FH Ulm
Labor für Schaltungsintegration

MULTIPROJEKT CHIP-GRUPPE

BADEN - WÜRTTEMBERG

WORKSHOP JULI 1992

FURTWANGEN

HERAUSGEBER: FACHHOCHSCHULE ULM

© 1992 Fachhochschule Ulm

Das Werk und seine Teile sind urheberrechtlich geschützt.
Jede Verwertung in anderen als den gesetzlich zugelassenen
Fällen bedarf deshalb der vorherigen schriftlichen Einwilli-
gung des Herausgebers.

I N H A L T S V E R Z E I C H N I S

8. Entwicklung eines hochintegrierten Bildverarbeitungs-Schaltkreises zur Echtzeit-Schwerpunktberechnung

H. Vogel, U. Jäger, H. Clauss
FH Heilbronn

9. Entwurf einer Schaltung zur Suche von Zeichensequenzen

H. Kreutzer
FH Reutlingen

W. Ludescher
FH Ravensburg-Weingarten

10. Entwicklung eines elektronischen Würfels und dessen Umsetzung in einen anwenderspezifischen Schaltkreis

B. Reinke, D. Jansen
FH Offenburg

11. Digitaler Phasenreglerkreis mit numerisch gesteuertem Oszillator als LCA-Microcontroller Kombination

D. Jansen, B. Reinke, A. Rendler
FH Offenburg

12. Entwurf eines Fernsteuer-IC's als Studienaufgabe

H. Töpfer
FHT Esslingen

TELEFUNKEN electronic

1. Moderne Testkonzepte für komplexe Schaltungen

Dr. W. Denner TELEFUNKEN electronic GmbH
7100 Heilbronn

Gliederung

TELEFUNKEN electronic

- Einleitung / Testproblematik
- Design for Testability (DFT)
- Scan-Design Technik
- Integrierter Selbsttest (BIST)
- Boundary-Scan Test (BST)
- DFT bei Analog-Schaltungen
- Zusammenfassung / Ausblick

Testproblematik

TELEFUNKEN electronic

- Steigende Komplexität
- Limitierte Pinzahl
- Größere sequentielle Tiefe
- Höhere Qualitätsanforderungen
- Überproportionaler Testkostenanstieg
- Sinkende Bauelementpreise

TELEFUNKEN electronic

Mittlere Fehlerhäufigkeit pro IC

$$FH = 1 - Y^{(1 - TC)}$$

FH: mittlere Fehlerhäufigkeit

Y : Ausbeute (Yield)

TC: Fehlerüberdeckung (test coverage)

Beispiele:	FH	Y	TC
	1,8%	70%	95%
	0,1%	70%	99,7%

TELEFUNKEN electronic

'single stuck-at' Fehlermodell

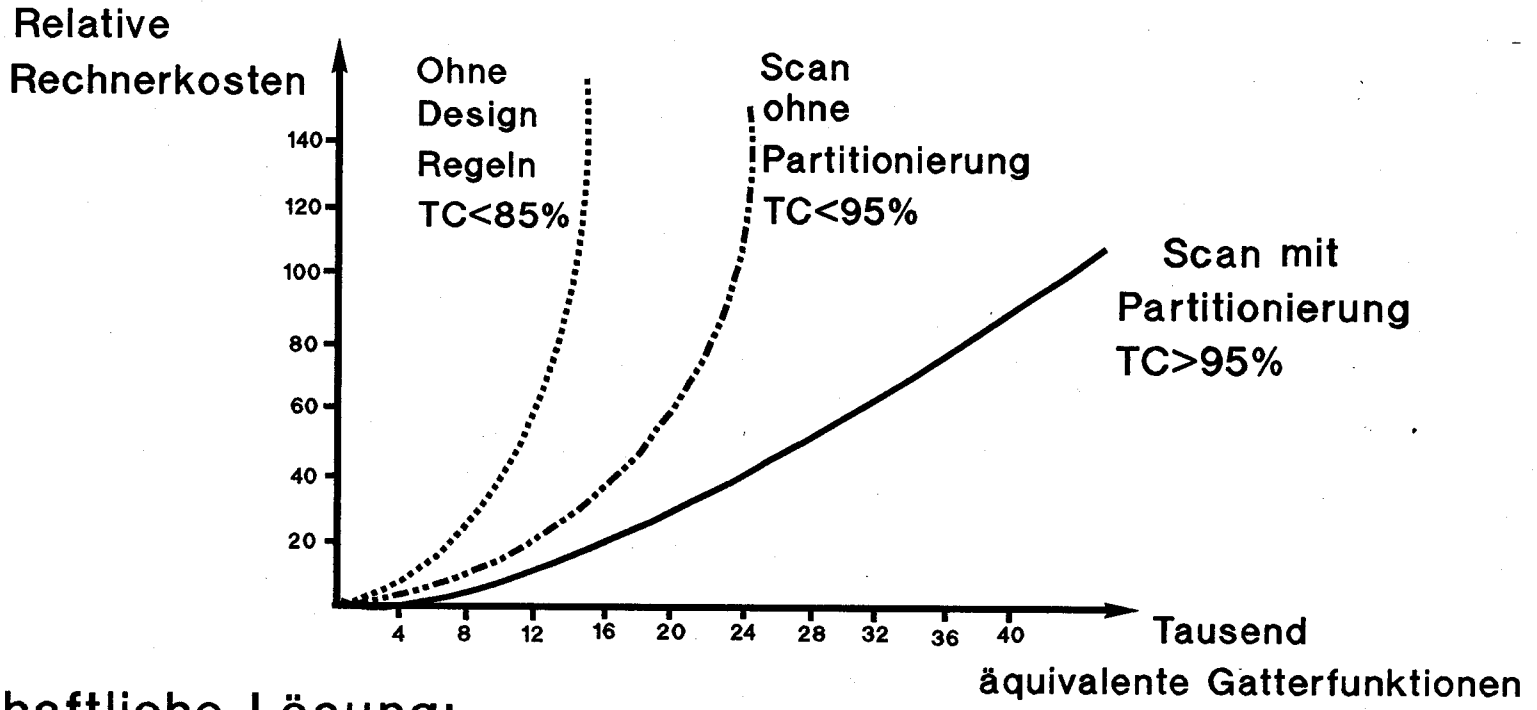
- Nur Berücksichtigung von Herstellungsfehlern, durch deren Auswirkung interne Knoten auf festem Potential gehalten werden.
  - Knoten konstant auf 'high' → 'stuck-at one' (sa1)
  - Knoten konstant auf 'low' → 'stuck-at zero' (sa0)

- Fehler treten nur als Einzelfehler auf.

Testproblematik

TELEFUNKEN electronic

Kosten für Prüfvorbereitung steigen exponentiell mit der Komplexität an.



Wirtschaftliche Lösung:

Prüffreundlicher Schaltungsentwurf

TELEFUNKEN electronic

Prüffreundlicher Schaltungsentwurf

Integration ***zusätzlicher*** logischer Funktionen, die den Test ***unterstützen*** oder erst ***ermöglichen,*** jedoch für die Systemfunktion ***unbedeutend*** sind.

TELEFUNKEN electronic

Konzepte des Prüffreundlichen Entwurfs

Verbesserung von: ***Beobachtbarkeit*** (observability)

Kontollierbarkeit (controllability)

Initialisierbarkeit (predictability)

Beobachtbarkeit: wie einfach kann ein internes Signal nach außen geführt werden?

Kontrollierbarkeit: wie einfach kann ein internes Signal von außen gesteuert werden?

Initialisierbarkeit: wie einfach kann eine sequentielle Schaltung in einen definierten Zustand gebracht werden?

einfach: wenig Testmuster; wenig Testzyklen; Testmuster leicht ableitbar

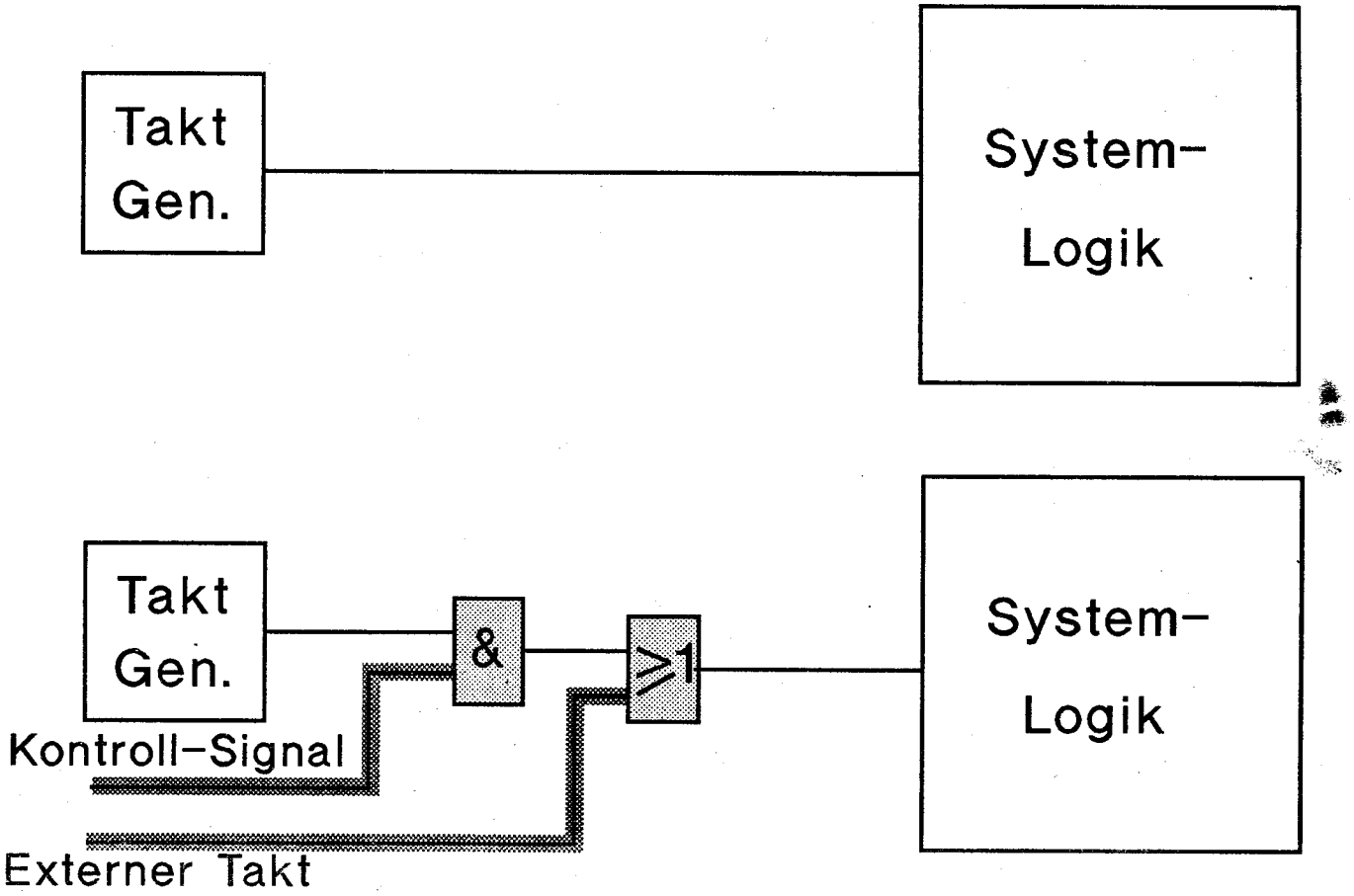
TELEFUNKEN electronic

Nachträglich einzubringende Maßnahmen

- Teilen nach elektrischen oder logischen Gesichtspunkten.
- Zusätzliche Prüfpunkte einführen.
- Multiplexer oder Schieberegister einbauen.

Geringe Effektivität
Teuer
Nicht automatisierbar

Teilen nach logischen Gesichtspunkten



Design for Testability

Vor dem Entwurf zu berücksichtigende Maßnahmen

- Partitionierung
- Busorientierte Architektur
- Reguläre Struktur / Random Logik
- Scan-Design Technik
- Selbsttest / Integrierte Testmuster-Generierung
- Boundary-Scan Technik

Große Effektivität
Automatisierbar

Preis für DFT

- größere Chipfläche
- größerer Designaufwand

Nutzen von DFT

- transparentere Schaltung
- vorhandene CAD-Tools einsetzbar
- höhere Testqualität
- kürzere Testzeiten

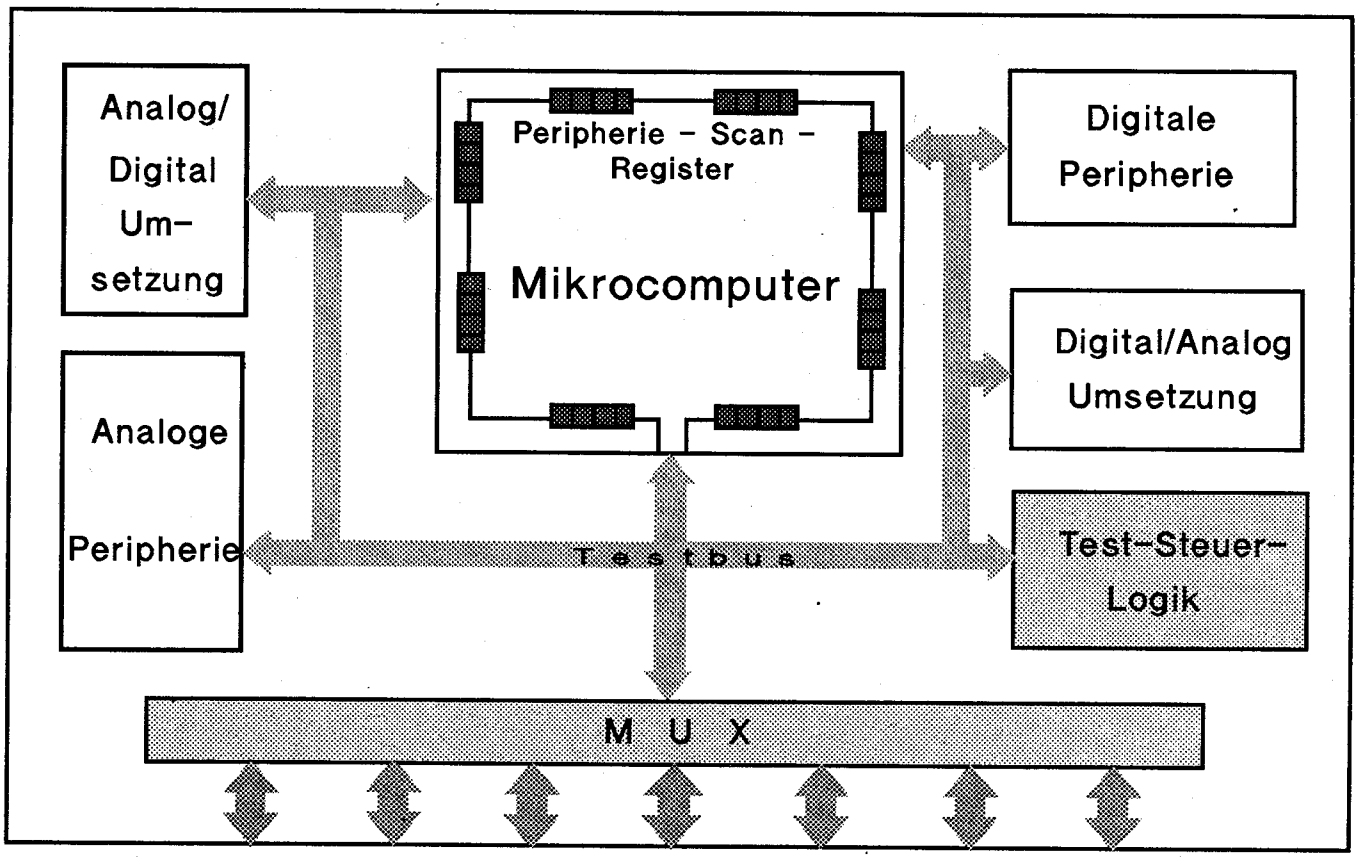
TELEFUNKEN electronic

Hauptkriterien für Partitionierung

- Analoge und digitale Bereiche müssen separat testbar sein.
- Speicherbereiche müssen zu Testzwecken separierbar sein.
- Jeder Teilbereich muß unabhängig von der umgebenden Logik testbar sein.
- Innerhalb eines Teilbereichs ist nur ***ein*** Systemtakt zulässig, der während des Tests extern steuerbar ist.

TELEFUNKEN electronic

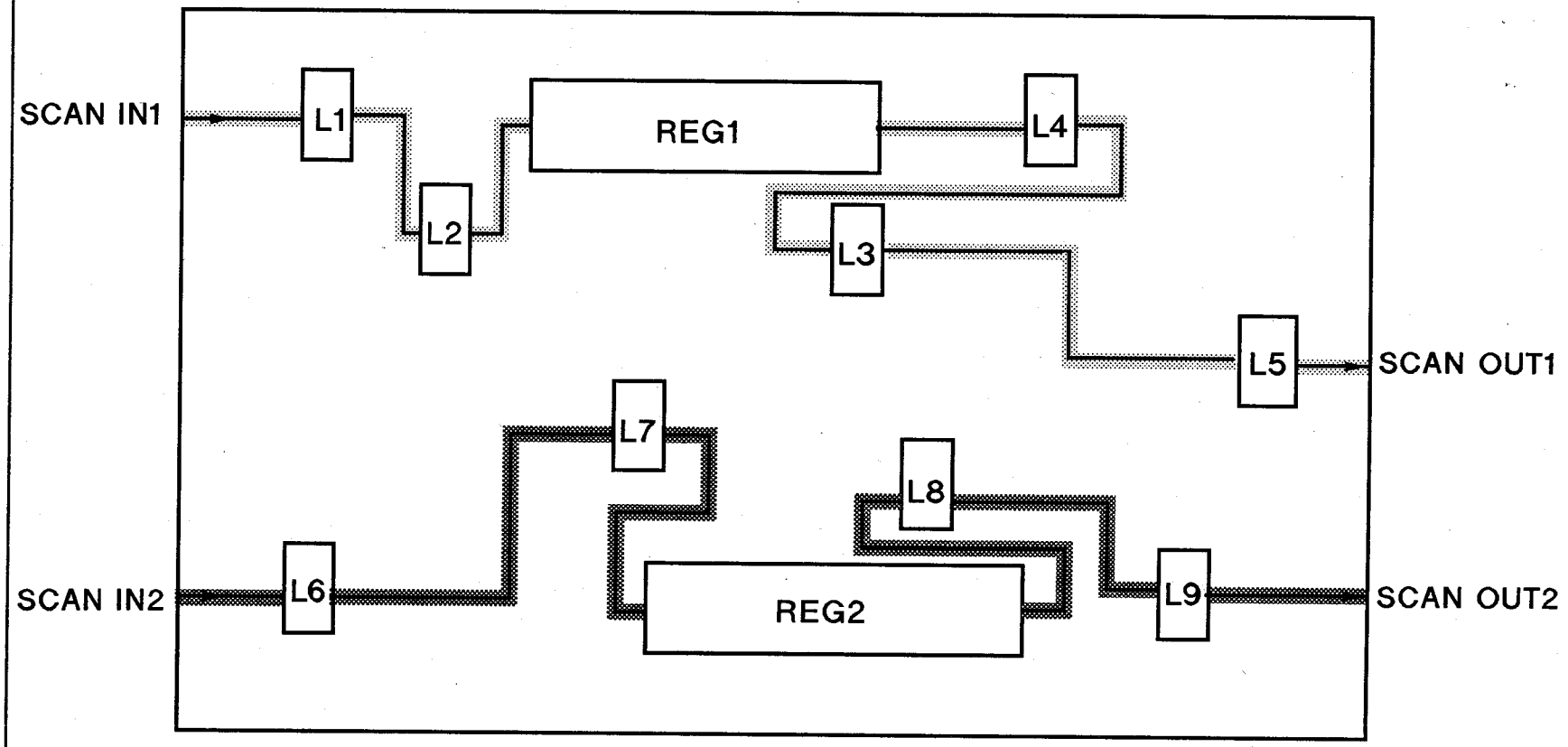
Blockschaltbild



Externe Pins

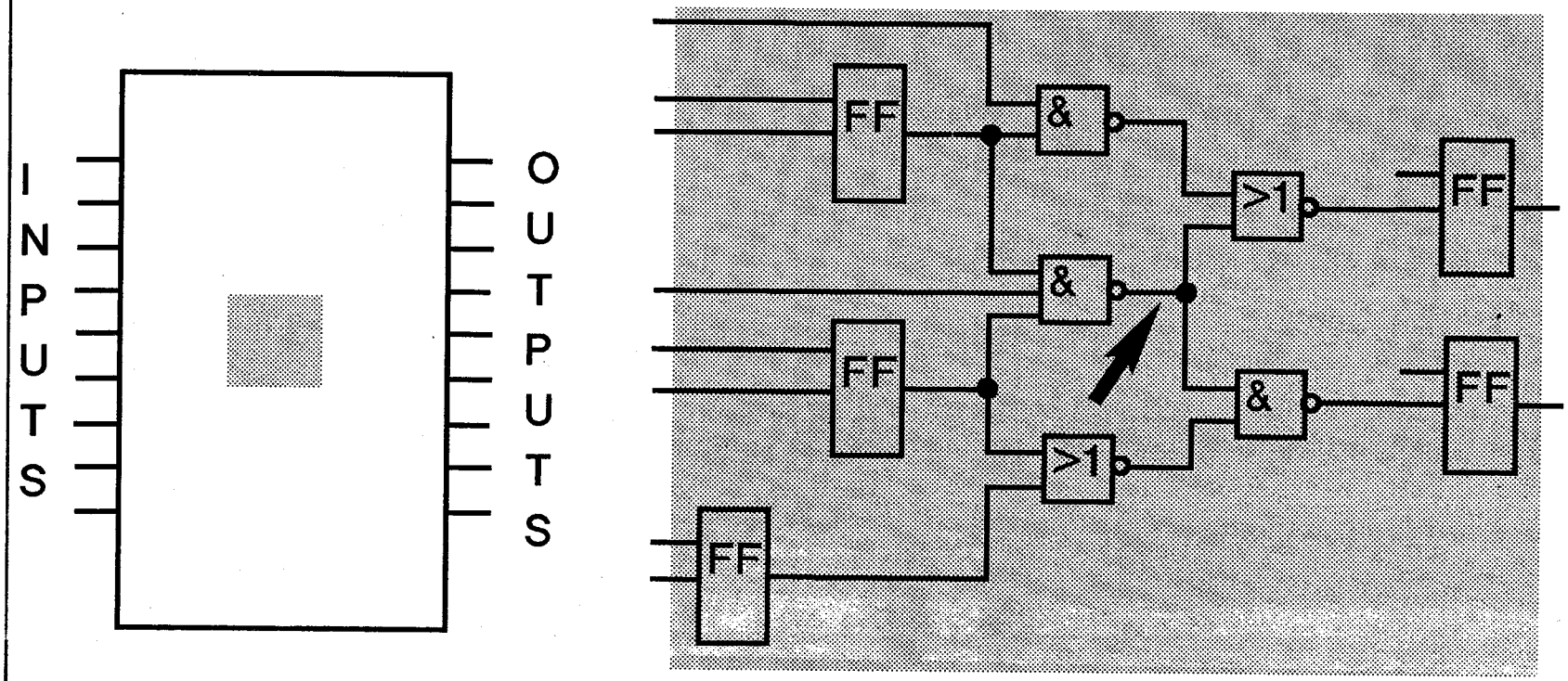
TELEFUNKEN electronic

Prinzip der Scan-Design Technik



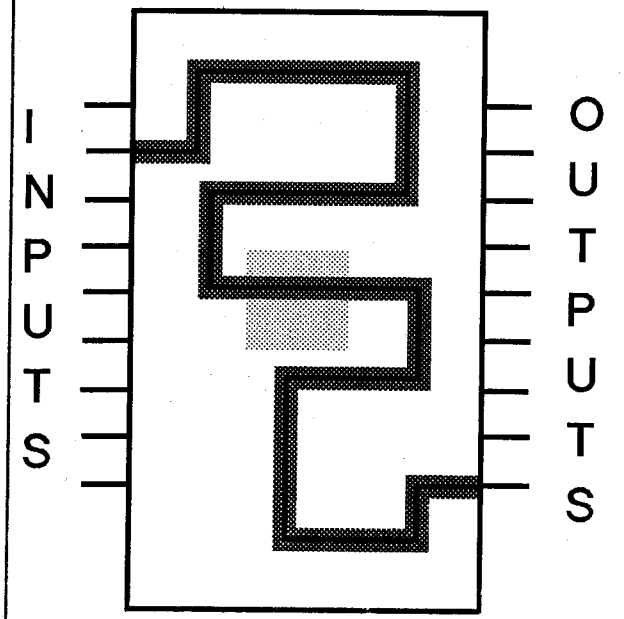
TELEFUNKEN electronic

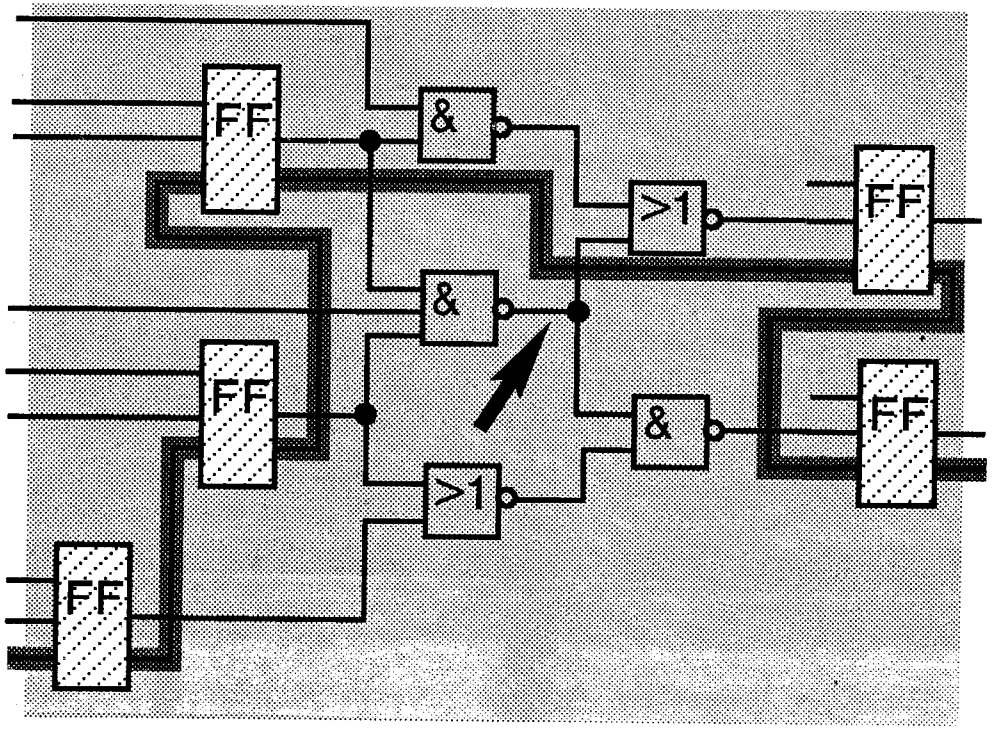
Einfluß auf Kontrollierbarkeit und Beobachtbarkeit



TELEFUNKEN electronic

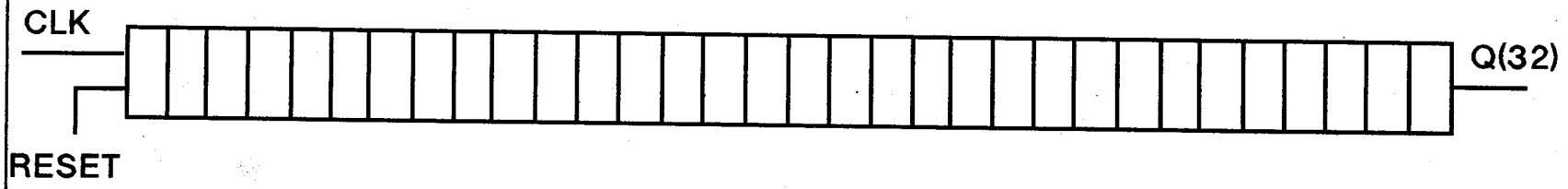
Einfluß auf Kontrollierbarkeit und Beobachtbarkeit



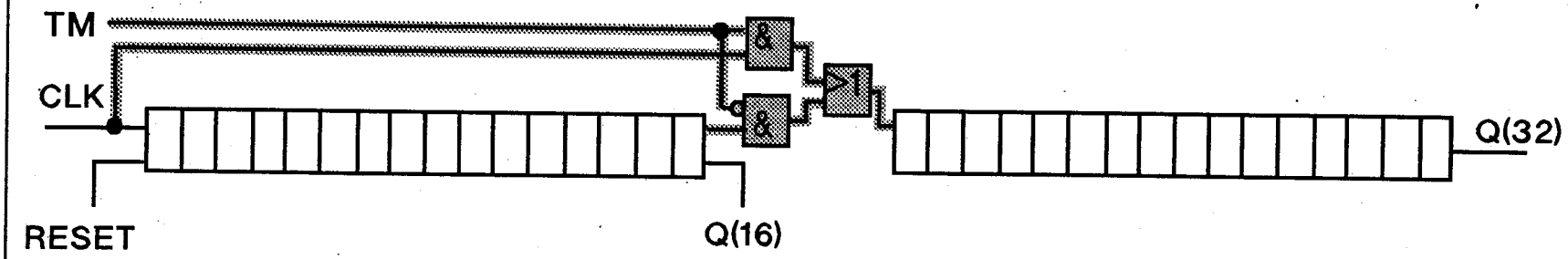


TELEFUNKEN electronic

Testkonzepte für 32-Bit Zähler



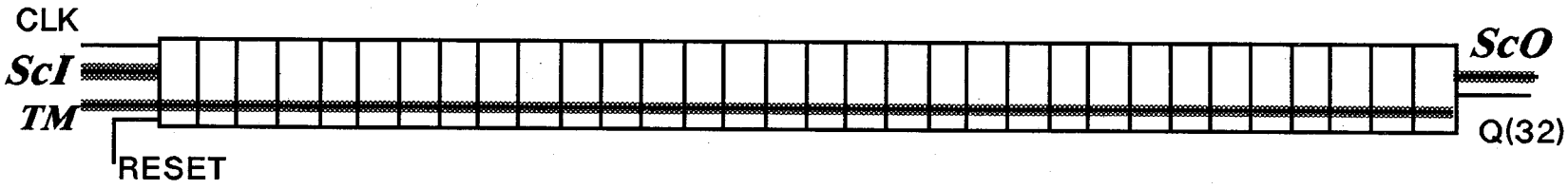
Test: 2^{32} *CLK Takte* ($4.295 * 10^{9}$ Takte) i.e. 429sec bei 10 MHz



Test: 2^{16} *CLK Takte* in TM, i.e 6,5msec bei 10 MHz

TELEFUNKEN electronic

Scan-Test bei 32-Bit Zähler



Test: ***920 CLK Takte*** bei Scan-Test, i.e. 92 usec (größere Zählerstufe)

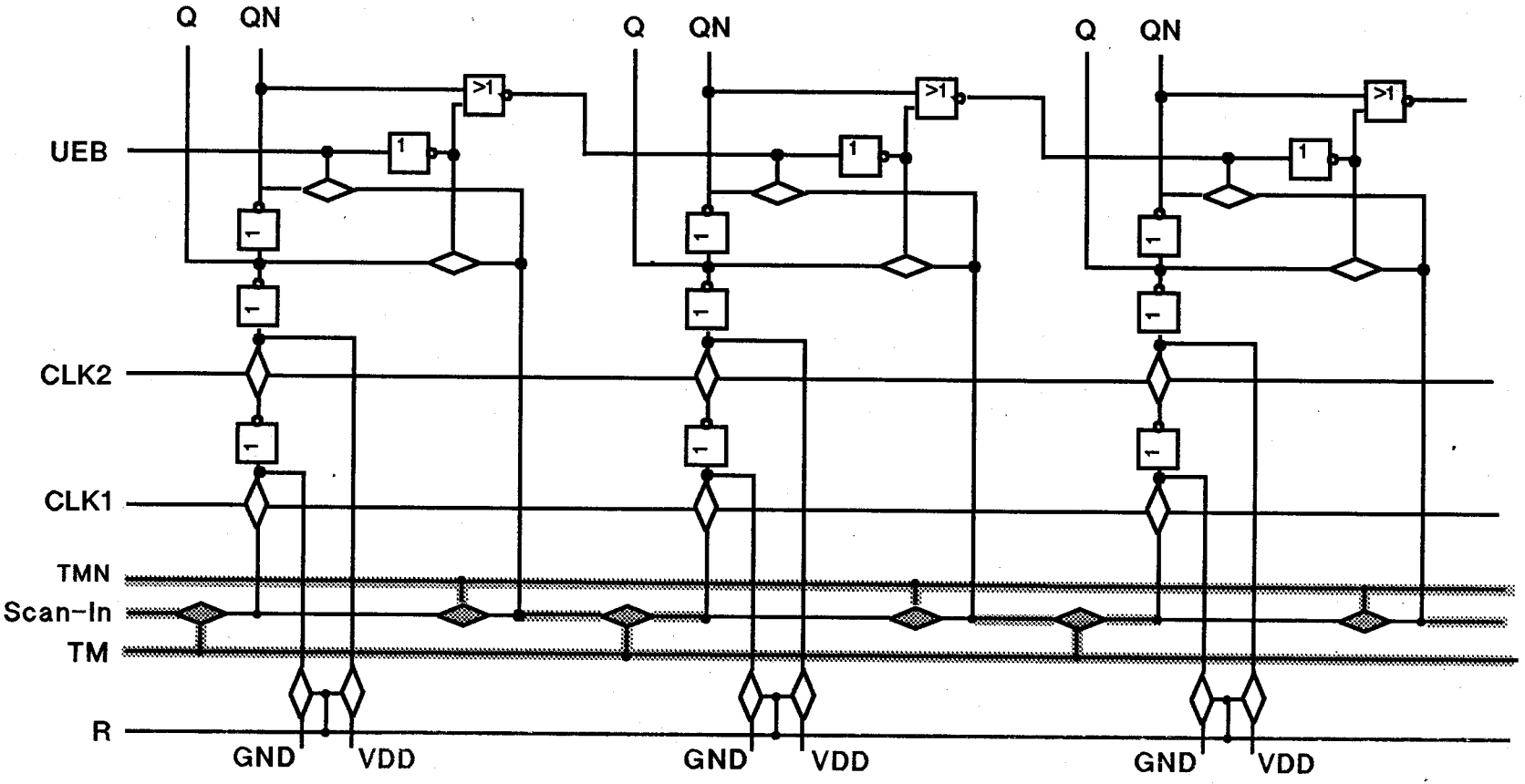
Testablauf:

TM	Zähltakte	Schiebetakte	Zählerinhalt
0	128		00000001000000000000000000000000
1		32	11111111000000000000000000000000
0	1		00000000100000000000000000000000
1		32	11111111100000000000000000000000
0	1		00000000010000000000000000000000
1		32	11111111110000000000000000000000

Taktzahl: 128 + 24*(32+1) = 920

TELEFUNKEN electronic

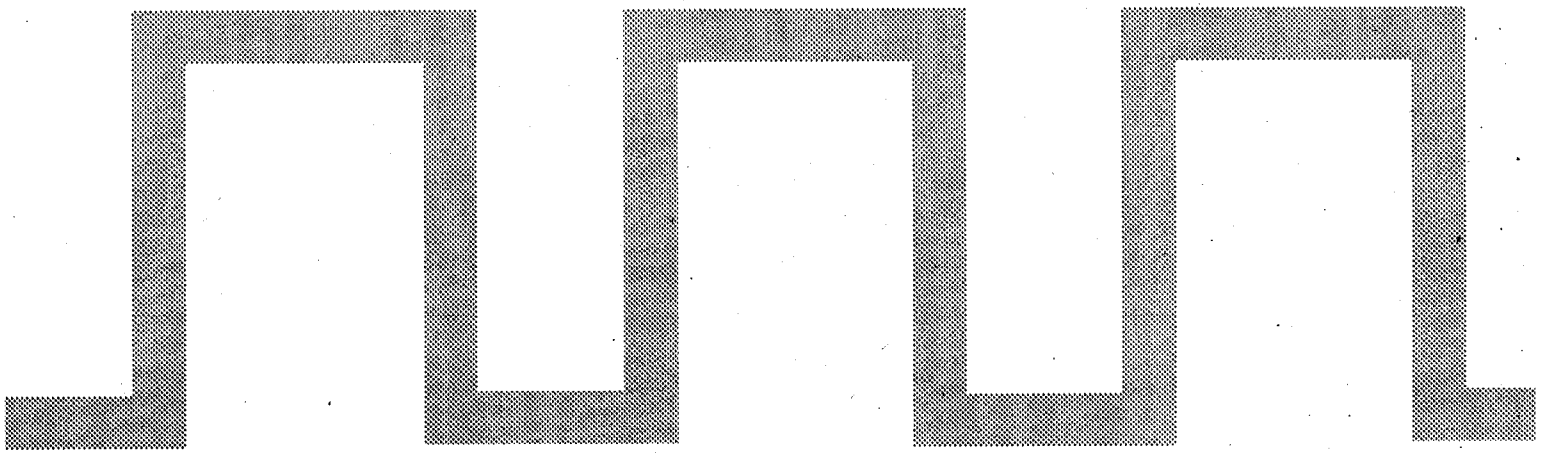
Beispiel für Scan-fähigen Zähler



Zusatz-Logik: ***2 Transistoren*** pro Zählerstufe

TELEFUNKEN
electronic

S



TELEFUNKEN electronic

Preis für DFT

- größere Chipfläche
- größerer Designaufwand

Nutzen von DFT

- transparentere Schaltung
- vorhandene CAD-Tools einsetzbar
- höhere Testqualität
- kürzere Testzeiten

TELEFUNKEN electronic

Selbsttest-Konzepte

- Geeignete Testsequenz speicherresident in 'on-chip / on-board' ROM
- Geeignete Testsequenz wird vor Test in 'on-chip / on-board' Speicher geladen
- Pseudo-random Patterngenerierung

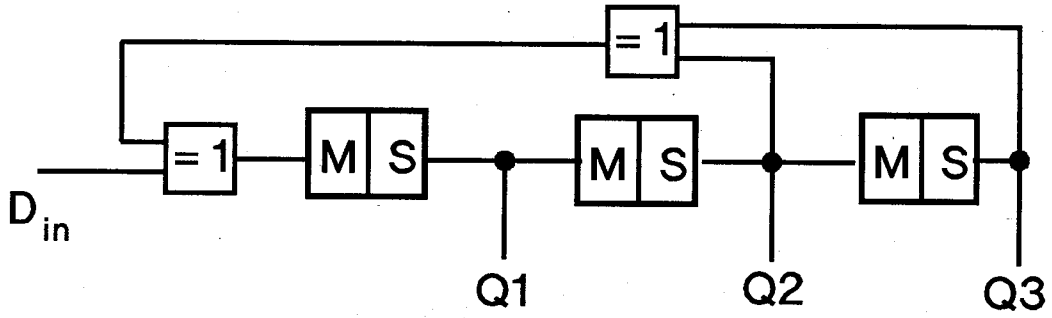
TELEFUNKEN electronic

'On-chip' Pseudo-random Patterngenerierung

- Sehr lange Testsequenzen
- Zu hohe Systemfrequenz für Testsysteme
- Einsatz von 'low-cost' Testsystemen erforderlich
- Selbsttest bei 'Power-up'

