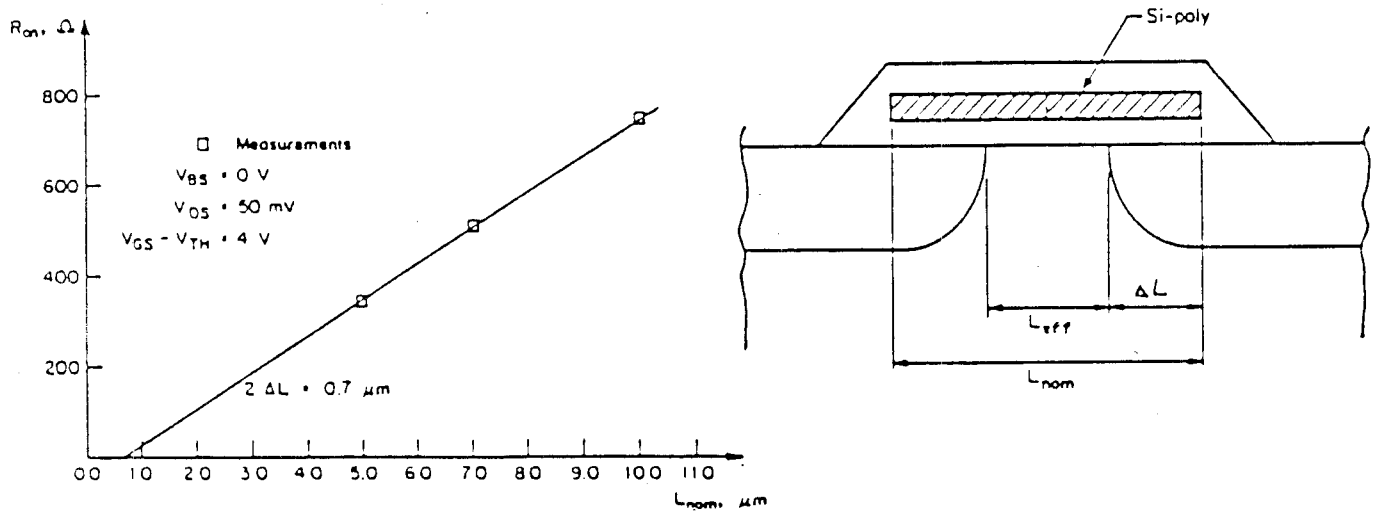



Bild 7: Bestimmung der Unterdiffusion DELTA L

Für [illegible] mußte die entstehende Gerade durch den Ursprung gehen, man sieht in Bild [illegible] daß dies nicht der Fall ist.
Die Abweichung entsteht durch DELTA L, wobei der Schnittpunkt der Geraden mit der Abszisse dem Wert 2 * DELTA L entspricht, siehe hierzu ebenfalls Bild 7.
Die Unterdiffusion betrifft natürlich die Kanalbreite des MOSFET gleichermaßen, dem wurde bei der Auswahl der Transistoren Rechnung getragen, indem eine möglichst große Kanalbreite (50 µm in beiden Fällen) gewählt wurde, der Fehler ist damit vernachlässigbar klein.
Bei unseren Messungen ergab sich eine Unterdiffusion von DELTA L ≈ 0,15 µm; bei 50 µm Gatebreite ein Fehler von 0,5 %, während sich bei $L_{nom}$ = 3,9 µm ein Fehler von rund 8 % ergibt.

Ein weiterer wichtiger Parameter ist die sogenannte Substratempfindlichkeit, d. h. die Abhängigkeit der Einsatzspannung eines MOSFET von der Spannung zwischen Source und Substrat (Bulk) des betreffenden Transistors. Dieser Zusammenhang wird beschrieben durch [5]

$$U_T = U_{T0} + k_1 \left( \sqrt{U_{SB} + 2\varphi_F} - \sqrt{2\varphi_F} \right) \qquad (3)$$

mit

| | |
|---|---|
| $U_{T0}$ | Schwellspannung bei $U_{SB}$ = 0 V |
| $U_{SB}$ | Spannung zwischen Source und Substrat (Bulk) |
| $k_1$ | Substrateffektkonstante |
| $2\varphi_F$ | Oberflächeninversionspotential, |

die Substrateffektkonstante $k_1$ ist abhängig von der Oxidschichtdicke $t_{OX}$ und der Störstellenkonzentration $N_B$ [3]:

$$k_1 = \frac{t_{OX}}{\varepsilon_{OX}} * \sqrt{2e * \varepsilon_{Si} * N_B} \qquad (4)$$

wobei

| | |
|---|---|
| $\varepsilon_{OX}, \varepsilon_{Si}$ | relative Dielektrizitätszahl |
| | $\varepsilon_{OX} = \varepsilon_{SiO2} = 3,9$ |
| | $\varepsilon_{Si} = 11,7$ |
| e | Elementarladung |
| $N_B$ | Störstellenkonzentration im Substrat |

Für den Betrieb von NMOS-Transistoren müssen die Source-Inseln gegenüber dem Substrat isoliert sein. Dies geschieht durch eine Sperrschicht-Isolation, also eine negative Substratspannung gegenüber der Source-Insel (für p-Substrat). Hierfür wird eine zusätzliche, negative Versorgungsspannung benötigt, gleichzeitig wird auch eine erhöhte Störsicherheit erreicht. Um mit nur einer Versorgungsleitung auszukommen, wird in dynamischen Speichern üblicherweise ein sogenannter Substrat-Vorspannungsgenerator eingesetzt, der bereits mit auf dem Chip integriert ist. Der Einfluß dieser Vorspannung auf die Einsatzspannung der Transistoren ist in Bild 8 dargestellt.

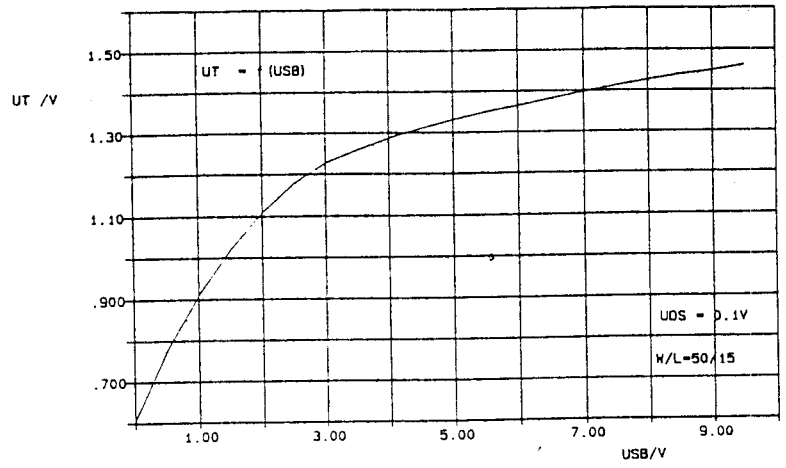



Bild 8: Substratempfindlichkeit

Zur anschaulichen Darstellung der Gleichmäßigkeit eines Fertigungsprozesses über der Fläche eines Wafers dient eine Darstellung nach Bild 9.

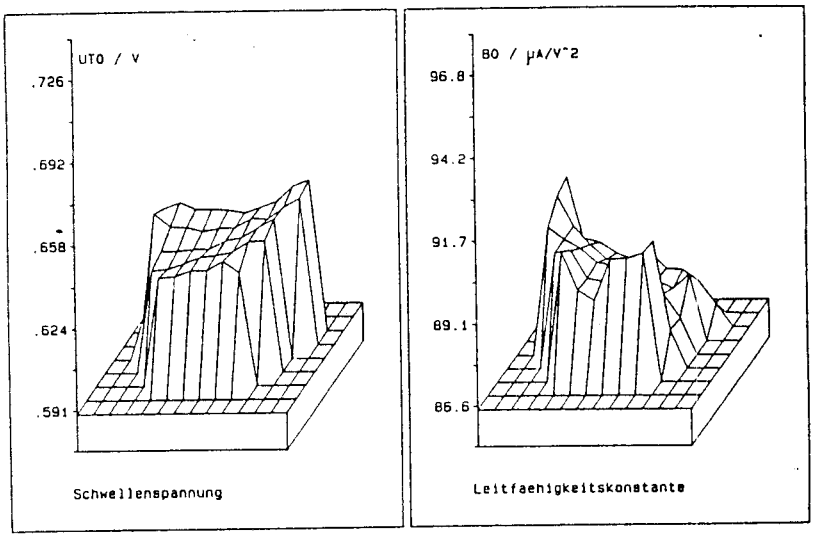



Bild 9: 3d - Darstellung von Einsatzspannung und Leitfähigkeitskonstante

Für diese Darstellung wurden über die Waferfläche verteilt jeweils ein Transistor mit gleichem W/L - Verhältnis pro Schnittfuge gemessen und die Einsatzspannung $U_{T0}$ bzw. die Leitfähigkeitskonstante $B_0$ errechnet und in ein 3d-Diagramm über der Waferfläche eingetragen. Daraus lassen sich anschaulich eventuelle Abweichungen eines Prozesses erkennen, z. B. könnte eine Belichtung durch eine schräg justierte Maske eine deutliche Parameter-Abweichung an einer Seite des Wafers ergeben.
Zu beachten ist, daß bei dieser Darstellung der Nullpunkt der jeweiligen Parameter-Achse unterdrückt wurde, die gemessenen Abweichungen sind in der Realität sehr gering!

## 3. Das Berufsfeld des Ingenieurs in der Halbleiterentwicklung

Friedrich Schmidtpott

ITT INTERMETALL, Leiter des VLSI Design-Zentrums

Als ich mich 1976 um eine Anstellung in einer Entwicklungsabteilung für integrierte Schaltungen bewarb, bekam ich die Stelle, weil in meinem Studienschwerpunkt Informationsverarbeitung mehrere Vorlesungen die Digitaltechnik zum Inhalt hatten. Von der Schaltungstechnik integrierter Schaltungen oder von der zugehörigen Prozeßtechnologie kam in meinem Studium nichts vor. Gerne hätte die Firma damals Ingenieure mit entsprechenden Kenntnissen eingestellt, aber eben solche waren rar. Es gab nur wenig Hochschulen, die in dieser recht neuen Fachrichtung Schwerpunkte setzten.

Dies ist jedoch eine typische Situation, in die eine Ingenieurwissenschaft gerät, wenn neue Technologien eine so rasante Entwicklung nehmen. Wie haben die Ingenieure und die Hoch- und Fachhochschulen diese Situation bewältigt?

Kommen wir auf die ersten Schaltungsentwickler für integrierte Schaltungen in den siebziger Jahren zurück. Dies waren zum großen Teil Ingenieure, die mit der Entwicklung von Schaltungen für die Konsum- und Industrieelektronik mit Transistoren, Operationsverstärkern und TTL-Bausteinen, Grundwissen über diese neuen Komponenten erlangten.

Die Ingenieure haben die Situation durch ihre Bereitschaft, neue Herausforderungen anzunehmen, gemeistert. Hinzu kommt nicht selten die Faszination über die technischen Möglichkeiten. Unbedingt herauszufinden, was mit einer neuen Technologie alles machbar ist, kennzeichnet den engagierten Ingenieur.

Nun aber zur Rolle der Hochschulen während dieser technisch innovativen Phasen. Hier ist wichtig, daß neben der Grundlagenvermittlung im ersten Teil des Studiums, im zweiten Teil Schwerpunkte gesetzt werden, die sich mindestens am "Stand der Technik" orientieren. In Verbindung mit den später in den Entwicklungsabteilungen gemachten Erfahrungen bietet der Ingenieur gute Voraussetzungen, auch neue technische Erkenntnisse in entsprechende Entwicklungen, sprich Produkte, umzusetzen. Denn in vielen Fällen, und so auch bei den Fortschritten der integrierten Schaltungen, baut eine neue Entwicklung auf der vorherigen auf.

Das bedeutet: wenn in der Ausbildung die elementaren Grundlagen gut vermittelt werden, ist dies die beste Nahrung, die dem Studenten für späteres Hinzulernen mit auf den Weg gegeben werden kann. Daß dieses Grundlagenwissen einem ständigen Anpassungsprozeß unterworfen werden muß, ergibt sich schon aus der Tatsache, daß die Studienzeiten nicht länger werden dürfen.

Meiner Meinung nach haben die Hoch- und Fachhochschulen diese Aufgabe gut bewältigt. Die Studenten, die heute als Ingenieure die Hochschulen verlassen, sind gerüstet, auch weitere Entwicklungsstufen in Angriff zu nehmen und mit neuem Inhalt zu füllen.

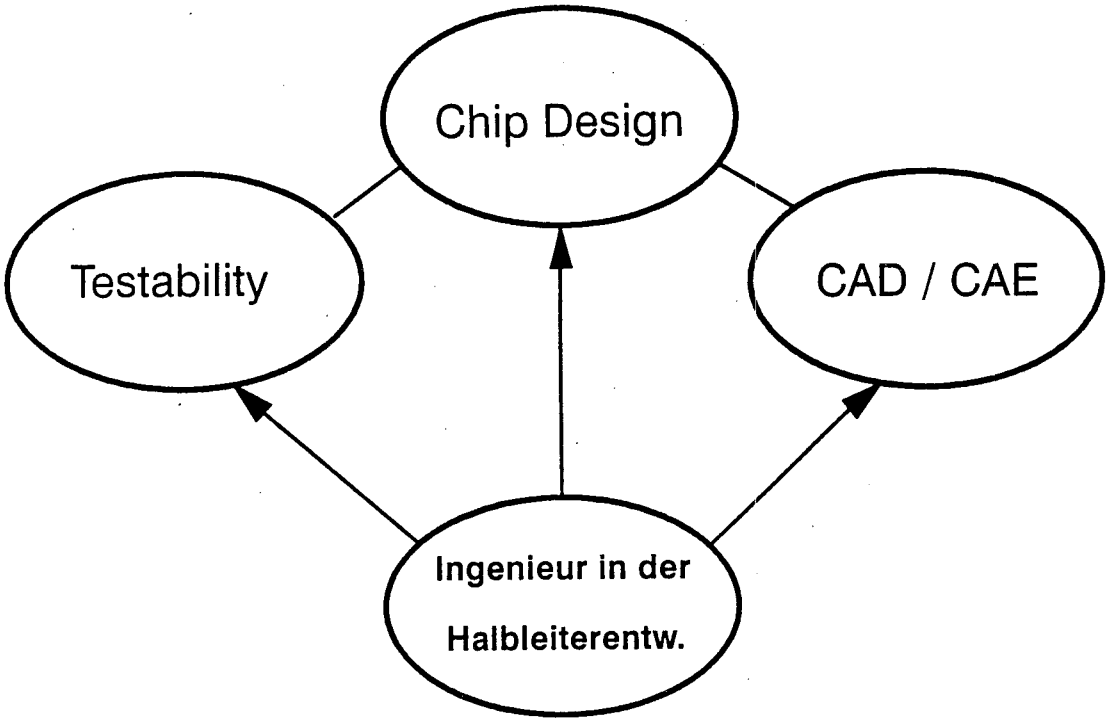
# AUSBILDUNGSRICHTUNGEN

- **Elektrotechnik**
  - Mikroelektronik
  - Nachrichtentechnik
  - Informationsverarbeitung
- **Physik**
  - Festkörperphysik
- **Informatik**
  - softwareorientiert
  - hardwareorientiert

ITT Intermetall

Der größte Teil der Schaltungsentwickler für integrierte Schaltungen kommt aus dem Studiengang "Allgemeine Elektrotechnik". Über die Beschäftigung mit der Prozeßtechnologie finden auch die Physiker den Weg zum Schaltungsentwurf. Sie sind bevorzugt in den Bereichen aktiv, die umfangreiche Technologiekenntnisse erfordern. Dazu zählen z.B. das D-RAM-Design und die Entwicklung von Analogschaltungen. Wegen der zunehmenden Softwareabhängigkeit, sowie der Realisierung verschiedener Rechnerarchitekturen auf Silizium, sind auch Ingenieure der Informatik in der Entwicklung vertreten.

# ARBEITSBEREICHE – HALBLEITERENTWICKLUNG




ITT Intermetall

Während noch vor etwa fünfzehn Jahren eine Ingenieurstätigkeit im Bereich der Halbleiterentwicklung eine sehr schmale Spezialisierung innerhalb der Elektrotechnik darstellte, existiert heute ein breites Arbeitsfeld in diesem Bereich. Zum Chip-Design sind die wichtigen Arbeitsbereiche CAD/CAE und Testability hinzugekommen. Die typischen Aufgaben dieser Bereiche werden nachfolgend genauer erklärt.

Die einzelnen Tätigkeiten dieser drei Arbeitsbereiche müssen nicht zwingend von verschiedenen Ingenieuren wahrgenommen werden. Je nach Größe und Organisationsstruktur der Entwicklungsabteilung ist auch eine andere Verteilung der Aufgaben denkbar. Dies ist durch die Verbindungen zwischen den Arbeitsbereichen dargestellt.

# CAD / CAE

**AUFGABE: Bereitstellung der Designwerkzeuge für jede Entwicklungsphase**

- **Betreuung und Wartung aller Software-Tools**
  - Freigabe neuer Software Versionen
  - Bug Report
- **Customize-Software entwickeln**
  - Interface Programme zur Hard- und Software entwickeln
  - Benutzeroberfläche anwenderfreundlich gestalten
- **Technologie-Files anlegen**
- **Modellentwicklung für Systemsimulation**
  - Funktionsbeschreibung von Blöcken
  - Hochsprachenmodelle
- **Modellentwicklung für Transistoren zur Circuit- Device- und Prozeßsimulation**
- **Parameterextraktion**
- **Leistungsfähigere Software-Tools zur Verfügung stellen**
  - Anbietermarkt beobachten
  - neue Programme (Programmpakete) bewerten
  - Anforderungen gegenüber Softwareanbietern definieren

---

ITT Intermetall

In den siebziger Jahren war der Schaltungsentwickler oft auch derjenige, der die wenigen, zur Designunterstützung zur Verfügung stehenden Programme wartete. Schaltbildeingabe und Layoutumsetzung erfolgten mittels Papier und Bleistift. Heute ist der Arbeitsplatzrechner - die Workstation - für jeden Entwicklungsingenieur selbstverständlich. Dort laufen Programme zur Schaltbildeingabe, zum Simulieren von Analog- und Digitalschaltungen, zur Layouterstellung, zur Layoutverifikation und vieles mehr. Alle anfallenden Arbeiten in der Softwareumgebung der Halbleiterentwicklung sind in dem Arbeitsbereich CAD/CAE zusammengefaßt.

# TESTABILITY

- **Teststrategie definieren**
  - Selbsttest
  - Testbus
  - Boundary scan
- **Testabdeckung sicherstellen**
  - Testpattern generieren
  - Fehlersimulation
- **Pin-zu-Pin Simulation**
  - Funktionsmodelle schreiben
- **Postprozessing für Produktions- und Entwicklungstester**
- **Support bei der Installation des Produktionstests**
- **Anforderungen an Testequipment formulieren**
  - Testfrequenz, Pinzahl, Speichertiefe
  - Mixed Signal-Tester
- **Effektivere Testmethoden erarbeiten**
- **Bewertung neuer Simulations- und Testsoftware**

ITT Intermetall

Mit der Zunahme der Komlexität der integrierten Schaltungen hat sich im Berufsfeld der Halbleiterentwicklung ein weiterer Arbeitsbereich etabliert, die Testability.

Die Qualität der Bauelemente ist heute gekennzeichnet durch Ausfallraten, die nur noch im ppm-Bereich liegen. Das erfordert die 100%ige Testabdeckung einer integrierten Schaltung. Jeder Transistor, jede Verbindung auf dem Chip, muß auf richtige Funktion getestet werden. Aus Kostengründen ist die Testzeit kurz zu halten. Beides ist nur zu erreichen,

wenn schon während der Systementwicklung entsprechende Teststrategien berücksichtigt werden, und in die Schaltungsentwicklung die notwendigen Testschaltungen einfließen.

Im Bereich Teststrategie muß, in Abhängigkeit von verwendeter Schaltungstechnik und Produktionsvolumen, der richtige Kompromiß zwischen Schaltungszusätzen auf Silizium oder Testaufwand in der Produktion gefunden werden.

Der Testability-Ingenieur führt parallel zum Schaltungsentwickler weitere Simulationen durch und erstellt die später im Produktionstest benötigten Testpattern. Dies bringt zwei Vorteile mit sich. Zum einen sorgt der Tetability-Ingenieur mit seinem Know-How für die optimale Abwicklung der Testproblematik, zum anderen verkürzt das parallele Arbeiten die Entwicklungszeit.