TELEFUNKEN electronic

Integrierte Testmuster-Generierung

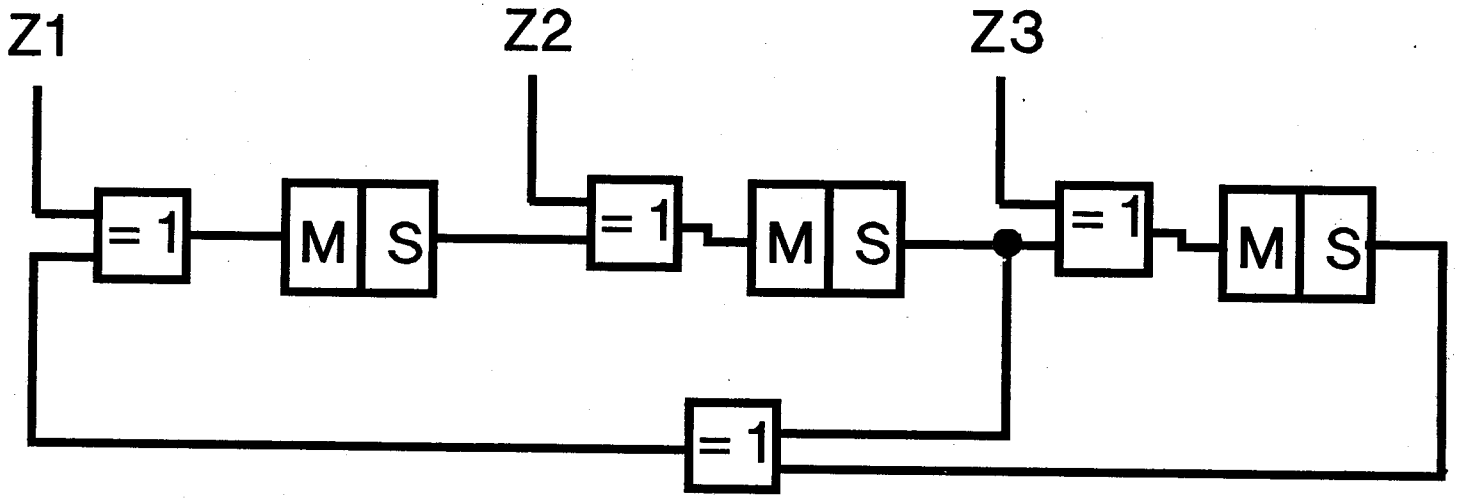


Testmuster-Sequenz:

Nr	Din	Q1	Q2	Q3
1	0	0	0	1
2	0	1	0	0
3	0	0	1	0
4	0	1	0	1
5	0	1	1	0
6	0	1	1	1
7	0	0	1	1
Absorbing state: (8	0	0	0	0)

TELEFUNKEN electronic

Signaturbildung



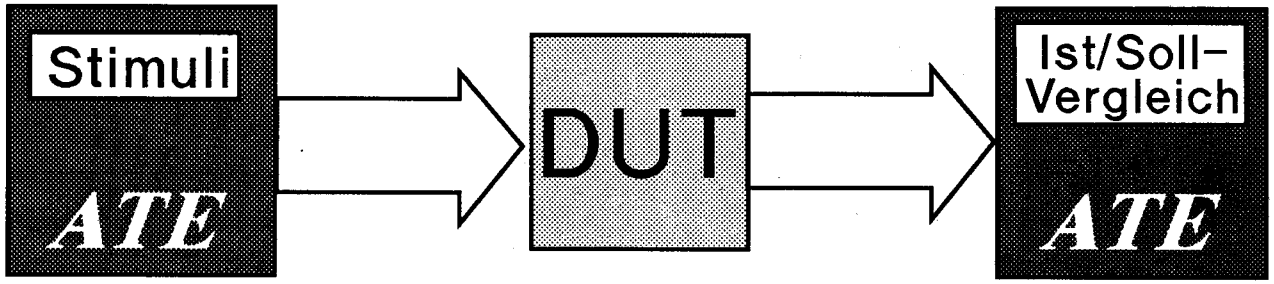
$$S = (1 - 2^{-n}) * 100\%$$

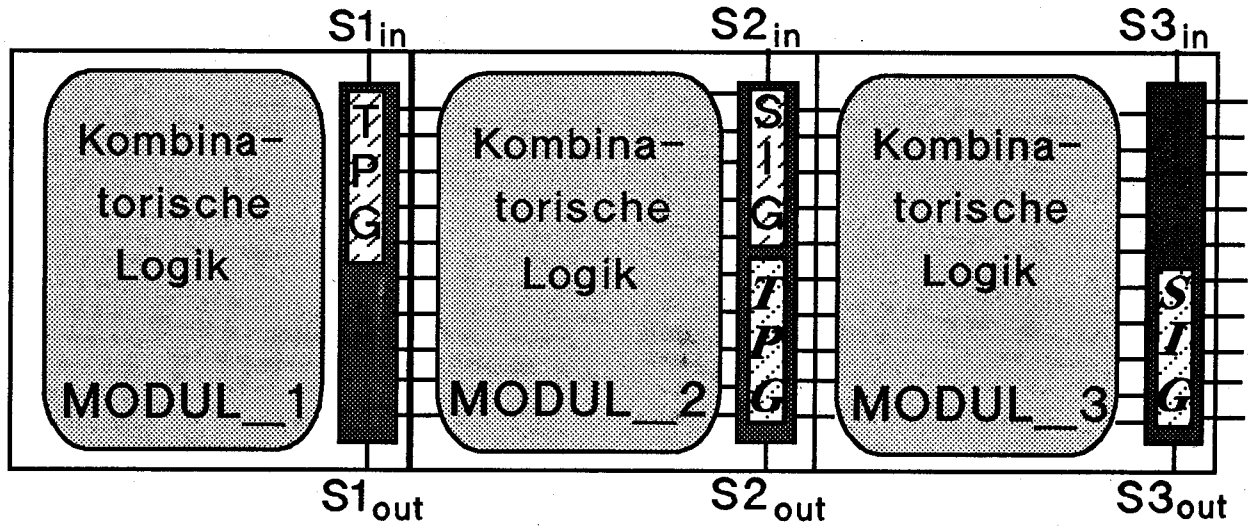
S: Aussage-Sicherheit n: Register-Länge

Beispiel: n = 8 → S = 99,6%

Integrierter Selbsttest

Selbsttest-Konzept





TELEFUNKEN electronic

Bisheriger Leiterplatten-Test

- Board einseitig bestückt
- Leitbahnabstand / -breite > 1mm
  (Dual In-Line Gehäuse)
- überschaubare Komplexität

- funktionaler Test ausreichend
- Kontaktierung mit Nadelbett-Adapter

TELEFUNKEN electronic

Zukünftiger Leiterplatten-Test

- Board beidseitig bestückt
- Leitbahnabstand / -breite < 0,5mm
  (SO-, PLCC-, PIN-Grid Gehäuse, MCM)
- sehr hohe Komplexität

- funktionaler Test unzureichend
- Kontaktierung mit Nadelbett-Adapter unmöglich

BOUNDARY - SCAN TEST

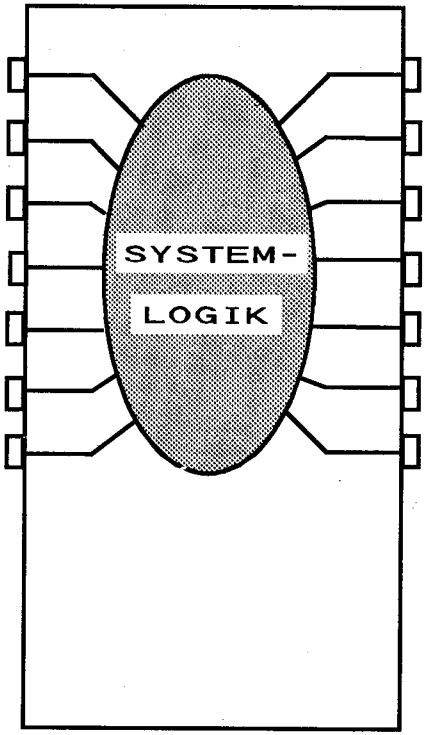
TELEFUNKEN electronic

Ziele des Boundary-Scan Test

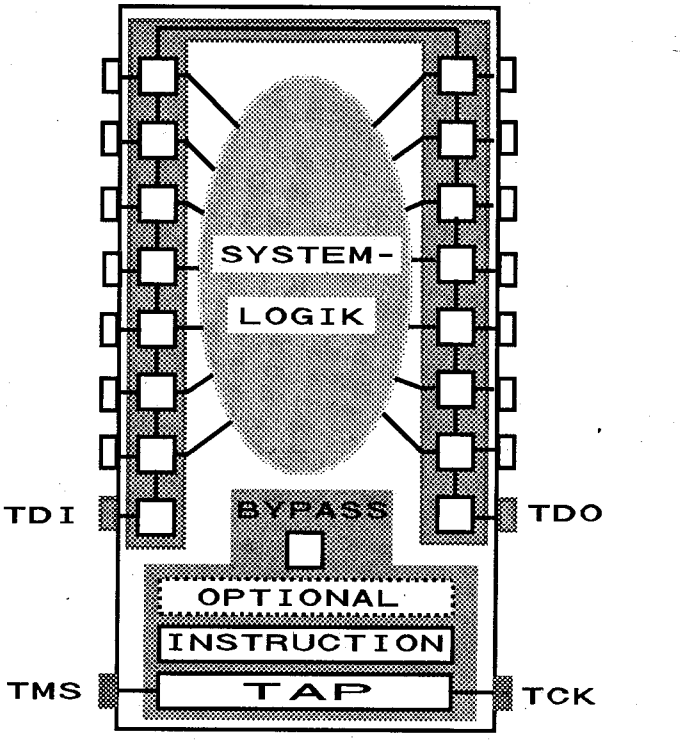
- Strukturierter Test anstatt funktionaler Test
- Höhere Fehlerüberdeckung und Diagnoseauflösung als bei In-Circuit Test
- Schnelle und einfache Entwicklung von Tests für Unterbrechung, Kurzschluß und 'stuck-at' Fehler
- Test der Bauteile im eingelöteten Zustand
- Isolation der Bauteile ohne Back-driving
- Einfache und schnelle Ausführung von Selbsttests

TELEFUNKEN electronic

Grundidee des Boundary-Scan Tests



Konventionelles Bauteil

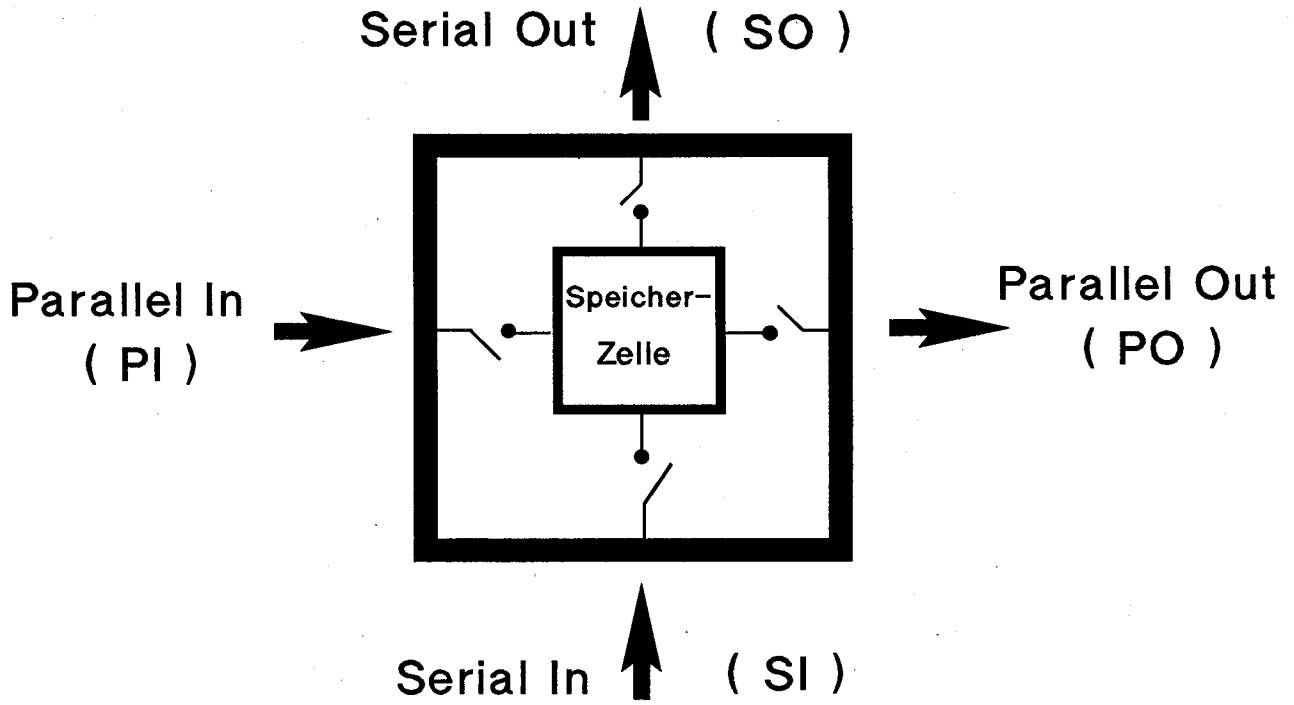


Zusatzlogik für BST Boundary-Scan Zelle

Boundary-Scan fähiges Bauteil

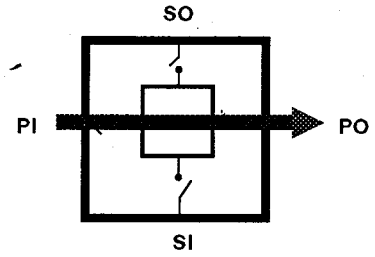
TELEFUNKEN electronic

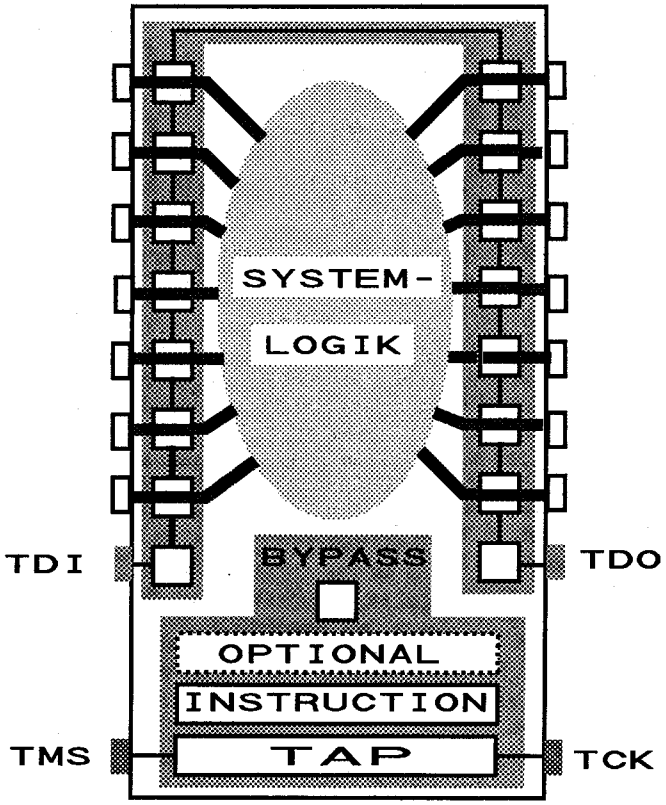
Boundary-Scan.Registerzelle



TELEFUNKEN electronic

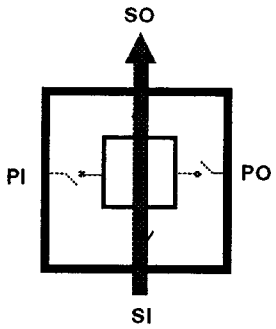
Normalmode

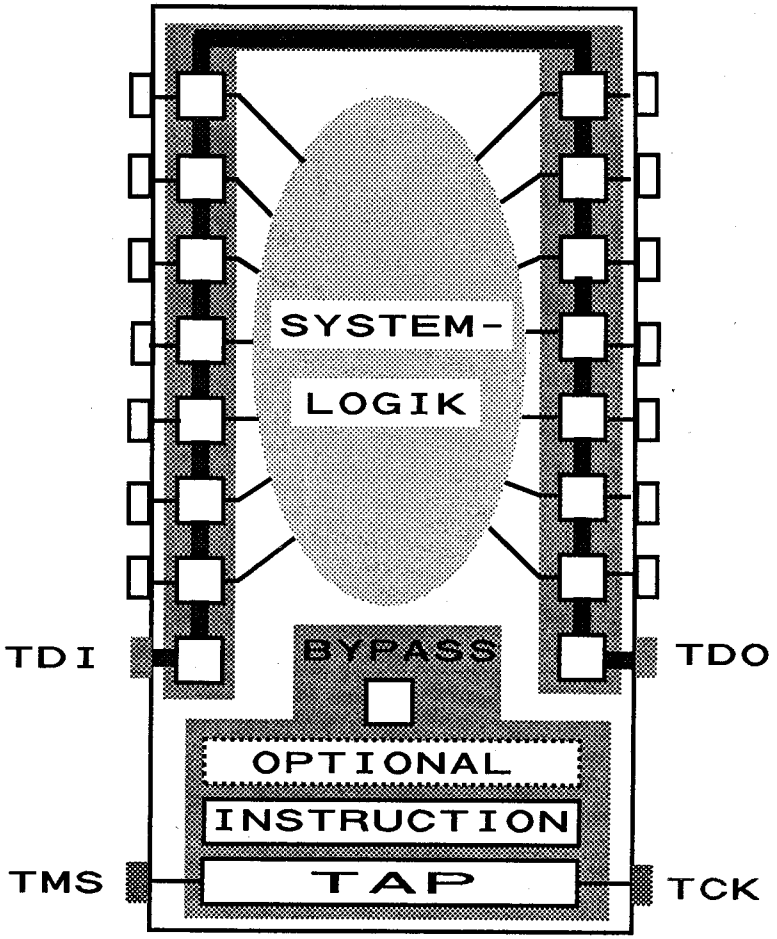




TELEFUNKEN electronic

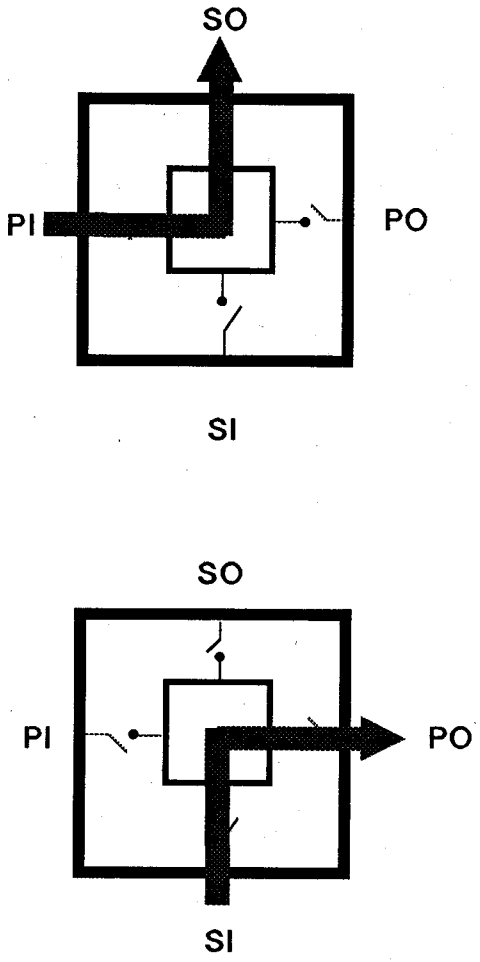
Scan-Mode

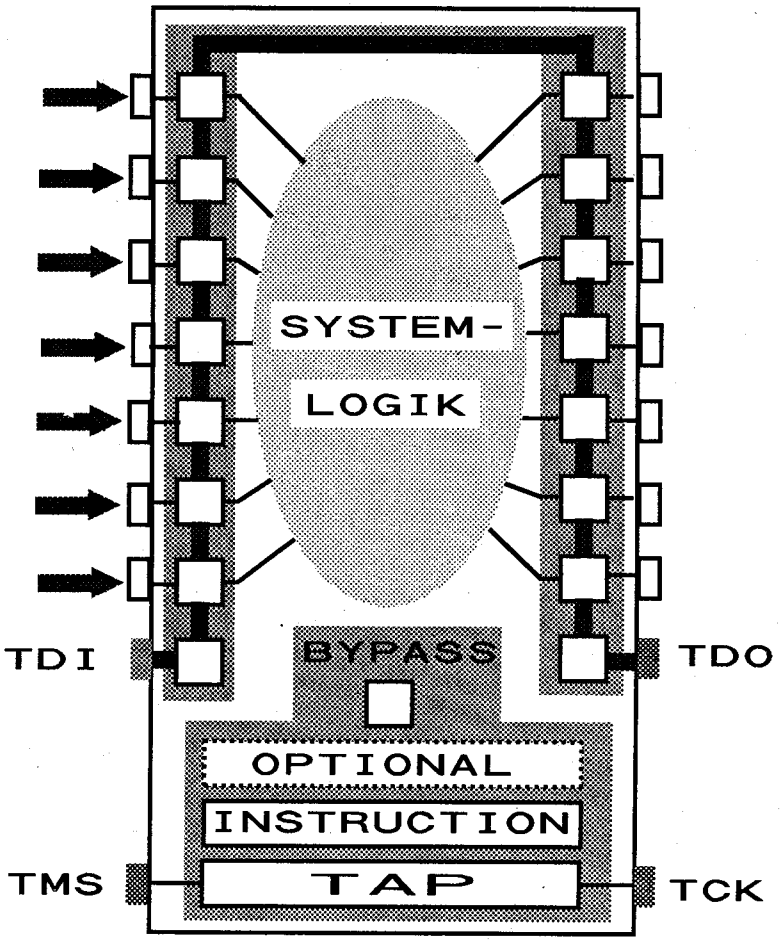




Boundary-Scan Technik

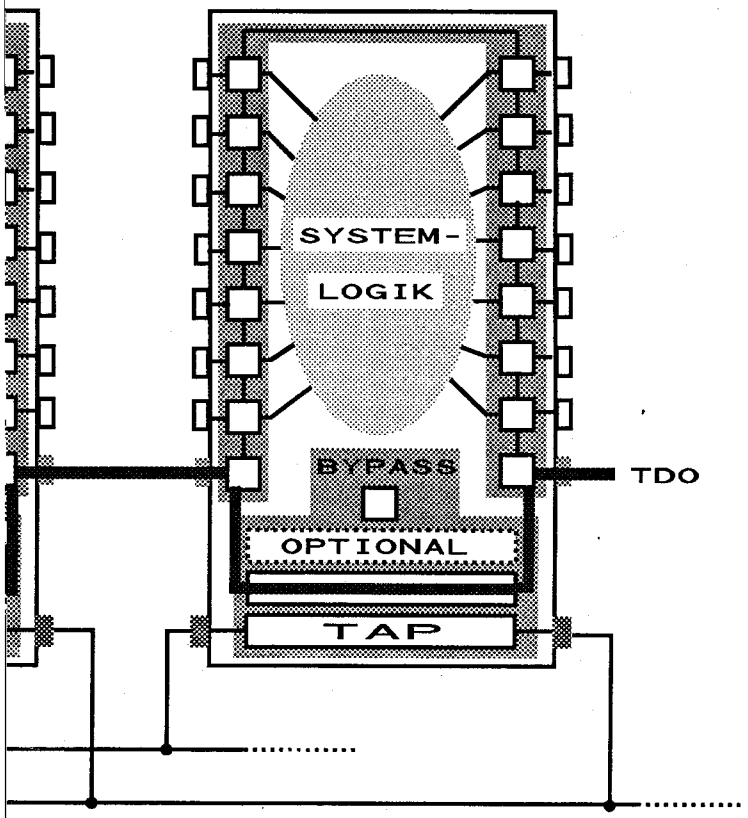
TELEFUNKEN electronic





TELEFUNKEN
electronic

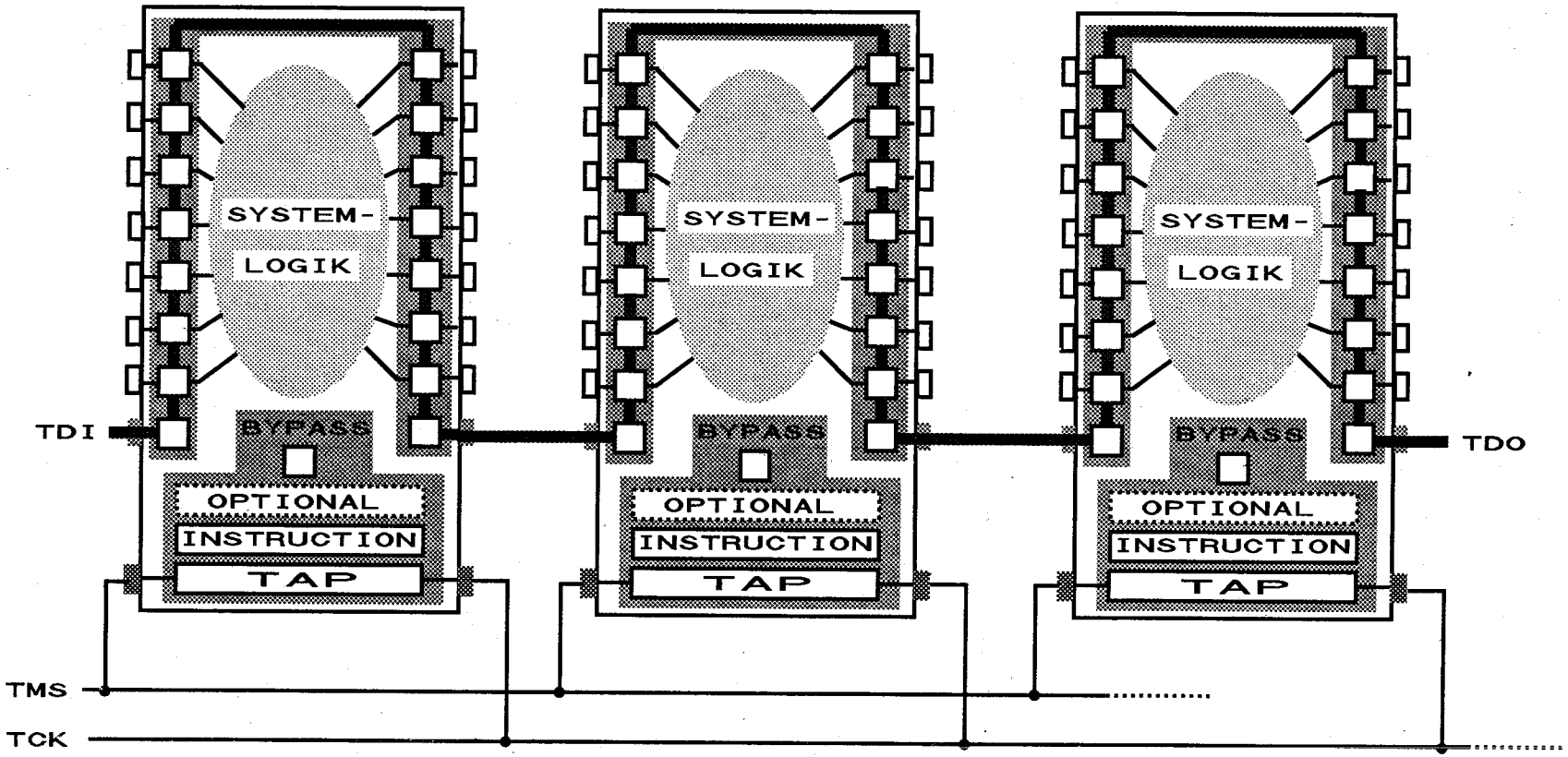
m Scan-Pfad



FURT39 W. Denner /7.92

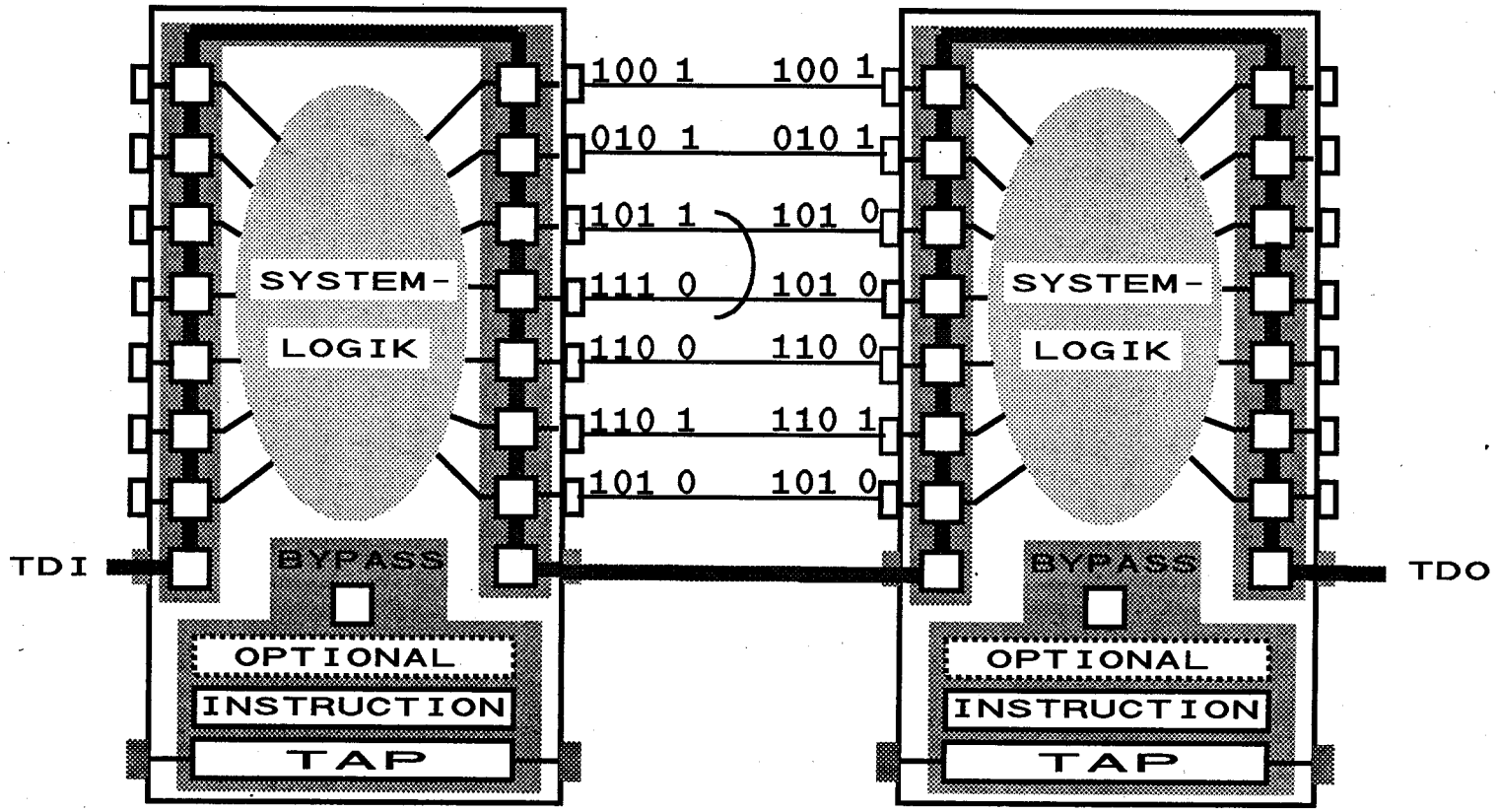
TELEFUNKEN electronic

Boundary-Register im Scan-Pfad



Boundary-Scan Technik

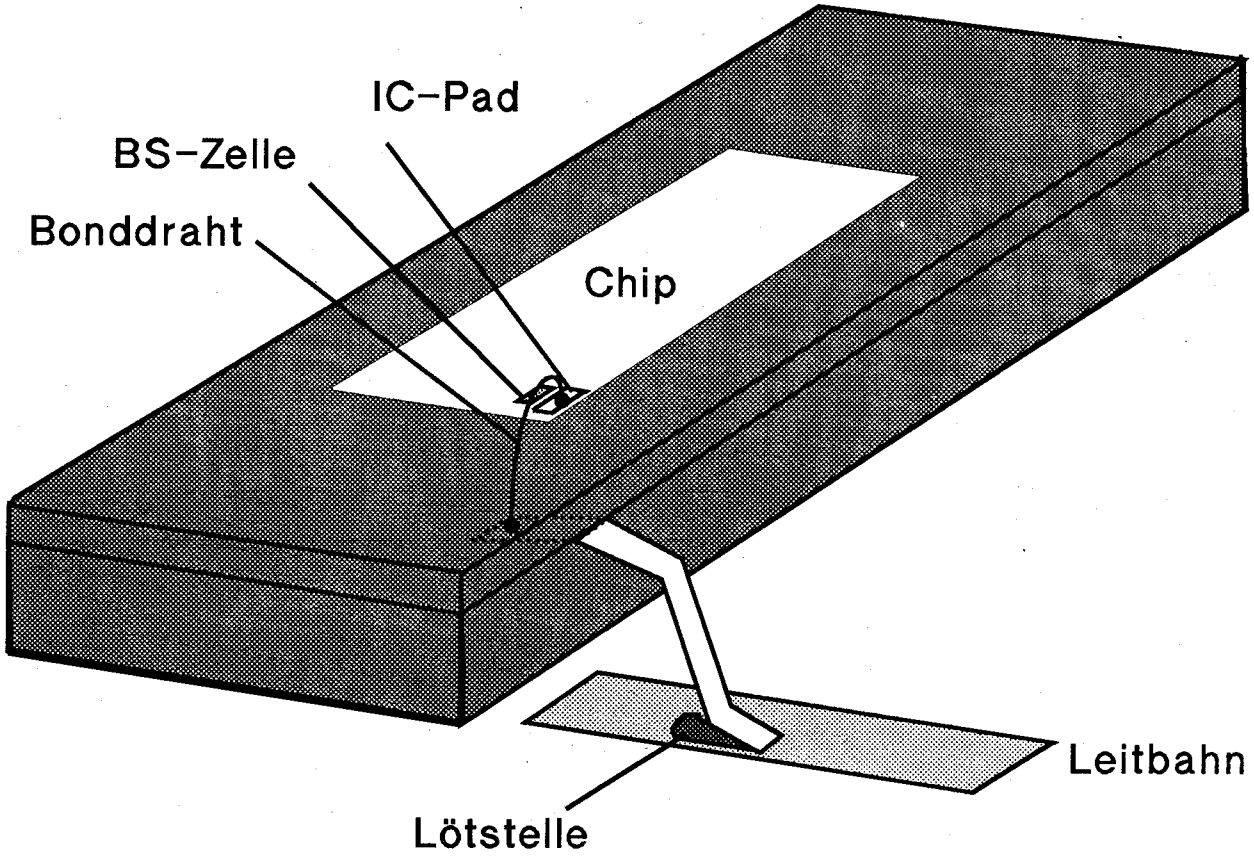
Kurzschluß-Test mit Boundary-Scan



Boundary-Scan Technik

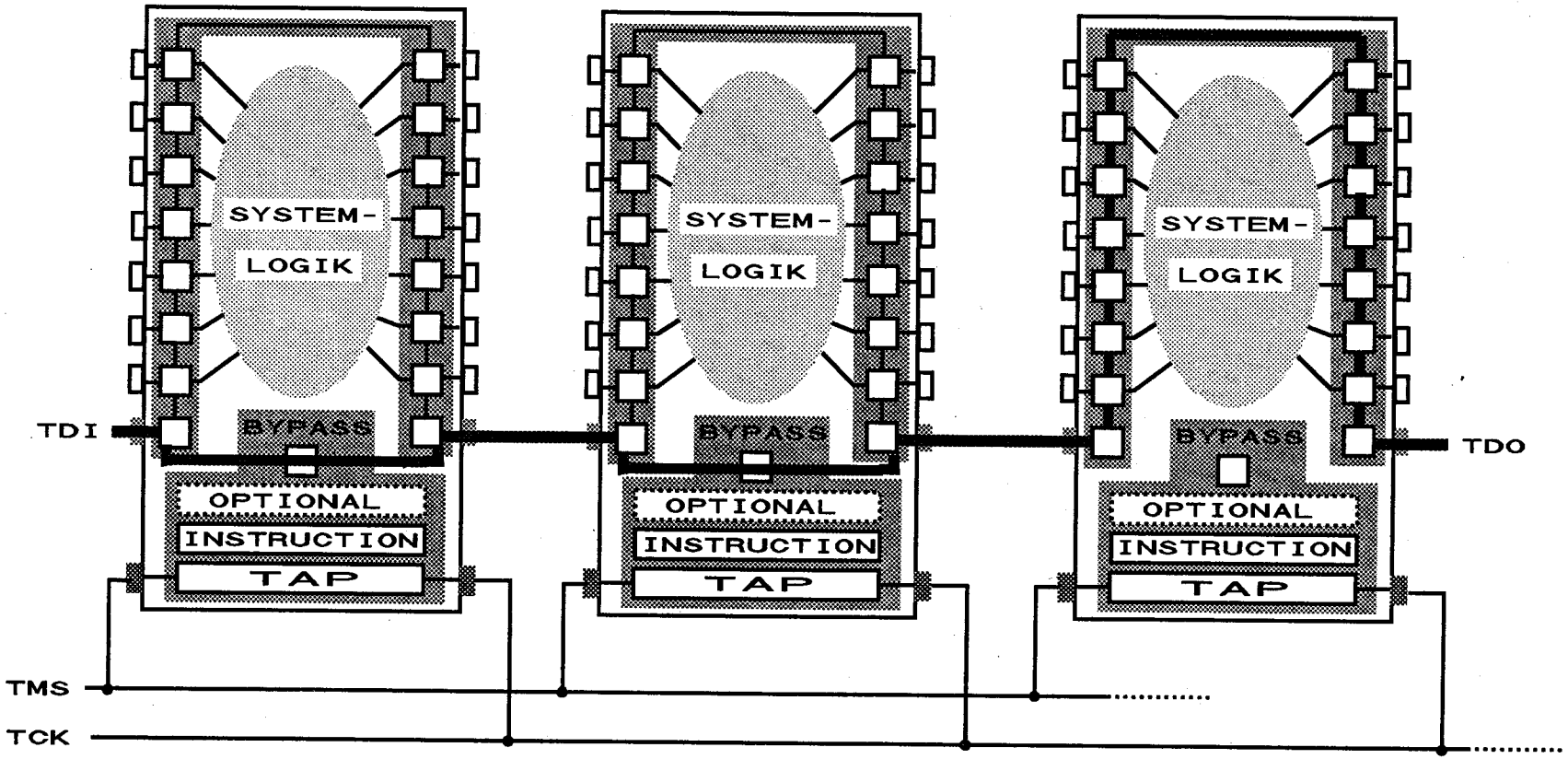
TELEFUNKEN electronic

BS-Zelle-Pad-Bond-Lötstelle-Leitbahn-Lötstelle-Bond-Pad-BS-Zelle



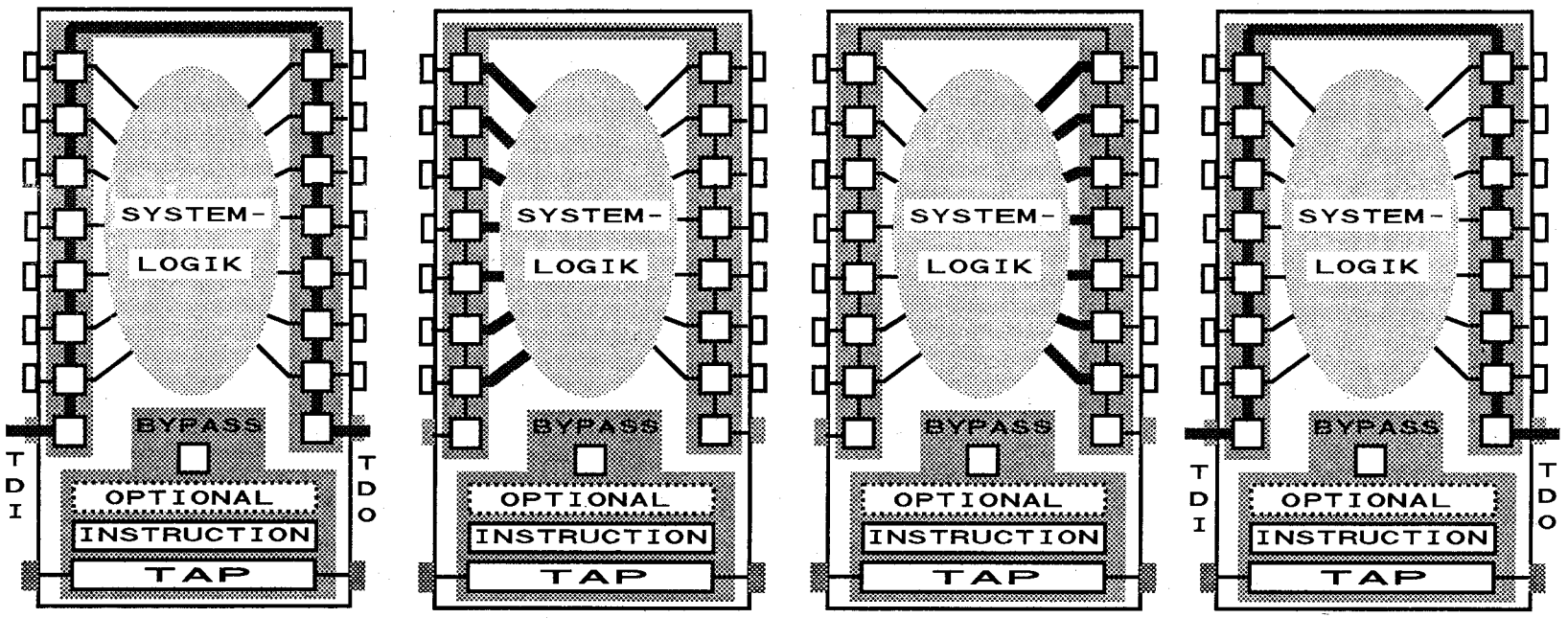
TELEFUNKEN electronic

Bypass-Register im Scan-Pfad



TELEFUNKEN electronic

IC - Test auf dem Board (INTEST)



TELEFUNKEN electronic

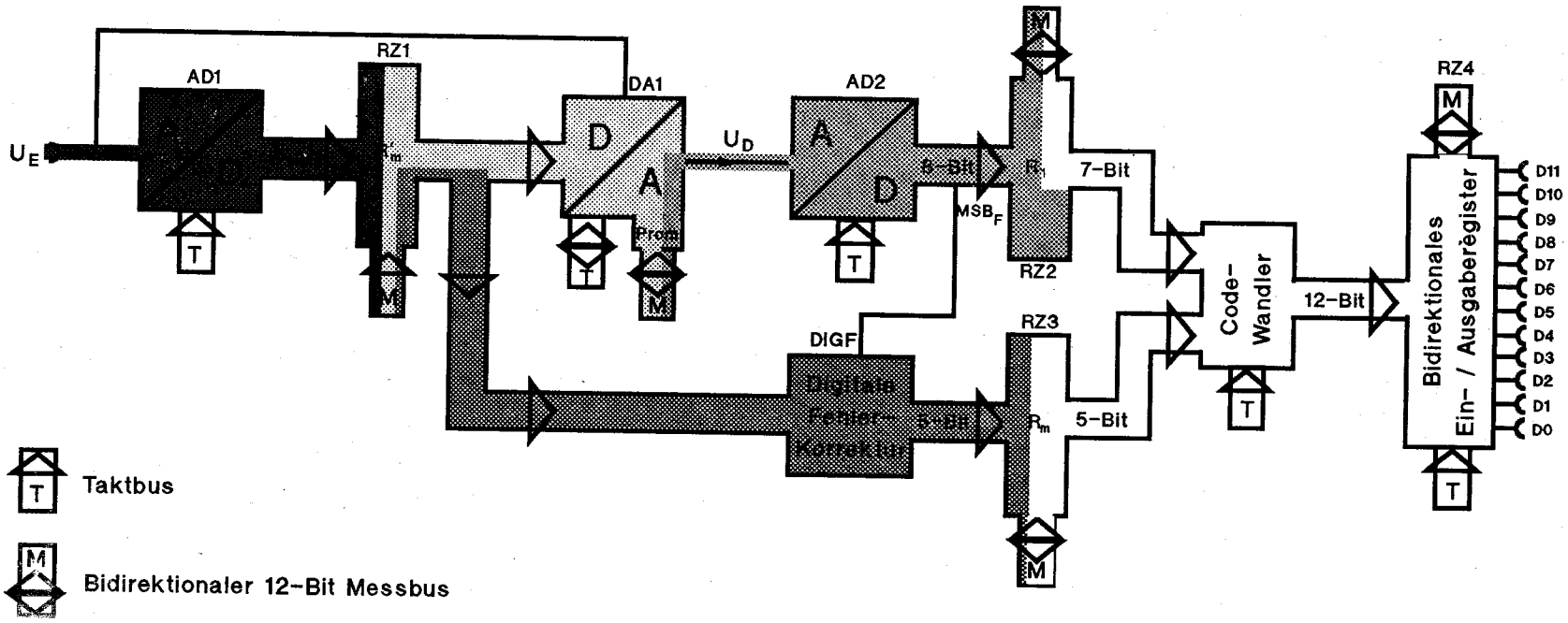
Situation bei Analogschaltungen

- Keine Konzepte für strukturierten Test
- Keine DFT-Regeln
- Keine erprobten Fehlermodelle
- Keine leistungsfähigen Fehlersimulatoren
- Keine bewährten Test-Resourcen

Ad-hoc Lösungen

TELEFUNKEN electronic

Partitionierung eines 12-Bit Pipeline-Kaskaden A/D-Umsetzers



TELEFUNKEN electronic

Zusammenfassung

- Komplexitätssteigerung verursacht neue Testprobleme
- Problemlösung durch prüffreundlichen Schaltungsentwurf, sofern Testkonzept rechtzeitig erarbeitet
- Preis für Testfreundlichkeit:

Scan-Design:	+ 10–15%	Si-Fläche
BIST	+ 17–23%	Si-Fläche
BST	+ 16–20	Gatter/Pin
	+ >160	Gatter für Interface
	+ 4	Pins

TELEFUNKEN electronic

Ausblick

- Selbsttestkonzepte für 'mixed-signal' Schaltungen
- Teststrategien für analoge Schaltungskomponeneten
- Leistungsfähige Simulationsumgebung für Analogschaltungen
- Kriterien für die Auswahl sensitiver Knoten

- BST und BIST effektiv kombinieren
- Hierarchischer Selbsttest
- BST und BIST für 'mixed-signal' Boards
- Verfeinerung der Diagnosemöglichkeiten

2. Vergleich der populären MOSFET - Modelle bei kleinen Geometrien.

Prof. Dr.-Ing. H. Khakzar
Labor Prozeßcharakterisierung der
Mikroelektronik
Fachhochschule Esslingen

Von den ca. 30 MOSFET-Modellen vergleichen wir folgende 5 populäre MOSFET-Modelle:

1. UCB MOS Level 2 - Modell
2. UCB MOS Level 3 - Modell
3. HSPICE MOS Level 6 - Modell
4. BSIM1 - Modell
5. BSIM2 - Modell

1. UCB MOS LEVEL 2 - Modell

Das UCB MOS LEVEL 2 - Modell von SPICE ist ein analytisches, eindimensionales Modell und wurde an der Berkeley Universität in Kalifornien entwickelt.

Folgende Transistoreigenschaften werden im MOS LEVEL 2 - Modell nachgebildet [8,10]:

* Abhängigkeit der Ladung Qb von der Spannung im Kanal.
* Abhängigikeit der Beweglichkeit von der normalen Oberflächenfeldstärke.
* Stromfluss in dem Zustand der schwachen Inversion (Subthreshold - Strom).
* Verkleinerung der Kanallänge im Sättigungsbereich.
* Die Zunahme der Schwellenspannung bei schmalen Kanalbreiten (Schmal-Kanal-Effekt) und die Reduzierung der Schwellenspannung bei kurzen Kanallängen (Kurz-Kanal-Effekt).
* Sättigung der Ladungsträgergeschwindigkeit.
* Überlappungskapazitäten
* Sperrschichtkapazitäten
* Die Temperaturabhängigkeit des Fermipotentials der Diffusionsspannung, der Beweglichkeit und des Sperrstroms.

Die physikalische Natur des Modells verursacht viele komplexe Gleichungen, die lange Simulationszeiten und Konvergenzprobleme mit sich bringen.
Die kleineren Geometrien bringen zusätzliche MOSFET-Effekte, die im LEVEL 2 - Modell nicht enthalten sind.

2. UCB MOS LEVEL 3 - Modell

Das LEVEL 3 - Modell ist strukturell gleich wie das LEVEL 2 - Modell.
Das LEVEL 3 - Modell ist halbempirisch, um einerseits eine höhere Recheneffizienz zu bekommen und anderseits die Konvergenzprobleme des LEVEL 2 - Modelles zu vermeiden.
Die Gleichungen des linearen Bereiches der Id-Uds-Kennlinie aus LEVEL 2 wurden durch eine Taylor-Reihen-Entwicklung vereinfacht.

Der Vorteil eines halbempirischen Modells wie LEVEL 3 ist seine Möglichkeit auch neue Effekte durch Curve-fitting zu berücksichtigen. Das Modell ist anpassungsfäger als ein rein physikalisches Modell.

Der Nachteil eines empirischen Modells ist, daß die Abbildung von physikalischen Eigenschaften verloren geht.
Diese Abbildung ist aber sehr von Nutzen, um Prozeßparameter und elektrische Parameter in Beziehung zu setzen.

Probleme mit dem SPICE MOS LEVEL 3 - Modell

1. Unstetiger Übergang Trioden - Sättigungsbereich

2. Schlechte Beschreibung der Kanallängenabhängigkeit.

3. Unstetiger Übergang schwache - starke Inversion, schlechte Modellierung schwache Inversion.

4. Keine Substratspannungsabhängigkeit der Beweglichkeit.

Parameterextraktion mit IC-CAP [9]

Figure 1: Ids of 20/2 μm NMOS Transistor

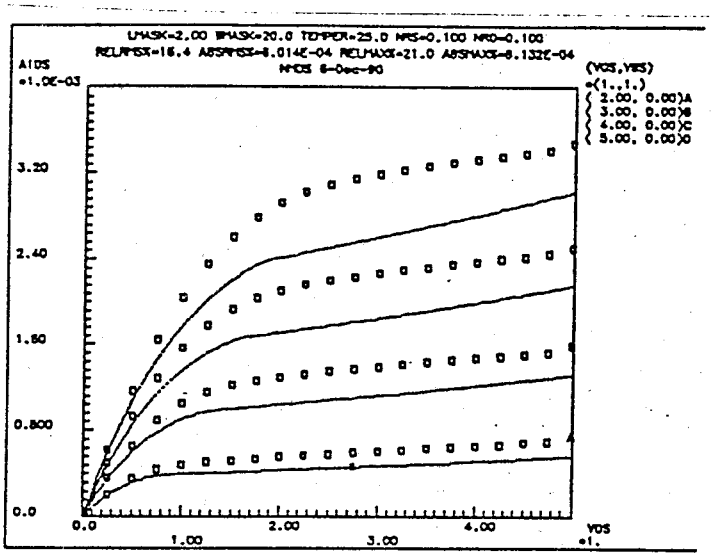
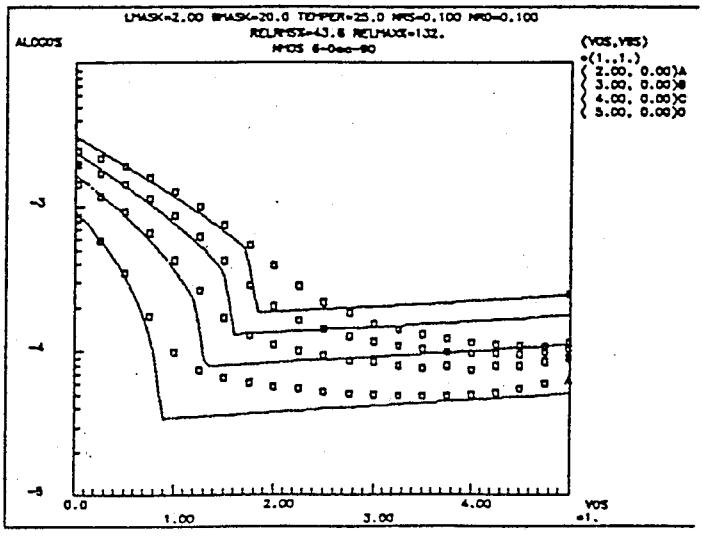


Figure 1: Gds of 20/2 μm NMOS Transistor



Parameterextraktrion mit IC-CAP [9]

Figure 2: Ids of 20/20 μm NMOS Transistor

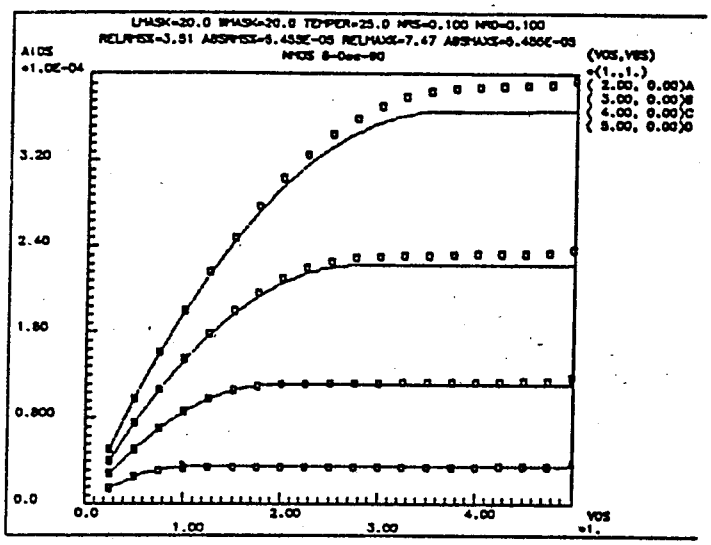
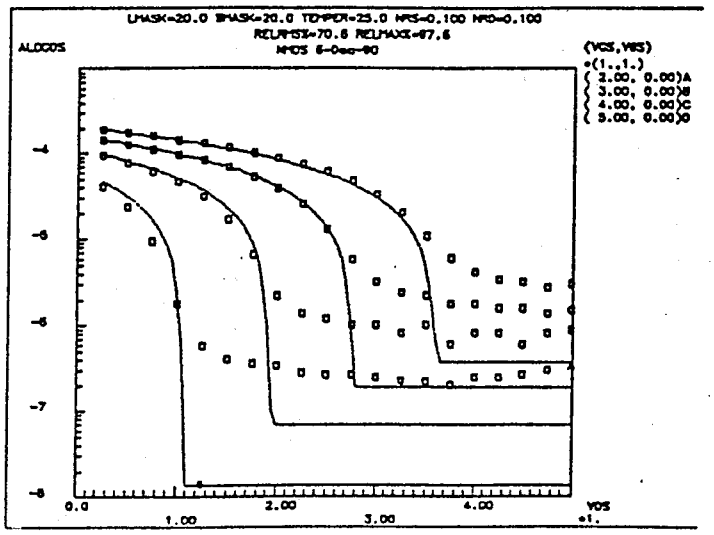


Figure 2: Gds of 20/20 μm NMOS Transistor



3. HSPICE MOS LEVEL 6 - Modell

Das HSPICE MOS LEVEL 6 - Modell ist ein modifiziertes UCB LEVEL 2 - Modell und basiert auf den Modellen der Simulatoren ASPEC, MSINC und ISPICE und ist von META-Software für HSPICE-Simulatoren entwickelt worden.

4. BSIM1

Das Berkeley Kurzkanal Modell IGFET Modell 1 (BSIM1) beinhaltet folgende Effekte:

* Abhängigkeit der Beweglichkeit von der Feldstärke.

* Sättigung der Ladungsträgergeschwindigkeit.

* Ladungsteilung der Verarmungsschicht zwischen Source und Drain.

* Nichtuniforme Dotierungsprofile für ionenimplantierte Transistoren.

* Kanallängenmodulation

* Stromfluss in dem Zustand der schwachen Inversion. (Subthreshold-Strom)

* Geometrieabhängigkeit der elektrischen Parameter.

5. BSIM2 - Modell

Das BSIM2 - Modell basiert auf dem BSIM1 - Modell und wurde entwickelt um MOSFET - Transistoren bis 0,2 μm Kanallänge zu modellieren.
BSIM2 bringt folgende Verbesserungen gegenüber BSIM1:

* genaue Modellierung des Ausgangswiderstandes

  - heiße Elektronen verkleinern den Ausgangswiderstand

  - Kanallängenmodulation

  - Eliminierung des negativen Ausgangswiderstandes

* Die Gleichung für die Beweglichkeit berücksichtigt den Einfluss der Source/Drain - Widerstände.

* Die Kapazität der Inversionsschicht wird modelliert.

* Genaue Modellierung des Gebietes der schwachen Inversion.

* Verbesserte Modellierung der Sättigungsgeschwindigkeit und Reduzierung der Beweglichkeit durch das vertikale Feld.

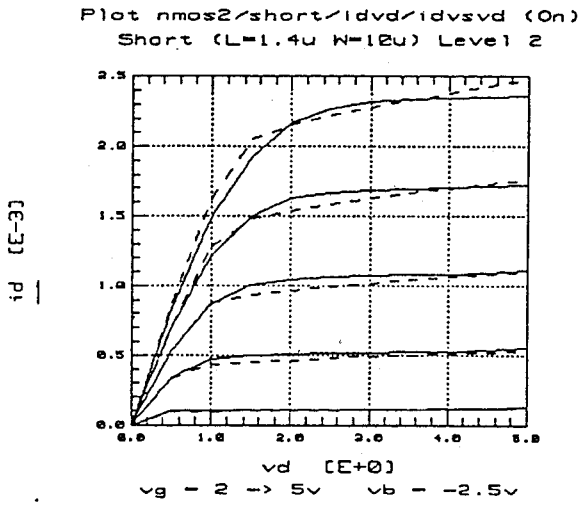
Die genaue Modellierung des Ausgangswiderstandes beim BSIM2 ermöglicht seinen erfolgreichen Einsatz bei Analogsimulationen und Charakterisierung des Submikron - MOSFET - Transistors.

6. Vergleich der Modelle anhand der Parameterextraktion mit IC-CAP

6.1 UCB MOSFET LEVEL 2 - Modell

Level 2 zeigt eine gute Genauigkeit bei Id = f(Ugs).
Einer der Gründe ist die gute Modellierung der Beweglichkeit.
Bei Id = f(Uds) ist die Genauigkeit jedoch nicht gut.
Bild 3 zeigt Id = f(Uds) bei einem MOSFET - Transistor mit der Kanallänge 1,2 μm und der Kanalbreite 10 μm.

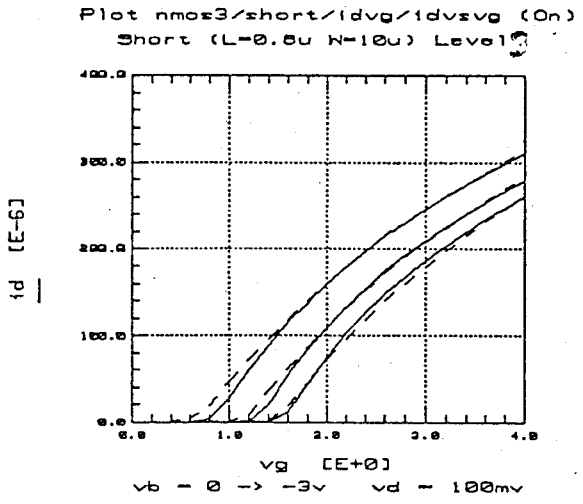
FIGURE 3. I_D vs V_{DS} Comparison for Level 2 model



6.2 UCB MOSFET LEVEL 3 - Modell

Bild 4 zeigt Id = f(Ugs). Die Übereinstimmung der gemessenen und der simulierten Kennlinie ist schlechter als beim Level 2 - Modell. Es zeigt sich, daß beim Level 3 der Drainstrom und die Beweglichkeitsdegradierung nicht gut modelliert sind.
Bei Id = f(Uds) ist die Genauigkeit jedoch gut, weil die Gleichung für den Sättigungsstrom flexibler ist. Bei diesen Kennlinien sind bis zu 0,8 μm gute Resultate erzielt worden.
Der extrahierte Wert für die Beweglichkeit ist jedoch nicht realistisch. Dies zeigt, daß die Parameter bei halbempirischen Modellen zum Curve-fitting benutzt wurden.

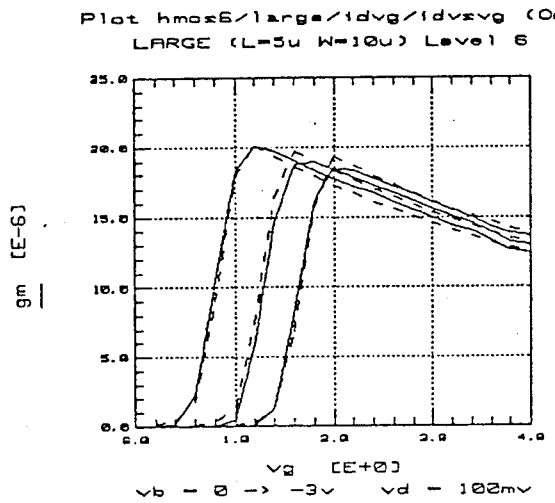
FIGURE 4. I_D vs V_{GS} Comparison for Level 3 model



6.3 HSPICE MOSFET LEVEL 6 - Modell

Das Level 6 - Modell zeigt eine ähnliche Genauigkeit wie Level 2 bei Id = f(Ugs) und wie Level 3 bei Id = f(Uds) für Kanallängen von L = 1,4 μm.
Bild 5 zeigt die Vorwärtssteilheit Gm = f(Ugs). Die Modellierung ist besser als Level 2 und Level 3.

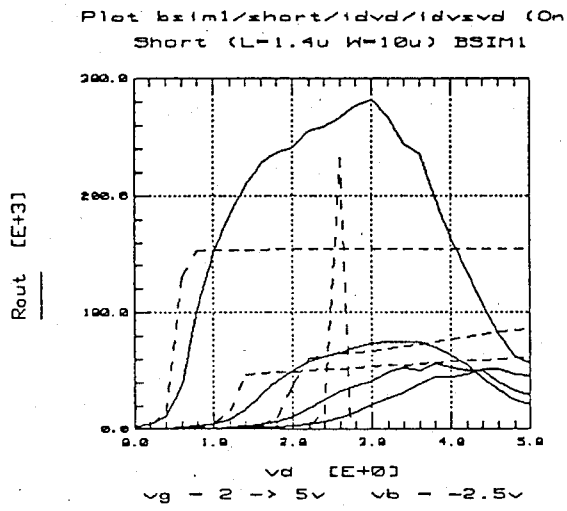
FIGURE 5. Transconductance Comparison for Level 6



6.4 BSIM1

Das BSIM1- Modell zeigt gute Ergebnisse bei Id = f(Ugs) und Id = f(Uds) bis zu Kanallängen von L = 0,5 μm.
Die Subthresholdmodellierung ist also bis zu Kanallängen von L = 0,8 μm gut.
Das Modell ist jedoch nicht in der Lage, die Abnahme des Ausgangswiderstandes bei hohen Drainspannungen zu modellieren (Bild 6). Das Modell ist demnach nicht für analoge Simulationen geeignet.

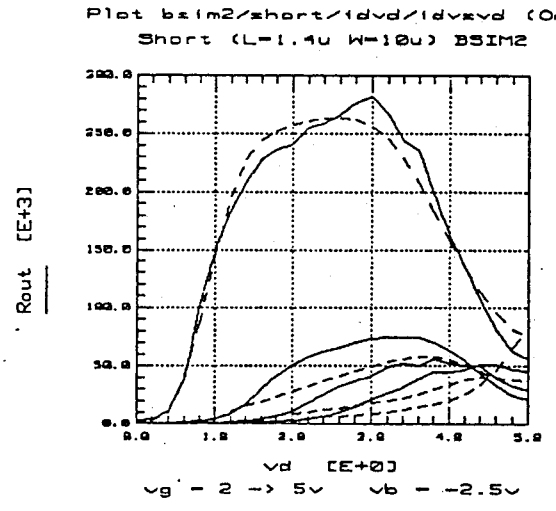
FIGURE 6. Output Resistance Comparison for BSIM1



6.5 BSIM2

Das BSIM2 - Modell zeigt gute Ergebnisse bei Id = f(Ugs) und Id = f(Uds) und Subthreshodmodellierung bis zu Kanallängen von 0,5 μm.
Das Modell ist in der Lage, den Ausgangswiderstand zu modellieren (Bild7), da die Gleichung für den Ausgangswiderstand die Effekte der Kanallängenmodulation und der heißen Elektronen beinhaltet.
Das BSIM2 - Modell stellt eine wesentliche Verbesserung gegenüber dem BSIM1 - Modell dar, und ist deshalb für analoge Simulationen geeignet.

FIGURE 7. Output Resistance Comparison for BSIM2



Gegenüberstellung der Modelle

Tabelle 1 zeigt die Genauigkeit der Modelle bei der Simulation von den 7 üblichen Charakteristiken.

Plot	Level 2	Level 3	Level 6	BSIM1	BSIM2
Id = f(Ugs)	gut	befriedigend	gut	sehr gut	sehr gut
log(Id) = f(Ugs)	befriedigend	befriedigend	befriedigend	gut	gut
Id = f(Uds)	befriedigend	gut	gut	sehr gut	sehr gut
r_{Ausg} = f(Uds)	mangelhaft	mangelhaft	mangelhaft	mangelhaft	gut
g_m= f(Ugs)	befriedigend	befriedigend	gut	befriedigend	gut
r_{DS} = f(L)	gut	gut	gut	befriedigend	befriedigend
g_{DS}= f(W)	befriedigend	befriedigend	befriedigend	befriedigend	befriedigend

Zusammenfassung

Da es kein MOSFET -Transistor - Modell gibt, das alle Effekte genau wiedergibt, muß man vor der Wahl des Modells die Anforderungen an die Schaltungen kennen, damit man die richtigen Kompromisse zwischen der Genauigkeit und der Rechenzeit schließen kann.

Literaturverzeichnis

1. B.J. Scheu, D.L. Scharfetter, P.K. Ko, BSIM: Berkeley Short-Channel IGFET-Modell for MOS Transistors, IEEE Journal of Solid-State Dircuits, Vol. SC-22, No. 4, August 1987, pp. 558-566.

2. P. Antognetti, G. Massobrio, Semiconductor Device Modelling with SPICE, McGraw-Hill, New York, 1988, p.148.

3. P. Antognetti, G. Massobrio, Semiconductor Device Modelling with SPICE, Mc Graw-Hill, New York, 1988, p.158 - 184.

4. P. Antognetti, G. Massobrio, Semiconductor Device Modelling with SPICE, McGraw-Hill, New York, 1988, p. 185.

5. HSPICE User's Manual, Meta-Software Inc., Campbell, CA, 1990.

6. B.J. Scheu, D.L. Scharfetter, P.K. Ko, BSIM: Berkeley Short-Channel IGFET Model for MOS Transistors, IEEE Journal of Solid-State Circuits, Vol. SC-22, No.4, August 1987, pp. 558-566.

7. J.S. Duster, M.C. Jeng, P.K. Ko, C. Hu, User's Guide for the BSIM2 Parameter Extraction Program and the SPICE3 with BSIM Implementation, ERL Memorandum, Department of Electrical Engineering and Computer Science, University of California at Berkeley, May 1990.

8. Ira Leventhal and Hitoshi Aoki, A Comprehensive Comparison of Small-Geometry MOSFET Models, February 10, 1992, Hewlett Packard Circuit Modeling Seminar.

Literaturverzeichnis

9. H. Richter, Einfache Erweiterung des SPICE MOS LEVEL 3 - Modells, Arbeitskreis MOS-Modelle und Parameterextraktion, 24.1..91, IMS Stuttgart.

10. H. Khakzar, A. Mayer, R. Oetinger, Entwurf und Simulation von Halbleiterschaltungen mit SPICE, EXPERT - Verlag 1991.

11. Höfer, E.E., Nielinger, H., SPICE,Springer - Verlag 1985.

3. Software zur Analyse von Sea-of-Gates Layouts*

W. Rülling
Labor für IC-Entwurf, Fachhochschule Furtwangen

Zusammenfassung

In der Arbeit wird vorgestellt, wie sich die Korrektheit manuell erstellter Verdrahtungen für Sea-of-Gates Layouts überprüfen läßt. Dazu wird ein besonders effizienter Netzextraktor beschrieben, der eine vorgegebene regelmäßige Masterstruktur berücksichtigt und deshalb nur die kundenspezifischen Verdrahtungsebenen analysieren muß. In Kombination mit einem Simulator kann der Extraktor gut zur Überprüfung von Sea-of-Gates Grundzellen eingesetzt werden.
Für die Überprüfung sehr komplexer Layouts wird ein Verfahren vorgestellt, das einen sehr schnellen Konsistenztest zwischen einer extrahierten Transistornetzliste und einer vorgegebenen Gatternetzliste realisiert.

1 Problemstellung

An der Fachhochschule Furtwangen wird im Rahmen des IIT-Projekts "Algorithmen für intelligente Sensoren" ein Chip zur schnellen Berechnung von Faltungsintegralen entworfen ([Rül91]). Ein Prototyp des Chips wird in einer Sea-of-Gates Technologie am IMS (Institut für Mikroelektronik, Stuttgart) gefertigt.
Bei den Entwurfsarbeiten hat sich gezeigt, daß sich aufgrund von Besonderheiten des verwendeten Schaltungsentwurfs praktisch keine Semi-Custom Entwurfswerkzeuge einsetzen lassen. Die Hauptursache für dieses Phänomen besteht darin, daß es in der Schaltung Steuersignale gibt, die an eine große Anzahl von Schaltungskomponenten angeschlossen werden müssen.
Dazu ist es erforderlich, die Steuersignale jeweils lokal für Gruppen benachbarter Komponenten zu verstärken. Würde man diese gemeinsam versorgten Gruppen willkürlich bilden, ergäbe sich ein unnötig großer Verdrahtungsaufwand und damit verbunden auch große Signallaufzeiten. Offensichtlich muß die elektrische Schaltung des Chips also unter Berücksichtigung der Layoutgeometrie entworfen werden. Diese Abhängigkeit der Schaltungsnetzliste vom Layout wird besonders deutlich, wenn man beachtet, daß die Anzahl der erforderlichen Treiberstufen und damit die Anzahl der Schaltungskomponenten von der Länge der benötigten Verdrahtungsleitungen abhängt.

*Die Arbeit wurde im Rahmen des IIT-Projekts *Implementierung von Algorithmen für intelligente Sensoren* durch das Schwerpunktprogramm des Landes Baden-Württemberg gefördert.

Für eine Reihe von Layoutabhängigkeiten der Netzliste werden in Semi-Custom Systemen Sonderbehandlungen angeboten. Beispielsweise wird die Reihenfolge der Flip-Flops in einem automatisch generierten Scan-Path typischerweise durch die bei der automatischen Plazierung entstehende geometrischen Anordnung ermittelt. D.h. hier wird die Netzliste des Scan-Path aus dem entstehenden Layout konstruiert. Ähnliches gilt bei einer automatischen Realisierung von Clock-Signalen. Hier können beispielsweise die Zellen innerhalb einer Standdardzellenreihe jeweils gemeinsam von zusätzlich eingesetzten Treibern versorgt werden.
Leider fehlt eine entsprechende Sonderbehandlung für sonstige "große Netze", wie etwa den oben erwähnten Steuersignalen. Für diese Signale muß der Designer also eine detaillierte Netzliste eingeben, obwohl er in der Regel keine Vorstellung von dem aus der Netzliste durch Plazierungs- und Verdrahtungsheuristiken generierten Layout besitzt. Da der Designer außerdem beim Semi-Custom Entwurf meist auch keine Vorgaben für die Plazierung von Layoutkomponenten machen kann, hat er praktisch keine Möglichkeit die Layoutabhängigkeit der Netzliste zu berücksichtigen.
Diese Schwierigkeiten haben dazu geführt, daß im Projekt auch recht große Layoutstrukturen manuell entworfen werden müssen. Da die Handarbeit jedoch relativ fehleranfällig ist, werden im Projekt besondere Tools zur Überprüfung von komplexen Handlayouts benötigt.
Zum einen benötigt man vor der Chipfertigung eine abschließende Prüfung des vollständigen Layouts. Zusätzlich sind aber auch während der Entwurfsarbeiten regelmäßige Überprüfungen der erstellten Teillayouts notwendig. Mit ihnen sollen eventuelle Verdrahtungs- oder Plazierungsfehler möglichst so früh entdeckt werden, daß die Fehler noch mit relativ geringem Aufwand beseitigt werden können.
Während die einmalige abschließende Prüfung im allgemeinen nicht besonders zeitkritisch ist, erfordern die regelmäßigen Prüfungen besonders effiziente Verfahren, damit sie die Entwurfsarbeiten nicht unnötig verzögern. Dazu wurden im Projekt eigene Tools entwickelt, die in den folgenden Abschnitten vorgestellt werden.

2 Netzextraktion für Sea-of-Gates Layouts

In diesem Kapitel wird die Arbeitsweise eines Netzextraktors beschrieben, der sich besonders zum Einsatz bei Sea-of-Gates Technologien eignet. Dazu werden im Abschnitt 2.1 zunächst die besonderen Anforderungen am Beispiel der Gate-Forest-Architektur des IMS (siehe [Beu89]) vorgestellt. Anschließend werden in Abschnitt 2.2 die für eine effiziente Verarbeitung benötigten Algorithmen skizziert. Im Abschnitt 2.3 wird der implementierte Netzextraktor schließlich an einem Layoutbeispiel demonstriert.

2.1 Besonderheit bei Sea-of-Gates Layouts

Die Sea-of-Gates Technologien sind eine Weiterentwicklung von Gate-Arrays. Wie bei Gate-Arrays werden sämtliche Transistoren kundenunabhängig auf einem sogenannten *Master* vorgefertigt. Die Realisierung der jeweils benötigten Gatter und der Verdrahtung zwischen den Gattern erfolgt nachträglich durch die Fertigung von einigen wenigen Personalisierungsebenen. Durch die Vorfertigung einheitlicher Masterstrukturen erreicht man eine drastische Reduktion der Fertigungszeit und dadurch auch eine Reduktion der Fertigungskosten.
Die Besonderheit der Sea-of-Gates Technologie gegenüber herkömmlichen Gate-Arrays besteht darin, daß auf dem Master kein spezieller Platz für Verdrahtungskanäle reserviert wird. Stattdessen ist der gesamte Master mit Transistoren gefüllt und es ist dem Designer überlassen, welche Transistoren für Gatter benutzt werden. Die benötigten Verdrahtungskanäle entstehen einfach dadurch, daß die Transistoren in manchen Regionen nicht kontaktiert werden und stattdessen Leitungen über diese Transistoren geführt werden. Auf diese Weise erreicht man bei Sea-of-Gates Layouts eine besonders hohe Transistordichte.
Beispielsweise besitzt der im Projekt verwendete GFxx2-Master des IMS (siehe z.B. [Beu89]) insgesamt 118.544 Transistoren. Je vier Transistoren sind in einer sogenannten *core*-Zelle zusammengefaßt (siehe Abbildung 1). Einer dieser Transistoren ist kleiner dimensioniert und wird in der Praxis nur zur Realisierung von Speicherelementen verwendet. Für die Realisierung von Gattern stehen dann pro core-Zelle ein P-Transistor und zwei N-Transistoren zur Verfügung.
Insgesamt befinden sich auf dem GFxx2-Master 62 Zeilen mit jeweils 478 solcher core-Zellen. Wie in Abbildung 1 dargestellt, sind die drei normal dimensionierten Transistoren zeilenweise untereinander kontaktiert, so daß sich Transistorketten bilden. Durch geeignete Belegung der Transistorgates mit den Signalen *VSS* und *VDD* sorgt man deshalb in der Praxis dafür, daß diese Ketten in unabhängige Teilketten zerlegt werden, so daß sich benachbarte Gatter nicht gegenseitig beeinflussen.
Wegen des für die Verdrahtung benötigten Platzes und der erforderlichen Sperrung vieler Transistoren, kann in der Praxis nur ein Bruchteil der 118.544 Transistoren tatsächlich aktiv verwendet werden. Trotzdem sind bei der Überprüfung von Layouts grundsätzlich alle Transistoren zu berücksichtigen, da insbesondere auch die Korrektheit der Sperrungen festgestellt werden muß.
Da es auch wesentlich größere Sea-of-Gates Master gibt, sollte der zu entwickelnde Netzextraktor in der Lage sein, Layouts mit mehreren 100.000 Transi-

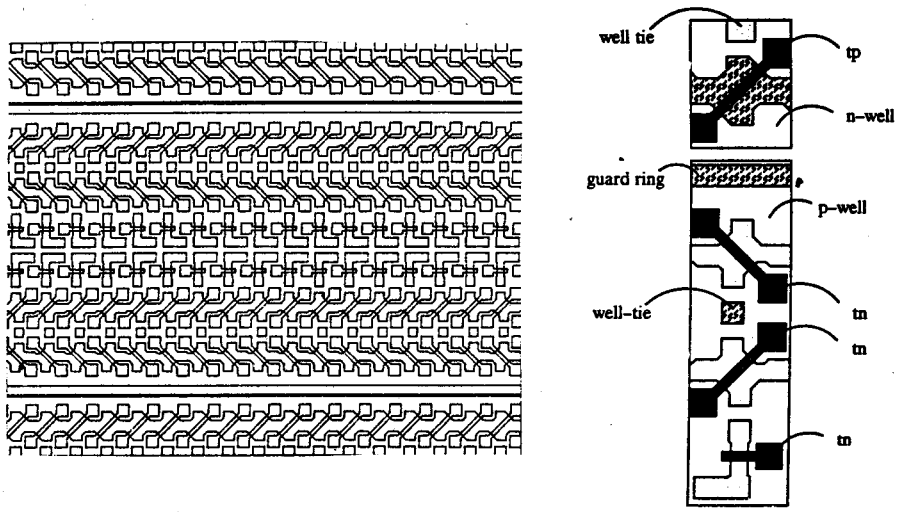


Abbildung 1: Ausschnitt aus einem Gate-Forest Master des IMS und eine einzelne aus 4 Transistoren aufgebaute Kernzelle

storen zu verarbeiten. Als Ausgabe sollte der Extraktor eine Transistornetzliste liefern, die alle für die jeweilige Schaltung relevanten Transistoren beinhaltet. Während die Transistoren auf dem Master relativ kompliziert aus nicht orthogonalen Strukturen aufgebaut sind, die relativ schwierig zu extrahieren sind, werden bei der Verdrahtung in den Personalisierungsebenen aus fertigungstechnischen Gründen nur vertikale und horizontale Leitungen zugelassen. Diese Besonderheit kann ausgenutzt werden, um eine besonders schnelle Netzextraktion zu erreichen.