# CHIP DESIGN

- **AUFGABE: Aus einer Systemspezifikation und einer definierten Technologie eine integrierte Schaltung entwickeln und auslegen**

- **ASIC Designer**
  - Gate-Array
  - Standard-Cell
  - Full-Custom

- **Memory Designer**
  - Dynamische RAMs
  - Statische RAMs
  - nichtflüchtige Speicher

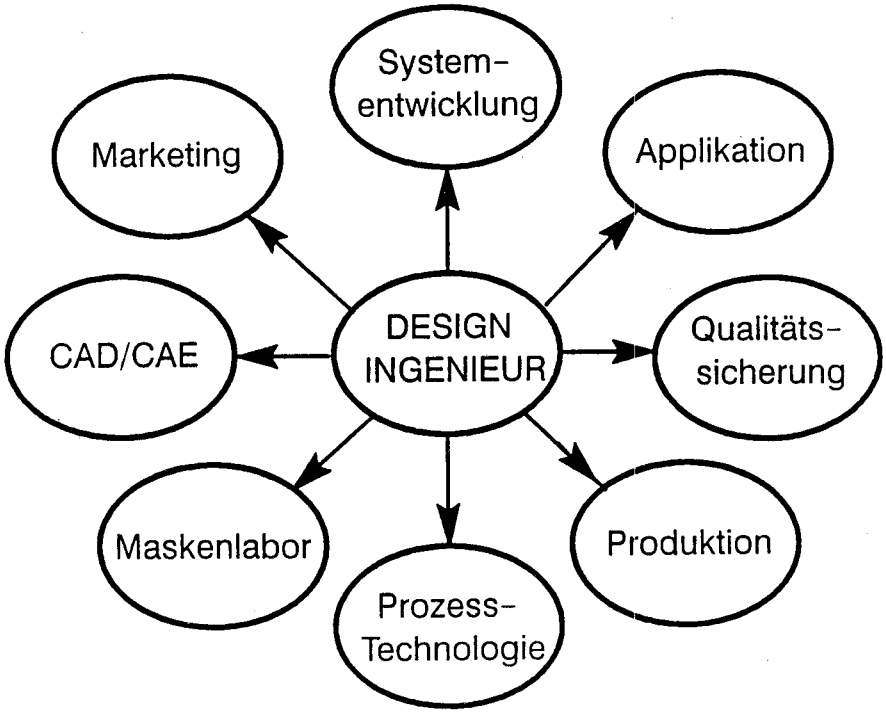
- **Prozessor Designer**
  - Mikroprozessoren und Controller
  - Signalprozessoren (Standard und Kundenorientiert)

- **Analog Designer**
  - CMOS Technologien
  - Bipolar Technologien
  - GaAs Technologien

- **Layout Designer**
  - Umsetzen der Schaltpläne in Maskenlayouts

Im Bereich Chip-Design arbeitet der größte Teil der Entwicklungsingenieure. In Abhängigkeit der zu lösenden Aufgabe, bietet sich dem Design-Ingenieur ein breites Arbeitsfeld. Er entwirft Schaltungen für verschiedene Technologien, in unterschiedlicher Schaltungstechnik. Die Bandbreite reicht vom fast vollautomatisierten Gate-Array-Design bis hin zum Analog-Design auf Transistorebene. Jedes Entwicklungsgebiet erfordert spezielle Methoden mit entsprechenden Kenntnissen, die jeder Entwicklungsingenieur erlernen muß.

## ARBEITSUMGEBUNG DES DESIGN-INGENIEURS

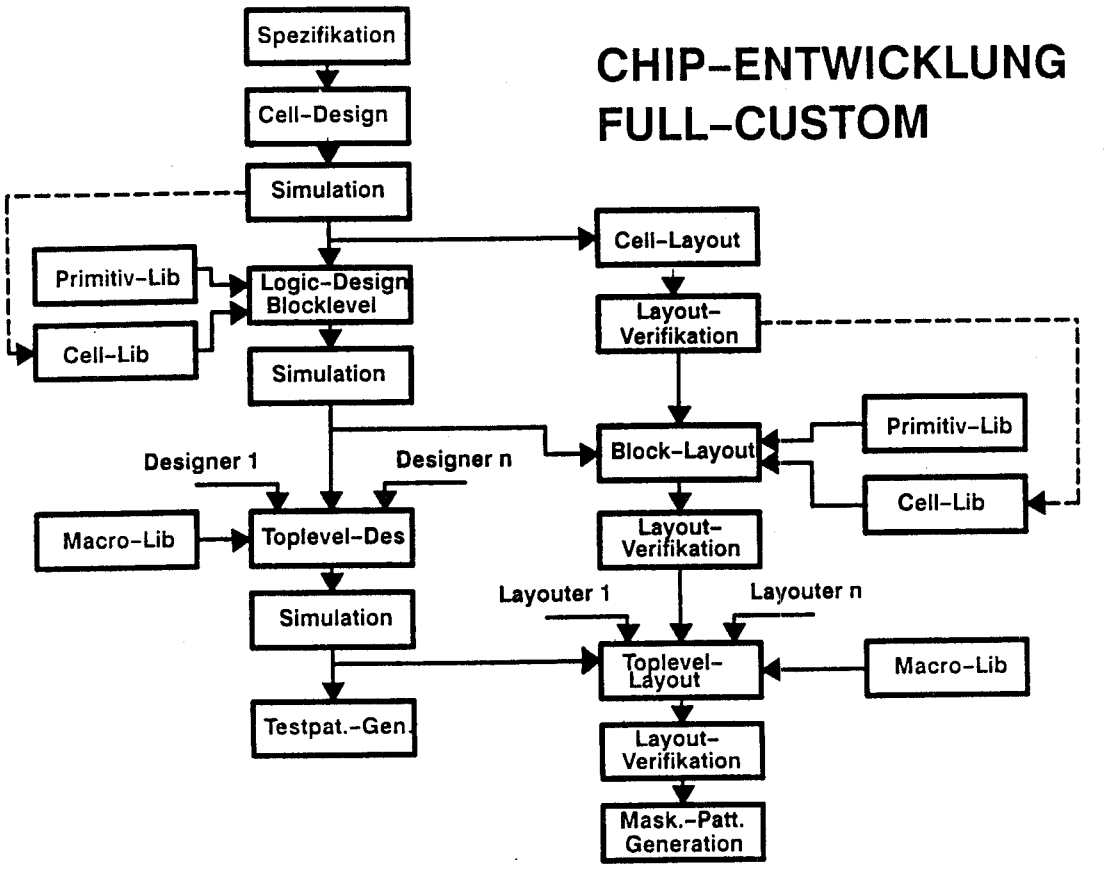



ITT Intermetall

Im Anschluß an die Beschreibung des Arbeitsbereiches eines Chip-Designers, möchte ich kurz auf jene Abteilungen eingehen, mit denen der Design-Ingenieur während der IC-Entwicklung früher oder später Kontakt bekommt.

Als ein Beispiel sollen hier die elektrischen Design-Rules genannt werden. Sie charakterisieren die verwendete Prozeßtechnologie, womit sie dem Design-Ingenieur als Grundlage für die Auslegung seiner Schaltung dienen. Hier ist denkbar, daß ein für die Schaltung kritischer Parameter gar nicht, oder ohne Toleranzen spezifiziert ist. Vielleicht kann der Prozeßentwickler den Sachverhalt sofort klären, oder er muß erst an Teststrukturen Messungen durchführen. Diese Zusammenarbeit trägt dazu bei, Redesignkosten und Zeitverlust zu vermeiden.

Für jede, in dem Bild dargestellte Abteilung ließe sich ein treffendes Beispiel finden, das die Wichtigkeit der Zusammenarbeit belegt. Denn das Ziel, in kürzester Zeit, ein im Sinne des Auftraggebers optimales Produkt zu entwickeln, haben alle, direkt oder indirekt an einer IC-Entwicklung beteiligten Mitarbeiter.




ITT Intermetall

Die typische Vorgehensweise bei der IC-Entwicklung findet man in der Literatur in ähnlicher Weise dargestellt, wie hier in diesem Bild. Die Besonderheit jedoch, liegt hier in einer stark parallelen Arbeitsweise im Schaltungsentwurf und in der Maskenauslegung, wie sie von der Firma INTERMETALL seit Jahren erfolgreich angewandt wird. Der Grund dafür ergibt sich aus der Tatsache, daß die Zeit von der Spezifikation eines Produkts bis zur Markteinführung so kurz wie möglich sein muß.

Nur so kann, speziell im Konsumer-Markt mit der relativ kurzen Einsatzdauer einer integrierten Schaltung, die Konkurrenzfähigkeit erreicht werden. Denn den Vorteilen einer Full-Custom-Entwicklung, die sich gegenüber Gate-Array und Standard-Cell-Verfahren, durch höhere Packungsdichte, höhere Geschwindigkeit und flexiblere Systemlösung darstellen, steht als Nachteil der größere Entwicklungsaufwand gegenüber. Dies darf aber nicht zu verlängerten Entwicklungszeiten führen.

Auf der linken Seite des Bildes ist der Schaltungsentwurf, auf der rechten Seite der Layoutablauf dargestellt. Es stehen drei Libraries zur Verfügung; die Primitiv-, die Cell- und die Macro-Library. Die Primitiv-Library enthält verschiedene logische Verknüpfungen wie NAND, NOR und Komplex-Gatter.

Die Cell-Library setzt sich zusammen aus Schieberegisterzellen, aus Latches, aus Halb- und Volladdiererzellen und ähnliches mehr. Wird für eine spezielle Entwicklung eine Schaltung benötigt, die in der Cell-Library nicht vorhanden ist, aber zu einer erheblichen Flächeneinsparung beitragen kann, wird diese entwickelt und nach der Simulation in die Cell-Library aufgenommen.

In der Macro-Library befinden sich ROMs, RAMs, Multiplizierer, Prozessor-Cores und weitere Funktionseinheiten, die häufiger in verschiedenen integrierten Schaltungen zur Anwendung kommen.

Die Entwicklung bis zum Blocklevel kann je nach Umfang des Projekts von mehreren Designern parallel durchgeführt werden. Dazu wird ein Projekt in sinnvolle Funktionseinheiten partitioniert. Im Toplevel-Design werden dann die einzelnen Entwicklungen des Blocklevel-Designs mit der noch fehlenden Peripherie zusammengefaßt. Den Abschluß der Schaltungsentwicklung bildet dann die Gesamtsimulation, deren Ergebnisse mit den Vorgaben der Spezifikation auf Übereinstimmung geprüft werden.

Auf der Layoutseite wird ähnlich wie im Schaltungsentwurf verfahren. Schon nach Fertigstellung der ersten Schaltungen wird mit dem Layout begonnen. Auch hier gibt es einen erheblichen Zeitgewinn durch paralleles Arbeiten. Schon kurze Zeit nach Abschluß der Schaltungsentwicklung liegt das fertige Chip-Layout, aus dem dann die Pattern für das Schreiben der Masken generiert werden, vor.

# DIGITALSCHALTUNGEN

- **Rechenwerke**
  - Addierer
  - Multiplizierer
  - Dividierer
  - Filter
  - ALU
- **Speicher**
  - RAM
  - ROM
  - EE-PROM
  - FIFO
  - Delay line
- **Steuerwerke**
  - Finite State Machine
  - Zähler mit Dekodierungen
- **Prozessoren**
  - 65C02 inkl. Peripherie
  - Fast Prozessor

ITT Intermetall

Diese Bild zeigt die wichtigsten Digitalschaltungen, die der Design-Ingenieur zur Entwicklung von integrierten Schaltungen für unterschiedliche Anwendungen benötigt. Diese Elemente sind vorwiegend in der Macro-Library einer Technologie enthalten. Hier können sie zur optimalen Lösung einer Aufgabe schnell angepaßt werden. Dies ist für die Bitbreiten von Addierern und Multiplizierern, sowie für die Größe und Organisation von Speichern besonders wichtig.

Hinweisen möchte ich noch auf den Punkt Prozessoren. Mit dem Core des 65C02 lassen sich durch Hinzufügen von anwenderspezifischer Peripherie sehr schnell leistungsfähige Controller-Bausteine entwickeln. Ferner steht für komplexe Entwicklungen mit einem hohen Anteil von Steuerungsaufgaben ein "Fast Prozessor" mit einer 12 bit ALU und RISC-Architektur zur Verfügung. Der Prozessor arbeitet in einer 1.2$\mu$ CMOS-Technologie mit einer Taktfrequenz von 40 MHz, RAM und ROM sind in der Speicherkapazität adaptierbar. Damit entfällt zum einen das langwierige Design von individueller Randomlogik, zum anderen erhält der Anwender durch ein maskenprogrammierbares ROM eine kostengünstige und flexible Systemlösung.

# ANALOGSCHALTUNGEN

- **A/D-Umsetzer**
  - Instrumentation (8 - 17 Bit Auflösung, SAR, Dual-Slope)
  - Video (7 - 9 Bit Auflösung, Two-Step-Flash-Converter)
  - Audio (14 - 15 effektive Bits, Sigma-Delta-Converter)

- **D/A-Umsetzer**
  - Instrumentation (R2R-Netzwerk)
  - Video (9 - 11 Bit, Kombination aus gewichteten und ungewichteten Stromquellen)
  - Audio (ca. 16 Bit, Sigma-Delta-Converter)

- **Oszillatoren**
  - RC-Oszillatoren
  - Quartz-Oszillatoren
  - PLL-Taktgeneratoren
  - abstimmbare Oszillatoren (VCO /DCO)

- **lineare Schaltungen**
  - OPAMPS
  - Verstärker mit wohldefinierter Verstärkung
  - aktive Filter

- **analoge Funktionsnetzwerke**
  - Multiplizierer
  - Logarithmierer
  - Begrenzer

ITT Intermetall

Das Bild Analogschaltungen zeigt die wesentlichen Entwicklungsaufgaben mit denen der Analog-Designer konfrontiert wird. Die Realisierungsmöglichkeiten sind stark technologieabhängig. Im ASIC-Bereich fällt der Anwendung von analogen und digitalen Schaltungen auf einem Kristall besondere Bedeutung zu. Deshalb dominiert hier die CMOS-Technologie. Zum Einsatz kommen A/D- und D/A-Umsetzer, die als analoges Front-und Backend benötigt werden.

Einfache Oszillatorschaltungen lassen sich in allen Technologien realisieren. Analoge Funktionsnetzwerke, also Schaltungen, welche die Form einer Kennlinie ausnutzen, sind eine typische Domäne der Bipolar-Technologien.

Zum Schluß noch ein paar Worte zur Ausbildung und zu den Anforderungen in der Praxis.

Aus der Sicht der Entwicklungsabteilungen sollte der Studienabgänger, der eine Tätigkeit in der Halbleiterentwicklung anstrebt, Kenntnisse über den physikalischen Aufbau der Transistoren, über Kennlinienfelder, über die wichtigsten Parameter, über Grundschaltungen in den verschiedenen Technologien, sowie über den Ablauf einer Chip-Entwicklung haben.

Ich will nicht bestreiten, daß auch ein Studienabgänger ohne Ausrichtung des Studiums auf die Mikroelektronik in der Lage ist, - bei entsprechendem Engagement - schon nach kurzer Zeit hier gleichwertige Arbeit zu leisten. Der Bewerber mit den Kenntnissen jedoch hat bereits einen Überblick über sein zukünftiges Arbeitsfeld, und man kann von einer gewissen Motivation des Bewerbers ausgehen, in diesem Bereich zu arbeiten. Dadurch ist das Risiko, eine falsche Wahl getroffen zu haben für beide Seiten geringer.

An den Bewerbungen von Studienabgängern mit dem Schwerpunkt Mikroelektronik ist auffällig, daß ca. 95% Schwerpunkte im Bereich Software und Digitaltechnik setzen. Es gibt meines Erachtens zu wenig Interessenten für die analoge Schaltungsentwicklung. Diese jedoch wird in den nächsten Jahren, gerade in Verbindung mit digitalen ASICS stark an Bedeutung gewinnen. Dank der Submikron-CMOS-Technologien werden auch schnelle Analogschaltungen machbar. Außerdem bietet sich damit die Möglichkeit an, auch komplexe Systemintegrationen durchzuführen, die dann an den IC- Schnittstellen analoge Signale haben. Diese Tendenz gilt für Standard Cell-Design ebenso, wie für Full Custom.

Analoges Design erfordert tieferes Wissen über elektrisches Verhalten und parasitäre Effekte des Transistors. Weiterhin sind die vielen speziellen Schaltungstricks, die zum stabilen Funktionieren einer Analogschaltung notwendig sind, nur durch Erfahrung erlernbar. In der Praxis gibt es auch eine Spezialisierung. Ein digitaler Schaltungsentwickler kann in der Regel nicht ohne entsprechende Einarbeitung einen A/D-Converter nach dem Sigma-Delta Prinzip entwerfen.

Grundsätzlich müssen in einer Entwicklungsabteilung eines Unternehmens "Produkte" entwickelt werden. Wie ich am Beispiel der Fullcustom-Chipentwicklung bereits erwähnt habe, entscheidet über den Erfolg eines Produkts am Markt sehr oft die Zeit, die eine Entwicklung von der Idee bis zur Serienreife benötigt. Also "Time to market". Um diese Zeit kurz zu halten, ist eine effiziente Zusammenarbeit aller an einem größeren Entwicklungsprojekt Beteiligten besonders wichtig. Nur mit dem kritisch hinterfragenden und kommunikativen Ingenieur ist dieses Ziel zu erreichen.

Es hat sich auch gezeigt, daß beim Umsetzen einer Spezifikation in eine integrierte Schaltung Systemfehler vermieden, bzw. oft bessere Systemlösungen gefunden werden, wenn der Designingenieur das zugrunde liegende Konzept gut verstanden hat und dafür auch Interesse entwickelt. Hier ist wieder der offene, über seinen Bereich hinausschauende, Ingenieur gefragt.

Oft leistet sich eine Entwicklung auch einen kleinen Bereich, in dem gewissermaßen "geforscht" wird, in dem nicht unbedingt produktreife Entwicklungen verfolgt werden. Hier ist durchaus der stärker theoretisch orientierte Tüftler und Einzelgänger am rechten Platz.

Abschließend möchte ich kurz auf die Frage eingehen, welchen Ingenieur die Entwicklung benötigt, den Hochschul- oder den Fachhochschulingenieur?" Ich meine, die Entwicklungsabteilungen benötigen sie beide. Der Hochschulingenieur mit seinem umfangreichen theoretischen Wissen und der stärker praktisch orientierte Fachhochschulingenieur ergänzen sich in der Entwicklungsarbeit hervorragend. Denn die Entwicklung einer integrierten Schaltung, von der Spezifikation bis zum Produkt, erfordert, wie ich zuvor schon aufgezeigt habe, nicht nur den Ingenieur mit dem theoretischen Denkansatz.

Der Erfolg eines Projekts hängt auch davon ab, wie gut die Arbeit koordiniert, organisiert und auch konzentriert wird. Hier zeigt die Erfahrung, daß in vielen Fällen schon nach kurzer Zeit kein Unterschied in der Arbeitsleistung erkennbar ist; auch nicht im Gehalt.

Voraussetzung dafür ist aber der Einzug der Mikroelektronik in die Studieninhalte der Fachhochschulen. Aus diesem Grunde begrüßen die Halbleiterfirmen die Einrichtung des ASIC-Labors auch in der Fachhochschule Offenburg.

# 4. Logic Cell Array (LCA) Entwicklungen für einen GPS - Empfänger

Hans Fiesel

Fachhochschule Offenburg

**Im Rahmen eines GPS-Projektes ist an der Fachhochschule Offenburg ein Konzept für einen experimentellen Navigationsempfänger entstanden. Hierfür wurde der digitale Teil entwickelt und aufgebaut. Für die Realisierung der Schaltung sollten benutzerprogrammierbare Gate Arrays von Xilinx (LCAs) verwendet werden, die sich schon bei einer anderen Arbeit an der Fachhochschule bewährt hatten.**
**Nachfolgend möchte ich dem Leser einen Überblick über das GPS-System und die Entwicklung der LCAs geben.**

## Einführung

Das GPS-System ist ein satellitengestütztes Navigationssystem, das ursprünglich für militärische Anwendungen entwickelt wurde.
Nach seiner Komplettierung werden 18 aktive und 3 Reservesatelliten um die Erde kreisen. Die Umlaufbahnen sind so gewählt, daß zu jedem Zeitpunkt und an jedem Ort der Erde mindestens 4 Satelliten empfangen werden können.
Zu einer dreidimensionalen Ortsbestimmung benötigt man die Entfernung dieses Ortes zu drei Satelliten und deren Positionen. Als Grundlage für die Entfernungsmessung werden die Laufzeiten der Satellitensignale herangezogen. Diese bestimmt man, indem der Zeitpunkt des Aussendens eines Signalereignisses mit dem Empfangszeitpunkt verglichen wird. Weil die Satellitenzeit mit der Zeit des Empfängers nicht genau übereinstimmmen wird, erhält man einen Meßfehler, der dem Produkt aus der Uhrzeitdifferenz und der Lichtgeschwindigkeit entspricht. Da es zu teuer wäre auch jeden Empfänger, wie schon die Satelliten, mit einer hochgenauen Atomuhr auszurüsten, mißt man zusätzlich die Entfernung zu einem 4. Satelliten. So erhält man ein System von 4 Gleichungen, aus denen man den dreidimensionalen Ort und die genaue Zeit bestimmen kann.