2.2 Algorithmen

Da der zu entwickelnde Netzextraktor bei relativ komplexen Layoutstrukturen eingesetzt werden soll, ist es wichtig, daß besonders effiziente Algorithmen eingesetzt werden. Deshalb wird für die Analyse der Verdrahtungsstrukturen ein Planesweep-Verfahren (siehe [Meh84b]) verwendet, das die jeweilige geometrische Anordnung der Layoutkomponenten ausnutzt, um durch eine geschickte Abarbeitungsreihenfolge die Rechenzeit zu reduzieren. Dabei werden lediglich die Personalisierungsebenen analysiert, so daß keine Rechenzeit zur Analyse der als bekannt vorausgesetzten Masterstrukturen anfällt.
Die Grundidee des Verfahrens besteht darin, mit einer vertikalen Linie, der Sweep-Line, von links nach rechts über die Layoutfläche zu fahren und dabei jeweils die unter der Linie befindlichen Objekte zu betrachten (siehe Abbildung 2).
Ein erwünschter Effekt dieser Vorgehensweise besteht darin, daß sich zu jedem Zeitpunkt nur relativ wenige Rechtecke unter der Sweep-Line befinden und deshalb auch nur relativ wenig Daten gemeinsam im Hauptspeicher gehalten werden müssen. Immer wenn ein linkes Intervall einer Box erreicht wird, werden die Daten der Box in den Speicher übernommen und wenn das rechte Intervallende erreicht wird, dann werden die Daten wieder aus dem Speicher entfernt.
Die Speicherung der Boxdaten geschieht nach Maskenebenen getrennt mit Hilfe von Intervallbäumen (siehe [Meh84b]). Auf diese Weise kann bei n Boxes auf

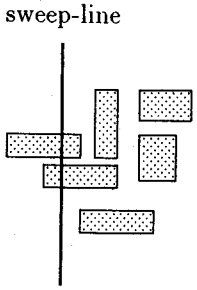
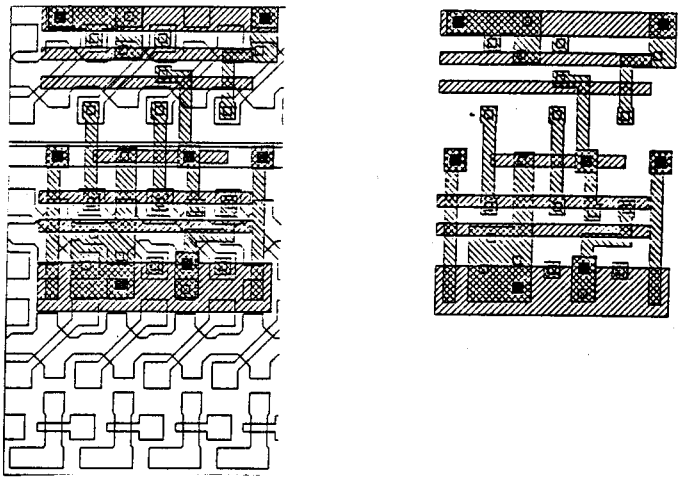


Abbildung 2: Momentaufnahme eines Planesweeps

der Sweep-Line die Anzahl der durch eine Box verursachten Überlappungen größenordnungsmäßig in der Zeit $O(k + \log n)$ ermittelt werden. Dabei ist k die Anzahl der gefundenen Überlappungen.
Die eigentliche Bewegung der Sweep-Line über das Layout geschieht ebenfalls auf recht effiziente Weise. Statt die Linie nämlich kontinuierlich zu bewegen, läßt man sie jeweils von der aktuell verwendeten x-Position direkt zur nächsten verwendeten Position springen. Dadurch hängt die Laufzeit des Verfahrens nicht von der Größe der Layoutfläche ab, sondern nur von der Anzahl der plazierten Objekte. Insbesondere können dabei leere Layoutstreifen übersprungen werden. Die Realisierung dieser Bewegung der Sweep-Line ist recht einfach. Man sortiert dazu die von der Layoutexpansion erzeugten Intervalle und Punkte nach ihren x-Koordinaten und arbeitete dann die sortierten Daten sequentiell ab. Dabei bestimmt die jeweils gelesene x-Koordinate die nächste Position der Sweep-Line.
In der Praxis wird die Laufzeit des Planesweep Verfahrens von der Zeit zum Sortieren der Layoutdaten dominiert. Gute Sortierverfahren benötigen dazu größenordnungsmäßig die Laufzeit $O(n \cdot \log n)$.
Während der Abarbeitung des Planesweeps werden bei jeder gefundene relevanten Überlappung von Rechtecken die betreffenden im Layout auftretenden elektrischen Signale aktualisiert. Dazu wird eine Union-Find Datenstruktur (siehe [Meh84a]) benutzt, mit der sich die Aktualisierungen im Mittel mit fast konstanter Laufzeit durchführen lassen. Der Zeitaufwand für diese Berechnungen ist daher vernachlässigbar. Die so während des Planesweeps konstruierte Transistornetzliste wird anschließend wahlweise in Form einer HILO- oder SPICE-Netzliste ausgegeben. Dabei wird bei jedem Transistor auch seine Position auf dem Master festgehalten.

2.3 Beispiel

In diesem Abschnitt soll der Netzextraktor an einem kleinen Beispiel demonstriert werden. Dazu verwenden wir das in Abbildung 3a dargestellte Layout *inan2a* eines 2-stelligen Nand-Gatters. Es sei in der Datei *inan2a.k* in KIC-Notation abgelegt. Wegen der besonderen Arbeitsweise des Netzextraktors, bei der lediglich die Personalisierungsebenen untersucht werden müssen, verwendet



a) b)

Abbildung 3: a) Vollständiges Sea-of-Gates Layout der Zelle *inan2a* und b) die Personalisierungsebenen der Zelle

der Extraktor vom Layout lediglich die in Abbildung 3b dargestellten Schichten. Prinzipiell kann der Extraktor ohne Vorgabe weiterer Informationen aufgerufen werden. Es ist jedoch sinnvoll, zusätzlich noch auf einer Datei *inan2a.pin* die Namen der Ein-Ausgabe Signale und ihre Anschlußpositionen auf dem Master anzugeben. Diese Signalnamen werden dann automatisch in der generierten Netzliste berücksichtigt. Abbildung 4 zeigt eine auf diese Weise erzeugte Netzliste des *nand*-Gatters.
Die logische Korrektheit der Layouts von Grundzellen läßt sich leicht durch Simulation der extrahierten Netzlisten überprüfen. Bei größeren Layouts scheitert diese Vorgehensweise aber an den erforderlichen extrem hohen Simulationszeiten. Deshalb wird im folgenden als Alternative ein Konsistenztest zwischen verschiedenen Netzlisten vorgestellt.

3 Konsistenztest zwischen Layout und Gatter-Netzliste

3.1 Problemstellung und Lösungsansatz

Das grundsätzliche Problem beim Konsistenztest besteht darin, zu entscheiden, ob zwei gegebenen Netzlisten die gleiche elektrische Schaltung beschreiben. Prinzipiell kann man dabei so vorgehen, daß man beide Netzlisten für umfangreiche Beispieleingaben simuliert und dann die Simulationsergebnisse miteinander vergleicht. In der Praxis ist dies allerdings wegen der erforderlichen hohen Simulationszeit nur für sehr kleine Schaltungen durchführbar. Ein weiterer Nachteil der Methode besteht darin, daß man im Fehlerfall keinen Hinweis darauf erhält, worin sich die beiden Schaltungen voneinander unterscheiden.
Als Alternative zur Schaltungssimulation gibt es die Möglichkeit, die strukturelle Äquivalenz beider Netzlisten direkt nachzuweisen. Dies bereitet jedoch ebenfalls große Schwierigkeiten. Zum einen sind im vorliegenden Fall verschiedene Arten von Netzlisten miteinander zu vergleichen, weil der Schaltungsent-

```
** HILO netfile extracted from layout inan2a
CCT inan2a (I1A, I1B, outp)
PTRANS
   PTRANS1x0y0 (s4, eins, eins);
NTRANS
   NTRANS1x0y0 (s5, null, null);
PTRANS
   PTRANS1x1y0 (eins, outp, I1A);
NTRANS
   NTRANS1x1y0 (null, s6, I1A);
PTRANS
   PTRANS1x2y0 (outp, eins, I1B);
NTRANS
   NTRANS1x2y0 (s6, outp, I1B);
PTRANS
   PTRANS1x3y0 (eins, s7, eins);
NTRANS
   NTRANS1x3y0 (outp, s8, null);

INPUT I1A I1B;
WIRE s4 s5 s6 s7 s8;
SUPPLY0 null;
SUPPLY1 eins;
```

Abbildung 4: Aus dem Layout *inan2a.k* erzeugte HILO-Netzliste eines 2-stelliges *nand*-Gatters

wurf auf der Gatterebene durchgeführt wird, die Netzextraktion jedoch auf der Transistorebene arbeitet. Zum anderen liefert ein Extraktor üblicherweise nicht die beim Schaltungsentwurf verwendeten Komponenten- und Signalnamen. Dadurch benötigt man für den Konsistenztest einen aufwendigen Strukturvergleich, der in der Praxis nur mit exponentieller Rechenzeit in Abhängigkeit von der Schaltungsgröße durchgeführt werden kann.
Die Situation sieht jedoch günstiger aus, wenn die gegenseitige Entsprechung der Schaltkomponenten beider Netzlisten klar ist. In diesem Fall lassen sich die Gatter beider Netzlisten einheitlich sortieren, so daß der Äquivalenzvergleich im Prinzip durch den direkten Vergleich der miteinander korrespondierenden Signalnamen durchgeführt werden kann. Im vorliegenden Fall liegt diese günstige Situation vor, da sowohl für die Gatternetzliste, als auch für die Transistornetzliste die Plazierungsdaten im realisierten Layout vorliegen.
Für den Konsistenztest zwischen Layout und simulierter Netzliste durchläuft man daher sequentiell die Plazierungsliste des Layouts und gleichzeitig die simulierte HILO-Netzliste. Dabei arbeitet man die an den Grundzellen anliegenden Signale der HILO-Netzliste in die aus dem Handlayout extrahierten Transistornetzliste ein und erhält so die Zuordnung der bei der HILO-Simulation verwendeten Signalnamen zu den geometrischen Layoutpositionen.
Bei dem beschriebenen Vorgang, der sich mit fast linearer Laufzeit implementieren läßt, entdeckt man gleichzeitig alle Widersprüche zwischen der simulierten HILO-Netzliste, den Simulationsvoraussetzungen für die Grundzellen und der extrahierten Netzliste. Daher lassen sich gleichzeitig die benötigten Fehlermeldungen generieren, wenn etwa ein Kurzschluß im Layout vorliegt, eine Leitung fehlt oder eine benötigte Gate-Isolation nicht realisiert wurde.

3.2 Beispiel

Bei dem skizzierten Konsistenztest zwischen Layout und Gatternetzliste entsteht ein Bearbeitungsprotokoll, wie es ausschnittsweise in der Abbildung 5 dargestellt ist. Aus dem Protokoll sind die Signalnamen der Gatternetzliste und die entsprechenden Signalnummern der extrahierten Netzliste ersichtlich. Beispielsweise liefert der erste 3-zeilige Block des Protokolls Informationen, über die Anschlußkonfiguration der Komponente *FAAO2I422*. Dieses Gatter wurde durch Plazierung der Zelle *FAAO2I* auf die Kernzellenposition (70,40) des Sea-of-Gates Masters realisiert. Die Darstellung der realisierten Anschlüsse zeigt beispielsweise, daß das Signal *s274* der simulierten Gatternetzliste im Layout die Nummer 150 erhält und das Signal *phi1* im Layout durch die Nummer 251 dargestellt wird. Bei der Abarbeitung aller Gatter ergibt sich schrittweise eine vollständige Umsetzungstabelle zwischen den Signalbezeichnern beider Netzlisten.
Am Beispiel des Gatters *SHIFTHALF425* erkennt man eine fehlende Verbindung im Layout. Hier tritt das Signal *phi1* gemeinsam mit der Nummer 244 auf, obwohl *phi1* zuvor mit dem Layoutsignal 251 identifiziert wurde. Daraus ergibt sich, daß die entsprechenden Pins der Gatter *SHIFTHALF425* und *FAAO2I422* im Layout nicht wie in der Gatternetzliste gefordert verbunden sind. Auf entsprechende Weise erkennt die Software am Gatter *SHIFTHALF426* eine Lay-

```
Komponente FAAO2I422 an Position 70 14
Layout-Netzliste: faao2i    ( 150,   236,   249,   238,   251)
Gatter-Netzliste: FAAO2I    (s274,  s404,  s504,  s403,  phi1)

Komponente IINV1A423 an Position 68 13
Layout-Netzliste: iinv1a    ( 231,   150)
Gatter-Netzliste: IINV1A    (s500,  s274)

Komponente IINV1A424 an Position 68 14
Layout-Netzliste: iinv1a    ( 249,   232)
Gatter-Netzliste: IINV1A    (s504,  s429)

Komponente SHIFTHALF425 an Position 77 13
Layout-Netzliste: shiftr1   ( 239,   249,   244,   243)
Gatter-Netzliste: SHIFTHALF (gval,  s504,  phi1, nphi1)

FEHLER: Fehlende Verbindung (HILO-Signal phi1) zwischen 251 und 244 !!!

Komponente SHIFTHALF426 an Position 81 13
Layout-Netzliste: shiftl1   ( 150,   239,    244,  243 )
Gatter-Netzliste: SHIFTHALF (s600,  gval,   phi1, nphi1)

FEHLER: Kurzschluss (Layoutsignal 150) zwischen s274 und s600 !!!
```

Abbildung 5: Auschnitt aus einem Protokoll beim Vergleich einer extrahierten Netzliste mit einer Gatter-Netzliste.

outverbindung zwischen unterschiedlichen Signalen der Gatternetzliste. Diese Situation entspricht einem Kurzschluß im Layout.
Man beachte auch, daß die im Protokoll verwendeten Zellnamen in den beiden Netzlisten unterschiedlich sein können. Beispielsweise wurde bei der Komponente *SHIFTHALF426* an Position (81,13) statt der in der Gatternetzliste vorgesehenen Zelle *SHIFTHALF* im Layout die Zelle *shiftl1* verwendet. Solche Abweichungen sind sinnvoll und notwendig, damit der Designer beim Layoutentwurf die Freiheit hat, je nach den aktuellen geometrischen Gegebenheiten unterschiedliche Varianten einer Zelle verwenden zu k nnen. Beispielsweise können die Zellvarianten bei gleichem logischen Verhalten unterschiedliche Pinpositionen oder ein unterschiedliches Längen/Breiten-Verhältnis besitzen. Die Äquivalenz der Varianten wird in der Praxis durch Simulation nachgewiesen.

3.3 Logische Korrektheit von Sea-of-Gates Layouts

Wie beschrieben, besteht das Ziel des Konsistenztests darin, die logische Korrektheit eines realisierten Layouts nachzuweisen. Die Grundidee besteht dabei darin, aus der Verwendung korrekter Grundzellen und der korrekten Realisierung der Gatternetzliste auf die logische Korrektheit des gesamten Layouts zu schließen.
Mit Hilfe der bisher skizzierten Tests wurde überprüft, ob das elektrische Netz auf den Pins der plazierten Komponenten mit dem Netz der vorgegebenen Gatternetzliste übereinstimmt. Auf die ähnliche Weise wird auch geprüft, ob zusätzliche Bedingungen, wie etwa korrekte Anschlüsse an die Stromversorgung oder die Sperrung von Transistoren für die Gate-Isolation-Technik, erfüllt sind. Sind alle diese Prüfungen erfolgreich verlaufen, dann ist nachgewiesen, daß die vorgegebene Gatternetzliste korrekt in eine *inter-cell* Verdrahtung des Layouts umgesetzt wurde. Die Korrektheit der *intra-cell*-Verdrahtung der einzelnen Gatter ergibt sich direkt aus der erfolgreichen Simulation der extrahierten Netzlisten der Grundzellen. Jetzt bleibt nur noch zu überprüfen, ob unzulässige "externe" Verbindungen zwischen zellinternen Signalen bestehen. Diese Situation ist in der Abbildung 6 skizziert. Beim beschriebenen Netzlistenvergleich ist diese Überprüfung leicht dadurch zu erreichen, daß die zellinternen (lokalen) Signale einer Komponente wie auch die Signale der Anschlußpins auf eine eindeutige Entsprechung im Layout getestet werden.
Damit sind die vom Konsistenztest durchgeführten Überprüfungen vollständig, so daß durch den erfolgreichen Konsistenztest tatsächlich die logische Korrektheit der Layoutrealisierung nachgewiesen ist.

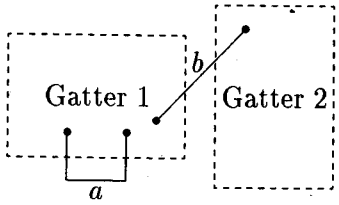


Abbildung 6: Fehlerhafte Verbindungen zwischen "inneren" Signalen von Gattern

4 Zusammenfassung und Ausblick

Mit den vorgestellten Programmen ist es möglich, sowohl Grundzellen, als auch relativ komplexe in Handarbeit erstellte Sea-of-Gates Layouts auf ihre logische Korrektheit zu überprüfen. Dabei ist es gelungen, die besonders laufzeitintensiven Probleme, wie etwa die Netzextraktion aus nicht orthogonalen Layoutstrukturen und den Netzlistenvergleich für unterschiedliche Abstraktionsebenen, so zu umgehen, daß sämtliche Programme mit einer fast linearen Laufzeit in Abhängigkeit von der Schaltungsgröße arbeiten.
Der praktische Einsatz im Entwurfsprojekt hat gezeigt, daß die gewählte Vorgehensweise wegen der günstigen Programmlaufzeiten auch für regelmäßige Überprüfungen von Teillayouts praktikabel ist und somit Entwurfsfehler sehr früh entdeckt werden können.
Bei der Implementierung wurde darauf geachtet, daß sich die Programme leicht für unterschiedliche Sea-of-Gates Prozesse verwenden lassen. Insbesondere kann der Netzextraktor mit Hilfe der in [Beu89] vorgestellten "Core-Cell-Description-Language" für spezielle Master konfiguriert werden. Damit wird das Programm auch für die neuen Sea-of-Gates Strukturen des IMS verwendbar sein. Es ist geplant, die Programme weiter auszubauen, so daß sie auch ohne besondere Vorkenntnisse leicht einsetzbar werden.

Literatur

[Beu89] M. A. Beunder. *Design and Analysis of Semi-Custom Architectures.* Dissertation, Institut für Mikroelektronik, Stuttgart, 1989.

[Meh84a] K. Mehlhorn. *Data Structures and Algorithms 2 – Graph Algorithms and NP-Completeness.* Springer Verlag, 1984.

[Meh84b] K. Mehlhorn. *Data Structures and Algorithms 3 – Multi-dimensional Searching and Computational Geometry.* Springer Verlag, 1984.

[Rül91] W. Rülling. *Ein VLSI-Chip zur schnellen Berechnung von Faltungsintegralen.* MPC-FH Workshop in Esslingen, 1991.

Entwurfsbaukasten für GateForest_1.2 Gate Arrays

Bearbeiter : Rathenow Markus
Fachbereich : Nachrichtentechnik
Betreuer : Prof. A. Führer
Fachhochschule : Ulm

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Schematisches Konzept des Baukastens

Hinter dem Begriff des Baukastens verbergen sich Programme und beschreibende Dateien. Sie ermöglichen dem Anwender des Baukastens auf einfache Weise eine logische Schaltung zu simulieren, um sie anschließend auf einem Chip der GateArray Familie GateForest_1.2 des IMS zu realisieren.
Die Dateien und Dienstprogramme, die diese Aufgabe übernehmen sind in dem Verzeichnis **~/ims** untergebracht. (Bild 1-1)

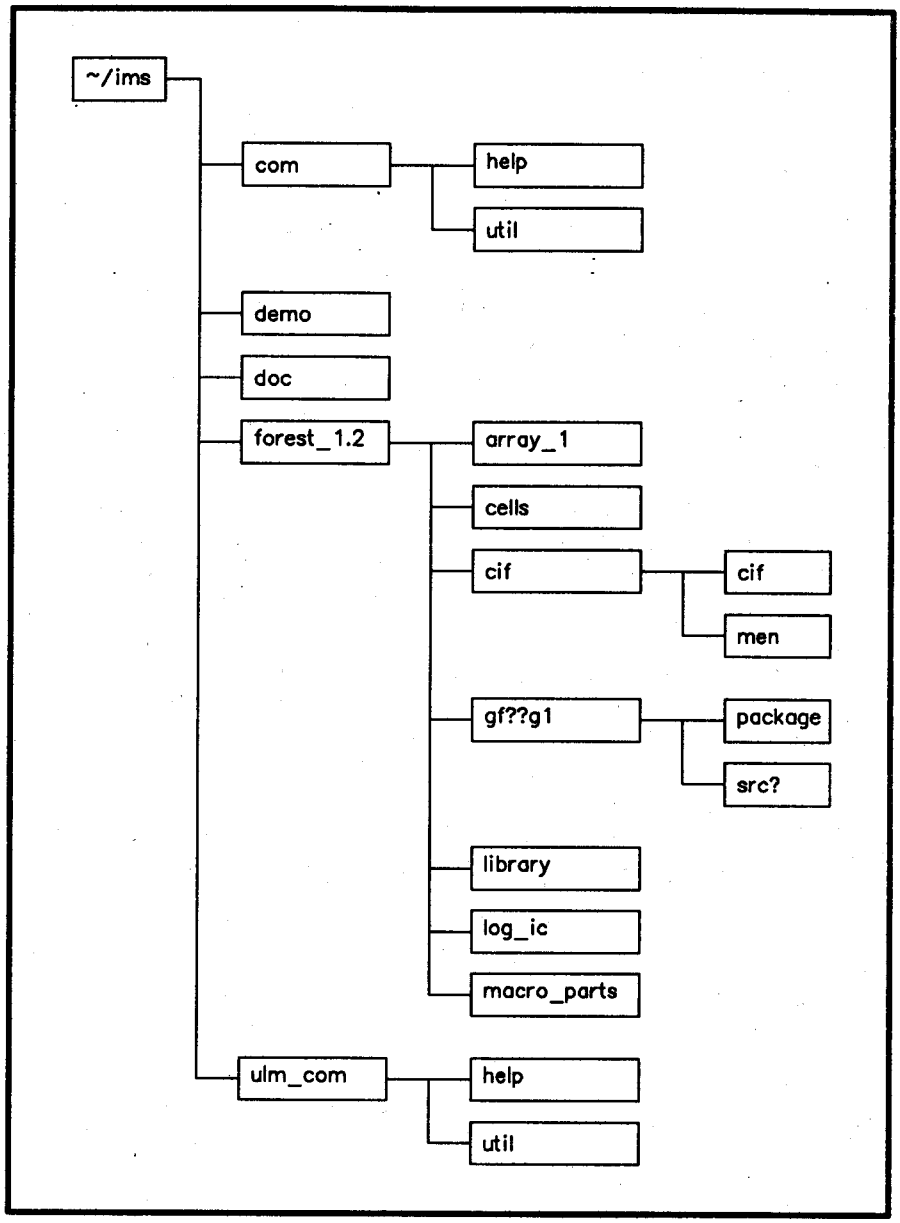


Bild 1-1 Verzeichnisstruktur von ~/ims

Die Struktur diese Verzeichnisses ist von der Firma Mentor Graphics vorgeschrieben und darf nur erweitert, nicht jedoch verändert werden.

Bei den **Dienstprogrammen des Baukastens** handelt es sich um Batches auf der Shell-Ebene des Betriebssystems AEGIS. In diesen Batches werden kompilierte und/oder interaktive Programme eingebunden. Sie werden dabei mit der richtigen Syntax aufgerufen und übernehmen die eigentlichen Aufgaben des Dienstprogrammes. Es wird damit das Prinzip verfolgt, daß der Benutzer diese Programme an seine Aufgaben (Technologie) anpassen kann: Der Aufruf von interaktiven Programmen mit angefügten Befehlsfolgen in der dafür notwendigen **Human-Interface-Macro-Language** ermöglichen dem Benutzer, Veränderungen in der Ausführung dieser Programme vorzunehmen.
Die Dienstprogramme sind in zwei Verzeichnisse untergebracht. Im Verzeichnis **~/ims/com** sind unter anderem Dienstprogramme des IMS untergebracht.
Die Programme des Verzeichnisses **~/ims/ulm_com** sind von der alten 2um-Technologie aus dem Verzeichnis ~/ims_kit übernommen und von mir an die neue Technologie angepaßt worden.

Die **Dateien** für die Dienstprogramme sind in dem Technologieverzeichnis **~/ims/forest_1.2** untergebracht. Weitere Informationen dazu stehen im Kapitel 4.

Im Verzeichnis **~/ims/demo** sind Demonstrationsbeispiele untergebracht.

Das **Doc**-Unterverzeichnis enthält Informationen zur Installation des Baukastens.

Angepaßt sind die Dienstprogramme des Baukastens an den Design Kit Creation Package (DKCP) /2/ von Mentor, dessen Prinzip in Bild 1-2 dargestellt ist.

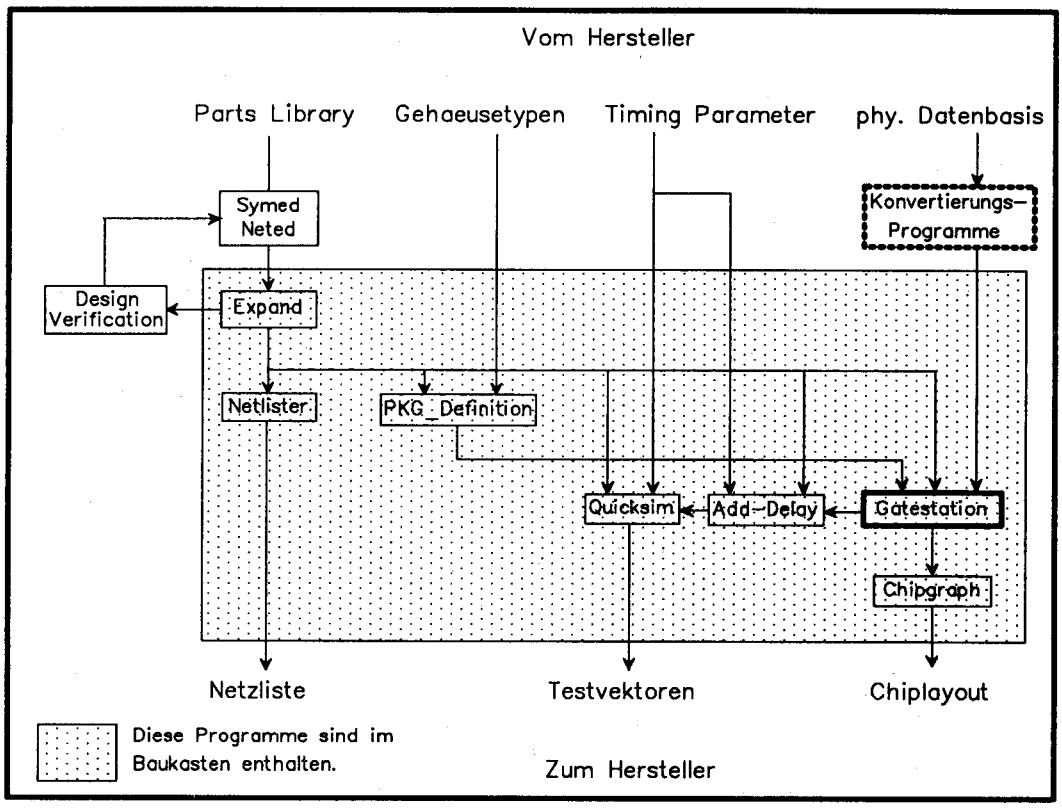


Bild 1-2 Konzept des Design Kit Creation Package

Erläuterungen zu dem Konzept in Bild 1-2:

Mit den beiden interaktiven Programmen **neted** und **symed** werden die logischen Stromlaufpläne eingegeben. Für die Entwicklung einer Schaltung auf dem IMS Gate Forest_1.2 werden dabei ausschließlich standardisierte Bauteile (siehe Kapitel 2) aus der Parts Library der 1.2um Technologie (~/ims/forest_1.2/cells) verwendet.

Das Programm EXPAND entspricht den beiden Dienstprogrammen **ims_expand_comp** und **ims_expand_design** des Baukastens. Sie stellen die notwendigen Daten aus dem Stromlaufaufplan für die weiteren Simulationen zusammen.

Hinter dem Programm NETLISTER stecken zwei weitere Dienstprogramme des Baukastens: **ims_mifnet** und **ims_smplnet** liefern zwei unterschiedliche Netzlisten.

Das Dienstprogramm **ims_pkg_def** übernimmt die Aufgabe der PKG-Definition. Die Aufgabe dieses Dienstprogramms ist die Vorplazierung der Anschlußzellen des IC's (Padzellen) und somit die Anpassung des IC's an seine Peripherie. Das Programm **ims_pkg_def** greift dabei auf die zur Verfügung stehenden Gehäusedaten zu.

Die für die Simulation notwendigen Programme sind QUICKSIM und ADD-Delay. Im Baukasten werden sie aus den Dienstprogrammen **ims_add_delay** und **ims_simlog** heraus aufgerufen.

Bei dem hervorgehobenen Begriff **gatestation** handelt es sich um ein umfassendes MENTOR-Graphics Software-Paket, welches die logischen und physikalischen bzw. geometrischen Daten einer Schaltung verarbeitet. Diese Software ist ausschließlich für die Layout-Erstellung verantwortlich. Sämtliche für diese Aufgabe notwendigen Programme der GateStation sind in dem Dienstprogramm **ims_route** zusammengefaßt:

> LOGIC_ENTRY: Es verbindet das Design mit der Technologie und erzeugt das physikalische Design.
> GATEPLACE plaziert die Macrozellen auf dem Chip.
> GATEROUTE verdrahtet die plazierten Macrozellen
> PREGRAPH erzeugt aus dem Design GATEGRAPH-Files, damit der Entwurf der Schaltung mit GATEGRAPH angeschaut und modifiziert werden kann.
> CHIPGRAPH_OUTPUT erzeugt aus dem Design CHIPGRAPH-Files, damit der Entwurf mit Chipgraph weiterverarbeitet werden kann.
> DDF_OUTPUT erzeugt ein DDF-File für ADD_DELAY

Mit Hilfe des Programms **chipgraph** können die aus der GATESTATION gewonnenen Chip- und Makrozellenstrukturen, sowie die verdrahteten Metallverbindungen in geometrisch korrekter Form dargestellt werden. Das Dienstprogramm **cg** erleichtert dabei den Aufruf von CHIPGRAPH.

Eine weitere Vereinfachung in der Handhabung des Baukastens gewährleistet das Dienstprogramm **ims_package**. Beim Aufruf dieses Programmes werden alle für die Simulation und Erstellung des Layouts notwendigen Dienstprogramme des Baukastens in der richtigen Reihenfolge und Syntax ausgerufen. (Siehe dazu Kapitel 4.1)

Der Block mit den Konvertierungsprogrammen beinhaltet alle Schritte die notwendig sind die vorliegenden Daten in die für die Gatestation abstrakte Form zu bringen.

2. Gate Array Technology

2.1 Allgemeines zum IMS GATE FOREST 1.2

Der IMS GATE FOREST ist eine Semicustom Entwurfsumgebung auf der Basis des Sea-of-gates Konzepts in CMOS Technologie. Das bedeutet, daß auf dem Wafer die Prozesschritte bis zum ersten Zwischenoxid vorgefertigt werden. Die anwendungsspezifische "Programmierung" geschieht dann durch die Verdrahtung der vorgefertigten Transistoren.
Beim Sea-of-gates Prinzip wird im Gegensatz zum normalen Gate Array der ganze Kernbereich des Chips mit Transistoren aufgefüllt. Die Transistoren sind dabei in Reihen angeordnet. Durch die "GateIsolation" Technik werden aktive Transistoren voneinander isoliert. (Zwischen den aktiven Transistoren befindet sich ein Transistor dessen Kanal gesperrt ist.).
Die Zahl 1.2 beim IMS GATE FOREST weist auf die Technologie hin und bedeutet daß die Länge des Kanals 1.2um lang ist.
Das Hauptanwendungsgebiet des IMS GATE FOREST ist die schnelle und kostengünstige Prototypen- und Kleinserienfertigung anwendungsspezifischer Schaltungen (ASIC's) basierend auf Zellen- und Macrobibliotheken.
Das Verdrahten erfolgt durch Plazieren der standardisierten Zellen aus der Macrobibliothek und anschließendem Routen der Verbindungen zwischen den Ein- und Ausgängen (Ports) der Macrozellen. Es stehen der Verdrahtung zwei Metallisierungsebenen zur Verfügung. Sie ist an ein Routing Grid (vorgegebenes Raster mit festen Abständen) gebunden. Die Mittelpunkte der Kontakte zwischen den Bahnen (VIAS) und zwischen der unteren Verdrahtungsebene mit dem Poly (CONT) liegen auf den Kreuzungspunkten des Grids. Die Leiterbahnen sind auf dem Grid mittenzentriert.

2.2 Masterbeschreibung der GFXXG1 Masterfamilie

Als Master bezeichnet man den schon vorgefertigten Teil einer Semicustom-Schaltung.
Der folgende Text soll die Layoutdaten und die damit verbundenen Layoutregeln zusammenstellen.
Dieser Abschnitt soll dazu dienen, daß weitere Master der GFXXG1-Familie in dem für die GateStation notwendigen Datenformat beschrieben werden können und somit für den IMS-Design-Kit zur Verfügung stehen.

Allgemeines:

Je nach Komplexität der Schaltungen stellt das IMS verschiedene Mastergröße zur Verfügung (Tabelle 2-1):
Von diesen Mastern sind der GFG-12, der GFG-9 und der GFG-4 für die Erstellung eines Layouts mit dem Baukasten von mir schon beschrieben worden.
Für diese 3 Master existiert zudem eine tabellarische Darstellung der physikalischen Werte (Tabelle 2-3). Von den restlichen zwei Mastern fehlten zur Zeit der Diplomarbeit die Daten.

Master	Realisierbare Gatter (NAND2)	Anzahl der Padzellen
GFG-16	650	32
GFG-12	1100	44
GFG-9	2000	56
GFG-4	5700	96
GFG-2	12700	144

TABLE 2-1 Digitale Master

Die Master lassen sich in zwei Bereiche gliedern. Dies ist zum einen der Kernbereich, der aus dicht plazierten Elementartransistoren aufgebaut ist (Sea-of-Gates Struktur).
Der andere Bereich ist der Padbereich. Er beinhaltet die Input/Output-Strukturen. Zwischen Kern- und Padbereich liegen keine aktiven Masterstrukturen (Transistoren).

Padbereich der Master:
Hierunter versteht man den äußeren Ring des Masters, der als Plazierungsbereich für die Padzellen (Input/Output-Strukturen) gedacht ist.
Die Anordnung dieser Padzellen ist in vier Bereiche (S, E, N und W) aufgegliedert. Die Durchnummerierung der möglichen Padplazierungen setzt sich aus einem dieser vier Buchstaben und einer dreistelligen Zahl zusammen. (Bild 2-1).

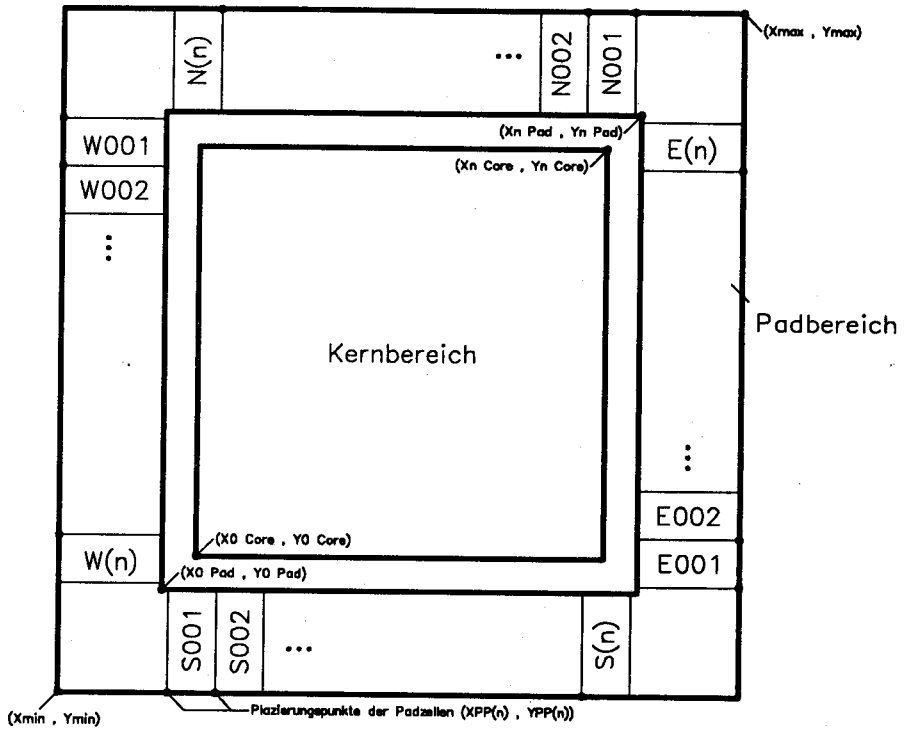


Bild 2-1 Aufbau der GATE FOREST Master

Corebereich (Kernbereich):
Dieser Bereich füllt den inneren Bereich des Masters aus und ist bezüglich der Außenabmessungen mittenzentriert. Er ist aus lauter dicht plazierten Elementar-Transistoren aufgebaut (Sea-of-Gates Struktur).
Die gesamte Struktur läßt sich vereinfacht durch die Vervielfachung und Spiegelung einer einzelnen Site-Zelle (nicht zu verwechseln mit der Makrozelle) beschreiben. Aufgebaut ist diese Zelle aus zwei unterschiedlich großen N-Kanaltransistoren und einem P-Kanaltransistor (Bild 2-2). An diese Site-Zelle sind bei der Entwicklung mehrere Anforderungen gestellt worden:

Portierung auf andere Technologien: Bei Technologien unter 2um Gatelänge ist ein lineares Skalieren der Geometrien (kleiner oder größere Strukturen) nach der Lamda Philosophie nicht mehr möglich. Deshalb wird für die Personalisierung ein Grid definiert. Beim Wechsel der Technologie wird der Abstand des Grids verändert (6um Gridabstand bei der 1.2um Technologie, 4um bei der 0.8um Technologie).

Unterstützung des Elektronenstrahl-Direktschreibens durch orthogonale Strukturen.

Da bei komplexen Schaltungen die Fläche für die Verdrahtung immer mehr Platz in Anspruch nimmt, wird mit *Plazierungs- und Verdrahtungsprogrammen* integriert.

Zur Plazierung der standardisierten Makrozellen müssen die Strukturen der Site-Zelle SITES. Unter einer Site versteht man die Plazierungsfläche, die eine Macrozelle belegen kann. Die Macrozellen des Corebereichs benötigen dabei mehrere benachbarte SITES. Vorstellen kann man sich eine Site als einen markierten Stellplatz auf einem Parkplatz. Die Definition einer Site des Corebereichs (man unterscheidet zwei Typen von Sites: Sites für den Corebereich und Sites für den Randbereich) fällt in etwa mit der äußeren Umrandung einer Mikrozelle zusammen (Bild 2-3). Die Punkte im Kernbereich auf die der Koordinatenursprung der Sitezelle (linke untere Ecke einer ungespiegelten Site) abgebildet wird (beim Aneinanderreihen und beim Spiegeln) sind Plazierungspunkte der Makrozellen im Kernbereich des Masters.
Den Aufbau des Kernbereichs zeigt Bild 2-3. Die linke untere Site hat die Orientierung 'N' (North). In vertikaler Richtung wird um eine horizontale Achse gespiegelt und man erhält somit die beiden nach oben fortlaufenden Orientierungen 'S' und 'N'.
Zu beachten ist hierbei, daß sich die Orientierung für die Makrozellen je nach Plazierung ändert. Bei der Plazierung einer Makrozelle auf den Ursprung einer 'S'-orientierten Site wird die Zelle ebenfalls gespiegelt dargestellt. Als Referenzpunkt bei der Plazierung wird die linke untere Ecke der Macrozelle verwendet.
Erkennbar ist auch, daß die horizontalen Spiegelachsen innerhalb der Sitezellen liegen. Es ergeben sich somit Überlappungen der Sites in vertikaler Richtung.
In horizontaler Richtung werden die Sitezellen nur um 12um verschoben plaziert.
Hinweis: Die Anzahl der horizontalen Sites wurde in der abstrakten Beschreibung CHIP.TDF für die Gatestation verkleinert. Der Grund liegt bei der möglichen überlappenden Plazierung der Macrozellen im Corebereich. (siehe Kapitel 3.1)

Zu beachten ist, daß der Rand des Corebereichs aus Randzellen besteht, die ihn nach außen hin isolieren. Diese Isolation ist notwendig, da von den Padzellen Störströme in den Corebereich gelangen könnten. Es fällt dadurch am linken und rechten Rand des Cores jeweils eine Spalte von SITES (12um) für die Platzierung weg.

Das zur Verdrahtung wichtige **Routing Grid** überdeckt den ganzen Corebereich bis hin zum Padbereich. Um die Padzellen anzuschließen ragt das Routing Grid etwas in den Padbereich hinein. (Bild 2-4)
Unter dem Routing Grid versteht man mehrere horizontale und vertikale Linien, sogenannte Tracks in Form eines Drahtgeflechts. Die Plazierung von Kontakten (Verbindung zwischen dem unteren Metall-1 und dem Substrat) und Vias (Verbindung zwischen Metall-1 und Metall-2), sowie die Metallbahnen sind strikt an das Routing Grid gebunden. Kontakte und Vias müssen zentrumsorientiert auf den Kreuzungspunkten der Tracks plaziert werden. Metallbahnen müssen mittenzentriert auf dem Grid verlaufen.
Es wird bei dieser Regelung automatisch der für diese Technik notwendige Abstand zwischen den Metallbahnen eingehalten.

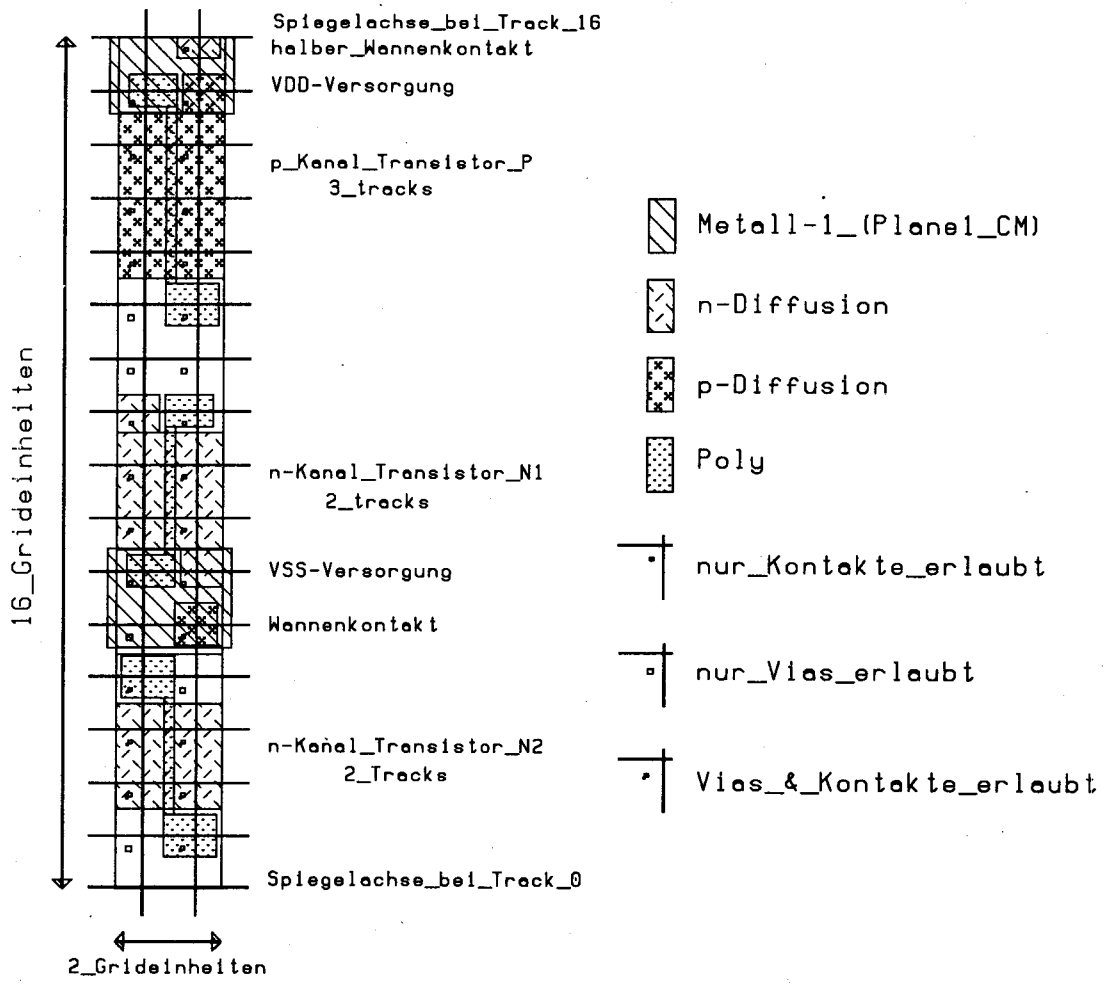


Bild 2-2 Site-zelle mit VSS/VDD-Spannung

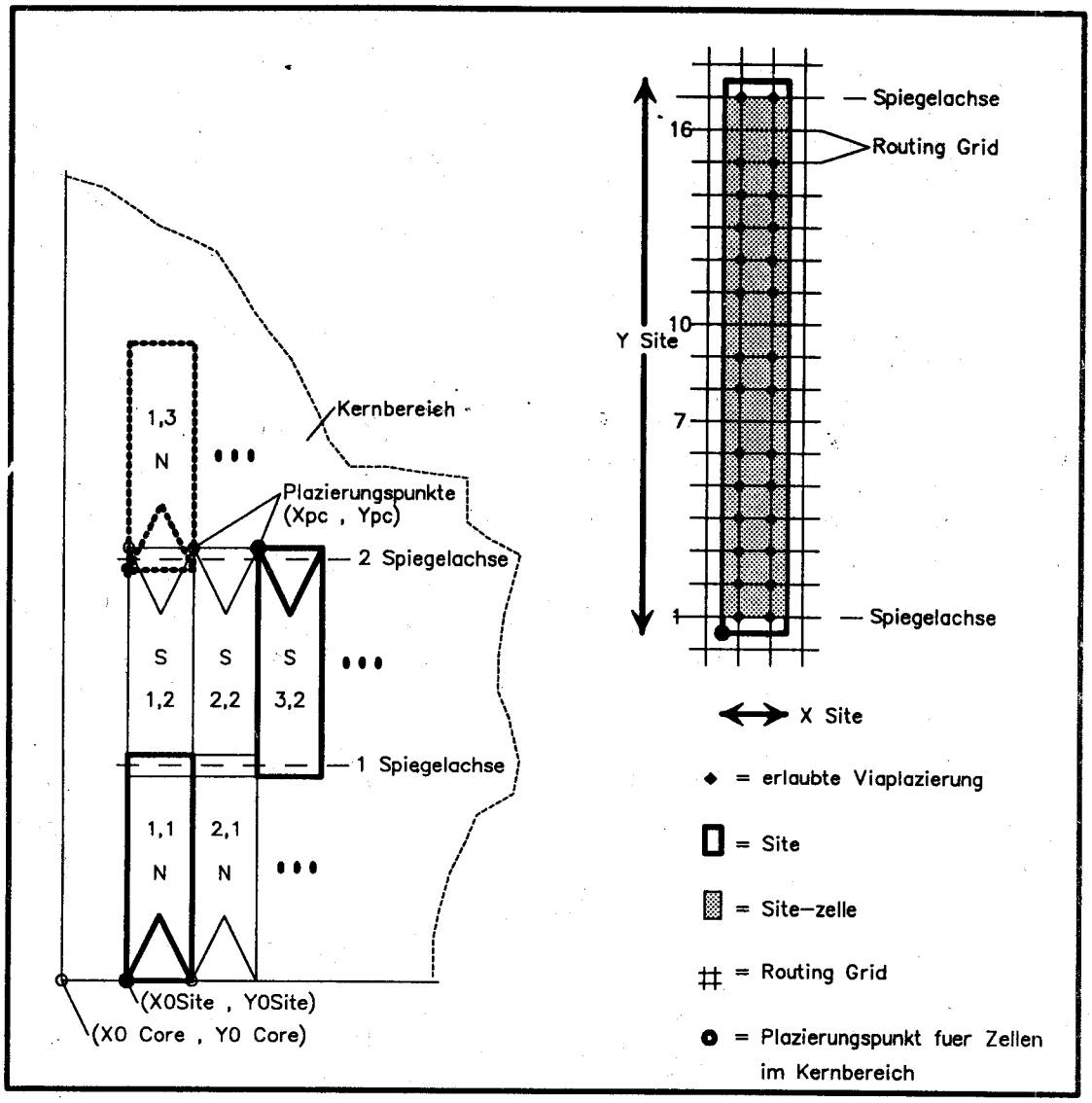


Bild 2-3 Aufbau des Kerns und Viaplazierung innerhalb einer Site

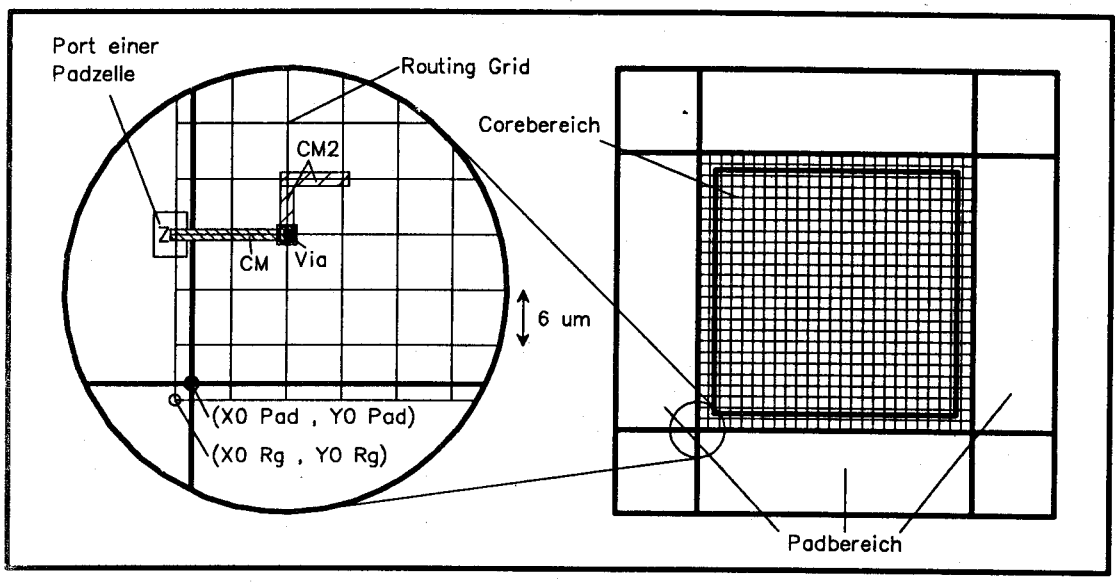


Bild 2-4 Lage des Routing Grids

In den folgenden Tabellen sind die Daten der beschriebenen Master angegeben. Vorgegeben waren die Daten des GFG-9 vom IMS. Die Werte der beiden anderen Master GFG-12 und GFG-4 sind von mir aus den Layoutdaten bestimmt worden. Die Werte der ersten Tabelle sind allgemeingültig und masterunabhängig.

Verbale Beschreibung	Symbol	Wert /um/
Breite einer Site	X_{Site}	12
Höhe einer Site	Y_{Site}	102
Überlappung der Sites horizontal	$O_{H\ Site}$	0
Überlappung der Sites vertikal	$O_{V\ Site}$	6
Breite der Sites horizontal	$d_{H\ Site}$	12
Breite der Sites vertikal	$d_{V\ Site}$	96
Abstand des Routing Grids (vertikal & horizontal)	d_{Grid}	6
Metallbahnbreite in Metall1 (untere Metallebene)	b_{CM}	2
Metallbahnbreite in Metall2 (obere Metallebene)	b_{CM2}	2.4

TABLE 2-2 Allgemeine Masterdaten

Verbale Beschreibung	Symbol	GFG-12 Wert [µm]	GFG-9 (Vom IMS) Wert [µm]	GFG-4 Wert [µm]
Linke untere Ecke des Masters	Xmin, Ymin	0 , 0	0 , 0	0 , 0
Rechte obere Ecke des Masters	Xmax, Ymax	2970 , 4290	4290 , 4290	6490 , 6490
Linke untere Ecke des Corebereichs	X0 Core, Y0 Core	603 , 606	603 , 606	611 , 650
Rechte obere Ecke des Corebereichs	Xn Core,Yn Core	2367 , 3684	3687 , 3684	5879 , 5840
Linke untere Ecke des Padbereichs (innen)	X0 Pad, Y0 Pad	495 , 495	495 , 495	495 , 495
Rechte obere Ecke des Padbereichs (innen)	Xn Pad, Yn Pad	2475 , 3795	3795 , 3795	5995 , 5995
Plazierungspunkte der Pads am südl. Rand	XPP(Sn), YPP(Sn)	385+n*220 , 0 n ∈ 1 , .. , 8	385+n*220 , 0 n ∈ 1 , .. , 14	385+n*220 , 0 n ∈ 1 , .. , 24
Plazierungspunkte der Pads am östl. Rand	XPP(En), YPP(En)	2970 , 385+n*220 n ∈ 1 , .. , 14	4290 , 385+n*220 n ∈ 1 , .. , 14	6490 , 385+n*220 n ∈ 1 , .. , 24
Plazierungspunkte der Pads am nördl. Rand	XPP(Nn), YPP(Nn)	2585-n*220 , 4290 n ∈ 1 , .. , 8	3905-n*220 ,4290 n ∈ 1 , .. , 14	6105-n*220 , 6490 n ∈ 1 , .. , 24
Plazierungspunkte der Pads am westl. Rand	XPP(Wn),YPP(Wn)	0 , 3905-n*220 n ∈ 1 , .. , 14	0 , 3905-n*220 n ∈ 1 , .. , 14	0 , 6105-n*220 n ∈ 1 , .. , 24
Plazierungspunkte für Zellen im Kernbereich	XPC(n), YPC(m)	603+n*12 , 414+m*192 und 603+n*12 , 612+m*192 n ∈ 1 , .. , nxSites m ∈ 1 , .. , nySites/2	603+n*12 , 414+m*192 und 603+n*12 , 612+m*192 n ∈ 1 , .. , nxSites m ∈ 1 , .. , nySites/2	611+n*12 , 458+m*192 und 611+n*12 , 656+m*192 n ∈ 1 , .. , nxSites m ∈ 1 , .. , nySites/2
Linke untere Ecke der linken unteren Site	X0Site, Y0Site	615 , 606	615 , 606	623 , 650
Rechte obere Ecke der rechten oberen Site	XnSite, YnSite	255 , 3684	3675 , 3684	5867 , 5840
Anzahl der Sites in horizontaler Richtung	nxSites	145	255	437
Anzahl der Sites in vertikaler Richtung	nySites	32	32	54
Linke untere Ecke des Routing Grids	(X0 Rg, Y0 Rg)	492 , 489	492 , 489	488 , 491
Rechte obere Ecke des Routing Grids	(Xn Rg, Yn Rg)	2478 , 3801	3798 , 3801	6002 , 5999

TABLE 2-3 Digitale Master

2.3 Macrobibliothek

Die Zell- oder Macrobibliotheken bestehen aus standardisierten Zellen der Digitaltechnik (AND, OR, MUX, FF ...). Der Vorteil bei der Verwendung von standardisierten Zellen ist, daß diese Zellen eine Vorverdrahtung auf dem Hintergrund darstellen und somit in ihrem logischen und zeitlichen Verhalten schon beschrieben sind.

Die zugehörigen Informationen der Zellen werden im Baukasten in drei Verzeichnissen abgelegt.

Das *logische Verhalten* ist im Verzeichnis ~/ims/forest_1.2/cells :
Zu jeder Zelle (Unterverzeichnis) ist ein Symbol für den Stromlaufplan (in NETED) angegeben. Hinzu kommt das Simulationsmodell (mit wenigen Ausnahmen Quickpartmodelle)

Die *Technologiedaten* für die GateStation sind im Verzeichnis ~/ims/forest_1.2/gf??g1/src? untergebracht:
Die Beschreibung des Masters ist in dem kompilierten Datenfile cadi.chp abgelegt. Die Daten über die Macrozellen sind in dem ebenfalls kompilierten Datenfile cadi.blc enthalten. Die Datengrundlagen für die beiden abstrakten Beschreibungen ist die physikalische Datenbasis.

Die ***physikalische Datenbasis*** der Macrocellen steht im Verzeichnis ~/ims/forest_1.2/library:
Layout der Zellen im Chipgraphformat

Auf diese physikalische Datenbasis greift der Anwender bei der Verwendung des Baukastens zur Erstellung des Layouts zu.

Es gibt drei verschiedene Arten von Macrozellen:

- standardisierte Zellen des Corebereichs
- standardisierte Padzellen
- masterabhängige Powerzellen (Stromversorgungszellen).

2.3.1 Standardisierte Zellen des Corebereichs

Die größte Gruppe bilden die **standardisierten Zellen des Corebereichs.**
Sie werden mit ihrem Koordinatenursprung (0,0) auf die Plazierungspunkte im Kernbereich plaziert.. Die Zellen können dabei auf ungespiegelte und/oder gespiegelte Sites plaziert werden; je nachdem wieviele **Tracks** (Linien des Routing Grids) für die Verdrahtung benötigt werden. Bei der Plazierung der Macrozellen werden je nach Größe mehr oder weniger Sites belegt. Die belegten Sites dürfen dann von einer anderen Macrozelle nicht mehr verwendet werden. (Bild 2-5). Werden entweder nur gespiegelte oder ungespiegelte Zellen plaziert, so stehen 20 freie Tracks zur Verfügung. Werden beide Arten der Plazierung verwendet, so verringert sich der Platz zwischen den Zellen zum Routen (**Routing Channel**) auf 9 Tracks (Bild 2-6). Weitere Tracks stehen eventuell in den Zellen selber zur Ausnutzung bereit.

In Bild 2-7 ist eine Zelle zur Veranschaulichung dargestellt. Bei dieser Darstellung sind nur die beiden Metallagen zu sehen.

Zellenbibliothek der Corezellen (~/ims/forest_1.2/library):

Addierer:	ad01d1
Logische Gatter:	an02d1, an03d1, an04d1, an05d1, an06d1, an07d1, an08d1 nd02d1, nd03d1, nd04d1, nd05d1, nd06d1, nd07d1, nd08d1 nr02d1, nr03d1, nr04d1, nr05d1, nr06d1, nr07d1, nr08d1 or02d1, or03d1, or04d1, or05d1, or06d1, or07d1, or08d1
Buffer:	ni01d2, ni01d4, ck01d2, ni02d4, nt02d4
Inverter:	in01d1, in01d4, it02d1
Multiplexer:	mx21d1, me21d1
Dekoder:	de12d1, de24d1
Latches:	labfnb,lacfnb, lanfnb, lapfnb, rsbnnb
Flipflops:	dfbtnb, dfctnb, dfntnb, dfptnb, mfbtnb, mfctnb, mfptnb, jkbtnb jkctnb

Die logischen Beschreibungen zu den Zellen sind im technischen Bericht IMS-TB-3/91 (IMS 1.2um GateForest Zellenkatalog Version 1.1) nachzulesen.

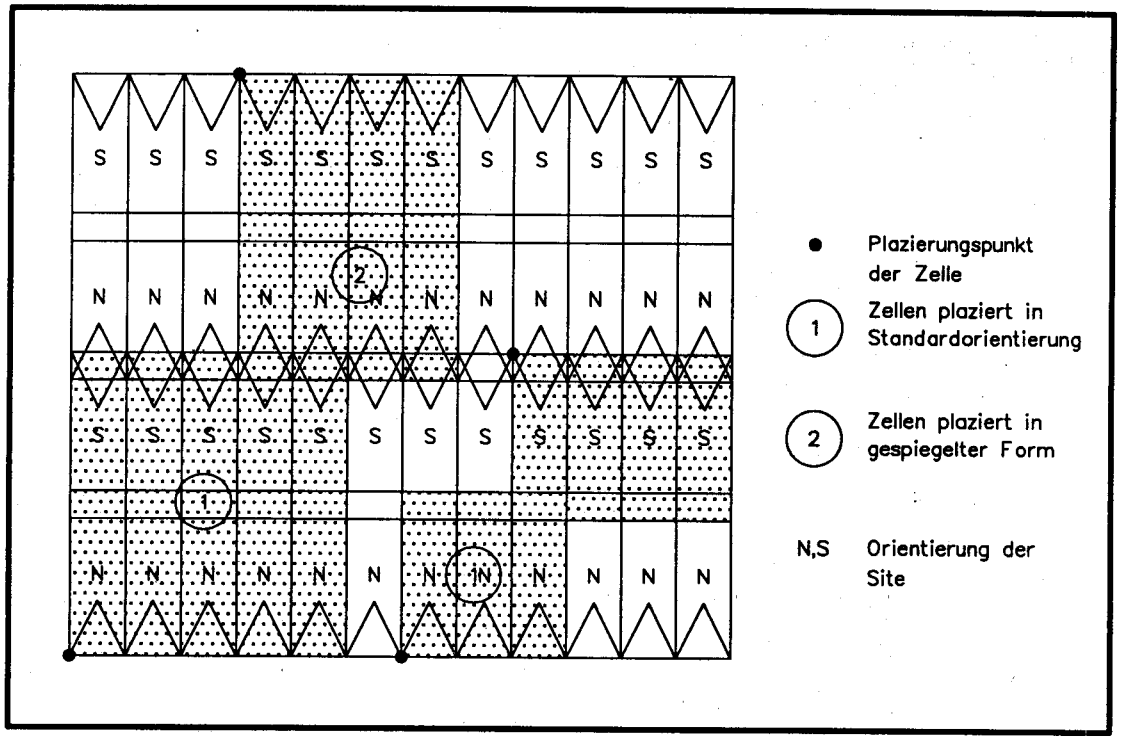


Bild 2-5 Plazierung ungespiegelter IMS GATE FOREST Zellen im Kernbereich

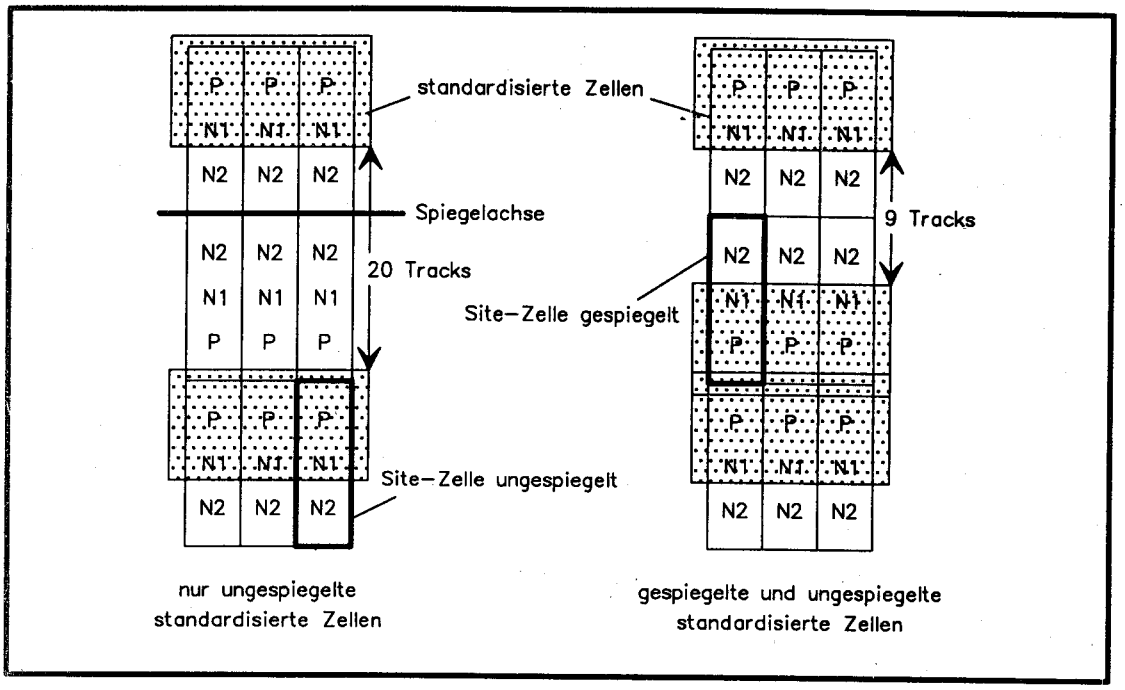


Bild 2-6 Freie Tracks im Verdrahtungskanal

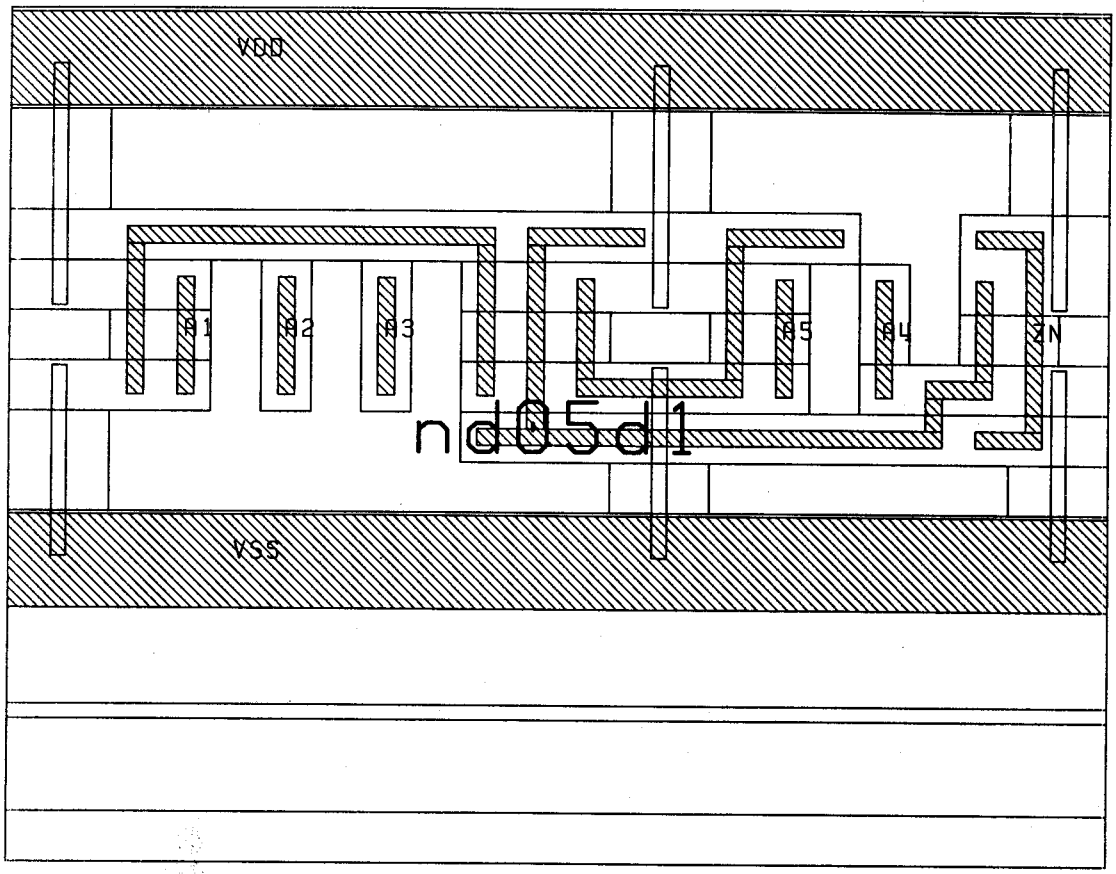


Bild 2-7 Macrozelle ND05D1 (NAND mit 5 Eingaengen)

Erstellung eigener Macrozellen:

Neben den existierenden Macrozellen kann der Benutzer eigene Zellen erstellen oder mehrere schon vorhandene Zellen zu einer neuen Zelle verschalten (z.B. Mux mit vier Eingängen oder SCAN-Path-FF).
Die Art der Erstellung von eigenen Leiterbahnen, Kontakten und Vias ist jedoch nicht frei wählbar. Bei der Erstellung ist die Verwendung von standardisierten Microzellen zwingend vorgeschrieben (siehe Bild 2-8). Die Macrozellen bestehen daher aus lauter zusammengesetzten standardisierten Microzellen. Durch Änderung der Referenzen auf die Microzellen kann ein einfacher Technologiewechsel z.B. auf 0.8um durchgeführt werden. Dabei wird nur der Inhalt dieser Zellen geändert, d.h. das Design als solches bleibt erhalten.

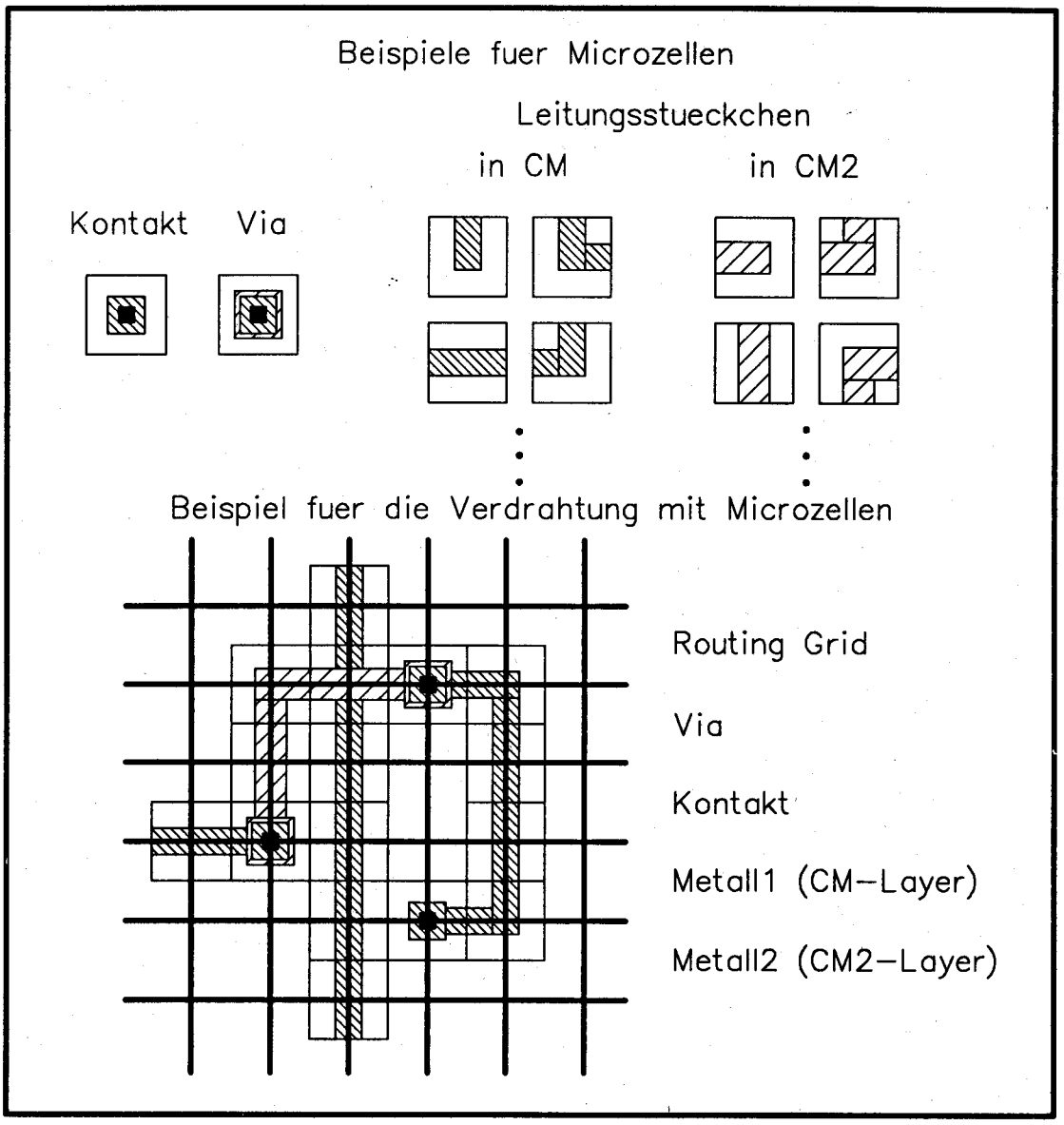


Bild 2-8 Verdrahtung mit Microzellen

2.3.2 Standardisierte Padzellen

Die **standardisierten Padzellen** werden an den Rändern plaziert. Dabei ergeben sich vier verschiedene Orientierungen:

Am südlichen Rand -->Orientierung = N (normal)
Am östlichen Rand -->Orientierung = R (um 90 Grad rotiert)
Am nördlichen Rand -->Orientierung = MXY (gespiegelt an x und y Achse)
Am westlichen Rand -->Orientierung = RMXY (um 90 Grad rotiert, gespiegelt an x und y Achse)

Im Gegensatz zu den Macrozellen des Corbereiches benötigen sie jeweils nur eine Sitezelle am Rand.
Bei der Plazierung gibt es Einschränkungen, da die Power_zelle gewisse Sites mit bestimmeten Padzellen schon vorbelegt.
Bei der Bearbeitung der Diplomarbeit lagen keine ausführlichen Layoutbeschreibungen der Padzellen vor. Angegeben waren nur die Lage und Größe der internen Ports und der Metallblockierungen.

Zellenbibliothek (~/ims/forest_1.2/library)

Eingangspadzellen:	idin1, ibin1, ibinu1, ibind1, itin1
Ausgangspadzellen:	iout1, iout2, itri2
Bidirektionale Padzellen:	ibid2
Power-Padzellen:	ivsr1, ivdr1, ivsc1, ivdc1

Weitere Informationen sind im selben technischen Bericht wie bei den Corezellen zu erfahren.

2.3.3 Powerzelle

Die **Powerzelle** ist von Master zu Master unterschiedlich aufgebaut. Durch sie ergeben sich starke Einschränkungen beim Plazieren und Verbinden (routen) der Macrozellen. Die Metallbahnbreiten der Powerzelle entsprechen nicht genau dem Trackraster. Die abstrakte Beschreibung der Powerzelle für die GateStation erfolgt somit im CHIP.TDF, damit es keine Layoutverletzungen (Kurzschlüsse) zwischen den Metallbahnen der Powerzelle und den Leitungen der Schaltung gibt

Funktion der Powerzelle:
Die Sea-of-Gates-Transistoren werden über horizontale Leitungen in Metall1 (untere Metallschicht) versorgt. Beim p-Kanal Transistor P, wie auch beim n_Kanal Transistor N1 verlaufen die Leitungen über einen der beiden Gateanschlüsse, die aktiven Gebiete der Transistoren und die Wannenkontakte (Bild 2-2). Auf diese Weise können Transistoren platzsparend durch "Gate Isolation' isoliert, aktive Gebiete an die Versorgung angeschlossen und die Wannenkontakte realisiert werden.

Die Breite der VDD- und VSS- Versorgungsleitungen die über den Corebereich laufen:

VSS-Bus:	11.2 um
VDD-Bus einfach:	11.2um
VDD-Bus doppelt:	17.2um

Veränderungen der Powerzelle gegenüber der alten 2um-Technologie:
Die Powerzelle beinhaltet nicht nur die kammartigen Versorgungsleitungen, sondern auch die zur Versorgung benötigten Padzellen im Randbereich. Es ergibt sich somit eine automatische Vorbelegung dieser Sites bei der Plazierung der Powerzelle auf dem Master. Bei der Plazierung der schaltungsbedingten Macrozellen (Padzellen) muß darauf geachtet werden, daß diese Sites frei bleiben.

GF4G1:	IVSR1 (VSS-Versorgung für den Padring):	W007, S008, E007, N008 W008, E008
	IVDR1 (VDD-Versorgung für den Padring):	W017, S017, E017, N017 W018, E018
	IVSC1 (VSS-Versorgung für die Corezellen):	W009, S009, E009, N009
	IVDC1 (VDD-Versorgung für die Corezellen):	W016, S016, E016, N016
GF9G1:	IVSR1 (VSS-Versorgung für den Padring):	W004, S005, E004, N005
	IVDR1 (VDD-Versorgung für den Padring):	W011, S010, E010, N010
	IVSC1 (VSS-Versorgung für die Corezellen):	W005, E005
	IVDC1 (VDD-Versorgung für die Corezellen):	W010, E009
GF12G1:	IVSR1 (VSS-Versorgung für den Padring):	W005, E005
	IVDR1 (VDD-Versorgung für den Padring):	W011, E010
	IVSC1 (VSS-Versorgung für die Corezellen):	W006, E006
	IVDC1 (VDD-Versorgung für die Corezellen):	W010, E009

In Bild 2-9 ist die Powerzelle des GFG-12 Masters dargestellt. Man erkennt darin die kammartigen Versorgungsleitungen und die schon vorplazierten Powerpadzellen.

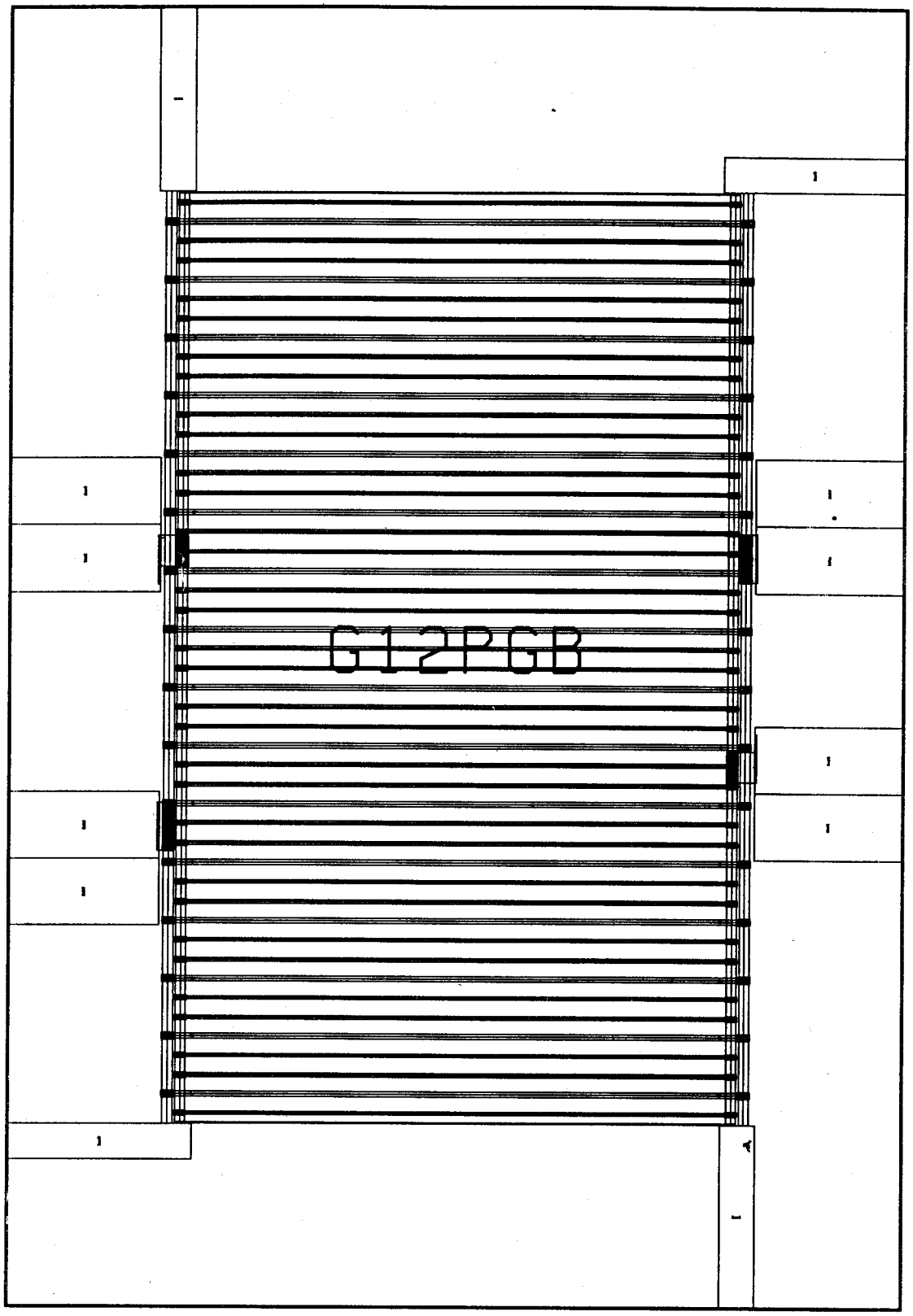


Bild 2-9 Powerzelle des GFG-12

3. Aufbau des IMS-Directories

Es ist darauf zu achten, daß diese Verzeichnisstruktur beibehalten wird. Sie darf auf keinen Fall geändert werden.
Sämtliche Programme des Baukastens sind durch die Verwendung von Links, Referenzen und Dateien an diese Struktur gebunden.