## Das Satellitensignal

Alle Satelliten senden ihre Daten auf den gleichen beiden Frequenzen :

$$f_1 = 1575.42 \text{ MHz}$$
$$f_2 = 1227.60 \text{ MHz}$$

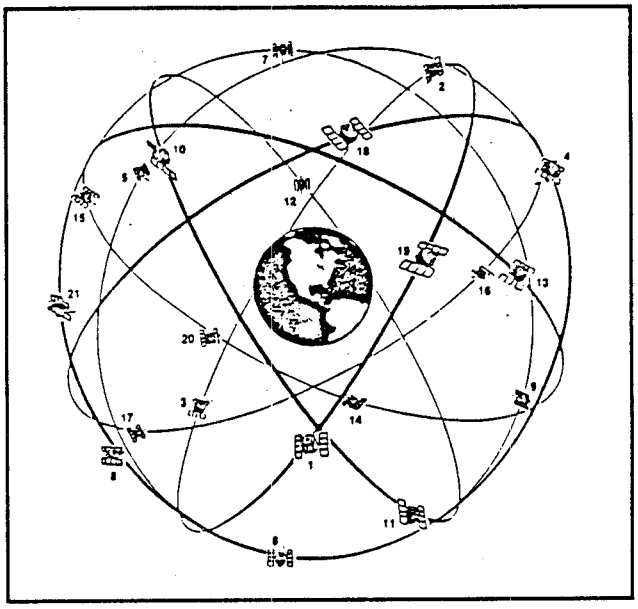


Bild 1 : Satellitenkonstellation :18 aktive, 3 Reservesatelliten; auf 6 verschiedenen Umlaufbahnen ; 20200 km über der Erde; 55° Inklination.

Zur Trennung der einzelnen Satellitensignale wird das Codemultiplex- (Spread Spectrum-) Verfahren angewandt. Die Phase des HF-Trägers wird hierbei nicht nur durch den digitalen Datenstrom moduliert, sondern auch durch eine digitale Codefolge, deren Bitfrequenz sehr viel höher ist. Dabei wird die Bandbreite des Nutzsignals auf die Bandbreite des Codes gespreizt. Das Empfangssignal muß mit dem gleichen, synchronen Code korreliert werden, um die Daten wieder auf die ursprüngliche Bandbreite zurückzusetzen.
Das GPS-System verwendet hierfür 2 verschiedene Codes; zum einen den C/A- (Coarse Aquisition) Code, der jedem Benutzer zur Verfügung steht und zum anderen den P- (Precise) Code, welcher militärischen Anwendungen vorbehalten bleibt.
Beide Codes werden mit den Daten durch eine Modulo-2-Addition (EXOR) verknüpft. Die Sendesignalerzeugung zeigt Bild 2.

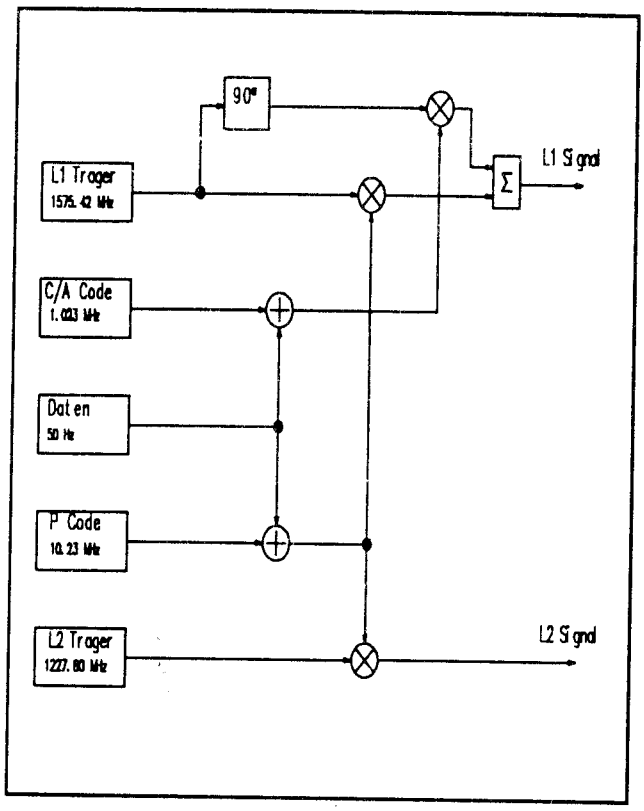



Bild 2 : Sendesignalerzeugung

## Verwendete Codes

## C/A Code

Der C/A-Code ist ein 1023 Bit Gold-Code, der aus der Modulo-2-Addition zweier 1023 Bit PRN Codes entsteht:

$$C/A_i(T) = G1(t) \ (+) \ G2(t + iT)$$

Die Folgen G1(t) und G2(t) werden von jeweils einem linear rückgekoppelten 10-Bit Schieberegister generiert.Die Generatorpolynome lauten:

$$G1 = X^{10} + X^3 + 1$$
$$G2 = X^{10} + X^9 + X^8 + X^6 + X^3 + X^2 + 1$$

Die Schieberegister werden mit 1,023 MHz getaktet und mit einem Synchronisierungssignal auf "alle 1" gesetzt (siehe Bild 3).
Zur Selektion benötigt jeder Satellit einen eigenen C/A-Code. Die verschiedenen C/A-Codes werden erzeugt, indem man die G2-Folge um eine ganzzahlige Anzahl von C/A-Takten verzögert. Dabei macht man sich die "Shift and Add" -Eigenschaft solcher Folgen zunutze, die besagt, daß die gleiche, verzögerte Codefolge entsteht, wenn man eine beliebige Anzahl von Parallelabgriffen des Schieberegisters miteinander EXOR-verknüpft.

Tabelle 1 zeigt die Zuordnung der Parallelabgriffe zu den entsprechenden Verzögerungen und den Codenummern.

| Satelliten-nummer | Signal-nummer | G2-Tab-Selektion | G2-Code Verzögerung |
|---|---|---|---|
| 1 | 1 | 2 (+) 6 | 5 |
| 2 | 2 | 3 (+) 7 | 6 |
| 3 | 3 | 4 (+) 8 | 7 |
| 4 | 4 | 5 (+) 9 | 8 |
| 5 | 5 | 1 (+) 9 | 17 |
| 6 | 6 | 2 (+) 10 | 18 |
| 7 | 7 | 1 (+) 8 | 139 |
| 8 | 8 | 2 (+) 9 | 140 |
| 9 | 9 | 3 (+) 10 | 141 |
| 10 | 10 | 2 (+) 3 | 251 |
| 11 | 11 | 3 (+) 4 | 252 |
| 12 | 12 | 5 (+) 6 | 254 |
| 13 | 13 | 6 (+) 7 | 255 |
| 14 | 14 | 7 (+) 8 | 256 |
| 15 | 15 | 8 (+) 9 | 257 |
| 16 | 16 | 9 (+) 10 | 258 |
| 17 | 17 | 1 (+) 4 | 469 |
| 18 | 18 | 2 (+) 5 | 470 |
| 19 | 19 | 3 (+) 6 | 471 |
| 20 | 20 | 4 (+) 7 | 472 |
| 21 | 21 | 5 (+) 8 | 473 |
| 22 | 22 | 6 (+) 9 | 474 |
| 23 | 23 | 1 (+) 3 | 509 |
| 24 | 24 | 4 (+) 6 | 512 |
| 25 | 25 | 5 (+) 7 | 513 |
| 26 | 26 | 6 (+) 8 | 514 |
| 27 | 27 | 7 (+) 9 | 515 |
| 28 | 28 | 8 (+) 10 | 516 |
| 29 | 29 | 1 (+) 6 | 859 |
| 30 | 30 | 2 (+) 7 | 860 |
| 31 | 31 | 3 (+) 8 | 861 |
| 32 | 32 | 4 (+) 9 | 862 |

Tabelle 1 : Wertetabelle der verschiedenen C/A-Codes

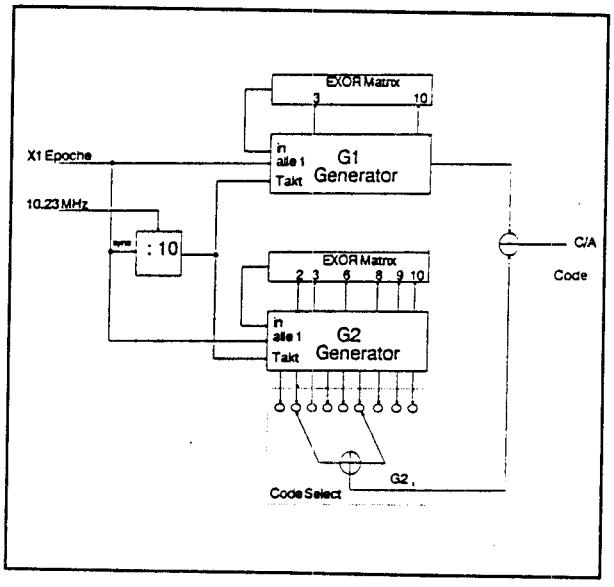



Bild 3 : C/A-Codeerzeugung

## Codesynchronisation

Das bei der Übertragung der Satellitensignale angewandte Codemultiplexverfahren verlangt, daß das Empfangssignal mit einem intern erzeugten Code korreliert wird, welcher dem Signal-Code entspricht und mit diesem in Phase ist. Das empfangene Signal wird also mit einem intern erzeugten Code multipliziert. Bei dieser Verknüpfung ergibt sich ein Signalanteil (Korrelationspegel) entsprechend der Verschiebung beider Codes zueinander. Den Korrelationspegel in Abhängigkeit von der Phasendifferenz der Codes gibt die Autokorrelationsfunktion an. Die angenäherte Autokorrelationskennlinie des C/A-Codes zeigt Bild 4 und 5.

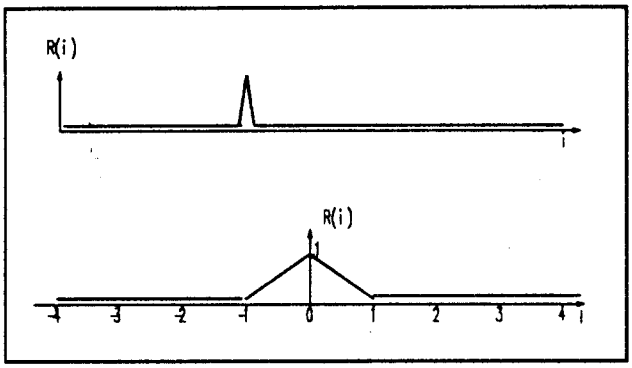



Bild 4 und 5 : Über die gesamte Codelänge ergibt sich nur bei der Phasendifferenz null eine Korrelationsspitze.

Zur Codesynchronisation wird der Empfängercode um +/- eine halbe Chiplänge verschobenen (early/late). Die jeweils entstehenden Korrelationssignale besitzen einen Anteil, welcher in einem engen Bereich der Codeverschiebung proportional ist (vgl. Autokorrelationskennlinie). Sind diese Anteile beider Signale gleich groß und größer als null, so ist der lokale Code mit dem Satellitencode in Phase.
In einem ersten Durchgang wird die Empfängercodephase in groben Schritten solange variiert bis ein Korrelatiossignal einen von Null verschiedenen Anteil besitzt (Aquisition). Danach wird die Differenz zwischen beiden Signalen auf null geregelt. Dafür wird bei dem Empfängerkonzept der FH Offenburg eine sogenannte Tau-Dither-Loop verwendet.

## Tau Dither Loop

Durch die zyklische Verschiebung des Codes (Dither) wird sich, sofern die Codephasendifferenz des Empfangscodes zu dem intern generierten Code kleiner gleich einer Taktlänge ist, ein der Autokorrelationskennlinie entsprechender Korrelationspegel einstellen (siehe Bild 7). Dieser Pegel wird nach dem Pegeldetektor PD entsprechend der Codephase mit +/- 1 bzw 0 bewertet. Ist der Mittelwert des Ausgangssignals dieser Bewertung gleich 0, dann sind die Pegel der vor- und der nacheilenden Codephase gleich groß und somit ist der Code in Phase.

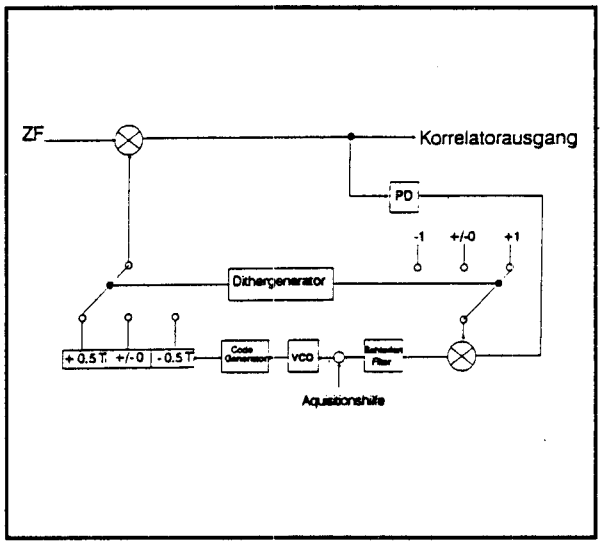



Bild 6 : Prinzipschaltbild Tau-Dither-Loop

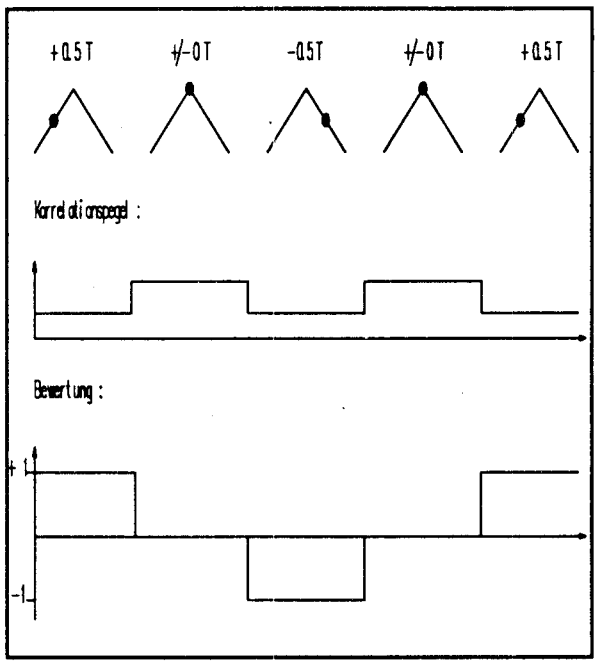



Bild 7 : Signale bei der Tau Dither Loop

## Das gesamte Empfängerkonzept

Das Empfangssignal wird zuerst bei der Antenne verstärkt, um die Leitungsdämpfung zu kompensieren. Nach der Umsetzung in die Zwischenfrequenz von 40,92 MHz wird das Signal mit dem Code korreliert und durch den VCXO auf eine 2. Zwischenfrequenz von 5,115 MHz umgesetzt. Eine Trägernachführungsschleife regelt dabei durch den spannungsgesteuerten Quarzoszillator die Dopplerverschiebung des Signals aus.

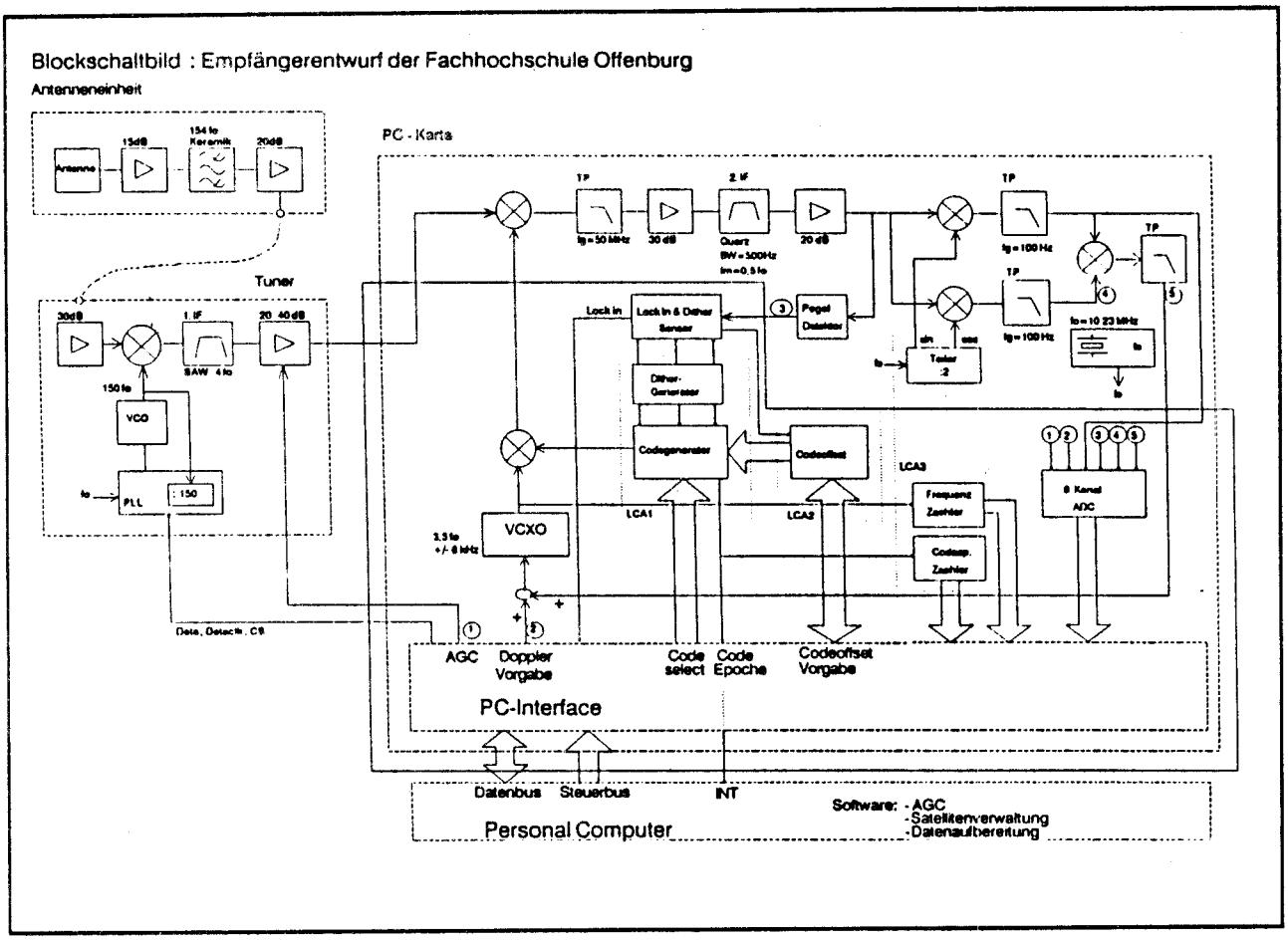



Bild 8 : Blockschaltbild des Empfängerkonzepts

Die Codekorrelation erfolgt im Zwischenfrequenzbereich. Dabei ist die C/A-Code-Nummer vom PC aus einstellbar. Die Codephase kann einerseits auch durch den PC vorgewählt und anderseits durch die Tau-Dither-Loop variiert werden.
Des weiteren besitzt der Entwurf einen Frequenzzähler zur Bestimmung der Dopplerverschiebung und einen Codeepochezähler, der als Zeitmaßstab verwendet werden soll.
Der Digitalteil dieses Empfängers ist mit Ausnahme des PC-Interfaces mit Hilfe von benutzerprogrammierbaren Gate Arrays verwirklicht worden. Insgesamt drei Bausteine enthalten den Codegenerator, einen Dithergenerator, einen 14-Bit Auf-/Abwärtszähler mit mehreren Steuerungsautomaten, welche für die Codeverschiebung verantwortlich sind, einen 16-Bit Frequenzzähler und einen 10-Bit Codeepochenzähler.

## Benutzerprogrammierbare Gate Arrays: Logic-Cell-Arrays (LCAs) von Xilinx

Xilinx bietet seit 1985 als erster Hersteller benutzerprogrammierbare Gate Arrays an, die es ermöglichen komplexe Logikentwürfe schnell und leicht modifizierbar in preisgünstigen Bausteinen unterzubringen. Es stehen mehrere LCA-Familien zur Auswahl, die sich in der Anzahl der internen Verbindungsmöglichkeiten und integrierbarer Logik unterscheiden. Die größten Bausteine besitzen eine Kapazität von 9000 Gattern. Bei allen Serien kann man zwischen drei Geschwindigkeitsstufen wählen : 50 MHz, 70 MHz und 100 MHz maximale Flip-Flop Taktrate. Diese Angaben beziehen sich aber auf nur eine FF-Durchgangs-

zeit. Daher werden die tatsächlichen Taktfrequenzen zwischen 5 und 20% dieser Werte liegen.

## Architektur der LCAs

LCAs bestehen aus drei verschiedenen Elementen:
- Logik-Blöcke
- Ein/Ausgabe-Blöcke
- Verbindungen

Eine zentrale Matrix aus einheitlichen Logik-Blöcken (Configurable Logic Blocks) wird von Ein/Ausgabe-Blöcken umgeben; dazwischen liegt ein Netz aus verschiedenartigen Verbindungsmöglichkeiten (siehe Bild 9). Die vom Anwender programmmierbare Logikschaltung wird durch die CLBs erzeugt, wobei die E/A-Blöcke die Schnittstelle zur Bausteinperipherie darstellen.

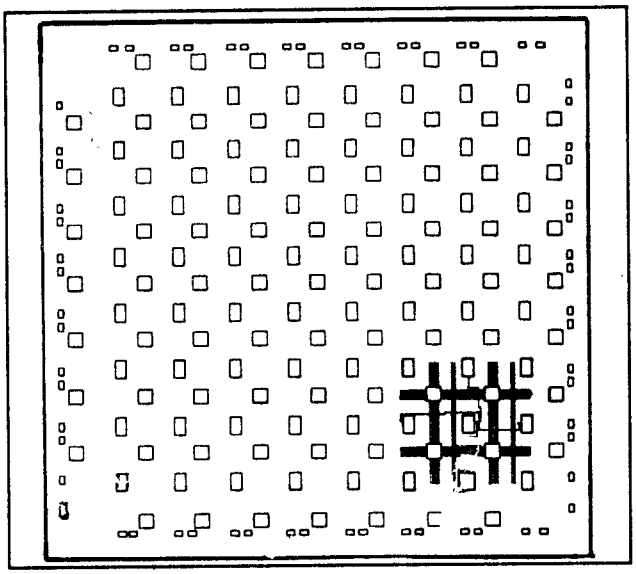


Bild 9 : Innerer Aufbau eines LCA

Alle LCA-Elemente und Funktionen werden durch ein Konfigurationsprogramm bestimmt, wodurch erst die eigentliche Anwenderschaltung entsteht. Das Programm wird während des Betriebes in einem On-Chip-Memory gespeichert und kann auf unterschiedliche Weise automatisch vom Baustein selbst aus einem externen PROM, oder gesteuert durch die Peripherie, geladen werden.
Zur Erzeugung dieses Programms bietet Xilinx ein komfortables Entwicklungssystem an.

## Das LCA-Entwicklungs-System von Xilinx

Das System besteht aus einem Programmpaket mit einer komfortablen Benutzer-oberfläche - dem Xilinx Design

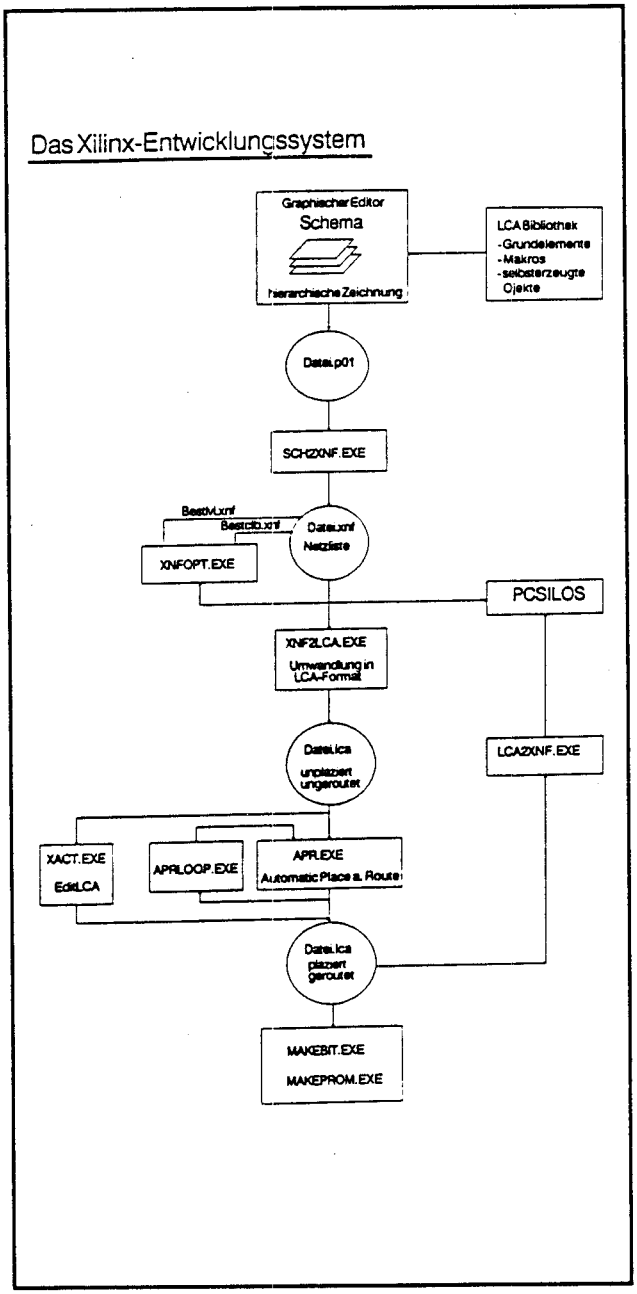



Bild 10 : Entwicklungssystem

Manager. Aus diesem lassen sich über Pull-Up-Menüs alle im Entwicklungsdurchlauf benötigten Hilfsprogramme mit den entsprechenden Optionen und Dateizusätzen aufrufen. Nachfolgend sind diejenigen Entwicklungswerkzeuge des Systems beschrieben, die ich selbst im Rahmen meiner Diplomarbeit benutzt habe. Das Softwarepaket besitzt im Bereich der Designeingabe einige weitere Optionen.
Erster Schritt der Designerzeugung ist die Erstellung eines Schaltplans mit Hilfe eines graphischen Editors ; beim System der Fachhochschule Offenburg ist dies Schema II

plus , aber das Xilinx-Programmpaket besitzt auch Schnittstellen zu anderen Editoren. Beim Zeichnen dieses Plans dürfen nur Symbole einer LCA-Bibliothek verwendet werden. Diese enthält sowohl Grundelemente wie zum Beispiel : Gatter , Puffer oder Flip-Flops als auch Makros (Zähler, Dekoder, Register...). Weiterhin besteht die Möglichkeit eigene Symbole zu kreieren und deren "Innenleben" in anderen untergeordneten Zeichnungen zu definieren. So entsteht ein hierarchischer Entwurf aus selbsterzeugten Symbolen und den dazugehörigen Zeichnungen, die in einer ersten Datei mit der Erweiterung .p01 abgelegt werden und von einem Konvertierungsprogramm (SCH2XNF.EXE) in eine Xilinx-Netzliste umgewandelt werden.
Diese Netzliste kann einerseits zur Simulation durch ein weiteres Hilfsprogramm in PC-Silos-Format umgewandelt werden und andererseits durch ein Optimierungsprogramm auf Geschwindigkeit bzw. Blockaufwand optimiert werden.
War die Simulation erfolgreich, kann die Netzliste in das LCA-Format übersetzt werden. Dabei werden die im Schaltbild verwendeten Symbole in LCA-Blöcke umgewandelt. Diese Blöcke müssen dann im LCA plaziert und dort elektrisch verbunden werden. Dazu steht neben einem LCA-Graghik-Editor, ein Programm zur Verfügung, welches diese Aufgabe übernimmt : APR (AUTOMATIC PLACE AND ROUTE).
Das Ergebnis dieses Programms hängt wesentlich von einer am Bearbeitungsanfang gewählten Zufallszahl ab, durch die die Anfangsplazierung bestimmt wird. Startet man APR mehrmals mit dem gleichen Entwurf, so wird man am Ende trotzdem unterschiedlich gute Designs erhalten. APRLOOP bietet daher die Möglichkeit automatisch den gleichen Entwurf mehrmals von APR plazieren und routen zu lassen, um sich hinterher das beste Ergebnis heraussuchen zu können. Wenn das Verbinden der Blöcke erfolgreich war, steht am Ende dieses Durchgangs eine Datei mit der Erweiterung .lca , welche durch zwei weitere Hilfsprogramme in ein PROM programmiert werden kann. Diese Datei kann ebenfalls in PC-Silos-Format umgewandelt werden, um in die Simulation auch die Signallaufzeiten im LCA miteinberechnen zu lassen.

Bild 10 zeigt den gesamten Entwicklungsablauf auf einen Blick.

## Erfahrungen mit LCAs und deren Entwicklungssystem

Bei meiner Diplomarbeit habe ich innerhalb dreier Monate zwei LCAs mit je 2000 Gattern und eines mit 3000 Gattern entwickelt und hardwaremäßig realisiert. Dabei hat die Einarbeitung in das System und das Austesten der Effektivität der einzelnen Optionen den größten Teil dieser Zeit in Anspruch genommen.
Alle drei Designs wurden von APR automatisch plaziert und geroutet. Dabei konnte ich allerdings einige Schwachpunkte erkennen:

1. Netze mit einer großen Anzahl von Anschlußpins führen zu hohen Laufzeitverzögerungen von 100ns und mehr. Auch wenn man diese Netze als "Critical" definiert, ergeben sich keine wesentlichen Verbesserungen.

2. Das Verhältnis von Logik-Blöcken und Verbindungsleitungen ist bei dem kleineren LCA (XC3020) günstig gewählt. Bei größeren Bausteinen, die zwangsläufig mehr Logik aufnehmen müssen, findet man aber die gleichen Verdrahtungsmöglichkeiten vor. Auf ein größeres LCA umzusteigen, um das Routen eines Designs zu erleichtern, hat deshalb keinen entscheidenden Vorteil gebracht. Aus diesem Grund wird auch das Routen eines Entwurfs, der fast sämtliche Logik-Blöcke des jeweiligen LCAs ausnützt, mit zunehmender Bausteingröße immer schwieriger.

3. Gibt man die Anschlußbelegung vor, erhält man wesentlich größere Durchlaufzeiten.

Mittlerweile hat Xilinx eine verbesserte und erweiterte Version des Autorouter-Pakets (ADI 3.0) ausgeliefert. Den schon vom Hersteller beschriebenen enormen Fortschritt gegenüber der älteren Version kann ich bestätigen. Ich habe diese neuen Programme auf ein Design angewendet, welches von dem alten System nur in 5% aller Versuche erfolgreich geroutet werden konnte. ADI 3.0 konnte diesen Wert auf 50% steigern. Dabei ergaben sich bei den größeren Netzen bis zu 5mal kürzere Verzögerungszeiten.
Mit dem Optimierungsprogramm XNFOPT habe ich keine positiven Erfahrungen gemacht. Die optimierten Entwürfe benötigten prinzipiell mehr Blöcke. Auch einen Geschwindigkeitsvorteil dieser Designs war nicht zu erkennen, da diese wesentlich mehr Verbindungsleitungen benötigten und dadurch die Signalverzögerungen anstiegen.
Die Schnittstelle zu PC-Silos liefert eine komfortable Möglichkeit, den Entwurf mit allen Durchlaufzeiten zu simulieren. Die Simulation hat sich als sehr exakt erwiesen, so daß die Umsetzung des Entwurfs in die Hardware nach erfolgreicher Simulation völlig unkritisch ist.

## Literaturverzeichnis

[1] Kromer, D., Landis, J., Military Agency for Standardization, Navstar Global Positioning System (GPS), System Characteristics 1987

[2] Mäusl, R., Digitale Modulationsverfahren Hüthig Verlag 2.Auflage 1988

[3] Meinke Gundlach, Taschenbuch der Hochfrequenztechnik Band 3 Springer Verlag 4.Auflage

1986

[4] Navstar GPS Joint Program Office, Los Angeles Air Force Station, US Air Force Space Division/ CWNI, Introduction To Navstar GPS User Equipment 1988

[5] Spilker, J. J., Global Positioning System : Signal Structure and Performance Characteristics, Stanford Telecommunications inc., 1979

[6] Xilinx, The Programmable Gate Array Data Book 1989

5. **Fachhochschule Ulm**
**Fachbereich Industrieelektronik**

# Steuerbaustein für eine Vierquadrantenregelung


Verfasser : Christoph Büchler
Betreuer : Prof. Arnold Führer


Präsentation : Workshop an der FH Offenburg
15. Juni 1990

# Inhaltsverzeichnis

## 1. Vorwort

Ich möchte mit diesem Bericht einen Einblick in meine Diplomarbeit im SS 1990 an der Fachhochschule Ulm geben.

Das Thema meiner Diplomarbeit ist die Entwicklung eines Steuerbausteins für eine Vierquadrantenregelung und die Integration auf einem Gate Forest Gate-Array vom Institut für Mikroelektronik in Stuttgart.
Als Entwicklungstools standen mir eine Apollo Workstation 3500 mit Mentor Graphics Software und sowie der LOG/iC Gates-Compiler der Firma ISDATA zur Verfügung.

Ulm, Juni 1990 Christoph Büchler

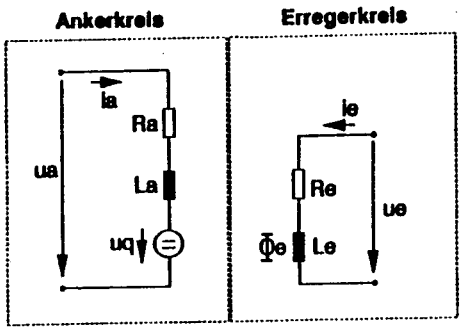
# 2. Allgemeine Grundlagen

## 2.1 Der Gleichstromantrieb

Die Gleichstrommaschine ist die Maschine für anspruchsvollste Drehzahlverstell- und Regelbedürfnisse.

Allgemein gesehen besteht die Gleichstrommaschine aus einem Ankerstromkreis und dem Erregerstromkreis. Ihr Ersatzschaltbild ist in Abb. 2.1 zu sehen.

ABB.2.1 ERSATZSCHALTBILD EINER FREMDERREGTEN GM




ua : Ankerkreisspannung
ia : Ankerkreistrom
Ra : Ankerkreiswiderstand
La : Ankerkreisinduktivität
uq : Ankerquell-Spannung

ue : Erregerkreisspannung
Re : Erregerkreiswiderstand
Le : Erregerkreisinduktivität
Φe : mag.Fluß Erregerwicklung

Nach Abbildung 2.1 gilt für den

Erregerkreis : $u_e = R_e * i_e + N * d\Phi_e / dt$

Ankerkreis : $u_a = R_a * i_a + u_q + L_a * di_a / dt$

$u_q = c_1 * n * \Phi_e$

$m_M = c_2 * i_a * \Phi_e$

Bei einem konstanten Erregerfeld ($\Phi_E$ = konstant) und im stationären Betrieb gilt für das Motormoment $m_M$ und für die Drehzahl n des Gleichstromantriebs folgende Proportionalität :

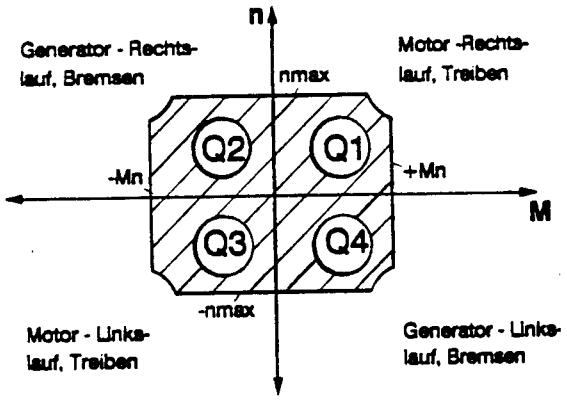
Gl.1 $m_M \approx I_a$

Gl.2 $n \approx U_a$

## 2.2 Quadrantenbegriff

Um einen Antrieb als Motor und Generator in beiden Richtungen betreiben zu können, benötigt man ein über vier Quadranten arbeitendes Stellglied.

ABB. 2.2 DARSTELLUNG DER 4 QUADRANTEN EINES ANTRIEBS



In den Quadranten Q1 und Q3 läuft der Antrieb im Motorbetrieb.

**Motorbetrieb** heißt :
Energiefluß vom Netz -> Antrieb,
oder Drehzahl n und Momentenrichtung M sind gleichsinnig.

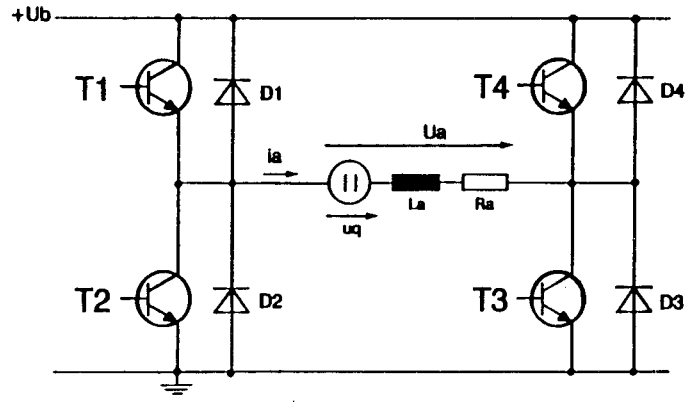
In den Quadranten Q2 und Q4 läuft der Antrieb hingegen im Generatorbetrieb, d.h. es liegt ein bremsender generatorischer Betrieb vor.

**Generatorbetrieb** heißt :
Energiefluß vom Antrieb -> Netz,
oder Drehzahl n und Momentenrichtung M sind entgegengesetzt.

## 2.3 Steuerverfahren bei Gleichstromstellern

Um einen Gleichstromantrieb in allen vier Quadranten betreiben zu können, müssen laut Gleichung 1 der Ankerstrom und laut Gleichung 2 die Ankerspannung beide Polaritäten annehmen können. Dies kann durch eine Brückenschaltung von Schaltgliedern erreicht werden.
Die Erklärung des Vierquadrantenbetriebs erfolgt am Beispiel eines Gleichstromstellers mit Transistor-Schaltgliedern. Die zugehörige Brückenschaltung ist in Abb.2.3 zu sehen.

ABB. 2.3 BRÜCKENSCHALTUNG FÜR VIERQUADRANTENBETRIEB




Da für den stationären Betrieb der Zusammenhang $n \approx U_a$ gilt, muß aus der konstanten Betriebsspannung $U_b$ eine variable Ankerspannung $U_a$ mit wechselbarer Polarität geschaffen werden.
Dafür gibt es im allgemeinen zwei verschiedene Steuerverfahren.

### 2.3.1 Steuerverfahren I

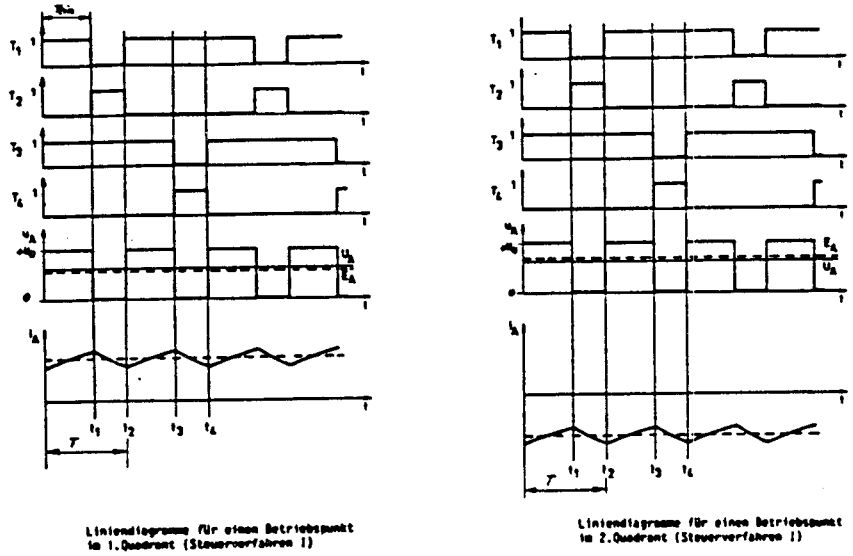
Durch das abwechselnde An- und Abschalten der Betriebsspannung $U_b$ vom Motor bildet sich ein Mittelwert der Ankerspannung, der durch Ändern des Tastverhältnisses in der Größe beeinflußt werden kann.

Dabei ergibt sich folgende Gleichung für den Mittelwert der Ausgangsspannung :

$$U_a = U_b * T_E / (T_A + T_E) = U_b * T_E / T$$

Die Abbildung 2.3.1 zeigt ein entsprechendes Liniendiagramm für einen Betriebspunkt im 1. bzw. 2.Quadranten, also Rechtslauf.