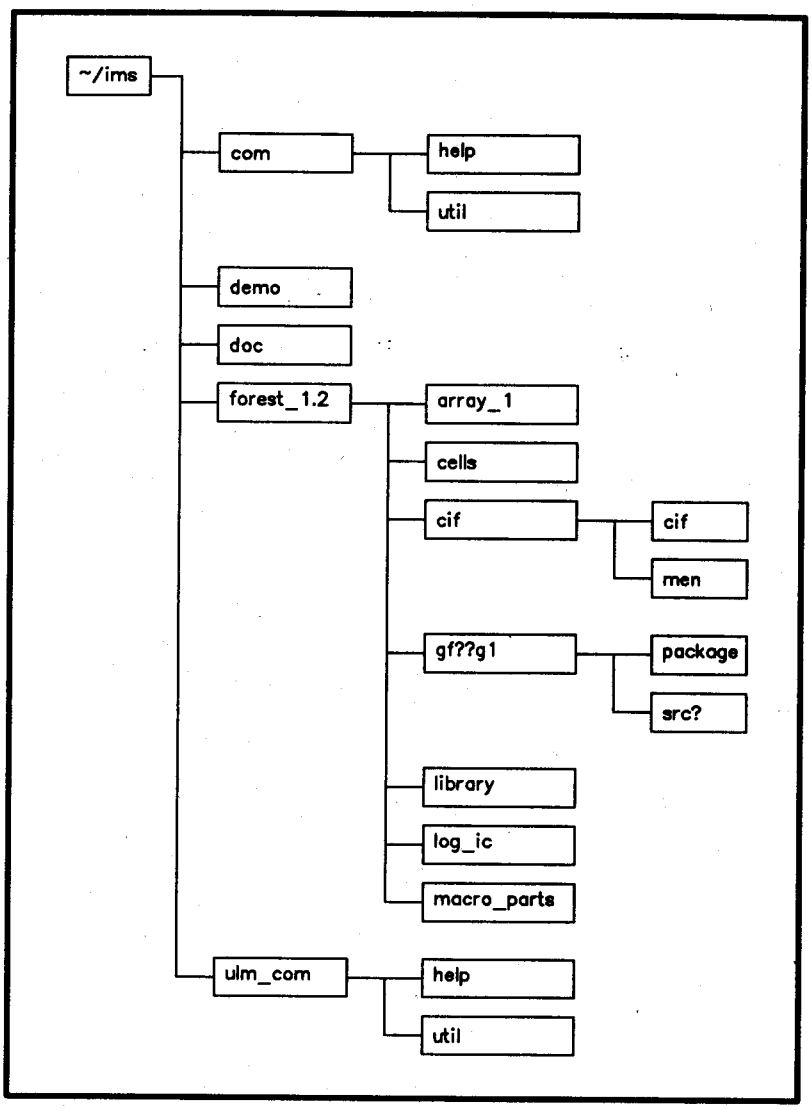


Bild 3-1 Verzeichnisstruktur des IMS-Directories

Kurze Beschreibung der Verzeichnisse:

(Das ~/ entspricht dem **/user/.**)

~/ims/
Hauptverzeichnis des IMS-DESIGN-KITS

~/ims/com/

Dieses Verzeichnis enthält die vom IMS gelieferten Batchprogramme. Auf einen Teil dieser Programme wird in den folgenden Kapiteln noch eingegangen. Es enthält zudem zwei Unterverzeichnisse.

~/ims/com/help/

Jeder Batch (IMS_...) beinhaltet die Möglichkeit, daß bei seinem Aufruf mit der Option -h ein Hilfetext angezeigt wird. Diese Texte sind im HELP-Verzeichnis abgelegt.

~/ims/com/util/

Verzeichnis für einen Teil der in den Batchprogrammen aufgerufenen kompilierten Programme.

~/ims/demo/

Das Verzeichnis enthält eine Demoschaltung im NETED-Format, die alle Macrozellen zum Testen bereithält.

~/ims/doc/

Hier sind die Releasenotes und Installationsanweisungen niedergeschrieben.

~/ims/forest_1.2/

Neben den beiden COM-Verzeichnissen ist dieses TECHNOLOGIE-Verzeichnis das wichtigste. Es enthält alle Informationen über den IMS-GATE-FOREST.

-physikalische und logische Beschreibungen der Macrozellen
- Beschreibung der Master
- ...

~/ims/forest_1.2/array_1/

Muster für die Master. Enthält alle Files die notwendig sind Simulationen mit den Programmen des IMS durchzuführen.

dly.dat = Fanoutabhängige Verzögerungszeiten an den Ausgängen der Macrozellen
version = Versionsangabe der Daten
wlen.dat = Masterspezifische Kapazitätsdaten der Leitungen

von den Unterverzeichnissen ~/ims/forest_1.2/gf??g1 zeigen gleichnamige Links auf diese Files.

~/ims/forest_1.2/cells/

Hier liegen alle Bibliothekselemente als Symbole für die Schaltungseingabe mit NETED. Dazu gehören alle Properties und Timingfiles der Elemente. Die Beschreibung des Zeitverhaltens mit BLM, Quickparts und Timingfiles befindet sich hier.

~/ims/forest_1.2/gf??g1/ (gf??g1 steht für die Master gf4g1, gf9g1 und gf12g1)

Technologieverzeichnise für die Master gf??g1 der IMS GateForest_1.2-Familie.
Diese Verzeichnisse enthalten alle masterspezifischen Datenfiles für den DCKP und die abstrakte Beschreibung der Master (CADI.CHP) und Makrozellen (CADI.BLC) für die GateStation.

~/ims/forest_1.2/gf??g1/src?/
Diese Verzeichnisse enthalten die entsprechende TDF-Beschreibung der Master und der Macrozellen für den Layoutentwurf mit der GateStation.
Es gibt für jeden Master mehrere Unterverzeichnisse, da die Möglichkeit besteht die Macrozellen unterschiedlich plazieren und verdrahten zu lassen.

~/ims/forest_1.2/gf??g1/package/
Hier sind die Bondplan-Symbole der Master abgelegt. Information über die Vorplazierungen der Padzellen sind hier in Files enthalten. Dieses Verzeichnis dient dem Program IMS_PKG_DEF als Datengrundlage.

~/ims/forest_1.2/library/
Hier befinden sich alle Macrozellen und die Spannungsversorgungsnetze der verschiedenen Master als Chipgraphzellen.

~/ims/forest_1.2/log_ic/
In diesem Verzeichnis befindet sich eine GATE-Bibliothek für LOGIC sowohl im ascii-Format (.dgl), als auch als kompiliertes binäres File (.bgl).

~/ims/forest_1.2/macro_parts/
Das Verzeichnis enthält unter anderem das PROCESS DEFINTION FILE (sowohl in ASCII als auch in binärer Form). Die Microzellen, aus denen neue Macrozellen aufgebaut werden können, sind hier abgespeichert.

~/ims/ulm_com/
Dieses Verzeichnis enthält alle Programme die für den Baukasten notwendig sind. Erklärungen zu den Programmen befinden sich im folgenden Kapitel.
Wie beim Verzeichnis ~/ims/com/ gibt es hier auch ein Help-Verzeichnis und ein UTIL-Verzeichnis mit den verwendeten kompilierten Programmen.

3.1 Genauere Beschreibung des Verzeichnisses FOREST 1.2

Neben den Unterverzeichnissen ~/ims/forest_1.2/gf??g1 sind auch mehrere Datenfiles in dem Verzeichnis forest_1.2 untergebracht da sie nicht masterspezifisch sondern allgemeingültig für diese Technologie sind. Aus den Unterverzeichnissen der Master (GF??G1) sind gleichnamige Links auf diese Files gerichtet. Diese Links sind notwendig, da mehrere der Dienstprogramme des Baukastens die Programme eigentlich hier erwarten.

Datenfiles:

netlist.config: Dieses Konfigurationsfile wird von LOGIC_ENTRY (im Dienstprogramm ims_route) verwendet: In NETED verbundene Ports mit VCC- oder GROUND-Komponenten werden durch dieses File in der GateStation richtig angeschlossen. Externe Signale werden in die Design Nets Section übernommen.

scio_data: Datenfile für IMS_PKG_DEF

technology: Technologiespezifische Datengrundlage für ADD_DELAY

Unterverzeichnisse der Master (GF??G1):

Links auf gleichnamige Verzeichnisse oder Daten:

cadi.blc	"~/ims/forest_1.2/gf9g1/src1/cadi.blc"
cadi.chp	"~/ims/forest_1.2/gf9g1/src1/cadi.chp"
cells	"~/ims/forest_1.2/cells"
design_properties	"~/ims/forest_1.2/design_properties"
dly.dat	"/user/ims/forest_1.2/array_1/dly.dat"
scio_data	"~/ims/forest_1.2/scio_data"
technology	"~/ims/forest_1.2/technology"
technology_properties	"~/ims/forest_1.2/technology_properties"
wlen.dat	"~/ims/forest_1.2/array_1/wlen.dat"cadi.blc

Datenfiles:

dsn_chk.config: masterspezifisches Konfigurationsfile für das Dienstprogramm **ims_design_checker.**

Unterverzeichnisse:

package: Hier sind Symbolschaltbilder der Gehäuse und Files mit der Information über die Vorplatzierung von Padzellen für das Dienstprogramm **ims_pkg_def** abgelegt.

src?: Es enthält die abstrakte Beschreibung der Master und der Macrozellen in kompilierter und in lesbarer (ASCII) Form für die GateStation.
src1: Hier ist die Beschreibung, die vom IMS offiziell unterstützt ist.
src2: Hier ist der Platzbedarf der Macrozellen des Corebereichs um eine Sitebreite (12um) an der rechten Seite der Zelle verkleinert angegeben. Bei der automatischen Plazierung tritt jetzt eine Überlappung der Macrozellen in horizontaler Richtung auf. Dies dürfte zulässig sein, da sich dabei nur die Isolationstransistoren der benachbarten Zellen überlappen (mit dem IMS noch nicht abgesprochen).
src3: Hier sind von src1 ausgehend die Ports der Padzellen so verkleinert, daß die GateStation nur noch einen Pin für den Anschluß zur Verfügung hat.
Der Sinn der dahinter steckt ist der, daß durch diese Einschränkung auf nur einen Pin, die Anzahl der Fehlverdrahtungen an den Ports der Padzellen verringert wird.

4. Werkzeuge des IMS-Design-Kits

Unter dem Begriff Werkzeug versteht man einzelne Programme, die nur einen Teil der Arbeit übernehmen. Erst der Einsatz mehrerer Programme hintereinander bildet dann das fertige Produkt. In diesem Fall hier ist das Produkt zweigeteilt. Zum einen möchte man nach Eingabe der Schaltung mit NETED Simulationsergebnisse mit QUICKSIM erhalten und zum anderen das fertige Chiplayout erstellt haben. Siehe Bild 1-2.
Beide Aufgaben werden durch den Batch IMS_PACKAGE realisiert, der die notwendigen Dienstprogramme des Baukastens (Bild 4-1) in der notwendigen Reihenfolge und mit den entsprechenden Optionen aufruft.
Bei den Programmen handelt es sich um Batches deren Zweck es ist, die kompilierten Programme mit den richtigen Optionen aufzurufen, Aufruf und Systemfehler (Files nicht vorhanden oder falsches Verzeichniss) abzufangen, und die gewonnenen Daten der kompilierten Programmen richtig einzuordnen.
Die Programme stehen in dem Verzeichnis ~/ims/ulm_com. Die kompilierten Programme sind zum Teil im Unterverzeichnis ~/ims/ulm_com/util abgelegt.

Nachfolgend werden die einzelnen Programme des Baukastens beschrieben.

4.1 IMS PACKAGE

Schaltungsentwicklung aufgebaut auf den DCKP und der GATESTATION.

Syntax:

ims_package <design> <technology> <GA_base_array>
[-CONT | -NODISP | -BA] [-SMp] [-MIF] [-QSim] [-Help] [-NO_Exp]
[-NO_Check] [-NO_Delay] [-NO_SIml] [-NO_Pkg]
[-MIN | -TYP | -MAX] [-TEST]

Beschreibung der Parameter:

design: Verzeichnisname der logischen Schaltung. (Muß immer angegeben werden.

technology: Verzeichnisname der Technologie: Zur Zeit nur forest_1.2. (Beim ersten Aufruf von IMS_PACKAGE nach der Schaltungseingabe muß die Technologie immer eingegeben werden)

GA_base_array: IMS GateForest, auf dem die Schaltung realisiert werden soll. Zur Zeit stehen nur die Master GF4G1, GF9G1 und GF12G1 zur Verfügung. (Wie die Technologie muß der Master beim ersten Mal angegeben werden; auch zu Simulationszwecken, da die Ausnutzung des Masters durch ein Checkprogramm untersucht wird.)

Die Reihenfolge der folgenden Switches (Schalter) ist nicht fest, sonder frei wählbar.

-CONT, -NODISP, -BA: Switches für den Batch IMS_PKG_DEF.

-SMp: Veranlaßt die Ausführung von IMS_SMPLNET (Sample Netlist).

-MIF: Veranlaßt die Ausführung des Programms IMS_MIFNET (Mentor Intermediate Format Netlist)

-QSIM: Veranlasst die Ausfuehrung von IMS_DESIGN_CHECKER, IMS_SIMLOG, IMS_ADD_DELAY (mit FANOUT ONLY) UND QUICKSIM

-Help: Listet das File ~/ims/ulm_com/help/ims_package.hlp auf.

-NO_Exp: Die Ausführung von IMS_EXPAND_COMP und IMS_EXPAND_-DESIGN wird unterdrückt.

-NO_Check: Unterdrückt die Ausführung von IMS_DESIGN_CHECKER

-NO_Delay: Unterdrückt die Ausführung von IMS_ADD_DELAY

-NO_SIml: Unterdrückt die Ausführung von IMS_SIMLOG

-NO_Pkg: Unterdrückt die Ausführung von IMS_PKG_DEF

-MIN, -TYP, -MAX: Erlauben eine Auswahl der KRISE- und KFALL-Werte im IMS_ADD_DELAY. Defaultwert ist -TYP.

-TEST: Gibt den Inhalt des zu bearbeitenden Verzeichnisses nach jedem Schritt aus.

Voraussetzung:

Bevor der Batch gestartet werden kann muß eine Schaltung mit NETED erzeugt worden sein. Hierbei ist zu beachten, daß die Schaltung vor dem Abspeichern **gecheckt** wird. Dieser Check ist notwendig, da in der Schaltung noch Fehler (Falsche Versionen der Symbole oder Schaltungsfehler) vorliegen können. Außerdem wird die Schaltung beim Checken hierarchisch strukturiert und es werden den Bauteilen fortlaufende Handelnummern (/I$xx) zugeordnet. (Wichtig für EXPAND).

Beschreibung des Programms:

Es werden je nach Angabe von Optionen Programme aufgerufen oder weggelassen.
Die wichtigste Option ist der Switch -QSIM. Mit ihm entscheidet man sich, ob man eine Simulation (-QSIM) oder ein Layout durchführen will (Default) (Bild 4-1). Die Auswirkungen der weiteren Switches sind in der Tabelle 4-2 zusammengestellt. Erklärungen zu den einzelnen Programmen folgen.
Der Sinn für die Existenz dieses Programmes ist die einfache Benutzung der für die Simulation und Erstellung des Layouts notwendigen Programme ohne das Wissen über deren Syntax und deren Voraussetzungen (Konfigurationsfiles).

Aufruf des Programms:

Beim **ersten Aufruf** müssen die ersten drei Parameter (Design, Technologie und Master) immer angegeben werden, egal ob man simulieren oder ein Layout erstellen will.
Die Angabe des Masters bei der Simulation ist desshalb notwendig, da man mit dem **ims_design_check** die Ausnutzung des Chips überprüft. Hier erkennt man, ob die Schaltung auf dem ausgewählten Master noch Platz hat. Bei weiteren Aufrufen muß nur noch der Name des Designs angegeben werden, da die anderen Informationen in einem Parameterfile abgelegt sind.

1. Simulation: Simuliert wird mit der Option -qsim.

2. Layout: Fehlt die Option -qsim, so wird ein Layout erstellt.
Befindet sich noch keine Angabe über das Gehäuse in dem Parameterfile, so wird im Programm danach gefragt. Für jeden Master stehen zwei Gehäusegrößen zur Verfügung.
Will man einen nachträglichen **Gehäusewechsel** durchführen, so müssen beim Aufruf die Technologie und der Master angegeben werden.

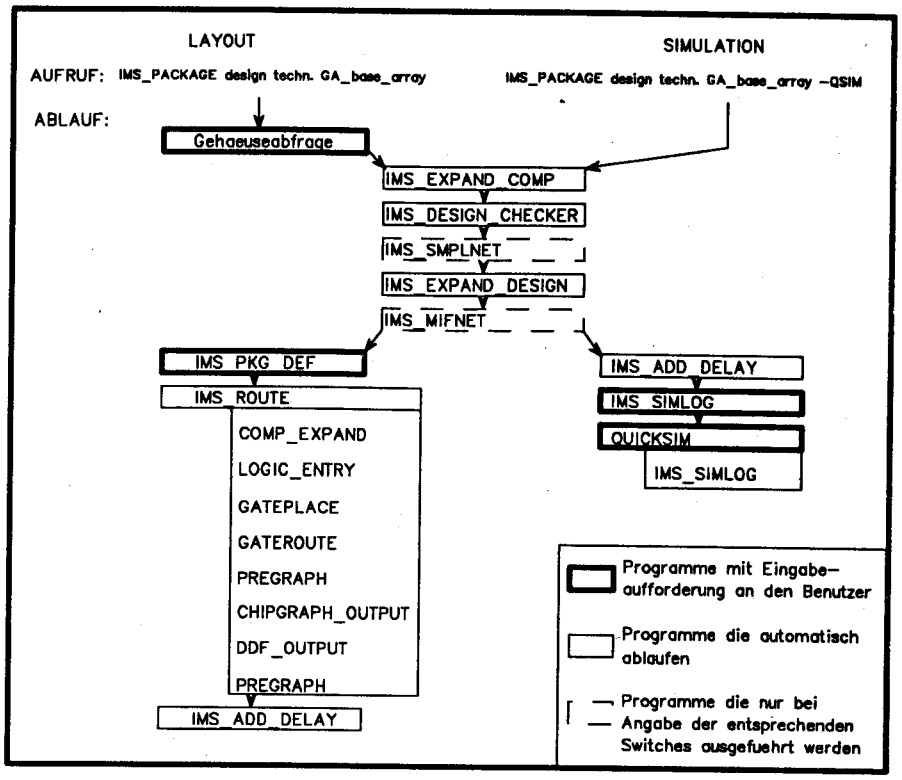


Bild 4-1 Struktur von IMS_PACKAGE

x = Programm wird ausgefuehrt o = Option hat Auswirkung auf Prog. – = Option hat keine Ausw. auf Prog.	ohne Optionen	-NO_Check	-NO_Delay	-NO_Pkg	-NO_Exp	-MIF	-SMp	-BA	-CONT -NODISP	-MIN -TYP -MAX	-QSIM	-NO_Sim
IMS_EXPAND_COMP	x	x	x	x		x	x		x	x	x	x	x
IMS_DESIGN_CHECKER	x		x	x	x	x	x		x	x	x	x	x
IMS_SMPLNET							x						
IMS_EXPAND_DESIGN	x	x	x	x		x	x		x	x	x	x	x
IMS_MIFNET						x							
IMS_PKG_DEF	x	x	x		x	x	x	o	o	x			
IMS_ROUTE	x	x	x	x	x	x	x		x	x			
IMS_ADD_DELAY	x	x		x	x	x	x		x	o	o	x	x
IMS_SIMLOG											x	x	
QUICKSIM											x	x	x

Bild 4-2 Optionen des IMS_PACKAGE

Verwendung des **ims_package** im Schaltungsentwicklungsblauf:

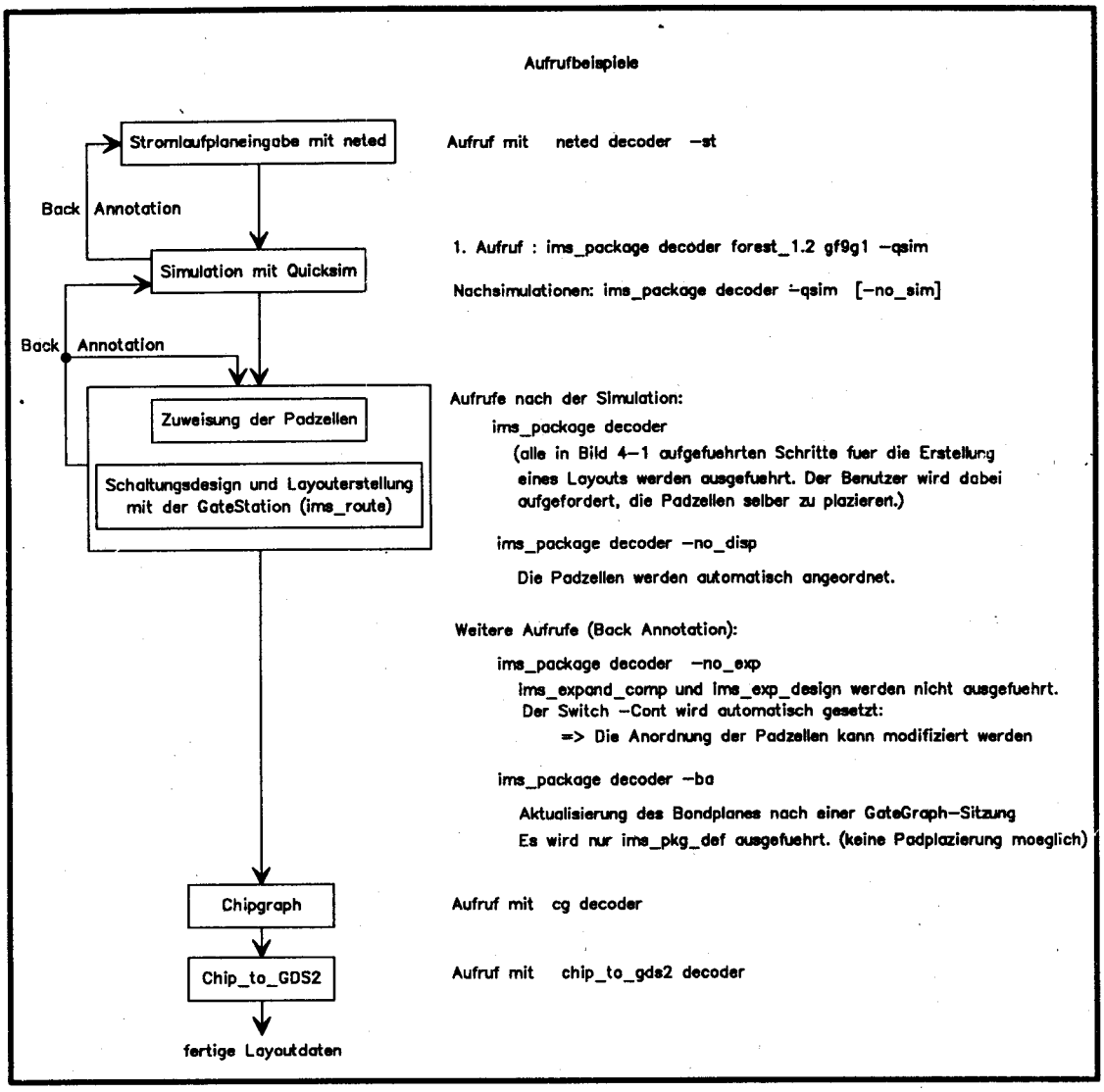


Bild 4-3 Designablauf

4.2 Erlaeuterungen zum EXPAND

Es gibt zwei IMS_EXPAND-Dienstprogramme, die vom **ims_package** aufgerufen werden (**ims_expand_comp** und **ims_expand_design**). Diese zwei Batches rufen wiederum das interaktive Programm EXPAND mit den notwendigen Befehlsfolgen auf.

Aufgabe von EXPAND

Mit EXPAND wird eine elektrische Schaltung, die mit Neted oder Symed erstellt wurde, für die nachfolgende Analyse oder Layouterzeugung vorbereitet.
---> Das EXPAND muß vor der Simulation oder der Erstellung des Layouts ablaufen.
EXPAND filtert die notwendigen Informationen über Designfunktionalität, Zeitverhalten und andere Charakteristiken aus dem netedfile.nrel oder Symedfile.srel heraus und stellt alles in ein einzelnes Designfile zusammen (name.erel_$X (Versionsnummer X zur Kontrolle)).
Der Defaultwert für das Designfile ist design.erel. Man kann diesen Namen jedoch frei wählen und erreicht somit, daß man mehrere Designfiles für einen Designtree erzeugen kann. Der Sinn der dahinter steckt ist der, daß man für die verschiedenen Arten von Simulationen jeweils ein File besitzt.

Aufruf:

expand <design> [Designfile_name]

Beschreibung der Parameter

design ist das Haupverzeichnis der hierarchisch aufgebauten Schaltung

Designfile_name Name des eingeebneten Files, in das die Informationen abgelegt werden. Der Defaultwert für Designfile_name ist design.

4.2.1 IMS_EXPAND_COMP

Erzeugung eines eingeebneten (flattened) Designfile comp.erel aus dem Netedfile design_name.nrel auf der Hierarchieebene der Macrozellen.

Syntax

ims_expand_comp <design> <technology>

Beschreibung der Parameter:

design ist das Haupverzeichnis der hierarchisch aufgebauten Schaltung

technology ist der Name der verwendeten Technologie. Hier: forest_1.2

Programmbeschreibung:

Die hierarchisch aufgebaute Netedschaltung enthält die gesamten Informationen zur Simaulation und Herstellung des Layouts in Form von Properties. Diese hierarchische Struktur muß vorher für die weiteren Programme aufbereitet werden. Diese Aufgabe übernimmt das im IMS_EXPAND_COMP aufgerufene interaktive Programm EXPAND, das die hierarchische Struktur bis auf die Höhe der Macrozellen (PHY_COMP-Prop.) einebnet und die Properties in ein "flattened" Designfile (COMP.EREL) zusammenstellt. Dieses Designfile ist die Grundlage für die weiteren Batches des IMS_PACKAGE zur Erstellung des Layouts.

Aufruf von EXPAND in IMS_EXPAND_COMP:

```
/idea/com/expand ^1 comp <<!
# Hier werden alle Properties aufgelistet, die in dem Design.file (comp.erel) enthalten sind. Der
# Property-Befehl wird dabei für jeden Ownertyp (INSTance, PIN, NET aufgerufen.
PROPERTY PIN  PINTYPE LOAD SIZE KRISE KFALL PHY_PIN
PROPERTY PIN  FANOUT FANIN PINCLASS SIGTYPE
PROPERTY NET  PINTYPE PRIO WTYPE VTYPE MAX_DELAY
PROPERTY NET  DC SIGTYPE
PROPERTY INST COMP PLACE SEED SC_IO GATECOUNT
PROPERTY INST COMPTYPE COMPSIZE COMPAREA POWERDISS
#
PARAMETER STD_CELL ^std_cell
#
# Mit PRIMITIVE wird EXPAND mitgeteilt, bis zu welcher Hierarchiestufe expandiert werden
 # soll (TOP->DOWN).
PRIMITIVE PHY_COMP
# Mit Run wird das Design_file expandiert, dabei werden die oben getroffenen Einstellungen
 # verwendet.
RUN -L
# Verlassen des Programms EXPAND mit BYE
BYE
# !!Wichtig: Hinter ! darf kein weiteres Zeichen stehen.
!
```

Voraussetzung:

Als Datengrundlage muß eine mit Neted gezeichnete Schaltung vorliegen.
Diese Schaltung muß in NETED gecheckt worden sein.

4.2.2 IMS_EXPAND_DESIGN

Erzeugung eines geebneten "flattened" Designfiles design.erel aus dem Netedfile design_name.nrel auf der Hierarchiestufe der COMP-Property.

Syntax

ims_expand_design <design> <technology>

Beschreibung der Parameter

design ist das Haupverzeichnis der hierarchisch aufgebauten Schaltung

technology ist der Name der verwendeten Technologie. Hier: forest_1.2

Programmbeschreibung

Im Gegensatz zum Batch IMS_EXPAND_COMP wird hier bis auf die COMP-Propertyebene expandiert. Das Designfile dient in den folgenden Batches als Grundlage für die Simulation mit Quicksim.
Durch die Eigenart, daß die Beschreibung der Logik und das Zeitverhalten einer Macrozelle nicht nur durch die Properties an den Pins erfolgt, sondern auch durch die Verwendung von Bauteilen der Gen-lib, Behavior Language Models und Quickparts (siehe Ende diese Kapitels) erweitert ist, wird nach dem Aufruf von EXPAND noch ein weiteres Programm (**extract**) aufgerufen.
Dieses Programm vervollständigt das desin_file (design.erel).

Aufruf von EXPAND und EXTRACT in IMS_EXPAND_DESIGN

```
/idea/com/expand ^1 design <<!
# Wie beim IMS_EXPAND_COMP erfolgt hier eine Auflistung aller Properties, deren Werte in
das Design_file übernommen werden sollen.
PROPERTY  INST  MODEL
PROPERTY  INST  MODELFILE MODELCODE -NOUPcase
PROPERTY  PIN   PINTYPE RISE FALL DRIVE STABLE
PROPERTY  NET   PINTYPE INIT DTIME DECAY WIREDEL
#
#  Set all parameters used in the designs...
PARAMETER STD_CELL ^std_cell
#
#  Allow for hierarchical expansions...
#INSERT    DESIGN
#
# Uncomment if design files are used for modelling...
# INSERT   SIM
#
# Set the expansion level and run...
PRIMITIVE MODEL
PRIMITIVE MODELCODE
RUN -L
```

```
BYE
!
/idea/com/extract ^1 design -i qp,hml
```

Voraussetzung:

Wie beim IMS_EXPAND_COMP muß eine mit Neted erstellte Schaltung vorliegen. Diese Schaltung muß in Neted geCHECKt worden sein.

Worterklärungen:

Behavior**L**anguage**M**odel: Ein BLM ist ein Pascalmodul das das logische und zeitliche Verhalten eines Bauteils bei der Simulation beschreibt. Es wird verwendet, wenn man das logische/zeitliche Verhalten von Macrozellen mit Kombinationen einfacher Gatter nicht mehr realisieren kann. (z.B. das Tristateverhalten von Padbuffern.)

Quickparts: Von einem Quickpart spricht man, wenn das logische Verhalten einer Zelle durch Kombination von einfachen Gattern, das zeitliche Verhalten jedoch durch ein Timing-File beschrieben wird.

Hybridquickpart: Unter diesem Begriff versteht man eine Schaltung, deren logisches Verhalten durch einen rückgekoppelten Quickpart beschrieben wird. Mehrere der Macrozellen sind als Hybridquickparts beschrieben. (~/ims/forest_1.2/cells)

4.3 IMS_DESIGN_CHECKER

Ausführung einer Designuntersuchung des expandierten Design_files (comp.erel) unter Berücksichtigung der Angaben in einem Konfigurationfile.

Syntax

ims_design_checker <design> <technology> <master>

Beschreibung der Parameter

design ist das Haupverzeichnis der hierarchisch aufgebauten Schaltung

technologie ist der Name der verwendeten Technologie. Hier: forest_1.2

master ist der Name des verwendeten Mastertyps bei der Erstellung des Layouts

Programmbeschreibung

Der IMS_DESIGN_CHECKER untersucht das Design_file comp.erel nach Verstößen gegen die logischen Designregeln.
Es wird untersucht:

- ob Fan-out und Fan-in Verstöße vorkommen (ob an einem Ausgang eine zu große Last angehängt ist)
- ob unverbundene Pins vorkommen
- wie hoch die Ausnutzung des Masters ist.
- ob ein Zusammenschluß externer und interner Netze vorliegt (Unter internen Netzen versteht man die Leitungen auf dem Chip. Externe Netze sind Leitungen zum Chip über die externen Anschlüsse der Padzellen.)

Die Arbeitsweise des Designcheckers wird durch drei Konfigurationsdateien (für jeden Master eine) bestimmt.
Als Ergebnis liefert der Designchecker das File **dsn_chk.log** im Designverzeichnis. Treten Fehlermeldungen auf, so wird das File im VIEW-Modus am Bildschirm dargestellt.

Beispiel eines Konfigurationsfiles für den GFG-9:

```
#-------------------------------------------
# Design_Checker configuration file for MGC
#-------------------------------------------
#
# Release 1.0
# Date   26-03-1992
# M.RATHENOW
#
# Define the technology...
TECHNOLOGY = FOREST_1.2
#
# Define the properties used with this library...
DEFINE_PROPERTIES
```

```
#
# Define the properties used on INSTANCES...
  COMP_PROP      = COMP
  COMP_SIZE_PROP = COMPSIZE ,  DEFAULT_VALUE = 1
  INST_PROP      = INST
  CELL_TYPE_PROP = COMPTYPE
  POWER_PROP     = POWERDISS
#
# Define the properties used on PINS...
  SIG_PROP       = SIGTYPE
  FANOUT_PROP    = FANOUT, DEFAULT_VALUE = 1
  LOAD_PROP      = LOAD,   DEFAULT_VALUE = 1
  PIN_TYPE_PROP  = PINTYPE
  PIN_CLASS_PROP = PINCLASS
  FANIN_PROP     = FANIN,  DEFAULT_VALUE = 1
  MAX_FANIN      = 24
#
#  MAX_FANIN ist die maximale Summe von Eingangslasten,
#  die von einer Bibliothekszelle (IN01D4) noch getrieben
#  werden kann.
#
# Define the properties used on NETS...
#   DC_SIGNAL_PROP = DC,     DEFAULT_VALUE = AC
#
# Describe the INSTANCE checking...
CHECK=INSTANCE
  COMPTYPE = INT, MAX_USED = 8128
  COMPTYPE = EXT, MAX_USED =  44
  COMPTYPE = INT AND EXT, MAX_USED = 8172

#  Die 44 ist auf das Gehäuse mit den 44 Anschluessen
#  zurueckzufuehren.
#  Fuer das kleinere Gehaeuse ist die Anzahl der Pads auf 36 beschraenkt.
#  Es ist daher Vorsicht geboten, da dieser Unterschied nicht abgefangen wird.

# Describe the NET checking...
CHECK=NET
  IF  (EXTERNAL_NET)  THEN     (PINCLASS=PAD)
  IF  (PINCLASS=PAD)  THEN_NOT (PINCLASS=INT)
  IF  (SIGTYPE=CLK)   THEN_NOT (DC)
#
  UNCONNECTED_INPUTS = ERROR
  UNCONNECTED_OUTPUTS = DONT_CARE
#
```

Beispiel für ein vom Design Checker erzeugten Report:

```
***** DESIGN_CHECKER REPORT *****
      ---------------------

DESIGN : PAD_12_TEST
```

TECHNOLOGY : FOREST_1.2

DATE : 25JUN92 18:40:27-MES

NUMBER	COMPONENT_NAME	OCCURRENCES
1	IOUT1	10
2	IBIN1	10
3	AN02D1	10

Total of 30 instances, 3 macro types.

COMPONENT TYPE	COUNT
INT AND EXT	70 out of 4644 (1.51 %).
INT	50 out of 4608 (1.09 %).
EXT	20 out of 36 (55.56 %).
TOTAL	70

Die Zahlen in der Spalte COUNT beziehen sich auf Sites.

POWER DISSIPATION.
AVERAGE POWER : 0.00 mW.

DESIGN_CHECKER completed successfully.

4.4 Netzlisten

Für die Simulation und die Erstellung des Layouts mit diesem Baukasten haben die Netzlisten keine Bedeutung. Sie bilden nur eine Schnittstelle zu anderen Softwares, wenn z.B. das Layout außerhalb des Hauses (beim IMS) erstellt wird.

4.4.1 EDIF-Netzliste

Mit dem Programm **edifnet** wird eine EDIF-Netzliste erstellt. Das Programm ersetzt das fehlerbehaftete Programm **ims_edif_net**.
Hinweis: EDIFNET wird nicht automatisch beim Durchlauf mit **ims_package** ausgeführt. Es kann nachträglich aufgerufen werden.

Syntax

edifnet <design>

Programmbeschreibung:

Es wird das File <design_name>/design.erel vom Batch IMS_EXPAND_DESIGN gelesen und verarbeitet. Als Ergebnis erhält man die EDIF-Netzliste in dem File <design_name>/edif.netlist. Diese Netzliste findet dann ihre Verwendung, wenn die Erstellung des Layouts dem IMS übertragen wird. Die notwendigen Informationen über die Schaltung sind in dieser Netzliste vorhanden.

4.4.2 IMS_MIFNET

Erzeugung einer Netzliste im ASCII-Format aus dem Designfile design.erel.

Syntax

ims_mifnet <design> <technology>

Beschreibung der Parameter

design ist das Haupverzeichnis der hierarchisch aufgebauten Schaltung

technology ist der Name der verwendeten Technologie. Hier: forest_1.2

Programmbeschreibung

IMS_MIFNET (Mentor Intermediate Format Netlister) erzeugt eine Netzliste <design_name>/mif.netlist, die Informationen über Properties, Instances und Verbindungen der mit Neted erstellten Schaltung enthält. Als Datengrundlage dient das Designfile design.erel.. Verwendet wird diese Netzliste zur Datenübertragung auf andere nicht Mentor spezifische Software.

Ausschnitt aus der Netzliste:

The Pascal source for MIFLIST is in /idea/sys/source/miflist.pas.

```
{NETLISTED: 05/08/1992 10:18}
%CIRCUIT /user/rathenow/test_ims_package/dec
%INSTANCE
/user/ims/forest_1.2/cells/idin1 sheet1(-255.0,80.0) [/I$61]
    MODELCODE:"qpfile.g5"
    MODEL:"$G5",
  PAD (Z3_OUT)
    PINTYPE:"IN",
  Z (Z3)
    PINTYPE:"OUT"
    FALL:"0.0"
    RISE:"0.0";

    .....

%ENDNET
%ENDCIRCUIT
```

Voraussetzung:

Eine mit Neted erstellte Schaltung muß mit IMS_EXPAND_COMP expandiert worden sein.

4.4.3 IMS_SMPLNET

Erzeugung einer Netzliste im ASCII-Format aus dem Designfile COMP.EREL

Syntax

ims_smplnet <design> <technology>

Beschreibung der Parameter

design ist das Haupverzeichnis der hierarchisch aufgebauten Schaltung

technology ist der Name der verwendeten Technologie. Hier: forest_1.2

Programmbeschreibung

Aus den Informationen des Designfile <design_name>/comp.erel wird eine Netzliste **sample.netlist** erstellt. Sie enthält weniger Informationen als die Netzliste des IMS_MIFNET.

Ausschnitt aus der Netzliste:

```
DESIGN PINS
INPUTS:

OUTPUTS:
```

```
SGELB_OUT SGRUEN_OUT SROT_OUT WGRUEN_OUT WROT_OUT
COMPONENTS
IDIN1 = (PAD:Z3_OUT ,)
IDIN1 = (PAD:Z2_OUT ,)
IDIN1 = (PAD:Z1_OUT ,)  ...
```

Voraussetzung:

Vor dem Batch IMS_SMPLNET muß eine mit Neted erstellte Schaltung mit dem Batch IMS_EXPAND_COMP expandiert worden sein.

4.5 IMS_PKG_DEF

IMS_PACKAGE sorgt für die richtige Plazierung der Padzellen und füllt die noch freien Padplätze mit Dummyzellen auf.

Allgemeine Syntax

ims_pkg_def <Design> <Technology> <Master> <Gehäuse>
[-BA | -CONT | -NODISP]

Beschreibung der Parameter:

design ist das Haupverzeichnis der hierarchisch aufgebauten Schaltung

technology ist der Name der verwendeten Technologie. Hier: forest_1.2

master ist der Name des verwendeten Mastertyps bei der Erstellung des Layouts

Gehäuse ist der Name des Bonplansymbols im Unterverzeichnis .../package des verwendeten Masters

Die beiden Switches -BA und -CONT können nur bei einem zweiten Aufruf von IMS_PKG_DEF gewählt werden, da sie sich auf bestehende Daten stützen.

-BA Mit diesem Switch ist es möglich die endgültige Plazierung der Padzellen eines vorherigen Durchlaufes des IMS_PACKAGE (Layout) anzuschauen. Diese Plazierung wird dabei im Bondplan-symbol <design_name>/package/sheet1.pic aktualisiert.
Veränderungen des Bondplan sind nicht möglich, sie werden beim Verlassen von **ims_pkg_def** ignoriert.

-CONT Mit diesem Switch kann man die Plazierung des vorhergehenden Laufs modifizieren. Mit diesem Switch verkürzt sich auch der Aufruf des Batches:
ims_pkg_def design_name -cont.

-NODISP Es kann keine Plazierung der Padzellen mit der Hand durchgeführt werden. Lediglich die Padzellen der Datei "package_name.nopads" werden vorplaziert. Erklärung siehe unten.