ABB. 2.3.1 LINIENDIAGRAMME STEUERVERFAHREN I



Für negative Drehzahlen (3. und 4.Quadrant) würden die Ansteuerungen von T1 und T2 bzw. T3 und T4 vertauscht. Die Ankerspannung würde dann negativ werden.

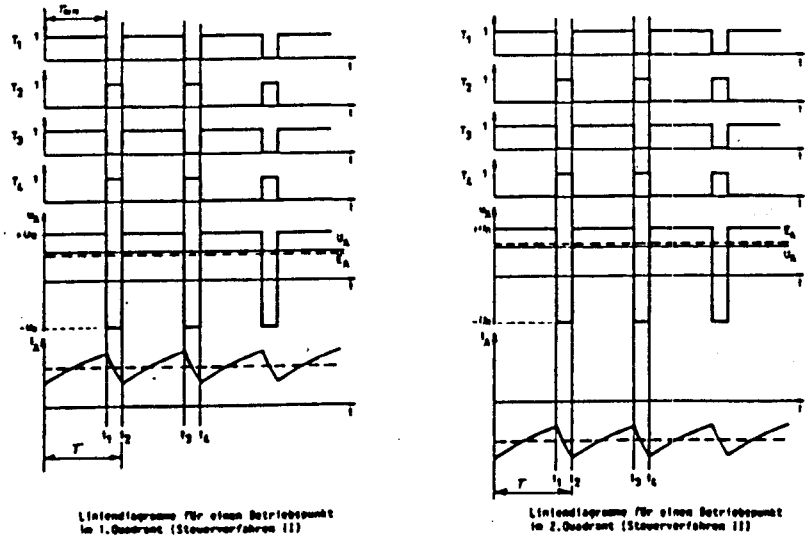
## 2.3.2 Steuerverfahren II

Der Unterschied zum Steuerverfahren I besteht darin, daß die Ankerspannung nicht zwischen 0V und +Ub, sondern zwischen +Ub und -Ub geschaltet wird.
Deshalb ergibt sich für den Mittelwert der Ankerspannung eine andere Gleichung. Es gilt :

$$U_a = U_b * (T_E - T_A) / T = U_b * (2*T_E - T) / T$$

Abbildung 2.3.2 zeigt entsprechende Liniendiagramme des Steuerverfahren II.

ABB. 2.3.2 LINIENDIAGRAMME STEUERVERFAHREN II



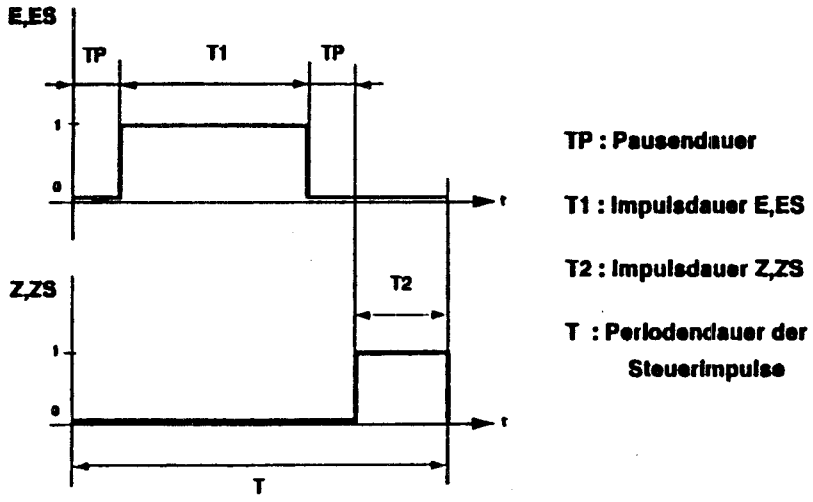
## 3. Anforderungen an den Steuerbaustein

Der Baustein soll eine Vierquadrantenregelung gemäß Steuerverfahren II Abschnitt 2.3.2 ermöglichen.

### 3.1 Funktionsbeschreibung

Die Steuersignale E,ES,Z und ZS zur Ansteuerung der Transistor-Schaltglieder T1,T3,T2 und T4 sollen einen Signalverlauf nach Abb. 3.1 haben.
Diese vier Steuersignale sind durch die zwei Signaleingänge NENABLE1 und NENABLE2 des Bausteins jederzeit hochohmig schaltbar.

ABB. 3.1 SIGNALVERLAUF DER STEUERSIGNALE E,ES UND Z,ZS




Folgende Einstellmöglichkeiten der Steuersignalausgänge sind zu realisieren :

**Periodendauer T** : **20 µs ... 500 µs**
-> Quantisierung : ≥ 360 Intervalle

**Pausendauer TP** : **0,2 µs ... 5 µs**

**Grenzen des Tastverhältnisses $T_1/T$ : 5% ... 95%**

Zudem soll beim Einschalten des Steuerbausteins ein voreingestelltes Tastverhältnis eingehalten werden, das erst durch ein Steuersignal des Rechners durch das vom Rechner vorgegebene Tastverhältnis ersetzt wird.

### 3.2 Programmierung des Bausteins

Eine Kommunikation des Bausteins soll mit einem Intel Mikroprozessor der Typen 8088/8086 und 80386DX(20MHz) möglich sein.
Programmiert werden kann die Pausendauer TP, die Impulsdauer T1 und die Periodendauer T des Steuersignals. Eine Überprüfung auf Richtigkeit der programmierten Daten muß vom Baustein erfolgen. Das Ergebnis der Überprüfung wird im Kontroll-/Statusregister Bit-Nr.7 dem Programmierer mitgeteilt. Bei einer fehlerhaften Programmierung des Bausteins wird das voreingestellte Tastverhältnis ausgegeben. Die Programmierung des Bausteins erfolgt über einen 8-Bit breiten bidirektionalen Datenbus.

## 3.3 Signalleitungen des Steuerbausteins

Der Baustein besitzt 18 Eingangssignale, 6 Ausgangssignale und 8 bidirektionale Signale.

### 3.3.1 Eingangssignale

| | |
|---|---|
| VCC | : + 5 Volt Versorgungsspannung |
| GND | : 0 Volt |
| CLK | : Taktsignal |
| NCS | : Chipselect (low aktiv) |
| | wirken unabhängig vom Chipselect: |
| R | : Master-Reset asynchron (high aktiv) |
| NCLR | : Clear synchron (low aktiv) |
| NT | : Umschaltung in Scan-Path-Betrieb (low aktiv) |
| TIN | : Testeingang für Scan-Path-Betrieb |
| NENABLE1 | : Signalausgänge E,ES aktivieren (low aktiv) |
| NENABLE2 | : Signalausgänge Z,ZS aktivieren (low aktiv) |
| C[2..0] | : Startmodus |
| | wirken nur bei aktivem Chipselect: |
| A[2..0] | : Adressen |
| NWR | : Zeiten/Kontrollwort schreiben (low aktiv) |
| NRD | : Statuswort lesen (low aktiv) |

### 3.3.2 Ausgangssignale

| | |
|---|---|
| E, ES | : Steuerimpulse der Dauer T1 |
| Z, ZS | : Steuerimpulse der Dauer T2 |
| TOUT | : Testausgang |
| T | : Periodenendesignal (LH-Flanke) |

### 3.3.3 Bidirektionale Signale

| | |
|---|---|
| D[7..0] | : Datenbus |

## 3.4 Testmöglichkeit des Bausteins

Um die Prüfbarkeit der Schaltung zu verbessern, ist ein Scan-Path (Prüfbus) zusätzlich zu integrieren.

## 4. Schaltungsrealisierung

Die Schaltung ist mit den Standardzellen der IMS-Zellbibliothek 3/88 ff für die Gate Forest Gate-Arrays GFxxx vom Institut für Mikroelektronik in Stuttgart realisiert.

### 4.1 Bildung des Steuersignals

Der Signalverlauf einer Periode T der Steuersignale E,ES,Z und ZS setzt sich nach Abb. 3.1 wie folgt zusammen :

$$T = TP + T1 + TP + T2$$

Da die Impulsdauer T2 der Steuersignale Z und ZS nicht programmiert werden kann, wird diese aus der bleibenden Differenz von T - 2*TP - T1 gebildet.

Die Pausendauer TP setzt sich als Vielfaches der Dauer eines Inkrements TI zusammen. Die Dauer eines Inkrements wird durch die Teilung des Systemtaktes (Periodendauer TT) mit einem programmierbaren Teiler (Teilerverhältnis 1:1 bis 1:31) erreicht, d.h. TI = nI * TT wobei $1 \leq nI \leq 31$.
Die Bildungsvorschrift der Pausendauer TP lautet somit :

$$TP = nP * TI = nP * nI * TT \text{ , Einstellmöglichkeit : } 4 \leq nP \leq 31$$

Die Impulsdauer T1 der Steuersignale E,ES wird ebenfalls aus dem Vielfachen der Dauer eines Inkrements gebildet. Für die Impulsdauer T1 gilt :

$$T1 = nD * TI = nD * nI * TT \text{ , Einstellmöglichkeit : } (0 \leq nD \leq 1023)$$

Da sich die Periodendauer T der Steuerimpulse aus folgender Gleichung bildet :

$$T = n * TI = n * nI * TT \text{ , Einstellmöglichkeit : } (256 \leq n \leq 1023)$$

ist zu beachten, daß für die maximale Impulsdauer T1 der programmierbare Faktor nD nur sinnvoll ist, wenn $nD \leq n-2*nP$ gilt.

Führt man dem Steuerbaustein einen Systemtakt mit der maximal möglichen Taktfrequenz von 20MHz zu, so ergeben sich folgende Einstellmöglichkeiten der Steuersignale :

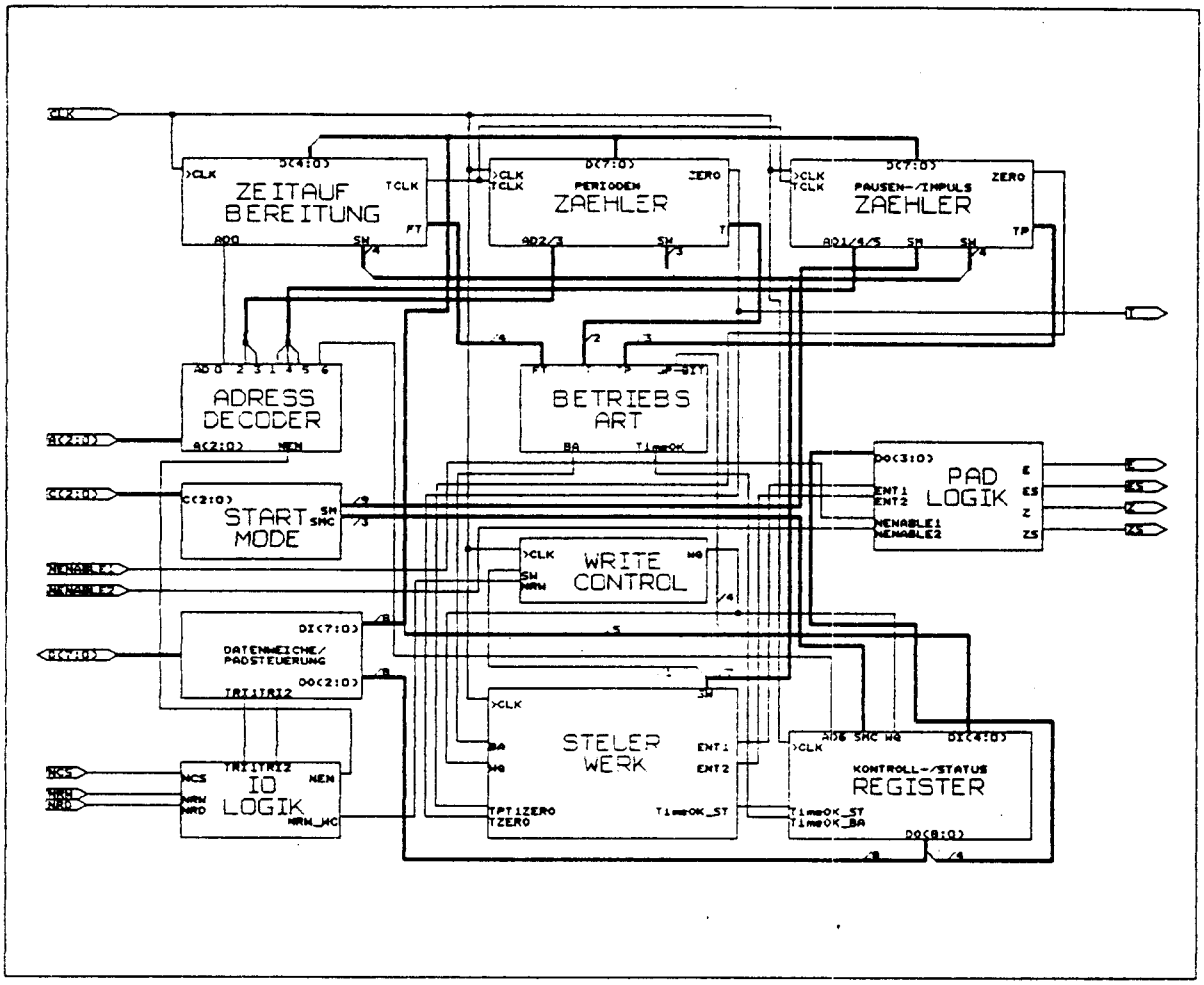
| | | |
|---|---|---|
| **Periodendauer T** | : | **12,8 µs ... 1580 µs** |
| -> Quantisierung | : | 3879 Intervalle |
| **Pausendauer TP** | : | **0,2 µs ... 48,05 µs** |
| -> Quantisierung | : | 135 Intervalle |

**Grenzen des Tastverhältnisses $T_1/T$ : 0% ... 99,2%**

## 4.2 Blockschaltbild

Die Abb. 4.2 zeigt ein vereinfachtes Blockschaltbild der gesamten Schaltung in der obersten Hierarchieebene. Deutlich zu sehen ist, daß ein synchrones Schaltungskonzept zugrunde liegt, d.h. der Systemtakt (CLK) wird allen Funktionsblöcken mit Speicherelementen zugeführt. Die Speicherelemente arbeiten also alle synchron mit dem Systemtakt. Dadurch wird eine sichere Funktion der Schaltung unter Berücksichtigung aller Toleranzen erreicht.

ABB 4.2 VEREINFACHTES BLOCKSCHALTBILD DER OBERSTEN ENTWURFSEBENE




Folgende Funktionsblöcke werden dabei unterschieden :

- ***ZEITAUFBEREITUNG***

Mit diesem Modul wird ein zum Systemtakt synchroner Einzelimpuls mit dem programmierten Teilerverhältnis erzeugt. Dieses Signal (TCLK) dient als Freigabesignal für die Funktionsblöcke *PERIODENZÄHLER* und *PAUSEN-/IMPULSZÄHLER*. Dadurch wird erreicht, daß die Zählermodule weiterhin mit dem ungeteilten Systemtakt getaktet werden und der Synchronismus der Gesamtschaltung erhalten bleibt.

- ***PERIODENZÄHLER***

Dieser Funktionsblock hat die Aufgabe, das Zeitsignal der Periodendauer T des Steuersignals zu liefern. Er besteht vor allem aus einem 10-Bit breiten programmierbaren Rückwärtszähler, der vom programmierten Wert n bis zum Zählerstand Null zählt. Beim Erreichen des Zählerstandes Null wird ein Meldesignal an das Steuerwerk geliefert. Der Zähler arbeitet mit dem Systemtakt (CLK), als Freigabesignal des Zählers wird unter anderem der erzeugte Einzelimpuls (TCLK) der Zeitaufbereitung verwendet.

- ***PAUSEN-/IMPULSZÄHLER***

Das Modul hat die Aufgabe, die Zeitsignale der Pausendauer TP sowie der Impulsdauer T1 des Steuersignals zu liefern. Um Chipfläche zu sparen, wurde ein gemeinsamer programmierbarer Rückwärtszähler (10-Bit breit) zur Bildung des Zeitsignales verwendet. Das Umschaltesignal, - ob die Pausendauer TP oder die Impulsdauer T1 über die Zähldauer des Rückwärtszähler erzeugt wird -, liefert das Steuerwerk.

- ***ADRESSDECODER***

Der Adressdecoder decodiert aus den drei Adressleitungen A[2:0] nachstehende sieben Adressen. Diese werden als Freigabesignal für die entsprechenden Register verwendet.

```
A2 A1 A0
0  0  0  :  Teilerverhältnis nI (5 bit)
0  0  1  :  Pausendauer nP (5 bit)
0  1  0  :  Periodendauer n ( 8 bit, niederwertig)
0  1  1  :  Periodendauer n ( 2 bit, höherwertig)
1  0  0  :  Impulsdauer nD (8 bit, niederwertig)
1  0  1  :  Impulsdauer nD (2 bit, höherwertig)
1  1  0  :  Kontroll-/Statusregister (8 bit)
```

- ***BETRIEBSART***

Dieses Modul enthält eine reine Kombinatorik, die die programmierten Faktoren n und nP der Zeiten T bzw. TP und das Teilerverhältnis nI auf eine mögliche Einstellverletzung (s. Kapitel 4.1) hin überprüft und daraus die Betriebsart (µP oder AUTO) des Bausteins bildet.

- ***PAD-LOGIK***

Der Funktionsblock Pad-Logik erzeugt die Ansteuersignale der Ausgangs PAD-Zellen der Steuersignale E,ES,Z und ZS. Er enthält eine Kombinatorik zur Auswertung des Kontrollregisterinhaltes Bit-Nr.[3..0] und Treiberstufen zur Ansteuerung der PAD-Zellen.

- ***START-MODE***

Das Modul decodiert das über die Signaleingänge C[2:0] voreinstellbare Tastverhältnis. Es kann zwischen folgenden drei Startmoden gewählt werden :

```
C2 C1 C0
0  0  1  : n = 512, nP = 31, nD = 224, nI = 1  -> (Ua = 0V     )
0  1  0  : n = 512, nP = 31, nD = 22 , nI = 1  -> (Ua ≈-0.8*Ub)
1  0  0  : n = 512, nP = 31, nD = 426, nI = 1  -> (Ua ≈+0.8*Ub)
```

- ***WRITE-CONTROL***

Der Funktionsblock liefert dem Steuerwerk die Information, ob während einer laufenden Steuersignalperiode T ein Schreibzyklus und damit eine Umprogrammierung des Bausteins durch den Prozessor stattgefunden hat. Durch diese Information wird der Ablauf des Steuerwerks optimiert.

- ***DATENWEICHE/PADSTEUERUNG***

Dieses Modul beinhaltet die notwendige Ansteuerlogik der bidirektionalen PAD-Zellen des 8-Bit breiten Datenbuses.

- ***IO-LOGIK***

Diese Kombinatorik bildet aus dem Schreib-/Lesesignal des Prozessors sowie dem Chipselect-Signal die notwendigen Steuersignale für das Modul *DATENWEICHE/PAD-STEUERUNG*, *ADRESS-DECODER* und *WRITE-CONTROL*.

- ***KONTROLL-/STATUSREGISTER***

Mit Hilfe des Kontrollregisters kann jeder Steuersignalausgang exklusiv freigeschaltet bzw. gesperrt werden (Bit-Nr.[3..0]).
Durch das Setzen von Bit-Nr.[4] wird der Baustein in den µP-Betrieb gebracht, d.h. das programmierte Tastverhältnis wird ausgegeben.

Das Statusregister (Bit-Nr.[7..5]) enthält codiert den eingestellten Startmodus und eine Information über die Richtigkeit der programmierten Daten.

ABB. 4.2.1 KONTROLL-/STATUSREGISTER

| | < STATUS- > | | | < KONTROLL-Info > | | | | |
|---|---|---|---|---|---|---|---|---|
| BIT Nr. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Bit Nr.7 : Zeiten korrekt (high aktiv)
Bit Nr.6,5 : Startmodus codiert 00=Nr.1, 01=Nr.2, 10=Nr.3
Bit Nr.4 : Kontrolle durch µProzessor (high aktiv)
Bit Nr.3 : Enable Z-Steuerausgang (high aktiv)
Bit Nr.2 : Enable ZS-Steuerausgang (high aktiv)
Bit Nr.1 : Enable E-Steuerausgang (high aktiv)
Bit Nr.0 : Enable ES-Steuerausgang (high aktiv)

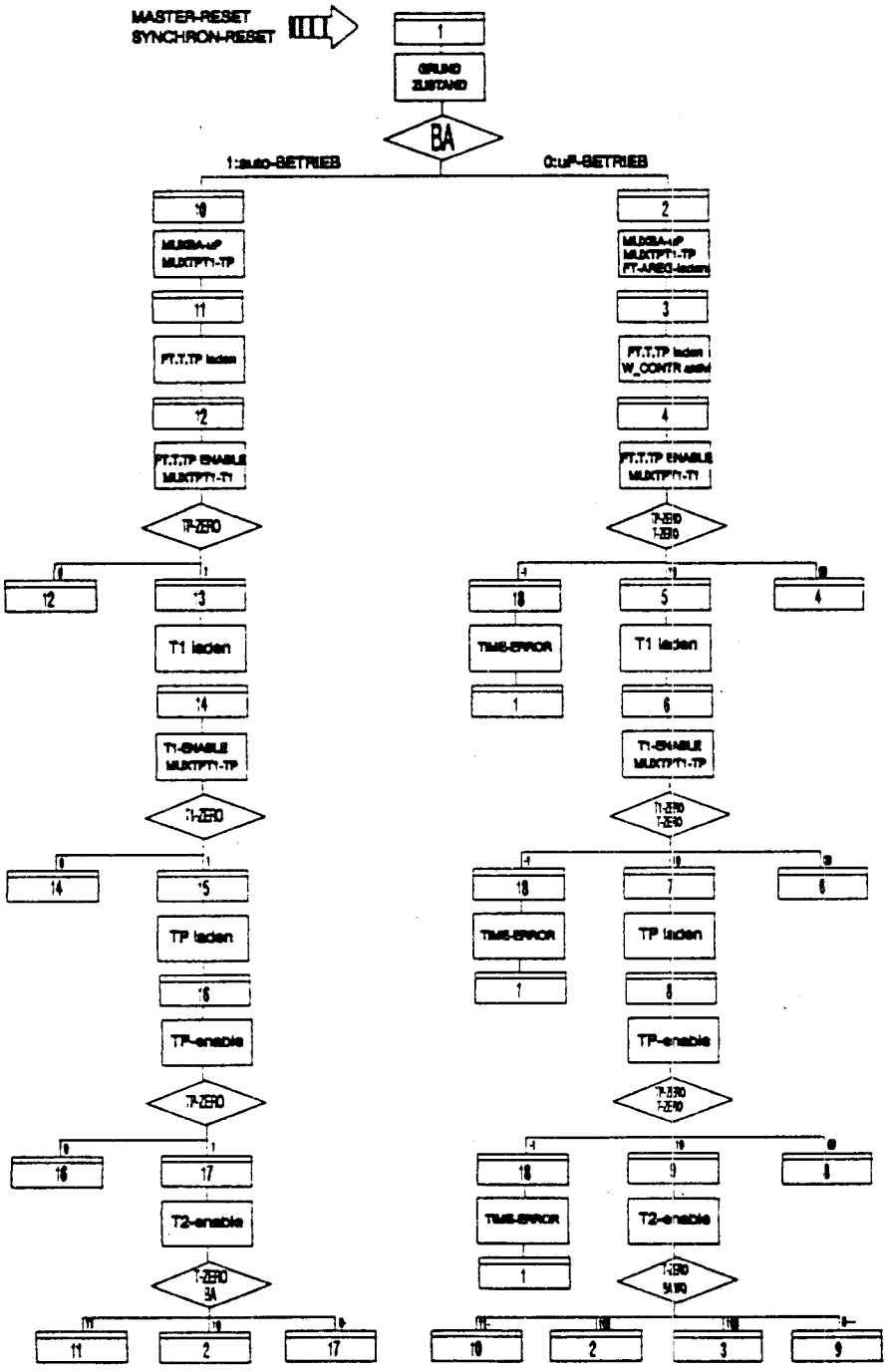
## 4.3 Steuerwerk des Bausteins

Das Steuerwerk ist das "Herz" dieser Schaltung. Es steuert das Zusammenspiel der im Kapitel 4.2 beschriebenen Funktionsblöcke. Realisiert worden ist es als Moore-Automat mit 18 Zuständen. Getaktet wird es mit dem Systemtakt CLK.
Die Entscheidungsvariablen des Steuerwerks sind die 5 Signale NCLR (synchroner Reset), TPT1ZERO (Zählerstand Null des *PAUSEN-/IMPULSZÄHLERs*), TZERO (Zählerstand Null des *PERIODEN-ZÄHLERs*), BA (ermittelte Betriebsart µP bzw. AUTO aus dem Funktionsblock *BETRIEBSART*) sowie WQ (Information aus dem Modul *WRITE-CONTROL* bei einem Programmierzyklus).

Als Steuervariablen liefert das Steuerwerk insgesamt 11 Signale. Diese Ausgangssignale dienen als Zähl-/Ladefreigabesignale (ENTFT,ENTPT1-/NLTFT, NLTPT1) für die Zähler in den Modulen *ZEITAUFBEREITUNG*, *PERIODENZÄHLER* und *PAUSEN-/IMPULSZÄHLER*, ein Registerfreigabesignal (ENFTR), die Freigabe der Impulsdauer T1 (ENT1) und T2 (ENT2), Steuersignale für Multiplexer (MUXBA, MUXTPT1), ein Löschsignal für die *SCHREIBÜBERWACHUNG* (W_CLR) sowie ein Meldesignal bei einer Zeitverletzung während einer laufenden Steuersignalperiode T (TimeOK_ST).

Die Funktion des Steuerwerks läßt sich durch folgendens Flußdiagramm beschreiben.

ABB. 4.3 FLUSSDIAGRAMM DES STEUERWERKS




Die entsprechende Ablauftabelle des Steuerwerks ist im Anhang 7.1 zu sehen.

### 4.3.1 Entwicklung des Steuerwerks

Die Umsetzung der Ablauftabelle in einen entsprechenden Stromlaufplan wurde mit Hilfe des LOG/iC Gates-Compilers der Firma ISDATA erreicht.

Für die Entwicklung der Gatterstruktur greift der LOG/iC Gates-Compiler auf eine Gatterbiliothek zurück, die die verfügbaren Gatter und ihre technische Eigenschaften, wie Laufzeiten, Kostenfaktoren und Fan-Out enthält.
Auf den Vorgang einer Gatter-Bibliothekserstellung möchte ich auf den Bericht [1] verweisen.

Damit der Gates-Compiler die erstellte Gatterbibliothek für die Stromlaufplanerstellung verwendet, muß diese im RUN-CONTROL Datenbereich des LOG/iC Eingabedatensatzes über das Schlüsselwort GATLIB = "Name der Gatterbibliothek" dem Compiler mitgeteilt werden. Der komplette LOG/iC-Eingabedatensatz für das Steuerwerk ist im Anhang 7.2 zu sehen.

Die vom Compiler generierte Schaltung ist 12-stufig und hat eine maximale Verzögerungszeit von 9,49ns.
Für die Schaltung werden folgende Standardzellen benötigt :

| | | |
|---|---|---|
| 2-Eingangs UND-Gatter IAND2 | : | 69 |
| 2-Eingangs ODER-Gatter IORO2 | : | 22 |
| 2-Eingangs NAND-Gatter INAND2A | : | 1 |
| 2-Eingangs NOR-Gatter INOR2A | : | 7 |
| Inverter IINV1A | : | 21 |
| Treiber IBUF4A | : | 1 |
| D-FF mit asynchron R und S | : | 12 |

Da noch keine Postprozessor-Phase, d.h. eine automatische Umsetzung der LOG/iC Netzliste in eine NETED-Netzliste, mit dem Gates-Compiler möglich war, mußte das von LOG/iC erzeugte Gatterschaltbild (20 DIN A4 Blätter !) von Hand in einen NETED-Stromlaufplan umgesetzt werden.

Der Stromlaufplan des Steuerwerks ist auf der nächsten Seite zu sehen.

### 4.3.2 Simulation des Steuerwerks

Die Funktionsüberprüfung des Steuerwerks wurde komplett mit dem Logiksimulator QUICKSIM durchgeführt.
Sehr hilfreich bei einer solch umfangreichen Simulation ist es, z.B. den Ausgangsvektor (hier 11 Signale) der Schaltung als 11-Bit breiten Bus zu declarieren. Definiert man dann zusätzlich einen 11-Bit breiten Bus als Soll-Ausgangsvektor und spricht diesen in dem Stimuli-File entsprechend an, kann eine Überprüfung der Steuerwerksfunktion durch den Vergleich IST-/ und SOLL-Ausgangsvektor recht schnell erfolgen.

Auf Seite 15 ist eine Teilsimulation des Steuerwerks zu sehen.

ABB. 4.3.1 STROMLAUFPLAN STEUERWERK

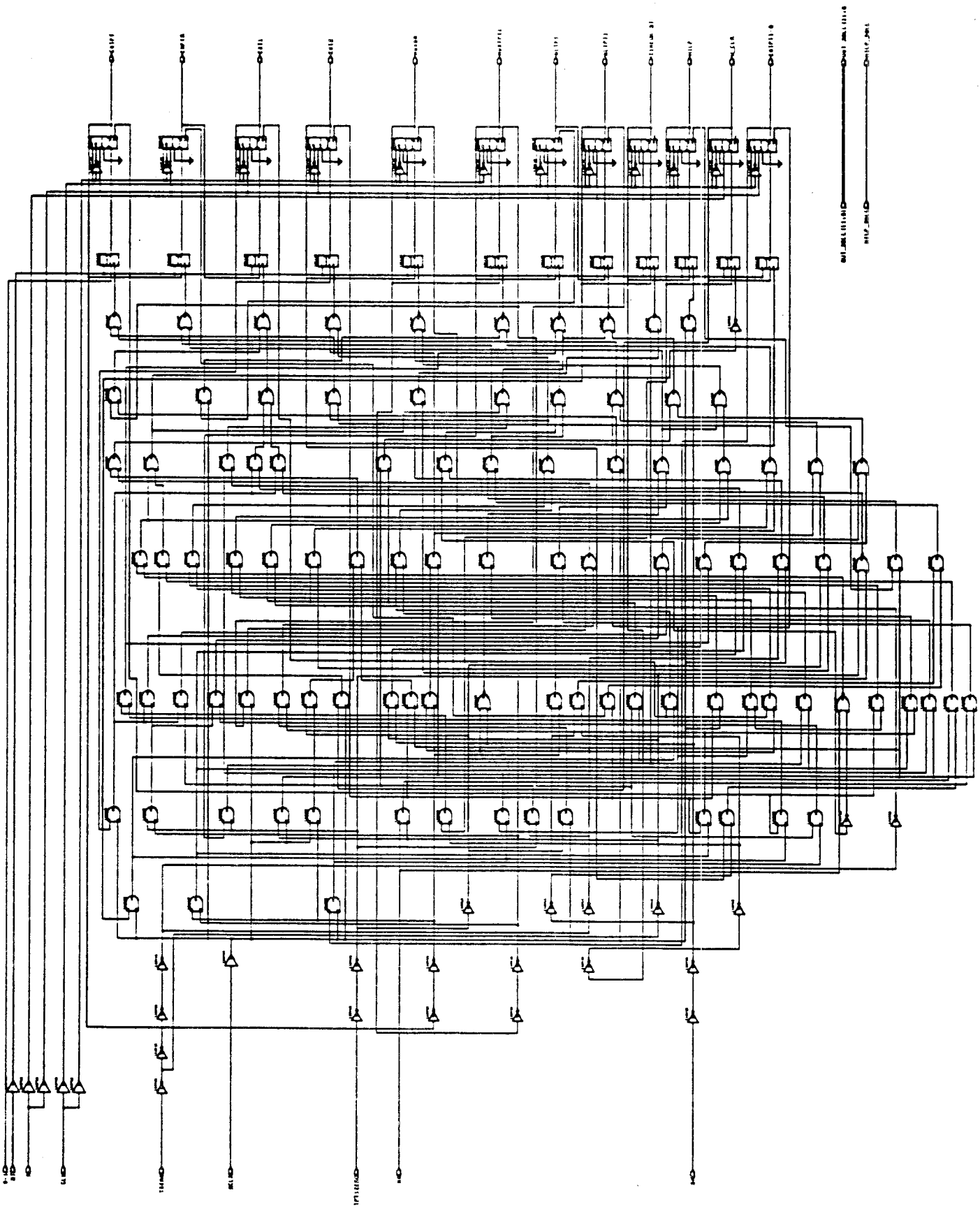
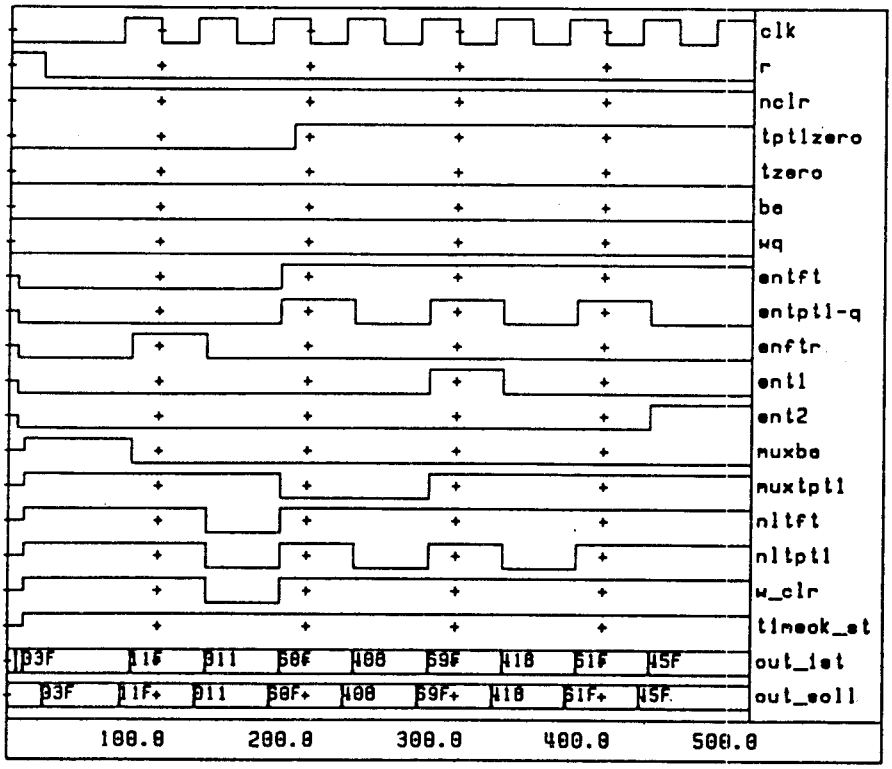


ABB. 4.3.2.1 STIMULI-FILE EINER TEILSIMULATION DES STEUERWERKS

```
# STEUERWERK PPBS
# Teilsimulation Nr.4, Mode = µP
# filename : sw_test4.sti
# create   : 28/05/90
# update   : 29/05/90

transcripting on -all
check -nospike
reset sim time

# system-clock (20MHz)
  clock period 50
  force clk         0   0      -NOAb
  force clk         0   50  -r -NOAb
  force clk         1   75  -r -NOAb

# default
  force nt          1   0      -NOAb
  force nclr        1   0      -NOAb
  force tptlzero    0   0      -NOAb
  force tzero       0   0      -NOAb
  force ba          0   0      -NOAb
  force wq          0   0      -NOAb
  force out_soll    03F 23     -NOAb

# power-on reset
  force r           1   0      -NOAb
  force r           0   22     -NOAb

# Zustand Nr.2
  force out_soll    11F   75   -NOAb

# Zustand Nr.3
  force out_soll    011   125  -NOAb

# Zustand Nr.4
  force out_soll    60F   175  -NOAb
  force tptlzero      1   190  -NOAb

# Zustand Nr.5
  force out_soll    40B   225  -NOAb

# Zustand Nr.6
  force out_soll    69F   275  -NOAb

# Zustand Nr.7
  force out_soll    41B   325  -NOAb

# zustand Nr.8
  force out_soll    61F   375  -NOAb

# Zustand Nr.9
  force out_soll    45F   425  -NOAb

run 500
```

ABB 4.3.3.2 SIMULATIONSERGEBNIS



## 5. Bestimmung der Chipkomplexität

Die Bestimmung der Chipfläche für die Gesamtschaltung läßt sich über die Anzahl und die Ausmaße der verwendeten Standardzellen ermitteln.

Folgende Standardzellen wurden für die Gesamtschaltung (Stand:12.06.1990) benötigt :

| Gatterbezeichnung | Breite | Höhe | Anzahl | Σ Sites |
|---|---|---|---|---|
| IINV1A | 2 | 1 | 343 | 686 |
| IBUF1A | 3 | 1 | 6 | 18 |
| IBUF4A | 7 | 1 | 87 | 609 |
| INAN2A | 3 | 1 | 21 | 63 |
| INAN3A | 4 | 1 | 19 | 76 |
| INAN4A | 6 | 1 | 10 | 60 |
| INAN5A | 7 | 1 | 1 | 7 |
| INOR2A | 3 | 1 | 24 | 72 |
| INOR3A | 5 | 1 | 3 | 15 |
| INOR4A | 6 | 1 | 6 | 6 |
| INOR5A | 7 | 1 | 1 | 7 |
| IAND2A | 5 | 1 | 165 | 825 |
| IORO2A | 5 | 1 | 22 | 110 |
| IMUX2A | 6 | 1 | 241 | 1446 |
| IDRS1B | 19 | 1 | 93 | 1767 |
| ITPB2A | 31 | 1 | 12 | 372 |
| | | | | Σ = 6139 |

Betrachtet man die Technologie-Statistiken der einzelnen IMS GF-Gate-Arrays, so stellt man fest, daß eine Integration der Schaltung auf dem Typ GFxx2 möglich ist.

Dieses Gate-Arrays bietet 14880 placement-sites. Da in meinem Fall nur 6139 placement-sites benötigt werden (entspricht einem Ausnutzungsgrad von 41%) kann auch von einer 100% Verdrahtung durch den Autorouter ausgegangen werden.

## 6. Literaturverzeichnis

[1] Aufbereitung einer Digitalschaltung mit dem LOG/iC Gates-Compiler für die Integration mit einem ASIC
Joachim Schmidt, MPC-Gruppe der FH Baden-Württemberg,
Workshop Juni 1989, Ulm

[2] Kapitel 1 : Unterlagen aus der Vorlesung Leistungselektronik
5.Semester Industrieelektronik FH Ulm, P. Fleischauer

# 7. Anhang

## 7.1 Ablauftabelle des Steuerwerks

| Z | Eingangsvektor | | | | | Ausgangsvektor | | | | | | | | | | | | FZ | BUS SOLL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NCLR | TPT1ZERO | TZERO | BA | WQ | ENTFT | ENTPT1 | ENTTR | ENT1 | ENT2 | MUXBA | MUXTPT1 | NLTFT | NLTPT1 | WCLR | TimeOKST | HILF | | |
| 1 | 0 | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 03F |
| 1 | 1 | - | - | 0 | - | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 2 | |
| 1 | 1 | - | - | 1 | - | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 10 | |
| 2 | 0 | - | - | - | - | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | - | 1 | 11F |
| 2 | 1 | - | - | - | - | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | - | 3 | |
| 3 | 0 | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | - | 1 | 011 |
| 3 | 1 | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | - | 4 | |
| 4 | 0 | - | - | - | - | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | - | 1 | 60F |
| 4 | 1 | - | 1 | - | - | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | - | 18 | |
| 4 | 1 | 0 | 0 | - | - | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | - | 4 | |
| 4 | 1 | 1 | 0 | - | - | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | - | 5 | |
| 5 | 0 | - | - | - | - | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | - | 1 | 40B |
| 5 | 1 | - | - | - | - | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | - | 6 | |
| 6 | 0 | - | - | - | - | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | - | 1 | 69F |
| 6 | 1 | - | 1 | - | - | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | - | 18 | |
| 6 | 1 | 0 | 0 | - | - | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | - | 6 | |
| 6 | 1 | 1 | 0 | - | - | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | - | 7 | |
| 7 | 0 | - | - | - | - | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | - | 1 | 41B |
| 7 | 1 | - | - | - | - | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | - | 8 | |
| 8 | 0 | - | - | - | - | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | - | 1 | 61F |
| 8 | 1 | - | 1 | - | - | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | - | 18 | |
| 8 | 1 | 0 | 0 | - | - | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | - | 8 | |
| 8 | 1 | 1 | 0 | - | - | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | - | 9 | |
| 9 | 0 | - | - | - | - | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | - | 1 | 45F |
| 9 | 1 | - | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | - | 2 | |
| 9 | 1 | - | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | - | 3 | |
| 9 | 1 | - | 0 | - | - | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | - | 9 | |
| 9 | 1 | - | 1 | 1 | - | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | - | 10 | |
| 10 | 0 | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 03F |
| 10 | 1 | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 11 | |
| 11 | 0 | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | - | 1 | 033 |
| 11 | 1 | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | - | 12 | |
| 12 | 0 | - | - | - | - | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | - | 1 | 62F |
| 12 | 1 | 0 | - | - | - | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | - | 12 | |
| 12 | 1 | 1 | - | - | - | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | - | 13 | |
| 13 | 0 | - | - | - | - | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | - | 1 | 42B |
| 13 | 1 | - | - | - | - | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | - | 14 | |
| 14 | 0 | - | - | - | - | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | - | 1 | 6BF |
| 14 | 1 | 0 | - | - | - | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | - | 14 | |
| 14 | 1 | 1 | - | - | - | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | - | 15 | |
| 15 | 0 | - | - | - | - | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | - | 1 | 43B |
| 15 | 1 | - | - | - | - | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | - | 16 | |
| 16 | 0 | - | - | - | - | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | - | 1 | 63F |
| 16 | 1 | 0 | - | - | - | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | - | 16 | |
| 16 | 1 | 1 | - | - | - | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | - | 17 | |
| 17 | 0 | - | - | - | - | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - | 1 | 47F |
| 17 | 1 | - | 1 | 0 | - | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - | 2 | |
| 17 | 1 | - | 1 | 1 | - | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - | 11 | |
| 17 | 1 | - | 0 | - | - | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - | 17 | |
| 18 | 0 | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | - | 1 | 03E |
| 18 | 1 | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | - | 1 | |