Programmbeschreibung

Mit dem Batch **ims_pkg_def** kann der Benutzer die Padzellen vorplazieren. Dies kann durch einen Einzelaufruf des Programms oder durch einen Aufruf aus dem **ims_package** erfolgen. Dabei hat die Angabe der Switches eine starke Bedeutung:
Beim ersten Durchlauf des Batches **ims_pkg_def** mit **ims_package** kann man den Switch -NODISP verwenden. Das Programm plaziert dabei alle Padzellen ohne Eingreifen des Benutzers.
Möchte der Benutzer die Padzellenanordnung selber bestimmen, so wird beim Aufruf von **ims_pkg_def** und **ims_package** keiner der Switches verwendet.
Bei weiteren Durchläufen von **ims_pkg_def** aus **ims_package** heraus wird automatisch der Switch -CONT gesetzt. Dem Benutzer wird dabei ermöglicht, weitere Veränderungen an der Padplazierung (Bondplan) vorzunehmen.
Werden mit dem Programm Gategraph (Interaktives Plazier- und Routingprogramm der Gatestation) Veränderungen am Bondplan vorgenommen, so kann man mit dem Switch -BA diese Veränderung bei einem erneuten Durchlauf des **ims_pkg_def** im Bondplan-symbol aktualisieren.

Je nach Wahl der Master werden dem Benutzer zwei Gehäusegrößen der Typen PLCC und CLCC angeboten (Tabelle 4-1). Beschrieben und somit verwendbar sind die Gehäuse der Master GFG-12, GFG-9 und GFG-4.
Die Symbole der Gehäuse sind in dem Unterverzeichnis ~/ims/forest_1.2/gf??g1/package des jeweiligen Masters abgelegt (Bild 4-4). Im gleichen Unterverzeichnis stehen auch die Files <gehaeuse_name>.nopads, in denen die Padzellen für die automatische Vorplazierung durch ims_pkg_def aufgelistet sind. Die Powerzelle wird bei der neuen Technologie nicht mehr vorplaziert.

Vorgehensweise bei der Padplazierung

Nach dem Aufruf des Programms hat man eine **neted** ähnliche Benutzeroberfläche vor sich. Zur Plazierung der Padzellen wählt man der Reihe nach mit der Maus die im Menue-Fenster aufgelisteten logischen Pins aus und ordnet sie den gewünschten physikalischen Pins des Masters im Edit-Fenster zu. Die Zuordnung erfolgt jedoch nur, wenn physikalische Pins des Masters mit dem Wert IONET angeklickt werden. Gesperrt für die Plazierung sind schon plazierte Pins, Anschlüsse für die Stromversorgung (VSS/VDD) und offene Anschlüsse (NC). Bei Wahl dieser Anschlüsse muß der logische Pin im Menue-Fenster nochmal angeklickt werden
Bei der Auswahl von **$_UNDO** wird die Vorplazierung rückwärts rückgängig gemacht.
Mit **$ADD_VDD_RING** und **$ADD_VSS_RING** können zusätzliche Verorgungspads für die Ausgangspads plaziert werden.
Sind die Eingaben richtig, so **muß** das Programm mit **$_EXIT** verlassen werden.
Mit **$_QUIT** verläßt man das Programm ohne Veränderung der Padplazierung.
Die Padzellen, die nicht vorplaziert wurden werden automatisch plaziert.

Nach dem Verlassen der Benutzeroberfläche werden die freien Padplätze mit Dummyzellen aufgefüllt.

Ergebnis des Programms

Das Programm liefert mehrere Files, die alle im Verzeichnis des Designs abgelegt werden:
Im **place.ddf** wird zu jeder logischen Padzelle der Ort auf dem Master angegeben (siehe Beispiel). Vorplazierte Powerpadzellen der Powerzelle werden hier nicht nicht angegeben.Die nach ims_pkg_def im Batch IMS_PACKAGE aufgerufene GateStation (**ims_route**) verwendet dieses Datenfile als weitere Plazierungsgrundlage.
Dem **pinout.report** entnimmt man den Padtyp und den logischen Signalnamen jeder Padzelle. Nicht im Schaltplan enthaltene Dummypads sind hier als "unassigned" gekennzeichnet.

Beispiel einer **place.ddf** Datei:

```
DDF
IDENTIFY
(* PACKAGE ID: CLCC/PLCC-44 *)          <---- Gehäuseangabe
(* ARRAY ID: GF12G1 *)                  <---- Masterangabe
BEGIN LAYOUT
BEGIN PLACE
# In den folgenden Zeilen sind alle vorplazierten Padzellen durch ihre Handlenummer
/I$xx angegeben Die Zuweisung der Padzelle zum Plazierungsort erfolgt mit SITE.
Die Option STATUS = FIXED gibt an, daß diese Zelle nicht verschoben werden
kann.
MAC BLOCK='/I$33' SITE=S001 STATUS=FIXED
MAC BLOCK='/I$34' SITE=S002 STATUS=FIXED
...
# Die weiteren Zeilen enthalten Angaben über Dummyzellen, die nicht im Schaltplan
vorkommen und vom System hinzugefügt wurden. Sie bekommen die Handelnummer
/NP$xx. Der Zellentyp muß hier angegeben werden, da hinter der Handelnummer
kein Schaltungsbauteil steht. STATUS=ASG gibt an, daß die Zellen während des
Layouts erzeugt wurden.
MAC BLOCK='/NP$01' TYPE=IDUM1 SITE=N005 STATUS=ASG
MAC BLOCK='/NP$02' TYPE=IDUM1 SITE=W007 STATUS=ASG
...
END PLACE
END LAYOUT
END DDF
```

Beispiel einer zugehörigen Pinbelegung in pinout.report:

DESIGN: PAD_12_TEST
PACKAGE: CLCC/PLCC-44
ARRAY: GF12G1

SIGNAL	MACRO	SITE	PACKAGE PIN	
IONET	-unassigned-	N005	1	# Ergänzung mit einer Dummyzelle
NC	-unassigned-		25	# Pin 25 nicht angeschlossen
VSS	-unassigned-	W005	10	# Versorgungsspannungen
VDD	-unassigned-	W010	14	# der Powerzelle
EIN8	IBIN1 S001		18	# Signalpin aus dem Schaltplan mit Typenangabe

Voraussetzungen:

- Die Schaltung muß mit **ims_expand_comp** expandiert worden sein.
- Im Unterverzeichnis des jeweiligen Masters ~/ims/forest_1.2/gf??g1/package sind die Symbolschaltbilder der Master und die Datenfiles <gehäuse_name>.nopads vorhanden.
- Auf das Datenfile SCIO_DATA im Verzeichnis ~/ims/forest_1.2 zeigen gleichnamige Links, die in den Unterverzeichnissen der Master stehen.

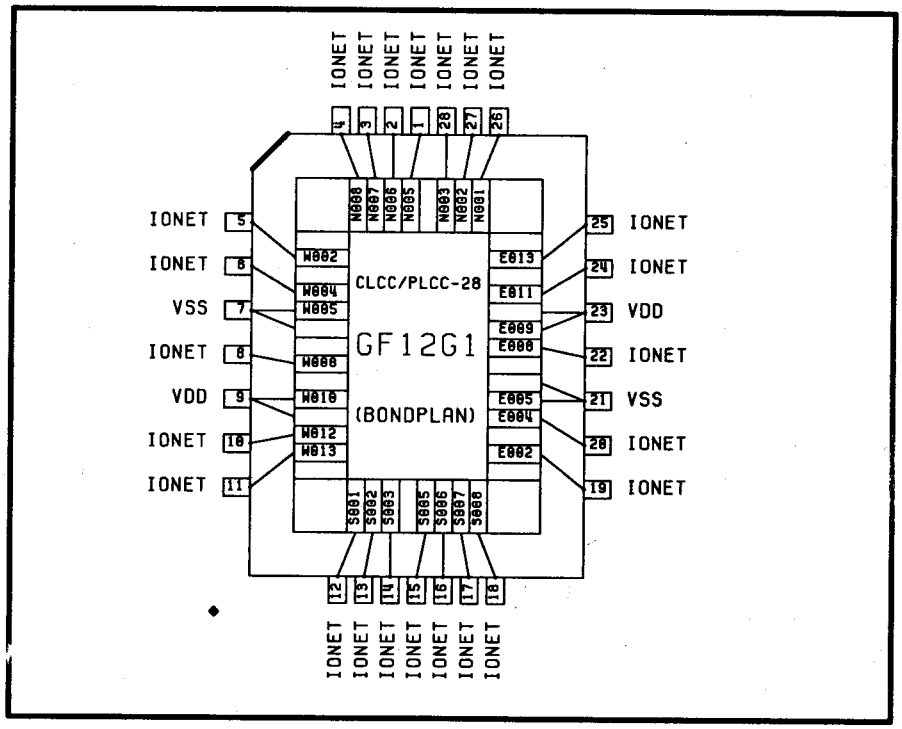


Bild 4-4 Bondplan fuer den GFG-12

Master	Pads	Gehäuse 1)		Gehäusebezeichnung	Signalpins (IONET)	POWERPINS (VSS/VDD)
GFG-16	32	28-Pin	CLCC	PB-C87651	24	4
			PLCC	02-28-2025 A/C/E		
		44-Pin	CLCC	PB-C85254-A	36	8
			PLCC	02-44-2200 C		
GFG-12	44	28-Pin	CLCC	PB-13046-A	24	4
			PLCC	02-28-2665 C/D/E		
		44-Pin	CLCC	PB-C85138	36	8
			PLCC	02-44-2230 B/C		
GFG-9	56	44-Pin	CLCC	PB-C85138	36	8
			PLCC	02-44-2230 B/C		
		68-Pin	CLCC	PB-C88141	44	9
			PLCC	02-68-3000 B/C		
GFG-4	96	68-Pin	CLCC	PB-C87671-B	59	9
			PLCC	02-68-3025 B/C		
		84-Pin	CLCC	PB-C87729	72	12
			PLCC	02-84-3010 B/C/T		
GFG-2	144	84-Pin	CLCC	PB-C86765	72	12
			PLCC	02-84-4210 B/C/L		
		144-Pin	PGA	CA-P14404-PU 2)	112	28

1) Alle Keramikgehäuse haben die Orientierung 'cavity up'
Alle Plastikgehäuse haben die Orientierung 'face up'
Alle CLCC-Gehäuse sind vom JEDEC Typ C

2) Gehäusebezeichnungen der Keramikgehäuse nach Unterlagen der Fa. KYOCERA
Ausnahme: 144-Pin PGA-GehÄuse nach Unterlagen der Fa. SHINKO ELECTRONIC INDUSTRIES

TABLE 4-1 Gehaeusetypen

4.6 IMS_ROUTE

Plazieren und Verdrahten einer logischen Schaltung, welche auf einem GateForest des IMS realisiert werden soll.
Durch Optionen kann die Ausführung mancher Programme unterdrückt bzw. veranlaßt werden.

Syntax

ims_route <design> <technology> <master>
[-B] [-C] [-NO_EXP] [-NO_LE][-P_ONLY] [-R_ONLY] [CO-ONLY]

Beschreibung der Parameter

design ist das Haupverzeichnis der hierarchisch aufgebauten Schaltung

technology ist der Name der verwendeten Technologie. Hier: forest_1.2

master ist der Name des verwendeten Mastertyps bei der Erstellung des Layouts

-B ist ein **GATEPLACE**-switch. Mehr dazu im Handbuch der GateStation

-C entspricht dem (-C)-Switch von **LOGIC_ENTRY**; er schaltet die Verwendung der Konfigurationsdatei **config.netlist** ein. Das **ims_package** verwendet diesen Switch beim Aufruf von **ims_route**.

-NO_EXP unterdrückt die Ausführung von **COMP_EXPAND**. Das **ims_package** verwendet diesen Switch, da die Schaltung schon mit **ims_expand_comp** expandiert wurde.

-NO_LE unterdrückt die Ausführung von **COMP_EXPAND** und **LOGIC_ENTRY**

-G_ONLY die Schaltung wird nur plaziert (**GATEPLACE**). Es findet kein Routen statt.

-R_ONLY die Schaltung wird nur geroutet. Dieser Switch kann verwendet werden, wenn eine Plazierung vorher stattgefunden hat.

-CO_ONLY Es werden nur Chipgraph-Zellen mit **CHIPGRAPH_OUTPUT** und dem interaktiven Designerwerkzeug **CHIPGRAPH** erzeugt

Programmbeschreibung

Dieses Dienstprogramm führt die notwendigen Programme der GateStation aus, die für die Plazierung und das Routen der Schaltung notwendig sind (comp_expand, **logic_entry, gateplace und gateroute**). Dabei greift **logic_entry** auf die Informationen des **place.ddf,** des **comp.erel** und der beiden Technologiedaten **cadi.chp** und **cadi.blc** zurück.
Weitere Schritte sind die Aufarbeitung dieser Daten für das Graphikprogramm der Gatestation **gategraph** mit **pregraph** und für Chipgraph mit **chipgraph_out-**

put. Mit **ddf_output** wird das DDF-file **netparm.ddf** erzeugt. Es dient dem folgenden Dienstprogramm **ims_add_delay** als Information über die Länge der Signalleitungen.
Auf die Programme der Gatestation gehe ich nicht weiter ein, da über sie in mehreren Texten schon geschrieben wurde.

Ergebnisse des Programms

Das Programm **ims_route** liefert viele Daten, die in Files oder Unterverzeichnissen abgelegt werden. Die wichtigsten bei der Verwendung des Baukastens:

netparm.ddf : Dieses DDF-File enthält Angaben über die Kapazitäten der einzelnen Netze in pF, über die Netzlängen in Metall1 und Metall2 und über die Anzahl der Vias in den einzelnen Netzen.

design_cell, macro_cell, routing_cell, power_cell, overflow_cell:
In diesen 5 Unterverzeichnissen sind die Layoutdaten für Chipgraph enthalten, wobei die **design_cell** aus den anderen Zellen außer der overflow_cell aufgebaut ist.
Hier unterscheidet sich auch die neue Technologie von der alten 2um Technologie: Die Powerzelle des Masters wird nicht mehr mit den anderen Macrozellen in die **macro_cell** eingebunden, sondern steht durch die Verwendung der **power_cell** auf der gleichen Hierarchiestufe in der design_cell wie die macro_cell selber. In der **routing_cell** sind die Verbindungen zwischen den Zellen als Pfade abgespeichert.

Designtree der **design_cell:**

```
design_cell
    macro_cell
        alle Macrozellen (Core- und Padzellen der Schaltung)
    power_cell
        powerzelle des Masters aus dem Verzeichnis
        ~/ims/forest_1.2/library
    routing_cell
```

4.7 IMS ADD DELAY

Es wird das Programm **add_delay** zur Berechnung der Verzögerungszeiten unter Berücksichtigung der Schaltungseinflüsse (Fanout und Netzkapazitäten) richtig aufgerufen.

Syntax

ims_add_delay <design> <technology> <master>
[-FO | -LO] [-MIN | -TYP | -MAX] [-BA]

Beschreibung der Parameter

design ist das Haupverzeichnis der hierarchisch aufgebauten Schaltung

technology ist der Name der verwendeten Technologie. Hier: forest_1.2

master ist der Name des verwendeten Mastertyps bei der Erstellung des Layouts

-FO Bei der Berechnung der Verzögerungszeiten wird nur der Einfluß des Fanout (Treibfähigkeit des Ausgangs) berücksichtigt. Die Leitungskapazitäten werden auf Null gesetzt. Dieser Switch wird vom **ims_package** automatisch gesetzt, wenn vor dem Aufruf von **ims_add_delay** noch kein Layout durchgeführt wurde.

-LO Es werden nur die Verzögerungen, die sich durch die Leitungskapazitäten ergeben berücksichtigt.

Fehlen beide Switches, so werden beide Einflüsse berücksichtigt.

-MIN, -TYP, -MAX Diese drei Switches sind von mir hinzugefügt worden, da bei der neuen Technologie für die beiden Properties **KRISE** und **KFALL** drei Werte (min, typisch, max) angegeben sind. Der Defaultwert ist -typ.

-BA Dieser Switch muß bei einem einzelnen Aufruf von **ims_add_delay** gesetzt werden, wenn noch kein Layout vorhanden ist.

Weitere Parameter sind im Mentorhandbuch **DesignKitCreationGuide** aufgeführt.

Programmbeschreibung

Ohne dem Programm **add_delay** würde eine Simulation mit den Verzögerungszeiten des unbelasteten Bauteils ausgeführt werden. Die tatsächliche Verzögerungszeit ist aber von der Schaltung abhängig, ihre Berechnung oder Schätzung geschieht mit **add_delay.**
Bei der Berechnung der zusätzlichen Verzögerungszeit spielen hauptsächlich zwei Faktoren eine Rolle: Der Fanout (kapazitive Belastung des Ausgangs durch nachfolgende Gatter) und die Kapazität der Verbindungsleitungen (Bild 4-5).
Diese beiden Faktoren können getrennt mit den oben genannten Switches (-FO |-LO) je nach Belieben berechnet werden.
Damit das Programm die von den Leitungskapazitäten verursachten Verzögerungszeiten ermittelt kann, müssen diese Leitungen bzw. Kapazitäten bekannt sein. Beim Aufruf von **add_delay** vor der Layouterzeugung verwendet das Programm Schätzwerte für diese Größen. (Sie sind in dem Konfigurationsfile enthalten.) Beim Aufruf nach der Layouterzeugung werden die Werte aus der Datei **netparm.ddf** (wird erzeugt durch ddf_output aus ims_route heraus) entnommen. Ist diese Datei nicht vorhanden, so muß bei einem einzelnen Aufruf von **ims_add_delay** der Switch **-ba** gesetzt werden.
Der Einfuß von Temperatur- und Versorgungsspannungsschwankungen auf die Verzögerungszeiten ist von der Technologie abhängig. Das Programm **add_delay** greift dafür auf die Angaben des Konfigurationsfile **~/ims/forest_1.2/technology** zurück.

Konfigurationsfile:

```
#-----------------------------------------
# Add_Delay Technology specification file.
#-----------------------------------------
# Release 2.0
# Date   22-03-1991
# A. Bamba
# 10.05.92 - M. Rathenow: An den Forest_1.2 angepasst
##
# Technology Type (default is CMOS Gate Array)...
STANDARD_CELL=FALSE;
# BIPOLAR=FALSE;
#
# Technology constants and defaults...
#  PFS_PER_LOAD ist die Standardlast. Sie entspricht
#  der Kapazitaet eines P/N-Gatepaares. (= 64 fF)
#
PFS_PER_LOAD=0.064;
KRISE_DEFAULT=1.0;
KFALL_DEFAULT=1.0;
#
# Temperature equation based on units of KELVIN...
# F(t) = 0.0000047(t**2) + 0.002565(t) - 0.18494
# Diese Temperaturkoeffizienten werden von MENTOR
# vorgeschlagen und sind nicht an den IMS-Prozess
# angepasst.
#
TEMPERATURE_COEF=-0.18494 0;
TEMPERATURE_COEF=0.002565 1;
TEMPERATURE_COEF=0.0000047 2;
#
# Define the valid temperature range (in CENTIGRADE)...
# Die folgenden Temperaturwerten sind mit IMS abgesprochen
#
TEMPERATURE_MAX=85.0;
TEMPERATURE_MIN=0.0;
#
# Voltage equation based on units of (1/VOLTS)...
# F(v) = 22.8(1/v**2) - 3.557(1/v) + 0.8202
# Diese Spannungskoeffizienten werden von MENTOR
# vorgeschlagen und sind nicht an den IMS-Prozess
# angepasst.
#
VOLTAGE_COEF=0.8202 0;
VOLTAGE_COEF=-3.557 1;
VOLTAGE_COEF=22.8 2;
#
# Define the valid voltage range (in VOLTS)...
# Die folgenden Spannungswerten sind mit IMS abgesprochen
#
VOLTAGE_MAX=5.5;
VOLTAGE_MIN=4.5;
#
# Delay modification due to processing conditions...
# Die als Kommentar bezeichneten Werten werden von MENTOR
# vorgeschlagen.
#
#  PROCESS_BEST=0.56;
#  PROCESS_NOMINAL=1.00;
#  PROCESS_WORST=1.75;
#
# Die folgenden Werte sind in Absprache mit dem IMS folgend
# festgelegt worden:
#    _BEST    : aus den FAST-Mittelwerten, berechnet bei: 5.5V, 0 C
#    _NOMINAL : berechnet bei: 5.0V, 27 C
#    _WORST   : aus den SLOW-Mittelwerten, berechnet bei: 4.5V, 85 C
#
#
PROCESS_BEST=0.60;
PROCESS_NOMINAL=1.00;
PROCESS_WORST=1.80;
#
# Estimated net capacitance based on connections (in PFS)...
# (A net with 3 connections will have a capacitance of 0.207 pfs)
# Diese Zusatzkapazitaeten werden von MENTOR
# vorgeschlagen und sind nicht an den IMS-Prozess
```

```
# angepasst.
#
# EXPECTED_NET_CAP=.058 1;
# EXPECTED_NET_CAP=.115 2;
# EXPECTED_NET_CAP=.207 3;
# EXPECTED_NET_CAP=.252 4;
# EXPECTED_NET_CAP=.370 5;
##
# Die folgenden Zusatzkapazitaeten sind Mittelwerte die aus
# einem Entwurf bestimmt worden sind. Sie sind darum noch mit
# grossen Unsicherheit behaftet.
#
EXPECTED_NET_CAP=.024 1;
EXPECTED_NET_CAP=.077 2;
EXPECTED_NET_CAP=.125 3;
EXPECTED_NET_CAP=.160 4;
EXPECTED_NET_CAP=.210 5;
EXPECTED_NET_CAP=.213 6;
EXPECTED_NET_CAP=.246 7;
#
# Estimated net resistance based on connections...
# (A net with 3 connections will have a resistance of 0.207)
# Diese Leitungswiderstaende werden von MENTOR
# vorgeschlagen. Sie werden zur Zeit nicht ausgewertet.
#
EXPECTED_NET_RES=.058 1;
EXPECTED_NET_RES=.115 2;
EXPECTED_NET_RES=.207 3;
EXPECTED_NET_RES=.252 4;
EXPECTED_NET_RES=.370 5;
#
```

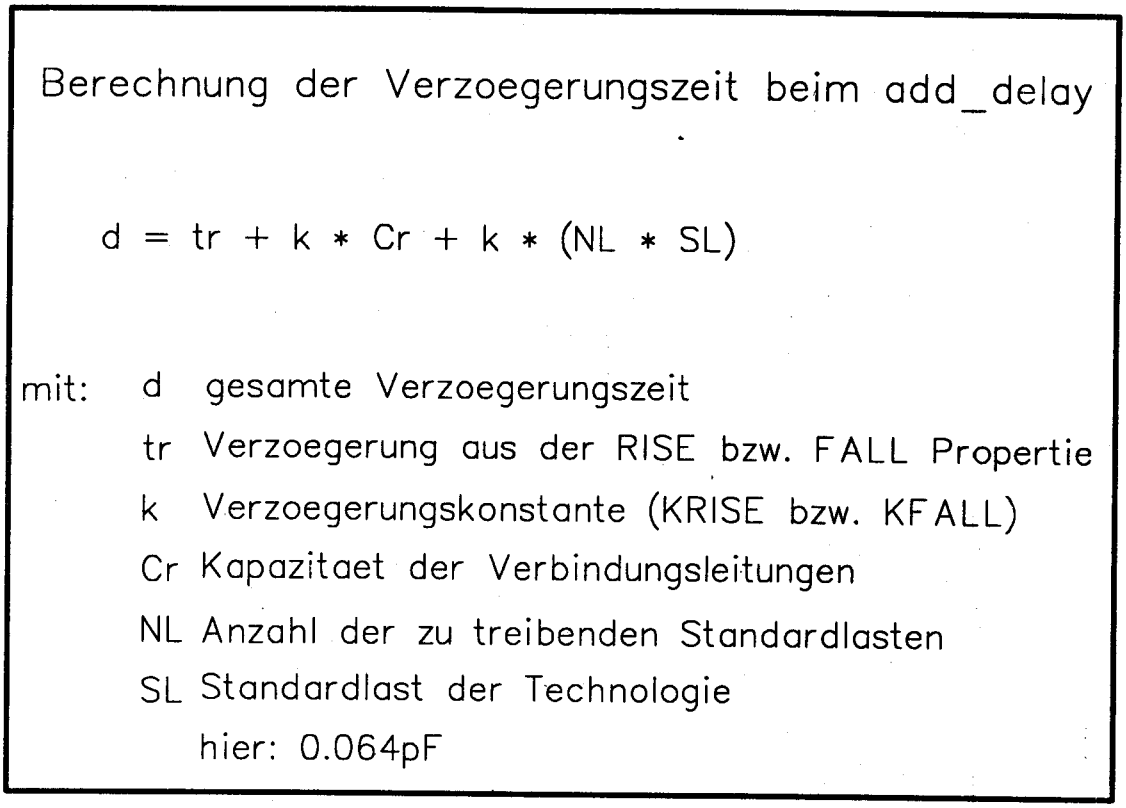


Bild 4-5 Berechnung der Verzoegerungszeit mit add_delay

4.8 IMS_SIMLOG & QUICKSIM

Nach der Erstellung des Stromlaufplanes mit **neted** wird mit **ims_package** unter Verwendung des Switchs **-qs** eine Simulation mit dem interaktiven Programm **quicksim** durchgeführt.
Vor der Simulation wird mit dem Dienstprogramm **ims_simlog** eine Rahmenstimulidatei **sim_start** erzeugt, die beim Aufruf von **quicksim** aus dem Programm **ims_package** automatisch ausgeführt wird. Bei der Erzeugung dieser Datei werden dem Benutzer Fragen über den gewünschten Inhalt der Datei gestellt:

- welche Länge die CLOCK-Periode bei der Simulation in ns haben soll
- ob für die Ein- und Ausgänge der Schaltung **(portin, portout)**
  - LOG-Kommandos
  - LIST-Kommandos
  - TRACE-Kommandos
  - MONITOR-Kommandos

in die Stimulidatei **sim_start** eingetragen werden sollen.

Beispiel für eine **sim_start** Stimulidatei:

```
# IMS QUICKSIM Command File
#
# Created on: Thursday, June 11, 1992  8:13:03 am (MES)
#
TRANSCRIPT ON
VIEW SHEET
#TEMPLATE FORCE WIRED
FORGET EXPRESSION  -A
FORGET MONITOR  -A
FORGET LOG  -A
FORGET TRACE       -A
FORGET LIST        -A
FORGET BUS         -A
FORGET BREAK  -A
#CHECK -Hazard -Race -NOSpike -NOTiming
#DEFINE BUS name  sig3 sig2 sig1 sig0  -c                          #Beispiel
#DEFINE BUS name_SOLL  sig3_SOLL sig2_SOLL sig1_SOLL sig0_SOLL -C   #Beispiel
#DEFINE EXPRESSION error1 ((x <> x_SOLL) AND (x_SOLL <> XXS))       #x_SOLL ist
eine Signalgruppe mit 5..8 Signalen
#DEFINE EXPRESSION error2 ((y <> y_SOLL) AND (y_SOLL <> XS))        #y_SOLL ist
ein einzelnes Signal
#DEFINE EXPRESSION error3 ((z <> z_SOLL) AND (z_SOLL <> XS))
#DEFINE EXPRESSION error  ((error1 OR error2 OR error3) AND (SCAN = 1S))
#BREAK error
SCALE USER TIME 1
SCALE TRACE TIME 100
PERIOD LIST 100
PERIOD TRACE 1000
# Die folgenden Kommandos von #BEGIN bis #END sollten in die Datei sim_forces.do
verlagert werden

# BEGIN
RESET SIM TIME
#TRANSCRIBE STIMULUS ÄpathnameÜ -Absolute -Replace
CLOCK PERIOD 100
# Die halbe Periodendauer wird auch in der Datei dec/sim.data abgelegt und von anderen
Programmen dort abgeholt
# FORCE CLK 0 0 -R
# FORCE CLK 1 50 -R
# FORCE SCAN 0 0
# FORCE SCAN 1 40 -R
# FORCE SCAN 0 49 -R
# Initialize all signals...
# END
#-------------------------------------------------------------------
## DO sim_forces.do

#-------------------------------------------------------------------
# Druckerausgabe: Auf eine DIN-A4 Seite gehen 8 Trace-Skalierungen
#               => Ausdruck auf ein Blatt mit n = (Laufzeit / 8)
```

```
# PLOT TRACE „ , , , n
#
# Abspeichern des Ausdrucks in das File TRACES.PIC
# Das n sollte dem n der Druckerausgabe entsprechen, zwecks spaeteren Ausdruck
# PLOT TRACE traces „ , , n -REplace
#
# Spaeterer Ausdruck mit: PLOT dir_name traces.pic -PR printer_name
#
# BYE
```

Weitere Files die von ims_simlog erzeugt werden sind:

io_net_names ist eine Liste mit allen Ein- und Ausgängen
sim.data enhält nur die Information über die halbe CLock-Periode

Nach **ims_simlog** beim Aufruf von **quicksim** wird die Stimulidatei **sim_start** im EDITier-MODUS am Bildschirm dargestellt. Der Benutzer kann jetzt individuelle Einstellungen für die gewünschte Simulation vornehmen:

-Der Bereich von #BEGIN und # END mit den FORCE-Kommandos muß mit der Hand (COPY und PASTE) in die FORCE-Datei **sim_forces.do** verlagert werden.
-Der Benutzer kann dabei weiter FORCE-Befehle in der Datei **sim_forces.do** hinzufügen
-Der Bereich zwischen #BEGIN und #END muß aus der Datei **sim_start** gelöscht werden.

Besonderheit der **sim_start** Datei:

Es wird dem Bunutzer eine Möglichkeit in Form eines Beispiels angeboten, wie er Laufzeitverletzungen durch Vergleich von IST und SOLL-Signalen unter Verwendung eines SCAN-Signales (Bild 4-6) erkennen kann:

```
#DEFINE BUS x  sig3 sig2 sig1 sig0  -c
#DEFINE BUS x_SOLL  sig3_SOLL sig2_SOLL sig1_SOLL sig0_SOLL -C
#Für jedes Signal wird eine einzelne Errorabfrage gemacht. Die Fehlermeldung ist aktiv, wenn
Ist- und Soll-Signal nicht gleich sind, und das Soll-Signal nicht deaktiviert ist.
#DEFINE EXPRESSION error1 ((x <> x_SOLL) AND (x_SOLL <> XXS))
#x_SOLL ist eine Signalgruppe mit 5..8 Signalen
#DEFINE EXPRESSION error2 ((y <> y_SOLL) AND (y_SOLL <> XS))
#y_SOLL ist ein einzelnes Signal
#DEFINE EXPRESSION error3 ((z <> z_SOLL) AND (z_SOLL <> XS))
# Die eigentliche Fehlermeldung erscheint in Quicksim, wenn zu dem Zeitpunkt des auf 1
gesetzten SCAN-Signals eine Fehlermeldung eines einzelnen Signals vorliegt.
#DEFINE EXPRESSION error  ((error1 OR error2 OR error3) AND (SCAN = 1S))
#BREAK error     #
```

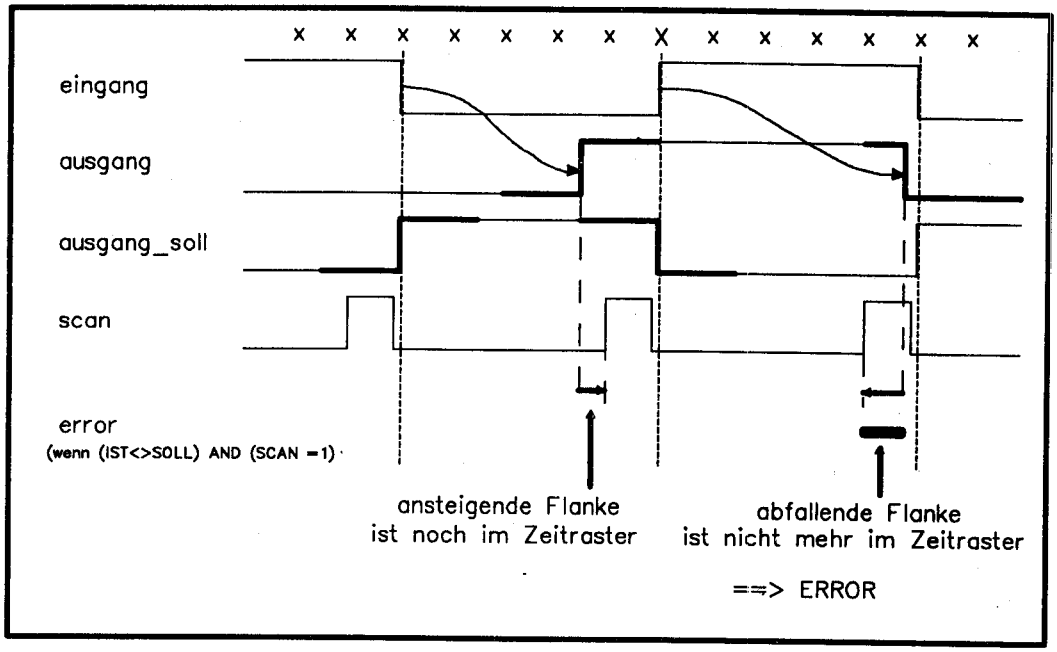


Bild 4-6 Ueberpruefung von Signalen mit Hilfe des SCAN-Signales

Nach Beendigung der Simulation mit Quicksim wird für einen Testautomaten beim IMS eine Datei (**sim.xverted**) erstellt. Dazu wird **ims_simlog** mit seinem zweiten Teil (ims_simlog ist in zwei Teile gegliedert) aufgerufen. Der Benutzer wird bei diesem Durchlauf nach der Periode des Testclocks abgefragt.
Dieser weitere Aufruf von **ims_simlog** erfolgt nur, wenn der Benutzer vor der Simulation auf die Frage, ob er LOG-Komandos erzeugt haben will, mit yes geantwortet hat. Antwortet er mit yes, so wird während der Simulation das LOG-File **sim.log** erzeugt.
In **sim.log** sind die bei der Simulation auftretenden Wechsel an den Signalen hintereinander aufgeführt.

4.9 Weitere Programme des Baukastens

Nach der Simulation des Stromlaufplanes und der Erstellung eines Layouts mit dem Batch ims_package und dessen Unterprogrammen, muß das Layout kontrolliert und gegebenfalls modifiziert werden, bevor es auf Magnetbändern abgespeichert und zur Fertigung ans IMS geliefert werden kann.
Die Kontrolle des Layouts richtet sich hierbei ausschließlich auf die Leitungsanschlüsse der Padzellenports.
Es tritt hier nämlich das gleiche Problem wie bei der alten 2um Technologie auf:
Die Leitungsanschlüsse liegen nicht immer exakt auf den Ports der Padzellen (Bild 4-7), sondern schneiden diese oft nur bzw. treffen sie gar nicht.
Solche Fehler müssen dann von Hand in CHIPGRAPH unter Berücksichtigung der Designregeln beseitigt werden (Routing Grid).
Um das fertige Design ansehen zu können benötigt man **gategraph** und **chipgraph.**
Für **gategraph** lautet der Aufruf: **gategraph** <design> <design>/tec
Mehr zu GATEGRAPH siehe GateStationUser's Manual.

Der Aufruf von **chipgraph** mit den richtigen Voreinstellungen übernimmt das im folgenden Kapitel beschriebene Dienstprogramm **cg.**

4.9.1 CG

Aufruf von CHIPGRAPH mit Anpassung an die hierarchische Struktur des mit **ims_package** erstellten Layouts, Tastenvorbelegungen und Darstellung der logischen Schaltung.
Dieses Programm ist hilfreich für die Beseitigung der Anschlußfehler an den Ports der Padzellen.

Syntax

cg <design_name> [master]

Beschreibung der Parameter

design ist das Haupverzeichnis der hierarchisch aufgebauten Schaltung

master ist der Name des verwendeten Mastertyps bei der Erstellung des Layouts

Programmbeschreibung

Nach dem Aufruf des Programms wird der zur Schaltung gehörende Master ermittelt und Chipgraph aufgerufen. Bei diesem Aufruf wird eine Tastenvorbelegung durchgeführt (Tabelle 4-2), das ProcessDefinitionFile gelesen, die hierarchische Schaltung (Tabelle 4-3) gepeekt und zusätzlichdie logische Schaltung in einem Fenster dargestellt.

Mit dem Programm lassen sich neben den Chipgraphdaten mit der speziellen hierarchischen Struktur aus dem **ims_package** auch normalen Chipgraphdaten anschauen.

Verbesserung der Anschlüsse an den Padzellen (Bild 4-7)

Nach dem Drücken der Taste '2' des rechten Tastenfeldes entspricht das Gridraster dem Routing Grid des verwendeten Masters.
Mit den Tasten '7' , '8' , '9' und '+' des rechten Tastenfeldes kann man in die einzelnen Hierarchiestufen des Designs springen. (siehe Tabelle).
Für Änderungen an den Verbindungsleitungen (PATH) muß man sich in der routing_cell befinden.
Diese Änderungen können mit dem Befehl aus dem POPUP-Menü EDIT - EDIT PATH - PATH MODIFY ausgeführt werden. (Siehe dazu Chipgraph Reference Manual).

Regeln bei der Verbesserung

- Die Verbindungsleitungen müssen auf dem Routinggrid liegen (Taste '2').
- Die dargestellten Blockierungen der Metallagen in den Padzellen dürfen nicht überfahren werden.
- Der Anschluß an den Port erfolgt ausschließlich im Metall1 (CM-Layer).
- Der Port darf nur senkrecht von innen (vom Corebereich) angefahren werden.

(**Keine horizontalen** (Padzelle alleine betrachtet) **Leitungen auf der Padzelle**).

- Nur das letzte Leitungsstückchen sollte wenn möglich modifiziert werden.

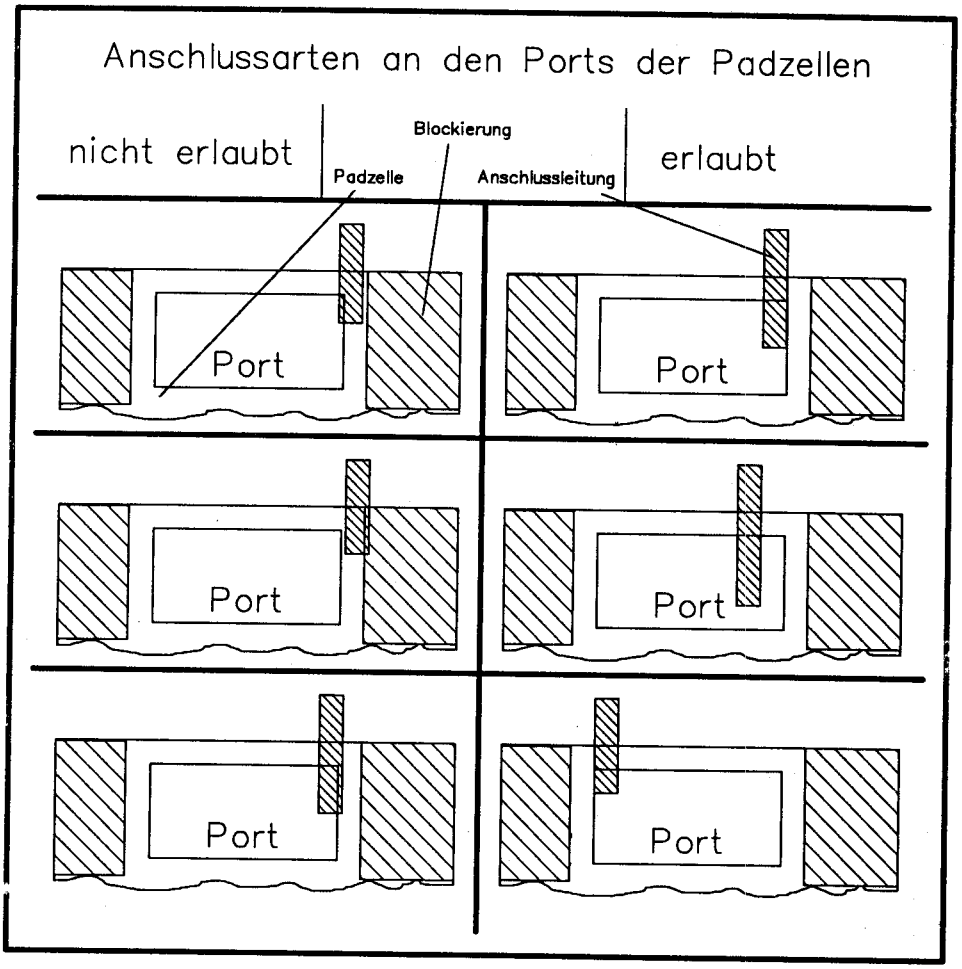


Bild 4-7 Anschlussarten des Routingprogramms

Taste	Funktion	Erläuterung
F0	FIT SEL	Zeigt selektiertes Teil groß auf dem Bildschirm
shift + F0	ZOOM OUT 2	Zeigt die doppelte Fläche an
ctrl + F0	ZOOM IN 2	Zeigt mehr Details
F9	INST CELL ...	Holt aktivierte Zelle in das EDIT-Window
shift +F0	ACT CELL ...	Aktiviert Zelle aus der Macrozellenlibrary
1 (Rechtes Tastenfeld)	GRID 0.1 0.1 1 1	Grid minimal einstellen Eignet sich zum Abstände messen
2 (")	**GRID ...**	Grid entspricht dem Routing-Grid des Masters. Muß vor der Verdrahtung mit der Hand eingestellt werden
5 (")	STATUS OBJECT	Zeigt die Eigenschaft des selektierten Bauteils an
6 (")	DO /user/ims/ulm_com/	Bereitet die Ausführung von Batches aus dem Verzeichnis ... vor.
7 (")	env name design_cell	Sprung in der Hierarchie auf die Designzelle
8	env name macro_cell	Sprung in die Macro_cell
9	env name power_cell	Sprung in die power_cell
+	**env name routing_cell**	Sprung in die routing_cell. Muß vor der Verdrahtung mit der Hand eingestellt werden.
Die hervorgehobenen Tasten sind für die Verdrahtungen an den Ports der Padzellen von Nutzen.		