## 7.2 LOG/iC-Eingabedatensatz für das Steuerwerk des Steuerbausteins

```
*IDENTIFICATION
Steuerwerk fuer PPBS
Christoph Buechler E8 SS1990
First Creation :  23/05/90
Update         :  25/05/90

*X-NAMES
NCLR,      !  0 : Grundzustand
!             1 : Normalbetrieb
TPT1ZERO,  !  0 : Pausen-/Impulszaehler Zaehlerstand <> 0
!             1 : Zaehlerstand = 0
TZERO,     !  0 : Periodenzaehler Zaehlerstand <> 0
!             1 : Zaehlerstand = 0
BA,        !  0 : uP-Betrieb
!             1 : auto-Betrieb
WQ;        !  0 : kein Schreibzyklus waehrend Periode
!             1 : Schreibzyklus

*Z-NAMES
;-------------------------------------------------------------------------
ENTFT,     !  0 : Perioden-/Frequenzteilerzaehler disable
!             1 : enable
ENTPT1,    !  0 : Pausen-/ImpulsdauerT1zaehler disable
!             1 : enable
ENFTR,     !  0 : Arbeitsregister Frequenzteiler disable
!             1 : enable
ENT1,      !  0 : ImpulsdauerT1- Ausgang disable
!             1 : enable
ENT2,      !  0 : ImpulsdauerT2- Ausgang disable
!             1 : enable
;-------------------------------------------------------------------------
MUXBA,     !  0 : Multiplexer BA : uP-Betrieb
!             1 : Multiplexer BA : auto-Betrieb
MUXTPT1,   !  0 : Multiplexer TPT1 : Impulsdauer T1
!             1 : Multiplexer TPT1 : Pausendauer TP
;-------------------------------------------------------------------------
NLTFT,     !  0 : Perioden-/Freuqenzteilerzaehler enable
!             1 : disable
NLTPT1,    !  0 : Pausen-/ImpulsdauerT1zaehler enable
!             1 : disable
;-------------------------------------------------------------------------
W_CLR,     !  0 : Schreibueberwachung aktivieren
!             1 : Ruhezustand
;-------------------------------------------------------------------------
TimeOK_ST,!  0 : Runtime-Error : T<TP, T<T1
!             1 : no Error
;-------------------------------------------------------------------------
HILF;      !    : Variable zur Unterscheidung der Zustaende

*Z-VALUES
S1  = 00000 11 11 1 1 0;
S2  = 00100 01 11 1 1 -;
S3  = 00000 01 00 0 1 -;
S4  = 11000 00 11 1 1 -;
S5  = 10000 00 10 1 1 -;
S6  = 11010 01 11 1 1 -;
S7  = 10000 01 10 1 1 -;
S8  = 11000 01 11 1 1 -;
S9  = 10001 01 11 1 1 -;
S10 = 00000 11 11 1 1 1;
S11 = 00000 11 00 1 1 -;
S12 = 11000 10 11 1 1 -;
S13 = 10000 10 10 1 1 -;
S14 = 11010 11 11 1 1 -;
S15 = 10000 11 10 1 1 -;
S16 = 11000 11 11 1 1 -;
S17 = 10001 11 11 1 1 -;
S18 = 00000 11 11 1 0 -;
```

```
*FLOW-TABLE
;-------------------------------------------------------------------------
RELEVANT = NCLR;
S[1..18], X 0 , F1; Reset
;-------------------------------------------------------------------------
RELEVANT = NCLR,TPT1ZERO,TZERO,BA,WQ;
;-------------------------------------------------------------------------
S1, X 1 --0- , F2  ; uP-Betrieb
S1, X 1 --1- , F10 ; auto-Betrieb
;-------------------------------------------------------------------------
S2, X 1 ---- , F3  ; MUXBA=uP-Betrieb, MUXTPT1=TP, FT-Arbeitsreg. laden
;-------------------------------------------------------------------------
S3, X 1 ---- , F4  ; FT,T,TP-Zaehler laden
;-------------------------------------------------------------------------
S4, X 1 -1-- , F18 ; Runtime-Error T<TP
S4, X 1 00-- , F4  ; TP-Zaehlerstand <> 0
S4, X 1 10-- , F5  ; TP-Zaehlerstand = 0
;-------------------------------------------------------------------------
S5, X 1 ---- , F6  ; Impulsdauer T1 laden
;-------------------------------------------------------------------------
S6, X 1 -1-- , F18 ; Runtime-Error T<T1
S6, X 1 00-- , F6  ; T1-Zaehlerstand <> 0
S6, X 1 10-- , F7  ; T1-Zaehlerstand = 0
;-------------------------------------------------------------------------
S7, X 1 ---- , F8  ; Pausendauer TP laden
;-------------------------------------------------------------------------
S8, X 1 -1-- , F18 ; Runtime-Error T<TP
S8, X 1 00-- , F8  ; TP-Zaehlerstand <> 0
S8, X 1 10-- , F9  ; TP-Zaehlerstand = 0
;-------------------------------------------------------------------------
S9, X 1 -101 , F2  ; T-Zaehlerstand = 0, uP-Betrieb, Schreibzyklus
S9, X 1 -100 , F3  ; T-Zaehlerstand = 0, uP-Betrieb, kein Schreibzyklus
S9, X 1 -0-- , F9  ; T-Zaehlerstand <> 0
S9, X 1 -11- , F10 ; T-Zaehlerstand = 0, auto-Betrieb
;-------------------------------------------------------------------------
S10,X 1 ---- , F11 ; MUXBA=auto-Betrieb, MUXTPT1=TP
;-------------------------------------------------------------------------
S11,X 1 ---- , F12 ; FT,T,TP-Zaehler laden
;-------------------------------------------------------------------------
S12,X 1 0---,  F12 ; TP-Zaehlerstand <> 0
S12,X 1 1---,  F13 ; TP-Zaehlerstand = 0
;-------------------------------------------------------------------------
S13,X 1 ----,  F14 ; Impulsdauer T1 laden
;-------------------------------------------------------------------------
S14,X 1 0---,  F14 ; T1-Zaehlerstand <> 0
S14,X 1 1---,  F15 ; T1-Zaehlerstand = 0
;-------------------------------------------------------------------------
S15,X 1 ----,  F16 ; Pausendauer TP laden
;-------------------------------------------------------------------------
S16,X 1 0---,  F16 ; TP-Zaehlerstand <> 0
S16,X 1 1---,  F17 ; TP-Zaehlerstand = 0
;-------------------------------------------------------------------------
S17,X 1 -10-,  F2  ; T-Zaehlerstand = 0, uP-Betrieb
S17,X 1 -11-,  F11 ; T-Zaehlerstand = 0, auto-Betrieb
S17,X 1 -0--,  F17 ; T-Zaehlerstand <> 0
;-------------------------------------------------------------------------
S18,X 1 ----,  F1  ; Fehlerbit setzen

*STATE-ASSIGNMENT
Z-VALUES;

*FANOUT
 $XALL=3;
 $YALL=6;

*RUN-CONTROLL
 STAGE-LIMIT = 2..20;
 STOP=BEST;
 TRANSFORMATION = NO;
 GATELIB = GF;
 LISTING = EQUATIONS, CROSS-REFERENCE, GATE-PLOT;
*END
```

6. **Fachhochschule Ulm**
**Fachbereich Industrieelektronik**

# Integration eines 4 Bit D/A-Wandlers auf dem Analog-Gatearray B500A

Verfasser : Georg Bisinger

Betreuer : Prof. Arnold Führer

Präsentation : Workshop an der FH Offenburg
15. Juni 1990

**Inhaltsverzeichnis**

## 1. Vorwort

Der Digital-Analog-Wandler ist das Bindeglied zwischen der digitalen Signalverarbeitung und der natürlichen Umwelt mit ihren rein analogen Signalen. Mit der immer stärker fortschreitenden Digitalisierung der gesamten Kommunikationstechnik werden die Aufgabengebiete von D/A-Wandlern immer vielseitiger und anspruchsvoller. Die Fortschritte auf dem Gebiet der hochintegrierten Schaltkreise, insbesondere die Mikroprozessortechnik, erweitern die Anwendungen der digitalen Signalverarbeitung erheblich. Oft werden analoge Anzeige und Steuersignale benötigt, die ein digitales System liefern muß.

Der im folgenden vorgestellte D/A-Wandler stellt sicherlich nicht das derzeitige Maximum in Bezug auf die Integration von D/A-Wandlern dar, von seinem Funktionsprinzip her ist er jedoch identisch mit vielen derzeit auf dem kommerziellen Markt erhältlichen Typen.
Durch seine modulare Struktur ist er, eine größere Chipfläche vorausgesetzt, problemlos nach allen Richtungen erweiterbar, so daß er leicht als eine Art Keimzelle für Weiterentwicklungen gelten kann.

Ulm, Juni 1990

Georg Bisinger
Fachhochschule Ulm

## 2. Aufgabenstellung

Es soll ein Digital/Analog-Wandler auf dem Analog-Gate-Array B500A der Firma AEG realisiert werden. Die Aufgabenstellung gliedert sich weiter grob in den Teil Minimalkonfiguration und weitergehende Funktionen des D/A-Wandlers.
Als Minimalkonfiguration sind folgende Punkte gefordert:

- Wortbreite 4 Bit
- Register außerhalb des Bausteines
- Referenzspannung wird von außen zugeführt
- als Operationsverstärker ist eine der bereits vorhandenen Schaltungen einzusetzen
- die Analogausgangsspannung liegt zwischen 0..5 V

Als erweiterte Funktionen sind genannt:

- Wortbreite 8 Bit
- Die Referenzspannung wird intern erzeugt
- Die Analogspannung liegt zwischen 0..10 V
- Abtasthalteglied am Ausgang

Leider ließ sich aus Platzgründen nur eine interne Referenzspannung realisieren.
Die Wortbreite von 8 Bit läßt sich aufgrund fehlender gleicher Widerstände in unmittelbarer Nähe für das R2R-Netzwerk nicht realisieren. Somit war es auch nicht sinnvoll die Ausgangsspannung auf 10 V zu erhöhen, da die Auflösung sonst zu grob gewesen wäre.

## 3. D/A-Wandler - verschiedene Arbeitsweisen

Es gibt 3 verschiedene Möglichkeiten einen D/A-Wandler zu realisieren.

### 3.1. Das Parallelverfahren

Prinzip: Es besteht aus einem Netzwerk mit dual gestuften Widerständen, das wiederum dual gestufte Ströme verursacht. Diese werden summiert und ergeben nach der I-U-Wandlung mit einem Operationsverstärker die analoge Ausgangsspannung.
Diese Variante eignet sich für integrierte Schaltkreise nur für sehr kleine Auflösungen, da bei der Integration nur ein relativ kleiner Widerstandsbereich zur Verfügung steht.

### 3.2. Das Wägeverfahren

Prinzip: Dieses Verfahren erlaubt ein Minimum an schaltungstechnischem Aufwand. Jedem Bit ist ein Schalter zugeordnet. Durch entsprechend gewichtete Widerstände wird der Ausgangsstrom erzeugt und durch einen als I-U-Wandler geschalteten Operationsverstärker in eine Spannung umgewandelt.
Durch Verwendung eines R2R-Netzwerkes lassen sich die Widerstandswerte auf zwei begrenzen. Aufgrund dessen ist dieses Verfahren für die Integration bestens geeignet und wurde im vorliegenden Fall verwendet.

### 3.3. Das Zählverfahren

Prinzip: Es erfordert nur einen einzigen Schalter. Er wird periodisch geöffnet und geschlossen. Sein Tastverhältnis wird mit Hilfe eines Vorwärtszählers so eingestellt, daß der Mittelwert der Ausgangsspannung den gewünschten Wert annimmt. Der große Nachteil dieses Verfahren ist, daß sich die Ausgangsspannung wegen des Tiefpaßes nur sehr langsam verändert. Deshalb wird es hauptsächlich in Frequenzmessern verwendet.

## 4. Das Analogarray B500A

Das B500A ist Mitglied einer 3er Familie von Analogarrays der Firma AEG. Die Familie besteht aus den Typen B250A, B500A und B1000A, wobei die Chips gleichartige Bauelemente aufweisen, aber in der Komplexität differieren.
Für die vorliegenden Arbeit stand nur das B500A zur Verfügung und deshalb sind im folgenden alle Angaben auf diesen Chip bezogen.

Auf bereits gefertigten Siliziumwafern stehen NPN-Bipolartransistoren mit einer Transitfrequenz von 500 MHz und PNP-Bipolartransistoren mit einer Transitfrequenz von 5 MHz, Widerstände, Z-Dioden und Kondensatoren in fester Anordnung zur Verfügung die mit einer speziell entwickelten Maske miteinander verbunden werden.

Die Bauelemente sind größtenteils in Gruppen zu Zellen zusammengefaßt. Es stehen 6 gleiche Standardzellen zur Realisierung von Grundschaltungen bzw. zur freien Verdrahtung zur Verfügung. Außerdem gibt es noch 3 Zellen die sich, aufgrund ihres Bauelementvorrats besonders zur Aufnahme einer Regierzelle, einer Bandgapzelle und einer Stromquellenzelle eignen. Prinzipiell gilt jedoch, daß der gesamte Chip absolut frei verdrahtet werden kann und nur durch einige Layoutregeln eine Beschränkung der freien Verdrahtung auftritt.

Ein großer Vorteil der Integration von Bauteilen auf einem Chip ist, daß alle gleichartigen Bauteile dieselbe Toleranz bzw. denselben Temperaturkoeffizienten aufweisen; dies ist in diesem Fall, vor allem beim R2R-Netzwerk, von außerordentlicher Bedeutung (siehe Seite 9).
Als großes Handikap bei diesem Array ist die niedere Transitfrequenz der PNP-Transistoren von 5MHz zu bewerten, die eine Realisierung von Schaltungen mit höheren Frequenzen ausschließt. Diese geringe Frequenz kommt daher, daß der Chipherstellungsprozeß auf NPN-Transistoren hin optimiert ist.

## 5. D/A-Wandler

### 5.1 Blockschaltbild

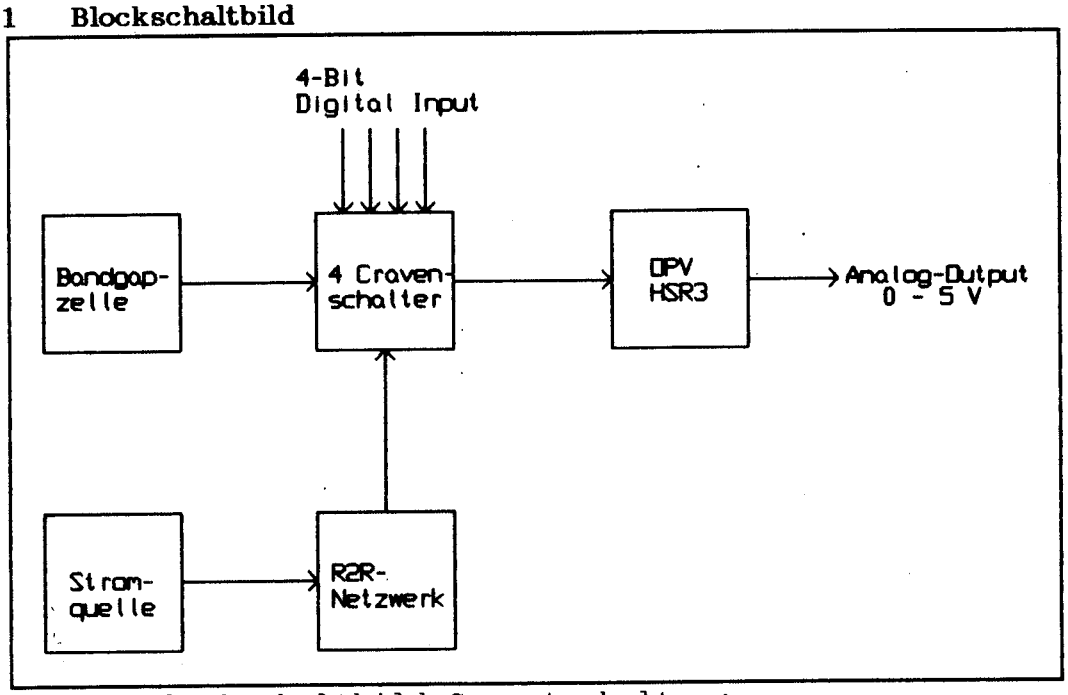



**Abb. 1** Blockschaltbild Gesamtschaltung

Schaltungsfunktion

Die Gesamtschaltung ist nach dem Prinzip eines R2R-Netzwerkes im stromschaltenden Mode mit einem nachgeschalteten Operationsverstärker als Strom-Spannungswandler aufgebaut .
Über die Cravenschalter (siehe Seite 12) wird, abhängig vom Wert des jeweils anliegenden Bits ausgewählt, ob ein Teilstrom der Stromquelle von Masse über den Operationsverstärkereingang oder an diesem vorbei gegen -10V fließt. Am Operationsverstärker addieren sich die Teilströme und werden von diesem in eine proportionale Spannung umgesetzt.

## 5.2. Gesamtschaltung

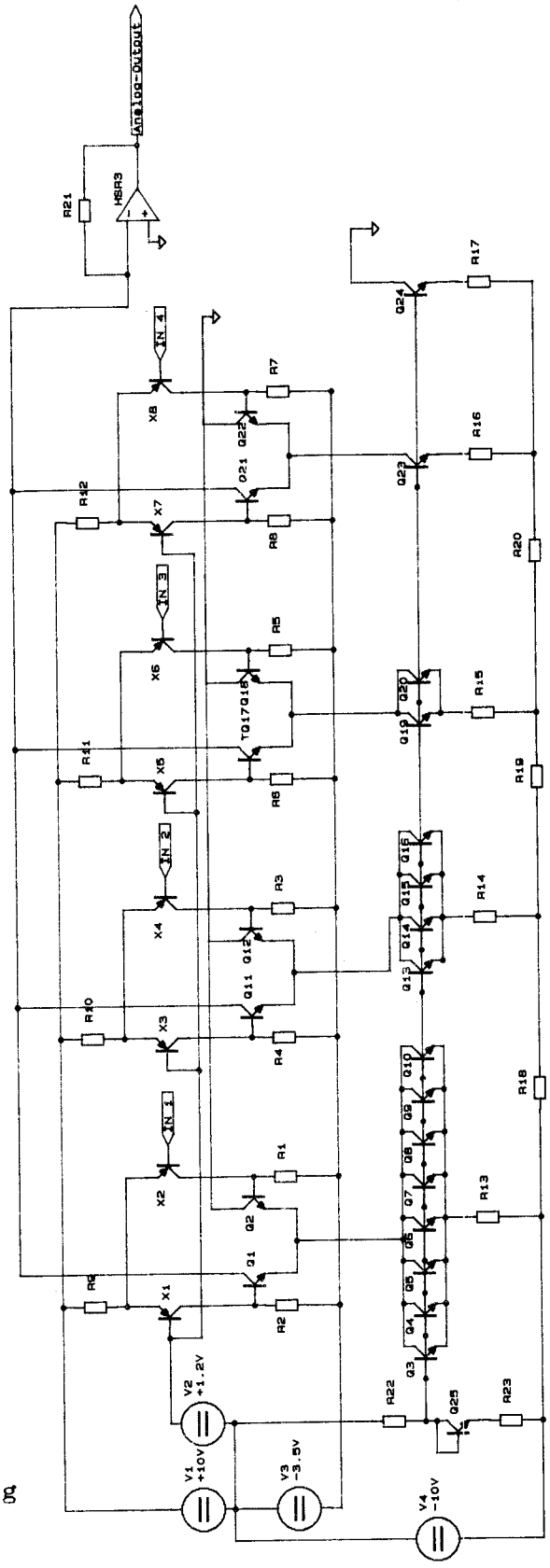



**Abb. 1** Gesamtschaltung

## 5.3 R2R-Netzwerk

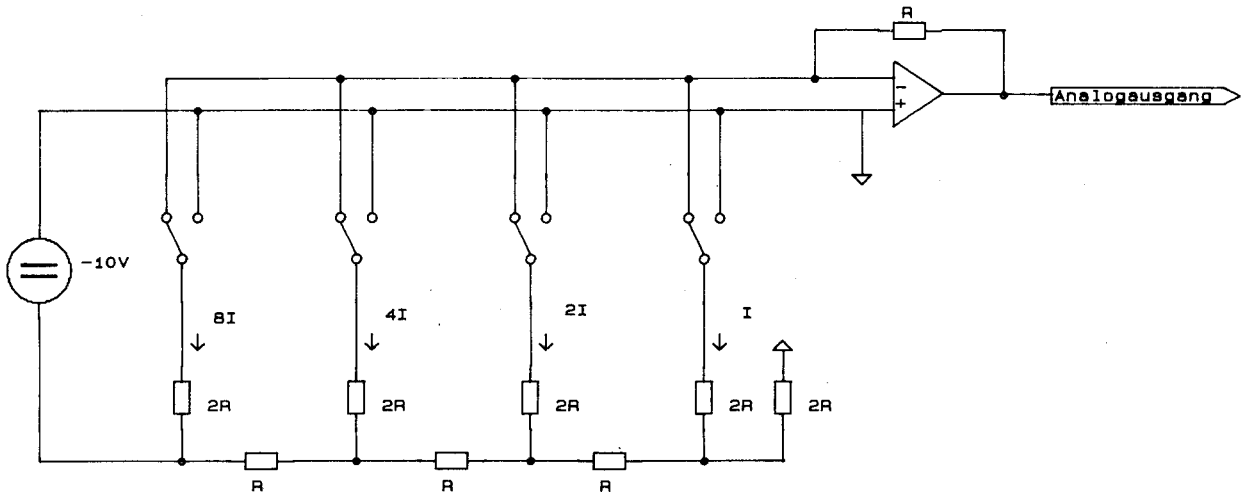



**Abb. 2** R2R-Netzwerk

Funktion

Das Grundelement des Netzwerkes stellt einen belasteten Spannungsteiler mit folgenden Eigenschaften dar:

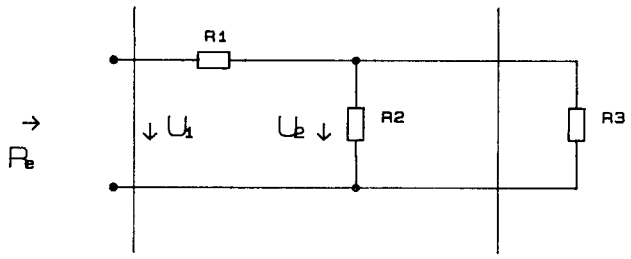



**Abb. 3** Aufbau eines Gliedes im Netzwerk

## 5.5 Bandgap

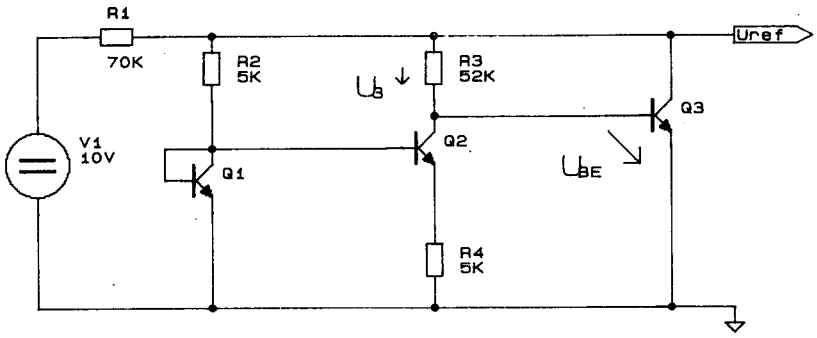



**Abb. 5** Bandgap-Referenzspannungsquelle

Funktion

Die Ausgangsspannung ergibt sich wie folgt:

$$U_{ref} = U_3 + U_{BE}$$

$U_{BE}$ hat negativen Temperaturkoeffizient
$U_3$ hat positiven Temperaturkoeffizient

Daraus ergibt sich bei geeigneter Dimensionierung für $U_{ref}$ einen TK = 0[1]

[1] Literaturhinweis: Buch [4]

## 5.6 Cravenschalter

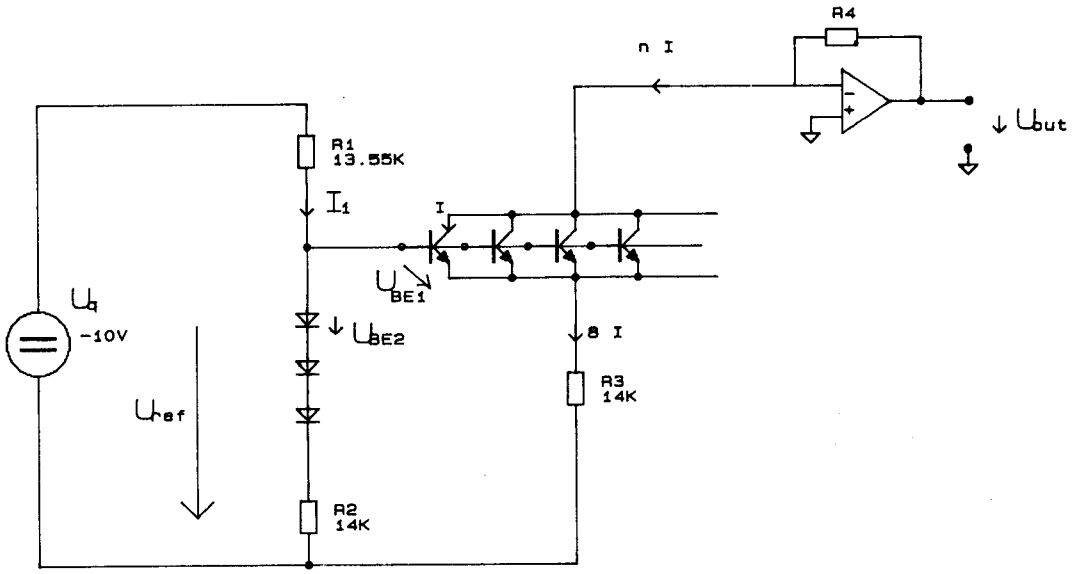



**Abb. 6** Cravenschalter

Funktion:

Die Cravenzelle dient als Schalter-Element. Ein NPN- und ein PNP-Transistorpaar in einer Differenzschaltung bilden den Stromschalter. Das PNP-Paar (X1,X2) dient zur Pegelanpassung und steuert das stromschaltende NPN-Paar (Q1,Q2). Jeder Stromzweig besitzt ein Schaltelement (Zelle) mit Ausnahme des Endwiderstandes im Leiternetzwerk. Der Transistor Q3 erzeugt einen gewichteten Strom entsprechend der Position im Leiternetzwerk. Der binär gewichtete Kollektorstrom (Q3) fließt zum Transistorpaar Q1-Q2, wobei nur ein Transistor leitend ist. Ist das Steuersignal am Logikeingang H, so leitet Q1. Ist das Steuersignal L, so leitet Q2 und Q1 ist gesperrt. Der Strom fließt über den Ausgang OUT2 bzw. Masse. Der Ausgang Out2 wird mit Masse verbunden damit die Stromquellenfunktion von der Schalterstellung unabhängig ist.

## 5.7 Operationsverstärker HSR3

Der Operationsverstärker HSR3 wurde in einer früheren Diplomarbeit entwickelt und stand für diese Aufgabe zur Verfügung. Er mußte jedoch in die Mentor-Entwicklungsumgebung übernommen und erneut simuliert werden um seine Funktion sicherzustellen.

Der HSR3 zeichnet sich für die Aufgabe als Strom-Spannungsumsetzer besonders durch seine hohe Slewrate von bis zu 17 V/µs bei einem Eingangssprung von 10.6 V/µs aus.

## 5.8 Technische Daten des D/A-Wandlers

| | | | |
|---|---|---|---|
| 1. | Ausgangsspannung | : | 0..5 V |
| 2. | Auflösung | : | 0,333 mV |
| 3. | Temperaturoffset | | |
| 4. | bei $\Delta T$ = 73 K | : | 2,56 mV |
| | entspricht | : | 0,035 mV/K |
| 5. | Offsetfehler | : | 2,211e-3 V |

-> Punkt 4 und 5 sind Simulationsergebnisse.

Der Verlauf der Ausgangsspannung ist in Anlage 1 [Analogausgangsspannung] dargestellt.
Aus Anlage 2 [Prozentuale Abweichung ...] geht die prozentuale Genauigkeit der Ausgangsspannung bezogen auf die Eingangsspannung hervor.

## 6. Entwicklungstools

Zur Verfügung stand eine komplette Entwicklungsumgebung der Firma Mentor Graphics.
Das System beinhaltet, bezogen auf diese Diplomarbeit, die Einzelprogramme Neted, Expand, Accusim und Chipgraph.
Die Programme sind für folgende Einsatzzwecke:

Neted : Programm zur Stromlaufplaneingabe.

Expand : Programm zum Expandieren der Schaltung.

Accusim : Programm zur Simulation von Analogen Schaltungen. Dieses Programm arbeitet mit denselben Parametern wie Spice.
Durch seine graphisch orientierte Benutzeroberfläche ist es in seinen Grundfunktionen relativ leicht erlernbar.
Information: Zur Berechnung des Verlaufs der Ausgangsspannung des D/A-Wandlers bei einer schrittweise Vergrößerung des Steuerwortes von 0000 auf 1111 bei T = 27°C und T = 100°C benötigt Accusim auf einer Apollo DN 3500 69 Minuten.

Chipgraph : Programm zum Erstellen der Layoutstruktur des Chips.
Hier ergibt sich jedoch folgendes Problem:
Von seiner Konzeption her ist Chipgraph zur Erstellung eines Full-custom-design gedacht. Somit mußte der B500A-Hintergrund zuerst erstellt werden um ein plazieren des Layout zu ermöglichen.
Leider gibt es bis zum jetzigen Zeitpunkt keinen Autorouter für einlagig metallisierte Analogschaltungen, so daß das gesamte routen von Hand zu erfolgen hat.

## 7. Ausblick auf Verbesserungs- und Weiterentwicklungsmöglichkeiten

Wie bereits in der Aufgabenstellung erwähnt ist die Priorität der Erweiterungen des D/A-Wandlers wie folgt:

- 8 Bit Wortbreite
- Die Analogspannung liegt zwischen 0..10 V
- Abtasthalteglied am Ausgang

Die erhöhte Wortbreite ist als wichtigster Punkt zu betrachten, da bei der derzeitigen Auflösung ein praktischer Einsatzzweck nur schwer zu finden sein dürfte. Mit der Realisierung der größeren Wortbreite ist natürlich auch die erhöhte Ausgangsspannung mit einer sinnvollen Auflösung zu realisieren.
Bei Vorgängen bei denen kein Überschwingen beim Datenwechsel auftreten darf ist der D/A-Wandler derzeitig aufgrund seiner Überschwinger nicht einsetzbar. Um diese störenden Einflüsse zu vermeiden ist am Ausgang ein Abtasthalteglied vorzusehen, was dann eine Beseitigung der Überschwinger gewährleisten würde.

## 8. Schwierigkeiten und Probleme

Das größte Problem bei der Schaltungsentwicklung war die Beseitigung der Temperatureinflüsse. Anfänglich bestand eine Spannungsänderung $\Delta U$ von $\approx$ 300mV bei einem Temperaturdifferenz $\Delta T$ = 73 K (27...100°C). Durch geeignete Beschaltung gelang aber dann die Reduzierung auf eine Spannungsänderung von $\approx$ 2,56 mV.

Anlagen:
- Ausgangsspannung D/A-Wandler
- Prozentuale Abweichung des D/A-Wandlers bezogen auf den Entwert

Literatur:

[1]- Halbleiter-Schaltungstechnik
Tietze-Schenk, Springer-Verlag

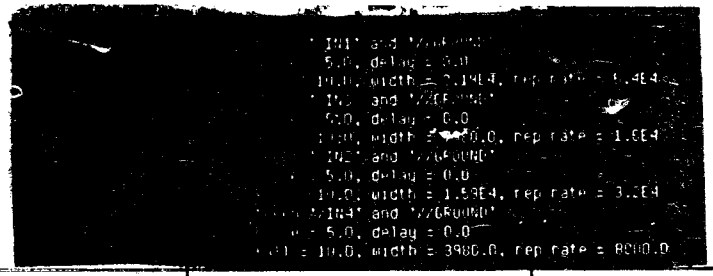
[2]- A/D- und D/A-Wandler
Eckl/Pütgens/Walter, Hüthig-Verlag

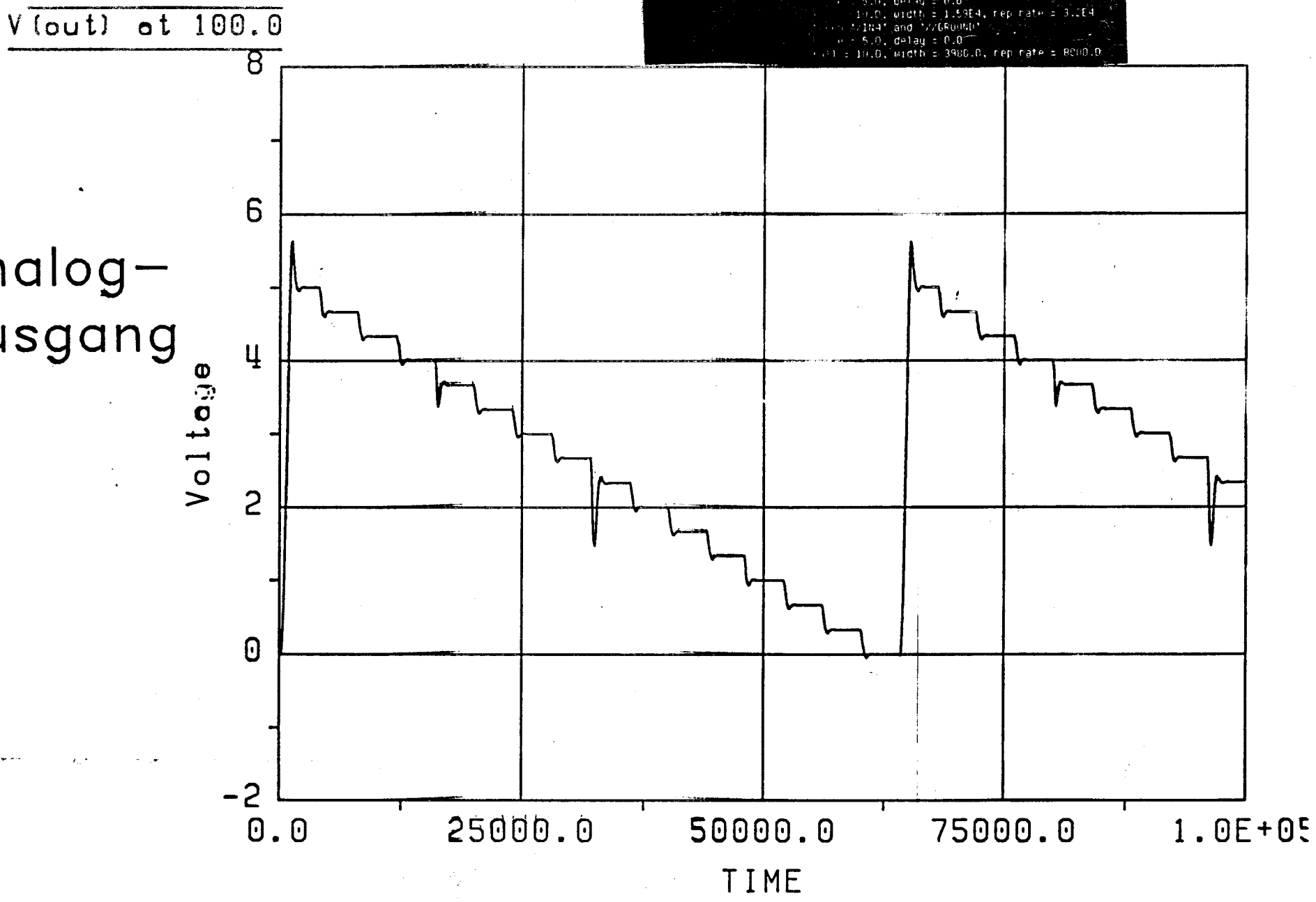
[3]- Analoge Schaltungen
Seifart, VEB-Verlag

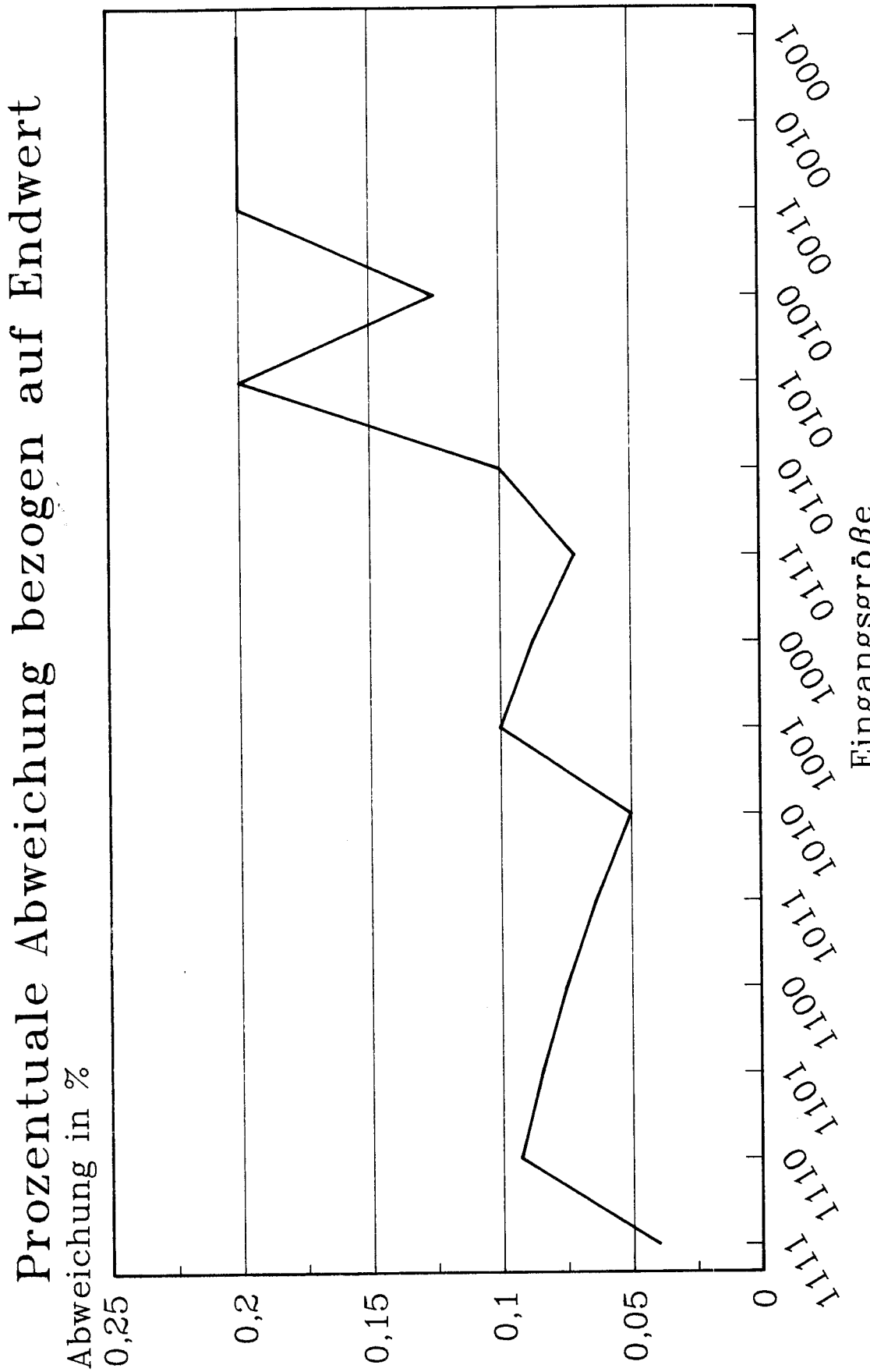
[4]- Elektronische Schaltungstechnik
Köstner/Möschwitzer, VEB-Verlag

[5]- Transistor-Elektronik
Rumpf/Pulvers, VEB-Verlag

# Analog–ausgang






Prozentuale Abweichung bezogen auf Endwert
Abweichung in %
0,25
0,2
0,15
0,1
0,05
0
1111 1110 1101 1100 1011 1010 1001 1000 0111 0110 0101 0100 0011 0010 0001 0000
Eingangsgröße
Berechnungsformel: Abweichung in % = (1- soll/ist) 100