TABLE 4-2 Tastenvorbelegung

master_chip im Verzeichnis /user_temp/		
design_cell im Verzeichnis .../<design>/		
macro_cell im Verzeichnis .../<design>/	power_cell im Verzeichnis .../<design>/	routing_cell im Verzeichnis .../<design>/
Hier sind alle auf dem Master plazierten Macro- zellen (Core- und Pad- zellen)	Hier ist die Powerzelle mit ihren zur Strom- versorgung notwendigen Powerpadzellen	Hier sind sämtliche Verbindungsleitungen zwischen den Macro- zellen.

TABLE 4-3 Hierarchie des Schaltungslayouts in Chipgraph

4.9.2 CHIP_TO_GDS2

Die Chipgraphdaten einer Schaltung werden für die Übergabe an das IMS vorbereitet.

Syntax

chip_to_gds2 <gds2_file_name> <design> <ort> <nummer>

Beschreibung der Parameter

gds2_file_name ist der Name des GDS2 -Files

design ist das Hauptverzeichnis der hierarchisch aufgebauten Schaltung

ort ist die Angabe über die Herkunft der Daten.

nummer dient der Trennung verschiedener Chipgraphdaten gleicher Herkunft

Programmbeschreibung

Bevor die fertigen Layoutdaten zum IMS geschickt werden, müssen noch mehrere Veränderungen an ihnen vorgenommen werden.

1. Bei der Fertigung werden mehrere verschiedene Schaltungen von unterschiedlichen Fachhochschulen) auf einem Wafer plaziert. Das dafür zuständige Programm beim IMS verlangt für jeden Schaltungstyp jedoch einen speziellen, nur einmal vorkommenden Namen.
Das Programm **cg** erweitert daher die Namen der Chipgraph-Zellen (design_cell, macro_cell, routing_cell und Power_cell) mit dem Herkunftsort **(ort)** und einer Nummer **(nummer)**, so daß dieser Name nur einmal vorkommt.
Bsp.: macro_cell ---> macro_cell_ulm_007
Der Inhalt des Referencfiles der design_cell wird entsprechend den neuen Namen für die macro_, power_, und routing_cell modifiziert.

2. Die mit erweiterten Namen versehenen Chipgraph-zellen werden mit dem Programm **translate write** in das GDS2-Format umgewandelt und dabei in das File **<home_verzeichnis>/gds2_data/gds2_file_name** abgelegt.

Voraussetzungen:

In dem Verzeichnis des Designs müssen die notwendigen Chipgraph-zellen (macro_cell,...) vorhanden sein.
Der Benutzer benötigt in seinem 'Home'-Verzeichnis ein Unterverzeichnis **gds2_data** in das die GDS2-Files geschrieben werden können.

5. Schaltungssynthese mit LOG_IC

Neben der Stromlaufplaneingabe ist für die neue IMS-GATE FOREST_1.2-Technologie auch eine Schaltungssynthese mit LOG_IC möglich. Die dafür notwendigen Dateien und Programme sind in dem Verzeichnis ~/ims/forest_1.2/log_ic untergebracht.

Beschreibung der Dateien:

ims_lib.dgl: Macrozellenbibliothek im ASCII-Format.
Sie enthält zu jeder Zelle Informationen über

- den Zellentyp, den Namen und über den Platzbedarf der Zelle
- die Anzahl, die Namen und Lage der Ports
- die Verzögerungszeit
- und Größe der Zelle in NETED

ims_lib.bgl: Kompilierte Macrozellenbibliothek für LOG_IC

vldnet.nld: Hier stehen die Informationen für den Postprozessor. Mit diesen Informationen erzeugt er ein Batchfile, das bei der Ausführung einen Stromlaufplan in NETED erzeugt.

logic.bin: Kompiliertes Startupfile für den Aufruf von LOG_IC mit voreingestelltenm SETUP-Werten.

Vorgehensweise bei der Schaltungssynthese mit LOG_IC:

1. Das Aufrufprogramm mit den notwendigen SETUPeinstellungen für LOG_IC /user/ims/forest_1.2/log_ic/**logic.bin** muß in das Arbeitsverzeichnis kopiert werden.
Es brauchen keine Veränderungen im Setupmenue mehr gemacht werden.

2. Aufruf von LOG_IC mit **logic** aus dem Arbeitsverzeichnis.

3. Editieren des WORKINGFILEs in LOGIC

und Anhängen eines *RUN_CONTROLLS an das Ende dieses Files:

```
*RUN_CONTROL
GATE-LIB = /user/ims/forest_1.2/log_ic/ims_lib.bgl
POST-PRO = vld.
```

GATE-LIB zeigt auf die kompilierte Bibliothek der Macro-zellen (ims_lib.bgl). Diese Angabe ist für die zweite Phase des Durchlaufs wichtig.

POST-PRO gibt den Namen des Postprozessors an. Mit Hilfe dieses Postprozessors und den Angaben im File vldnet.nld wird ein File mit dem Namen design_name.dat erzeugt.
Beim Ausführen dieses Files wird ein Stromlaufplan in NETED (design_name) erzeugt. Besonderheit dabei ist, daß die Verbindung der Pins nicht mit Leitungen, sondern mit der Netzproperty 'NET' beschrieben ist.

Ausschnitte aus der Macrozellenbiblithek:

Erläuterungen dazu befinden sich im Handbuch LOG/-IC 'Tools" (U33) unter dem Kapitel 3 Gates auf Seite A1-1 ff und im Handbuch LOG/-IC GatesPro "Tools" (U21) im Vorspann.

Kostenfaktor berechnet sich aus der COMPSIZE-Property
Lastfaktor 1 entspricht einer Kapazitaet von 64fF (Eingangskapazitaet des einfachen Inverters.)
Verzoegerungszeiten TP werden in Pico-Sekunden berechnet:
TP = To + (Lastfaktor * k) z.B.: TPLH = 650 + L * 189
mit To = Ti
k = (kapazitaet pro Lastfaktor) * m hier ist K = 64 * m
Ti und m sind im Datenbuch des Gate Array zu entnehmen

```
*GATE
TYPE       = ZINV;
NAME       = it02d1;
INPUTS     = 2;
COST       = 5;
SIGNALS:  I = 1[I(0,0)],
         /OE = 2[OEN(0,9)],
          Q = 1[ZN(18,0)];
TPLH:        650 + L * 189;
TPHL:        540 + L * 89;
LEFT-BOTTOM  = (0,-3);
RIGHT-TOP    = (18,12);
INSTANCE     = (0,-4);
...

*FLIPFLOP
TYPE       = JK-FLIPFLOP;
NAME       = jkbtnb;
COST       = 32;
SIGNALS: /PS = 3[SDN(-12,15)],
          J = 1[J(-12,-3)],
        CLK = 1[CP(-12,9)],
        /RS = 3[CDN(-12,12)],
          K = 1[K(-12,0)],
         /Q = 1[QN(6,-3)],
          Q = 1[Q(6,0)];
SETUP      = 1000, 800 ;
CQHL:        1100 + L * 58;
CQLH:        1100 + L * 96;
CNLH:        1000 + L * 102;
CNHL:        900 + L * 64;
LEFT-BOTTOM  = (-12,-6);
RIGHT-TOP    = (6,18);
INSTANCE     = (-12,-7);
...

*GATE
TYPE       = MUXPOS;
NAME       = me21d1;
INPUTS     = 2, 1, 1;
COST       = 9;
SIGNALS:  I = 1[I0(0,3),I1(0,0)],
          S = 2[S(0,12)],
         /E = 1[EN(0,9)],
          Q = 1[Z(18,3)];
TPHL:        830 + L * 70;
TPLH:        940 + L * 186;
LEFT-BOTTOM  = (0,-3);
RIGHT-TOP    = (18,15);
INSTANCE     = (0,-4);
...

*SPECIAL_ELEMENT
TYPE       = OUT_CON
NAME       = PORTOUT;
SIGNALS:   I = 1[X(0,0)];
LEFT-BOTTOM  = (0,-3);
RIGHT-TOP    = (6,3);
INSTANCE     = (0,-3);
```

Liste aller Macrozellen in der Bibliothek ims_lib.dgl:

an02d1	an03d1	an04d1	an05d1
an06d1	an07d1	an08d1	dfbtnb
dfctnb	dfntnb	dfptnb	in01d1
in01d4	it02d1	jkbtnb	jkctnb
me21d1	mfbtnb	mfctnb	mfptnb
mx21d1	nd02d1	nd03d1	nd04d1
nd05d1	nd06d1	nd07d1	nd08d1
ni01d2	ni01d4	nr02d1	nr03d1
nr04d1	nr05d1	nr06d1	nr07d1
nr08d1	nt02d4	or02d1	or03d1
or04d1	or05d1	or06d1	or07d1
or08d1	xn02d1	xo02d1	xo03d1

5. DESIGN UND LAYOUT EINER INTEGRIERTEN SCHALTUNG ZUR AUTOMATISCHEN TONSTANDARDIDENTIFIKATION

M. Rieger Thomson Consumer Electronics
S. Hauser Fachhochschule Furtwangen

ABSTRACT

Es wird eine Schaltung zur automatischen Tonstandardidentifikation vorgestellt, die im wesentlichen aus einem Signalverarbeitungsteil und einer PLL besteht. Im Signalverarbeitungsteil werden selektiv bestimmte Spektralanteile des zu untersuchenden Signals herausgefiltert und gleichgerichtet. Mit der Wahl der Frequenz des Ausgangssignales der PLL wird die Empfangsfrequenz des Signalverarbeitungsteils festgelegt.

1. EINLEITUNG

Heutige Fernsehempfänger arbeiten nach dem Überlagerungs-Prinzip (Bild 1): das an der Antenne anliegende TV-Signal, das im Frequenzbereich von etwa 50 MHz bis 850 Mhz liegt wird auf eine feste Zwischenfrequenz umgesetzt, so daß die Bildträgerfrequenz f_{BT} = 38.9 MHz beträgt. Der zwischenfrequente Tonträger liegt bei $f_{TT} = f_{BT} - F_T$, wobei F_T ein Frequenzoffset ist, der vom TV-Standard abhängt.

In Deutschland betägt F_T = 5.5 MHz, so daß f_{TT} = 33.4 MHz wird. In dem mit ZF-Demodulation bezeichneten Block wird das zwischenfrequente TV-Signal verstärkt und der Tonträger (bei f_{TT}) wird in einem Mischer auf F_T umgesetzt, wobei der mittels einer PLL regenerierte Bildträger als Lokaloszillator (bei f_{BT}) dient.

Am Ausgang des ZF-Demodulationsblockes stehen das Videosignal und das noch modulierte Tonsignal bei F_T zur Verfügung.

Tafel 1 zeigt für die in Europa üblichen TV-Standards die Lage von F_T (= sound carriers) und die zugehörige Modulationsart. F_T liegt zwischen 4.5 MHz und 6.5 MHz. Mögliche Modulationsarten sind Frequenzmodulation (FM), Zweiträger-Frequenzmodulation (FM/FM), Frequenzmodulation plus Nicam (FM/Nicam) sowie Amplitudenmodulation (AM).

Damit nun der Ton-Demodulationsblock (Bild 1) ordnungsgemäß arbeiten kann, müssen die Mittenfrequenz und die Demodulationsart entsprechend dem Standard des empfangenen Signales eingestellt werden. In heute üblichen Geräten müssen etliche Einstellungen dieser Art vom Benutzer in manueller Weise über die Fernbedienung eingegeben werden, was leider etwas unbequem und anfällig auf Bedienfehler ist. Die hier vorgestellte Schaltung zur Identifikation des TV-Standards soll diese Unzulänglichkeit beheben.
Die Frequenz F_T soll automatisch erkannt werden, was eine automatische Einstellung des Ton-Demodulationsblockes ermöglicht.

2. KONZEPT

Die vorgestellte Schaltung zur Tonstandardidentifikation arbeitet nach dem Prinzip eines durchstimmbaren Überlagerungsempfängers (Bild 2).
Die Schaltung besteht im wesentlichen aus einer PLL mit einstellbarem Teiler und einem Signalverarbeitungsteil. Das Eingangssignal 1 wird in einem Mischer abwärts gemischt in 2. Als Lokaloszillatorsignal dient das Ausgangssignal der PLL bei F_{ref}. Der Bandpass läßt nur Spektralanteile in der Nähe seiner Mittenfrequenz von 125 kHz durch. Das bedeutet, wenn der Empfang der Frequenz F1 gewünscht wird, muß F_{ref} = F1 - 125 kHz sein. Nach dem Bandpass wird das zu untersuchende Signal zur besseren Auswertbarkeit mittels eines Quadrierers gleichgerichtet. Ein nachfolgender Tiefpass unterdrückt unerwünschte Multiplikationsprodukte. Das Ausgangssignal wird in einem AD-Wandler digitalisiert und über Bus dem μP zur Beurteilung geschickt. Die Einstellung von F_{ref} erfolgt über einen programmierbaren Teiler, dessen Teilerverhältnis N vom μP eingestellt wird: F_{ref} = 15.625 kHz • N.

Es ist ersichtlich, daß die Schaltung nicht nur bei F = F_{ref} + 125 kHz ein Ausgangssignal liefert, sondern auch bei F = F_{ref} - 125 kHz (Spiegelfrequenz). Diese Eigenschaft ist jedoch hier nicht störend, weil die Identifikation der Tonträger (Tafel 1) verwechslungsfrei möglich ist.

In Bild 2 ist ein Beispiel dargestellt, bei dem am Eingang Signale bei 4.43 MHz (Farbträger), 5.5 MHz (1. Tonträger) und 5.74 MHz (2. Tonträger) anliegen. Wenn F_{ref} = 5.5 MHz - 125 kHz ist, fällt der 1. Tonträger in den Durchlassbereich des Bandpasses, während der Farbträger und der 2. Tonträger weitgehend unterdrückt werden.

3. SCHALTUNGSDESIGN

Nachdem das Konzept feststand, konnten die einzelnen Blöcke in Transistorebene designed und simuliert werden. Für die Simulation wurde PSPICE benutzt. DasSchaltbild wird in einem Graphic Editor von Valid eingegeben, wobei die Bauteile in einer Library vorliegen. Die Transistoren liegen mit ihren zugehörigen Parametern in der Bibliothek vor und können also z.B. in ihrer Größe nicht frei bestimmt werden. Mit Hilfsprogrammen, die vom Editor aus gestartet werden, kann man aus dem grafischen Schaltbild ein Textfile erzeugen, das als Spice-Eingabefile dient. Die Kommandos von PSPICE sind im Schaltbild enthalten. Von diesem Editor aus kann dann PSPICE gestartet werden, ebenso das Programm PROBE mit dem die Ergebnisse von PSPICE analysiert und dokumentiert werden können. PSPICE ist ein Simulator, der fast alle Transistorparameter berücksichtigt und auch mit angegebenen Toleranzwerten rechnet. Damit kann das Schaltungsverhalten sehr gut bestimmt werden.

3.1. SIGNALVERARBEITUNG

Im Zweig der Signalverarbeitung soll der Bandpaß näher beschrieben werden. Er besteht aus zwei Transkonduktanzverstärkern, einer Art Multiplizierstufe. Diese werden wie in Bild 3 gezeigt mit 3 Kondensatoren beschaltet. Mit dieser Anordnung, nur mit anderer Beschaltung der Kondensatoren, lassen sich auch Tiefpaß-, Hochpaßfilter und Bandsperren realisieren. Die Kondensatoren haben zueinander proportionale Werte (C0, m*C0 und (1-m)*C0) und deren Verhältnis zueinander beeinflußt die Güte des Bandpasses. Der Faktor m entspricht 1/Q, also der inversen Güte. Die Güte soll 5 betragen (± 15kHz 3dB-Bandbreite) und somit hat m den Wert 0,2.

Die Durchlaßfrequenz des Bandpasses soll bei 125kHz liegen und wird bestimmt durch das Gm der Transkonduktanzverstärker und dem Kondensatorwert C0. Die Transkonduktanz Gm der Verstärker wird durch das Verhältnis von 2 Strömen bestimmt. Der Ausdruck für die Mittenfrequenz F0 ist in Bild 3 enthalten. Bild 4 zeigt den Bandpaß auf Transistorebene und die Ströme I1 und I2 der Transkonduktanzstufen.

Um die Mittenfrequenz unabhängig von den Absoluttoleranzen der Bauteile zu machen, sollten die Ströme I1 und I2 abhängig von Re bzw. C0 sein. I1 wird abhängig von einem Widerstand erzeugt und das Produkt I1*Re bleibt somit konstant. Der Strom I2 wird im ersten Test von außen eingespeist und später wird ein auf dem Chip vorhandener C-abhängiger Strom verwendet. Die Absoluttoleranzen der Bauteile von ± 20% spielen nun keine Rolle mehr und die Relativtoleranzen sind mit ± 2% sehr klein und verändern nur die Güte des Bandpasses.

Wie die Simulation des Bandpasses zeigte nimmt die Dämpfung bei hohen Frequenzen merklich ab. Es treten aber auf jeden Fall Frequenzen im 10MHz-Bereich auf und es muß deshalb ein Vorfilter eingesetzt werden (siehe Bild 4). Dieser Tiefpaß ist ein Aktives Filter mit einer Eckfrequenz von 800kHz. Es wurde wegen der Integration so dimensioniert, daß die Werte der Kondensatoren und Widerstände nicht zu groß sind (2x5pF und 2x40kΩ). Das Simulationsergebnis des Bandpasses mit Vorfilter zeigt nun den gewünschten Frequenzgang (Bild 5).

3.2. PHASE LOCKED LOOP

Im zweiten großen Block, der PLL, steht der Voltage Controlled Oscillator im Vordergrund. Es wurde ein Relaxationsoszillator eingesetzt (Bild 6), der periodisch einen Kondensator über 2 Stromquellen umlädt. Die Schwingfrequenz hängt von dem Lade- und Entladestrom I1 und I2, dem Kondensator C und der Schwellspannung dU des Schmitt-Triggers ab. Der Strom I1 bleibt konstant, während der Strom I2 immer ein- und ausgeschaltet wird. Der Kondensator wird mit dem Strom I1 bei ausgeschaltetem Strom I2 aufgeladen, bis die Spannung am Kondensator den oberen Schwellwert des Schmitt-Triggers erreicht. Der Trigger schält nun den Strom I2 ein und der Kondensator wird mit dem Strom (I2-I1) wieder entladen, bis die Spannung am Kondensator den unteren Schwellwert des Schmitt-Triggers erreicht hat und der Strom I2 wieder ausgeschaltet wird. Damit wird der Kondensator von neuem geladen und der Vorgang wiederholt sich. Die Ausgangsspannung schwingt zwischen den beiden Schwellwerten des Schmitt-Triggers. Ist der Strom I2 doppelt so groß wie I1, dann sind Lade- und Entladezeit gleich und das Tastverhältnis des Ausgangssignals 1:1. Die Schwingfrequenz kann durch das Ändern der Ströme verschoben werden. Beide Ströme liegen an derselben Strombank und werden durch den Eingangsstrom immer um den gleichen Faktor geändert (Bild 7). Mit einem Eingangsstrom von 50 bis 250μA erhält man in der Simulation die gewünschten 2,5 bis 11MHz und die Frequenz ändert sich linear mit dem Eingangsstrom ohne Temperaturabhängig zu sein (Bild 6).

Bild 7 zeigt den VCO auf Transistorebene ohne zusätzliche Beschaltung. Für den Ausgang, der zum Teiler führt, mußte noch ein Level-Shifter eingesetzt werden, der die Ausgangspegel von 6,5V-7V auf 0-5V setzt. Der Teiler, der mit MOS-Gatter aufgebaut ist, kann nur 0-5V verarbeiten.

4. LAYOUT

Die Schaltung wird in einem BICMOS-Prozeß gefertigt, bei dem bipolare und CMOS-Bauteile in einem Prozeß hergestellt werden. Somit können digitale Schaltungsteile, wie z.B. Teiler, und analoge Schaltungsteile gemischt werden. Für das Layout stand ebenfalls der Graphic Editor zur Verfügung. Mit diesem Editor wurde das symbolische Layout gemacht, bei dem von den Zellen nur die Boundary und die Anschlüsse zu sehen sind und die Metallbahnen nur als einfache Wires gezeichnet werden. Man hat damit eine gute Übersicht und muß nur wenige Layoutregeln kennen. Auch aufgrund der Bedienungsfreundlichkeit des Editors fällt das Erlernen der Layout-Erstellung leicht. Es kann aber auch nur die aktuelle Ebene bearbeitet werden. Das Layout wird in mehreren Hierarchiestufen erstellt. Die Bauteile (Transistoren, Widerstände, Kondensatoren, Gatter) liegen als Standardzellen in einer Bibliothek vor und in einer Zelle können die Standardzellen plaziert und verbunden werden. Diese so entstandenen Zellen können in eine Bibliothek aufgenommen werden und dann als Zellen in Layouts höherer Stufe eingesetzt werden.

Die Transistoren liegen ja als Standardzellen vom Hersteller gegeben vor, ihre Größe und Ausführung ist fest und damit auch ihre Parameter. Die Transistoren gibt es mit verschiedenen Emittergrößen, sowie Doppelbasis- und Doppelemitter-Anschlüssen bei NPN-Transistoren und 2 bzw. 4 Kollektor-Anschlüsse bei PNP-Transistoren. Insgesamt stehen so 14 unterschiedliche Transistoren zur Verfügung. Eine Auswahl enthält Bild 8.

Es gibt 4 Arten von Widerständen, deren sheet resistance und deren Toleranzen unterschiedlich sind. Da aber alle aus einer p-dotierten Schicht bestehen, müssen sie in ein EPI-Gebiet gelegt werden, das an ein hohes Potential, meistens die Versorgungsspannung, gelegt werden muß. Je nach Wert und gewünschter Toleranzen wird eine Art des Widerstandes schon beim Design ausgewählt. Um geringere Toleranzen zu erreichen werden die Widerstände oft breiter als die übliche 6μm gemacht, da sich dadurch die Toleranzen der Geometrie, die Hauptverursacher sind, weniger auswirken.

Auch die Kondensatoren können aus 3 Arten ausgewählt werden, wobei meistens die Sandwich-Kondensatoren wegen ihrer hohen Kapazität verwendet werden. Die Toleranzen nehmen auch hier mit der Größe des Kondensators ab.
Die beiden Bauteile (Widerstand, Kondensator) werden mit Hilfe eines Programmes kreiert, bei dem die Art, der Wert und die Breite des Bauteils angegeben werden muß. Das Programm berechnet die Länge, kreiert das Bauteil und legt es in einer Bibliothek ab.

Die MOS-Transistoren können in ihrem W/L-Verhältnis frei festgelegt werden, wobei die Länge mindesten 2μm betragen muß. Das Standardverhältnis ist 6/3. Die NMOS-Transistoren müssen im Layout mit Substratkontakten umgeben und die PMOS-Transistoren in ein EPI-Gebiet gelegt werden, um sie zu isolieren.
Für die Verdrahtung hat man 2 Metallebenen, die jeweils eine Breite von 2μm haben. Diese Festlegung gilt nur für das symbolische Layout. Wird im Layout-Editor gearbeitet, kann die Metallbahn-Breite frei festgelegt werden, was beim Anschluß der PADs auch nötig ist. Im Layout-Editor sind alle Ebenen und Layer sichtbar und können bearbeitet werden.

Von den einzelnen Blöcken der Schaltung wurde das symbolische Layout anhand der Logik-Zeichnungen gemacht. Eine festgelegte Standardhöhe der Zellen soll eingehalten werden, da somit alle Zellen gleich groß sind und im TOP-Layout nebeneinander plaziert und die GND- sowie die VCC-Leitung durchgezogen werden können. Die GND- und VCC-Leitung sind jeweils 18μm breit. Durch die vorgeschriebene Zellenhöhe und durch die Standardzellen läßt sich das Layout nicht immer minimal gestalten, ist aber wesentlich schneller fertiggestellt.
Nach dem Erstellen des symbolischen Layouts einer Zelle wird ein Design Rule Check durchgeführt. Die Fehlermeldungen des DRC können als Kommentare an der Fehlerstelle in das Layout aufgenommen werden und erleichtern so die Fehlersuche. Nach dem DRC wird ein Extractor gestartet, der aus dem Layout eine Netzliste anfertigt. Diese Netzliste wird dann mit dem Programm Compare mit der Logik-Zeichnung verglichen. Diese Verifizierung des Layouts ist wichtig, um Fehler zu vermeiden.

Auch in diesem Abschnitt einige spezielle Dinge zum Bandpaß. Die Kondensatoren des Bandpasses werden alle außerhalb der Zelle im Gesamtlayout plaziert, da sie sehr viel Platz benötigen. Damit die Werte der Kondensatoren besser Matchen werden diese alle aus 3pF-Kondensatoren zusammengestellt. Die 2x5pF-Kondensatoren des Vorfilters werden jedoch in der Zelle plaziert. Beim Plazieren der Transistoren sollte darauf geachtet werden, daß zusammengehörende Bauteile nicht zu weit auseinanderliegen und dieselbe Ausrichtung haben. Dies ist vor allem bei Stromspiegeln oder Differenzstufen nötig. Für den ersten Test der Schaltung werden Widerstände zusätzlich zu anderen eingefügt und mit einer Metallbahn überbrückt. Diese Brücke kann mit einem Laser-Cutter aufgetrennt werden und damit der Wert des Widerstandes variiert werden. Dies ist wichtig, um eine Verstärkung oder den Aussteuerbereich einzustellen. Es werden Testpunkte gesetzt, das sind kleine Metall2-Flächen von mindestens 10μm Länge, auf die eine Nadel zum Messen gesetzt werden kann. Das Messen ist möglich, da die Test-Chips keine Passivierung besitzen. Bild 9 zeigt die Logik-Zeichnung und Bild 10 das symbolische Layout des Bandpasses in Schwarz/Weiß. Die farbige Darstellung auf dem Bildschirm ist wesentlich übersichtlicher.

Nachdem von allen Blöcken ein Layout erstellt war, wurden die Blöcke zusammengeschaltet, die noch fehlenden Kondensatoren eingesetzt und die Gesamtschaltung in ein PAD-Raster gesetzt. An die PADs werden die wichtigen Ein- und Ausgänge für den Test angeschlossen. Es sind mehr PADs, als später nötig sind (insgesamt 14), um die Funktionen der Schaltungsblöcke eventuell getrennt voneinander testen zu können. Der Chip wird nun gefertigt, gebondet und kommt als IC ohne Passivierung zurück. Dann wird getestet, verbessert (Redesign), das Layout abgeändert und von neuem getestet, bis ein befriedigendes Ergebnis vorhanden ist. Erst dann wird die Schaltung im Gesamtchip eingesetzt.

5. ZUSAMMENFASSUNG

Die Schaltung arbeitet als eine Art Spektrumanalyzer, der die Frequenzen 2-8MHz mit einer Rasterung von 31,25kHz abtastet und mit einer Auflösung von 31,25kHz arbeitet. Damit können die Frequenzen, die zur Tonstandard-Bestimmung wichtig sind, abgetastet werden.

Die Schaltung benötigt eine Fläche von 1,2mm^2 , was bei ca. 25 Pfennig/mm^2 einem Aufwand von 30 Pfennig entspricht.

Bild 1: Frequenzlage im TV-Empfänger

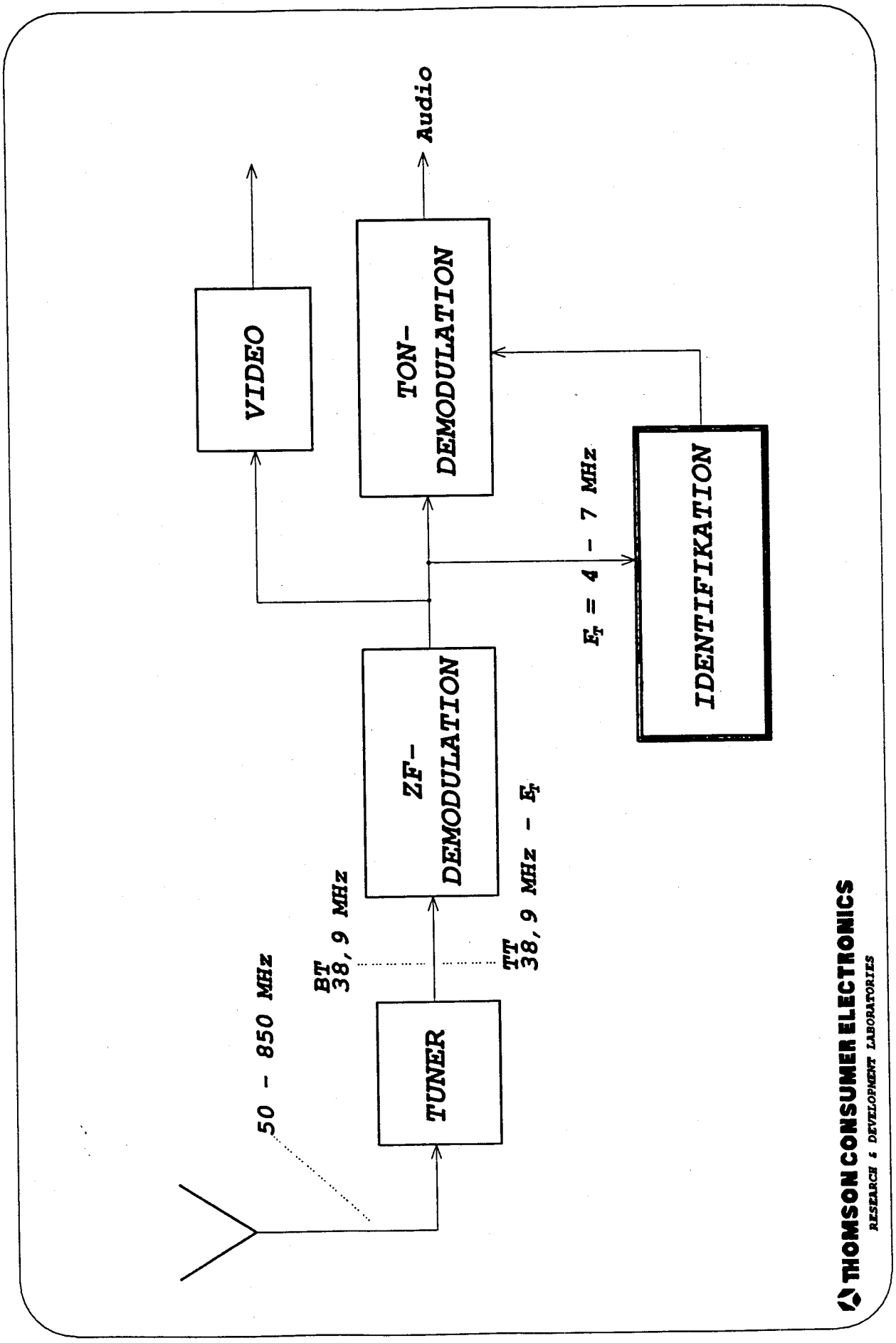
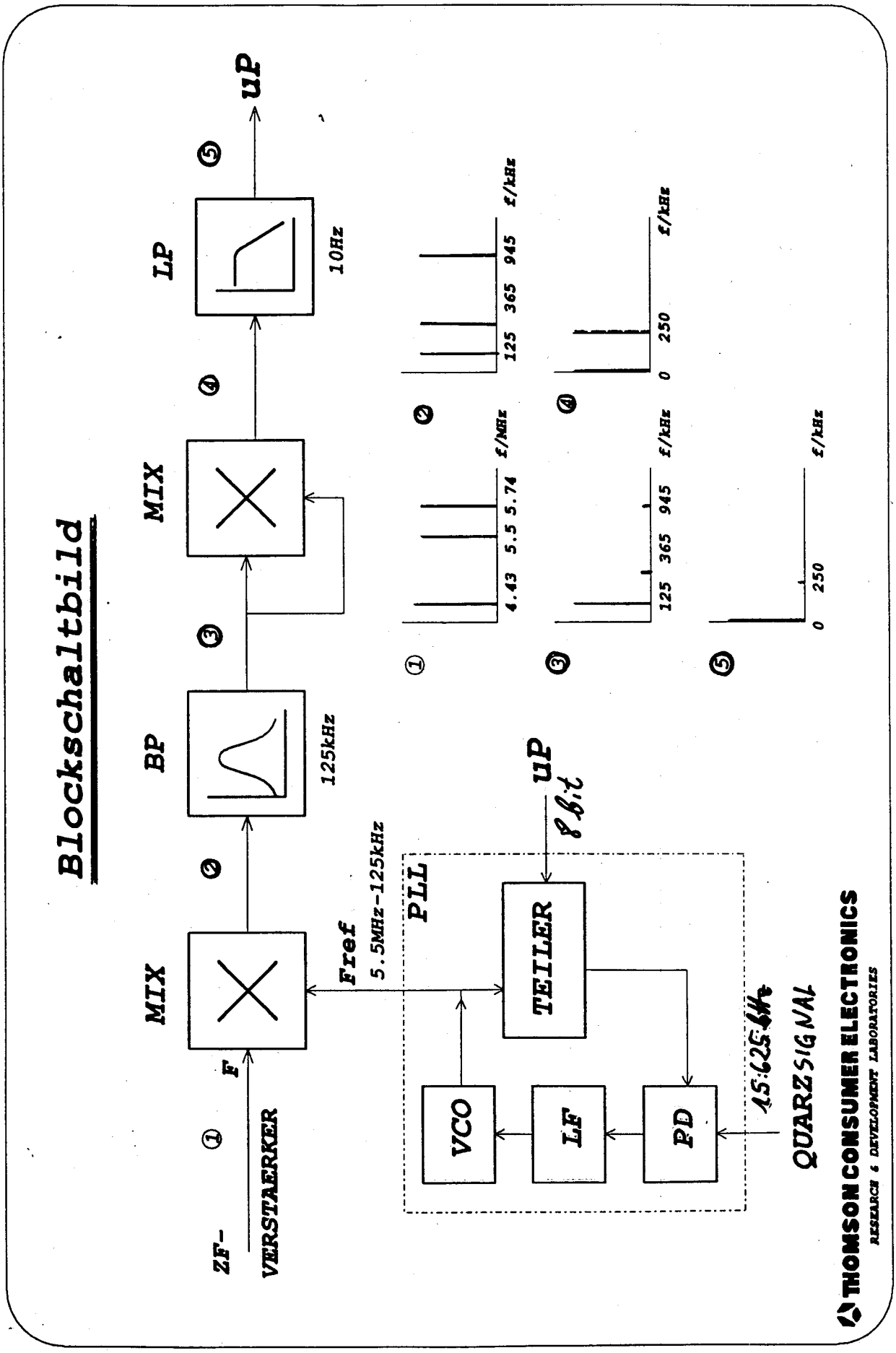


Bild 2: Blockschaltbild der Tonstandardidentifikation



TV Standards in Europe

Viewpoint: IF and Sound

System	Videoband-width [MHz]	Soundcarriers [MHz]	Sound-modulation
M,N	4.2	4.5	FM
B/G	5	5.5/5.74	FM/FM
B/G(Nicam)	5	5.5/5.85	FM/Nicam
I	5.5	6	FM
I(Nicam)	5.5	6.0/6.55	FM/Nicam
L	6	6.5	AM
D/K	6	6.5	FM

THOMSON CONSUMER ELECTRONICS
RESEARCH & DEVELOPMENT LABORATORIES

Tafel 1: TV-Tonstandards in Europa

Bild 3: BP-Blockschaltbild

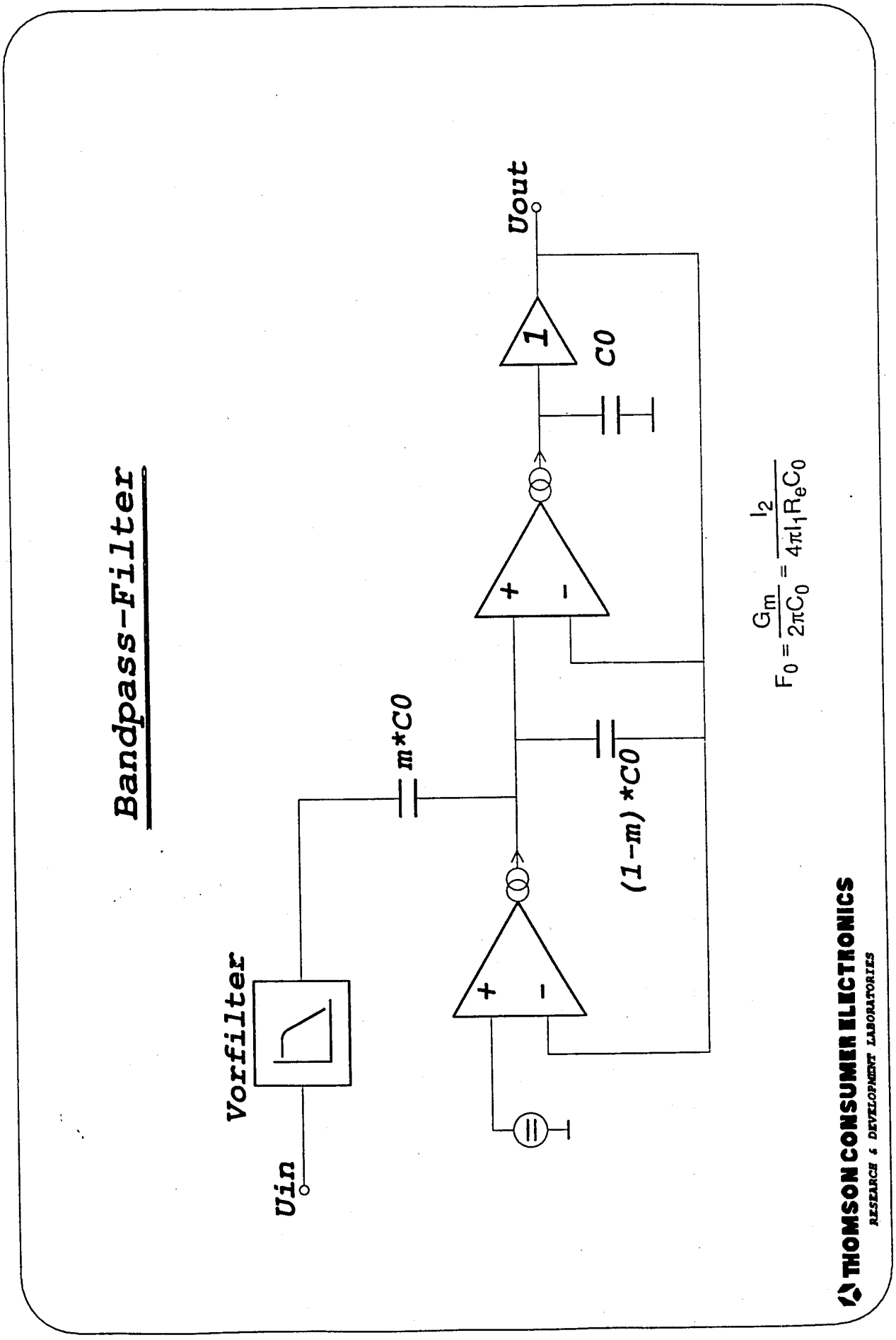


Bild 4: BP-Logik-Zeichnung

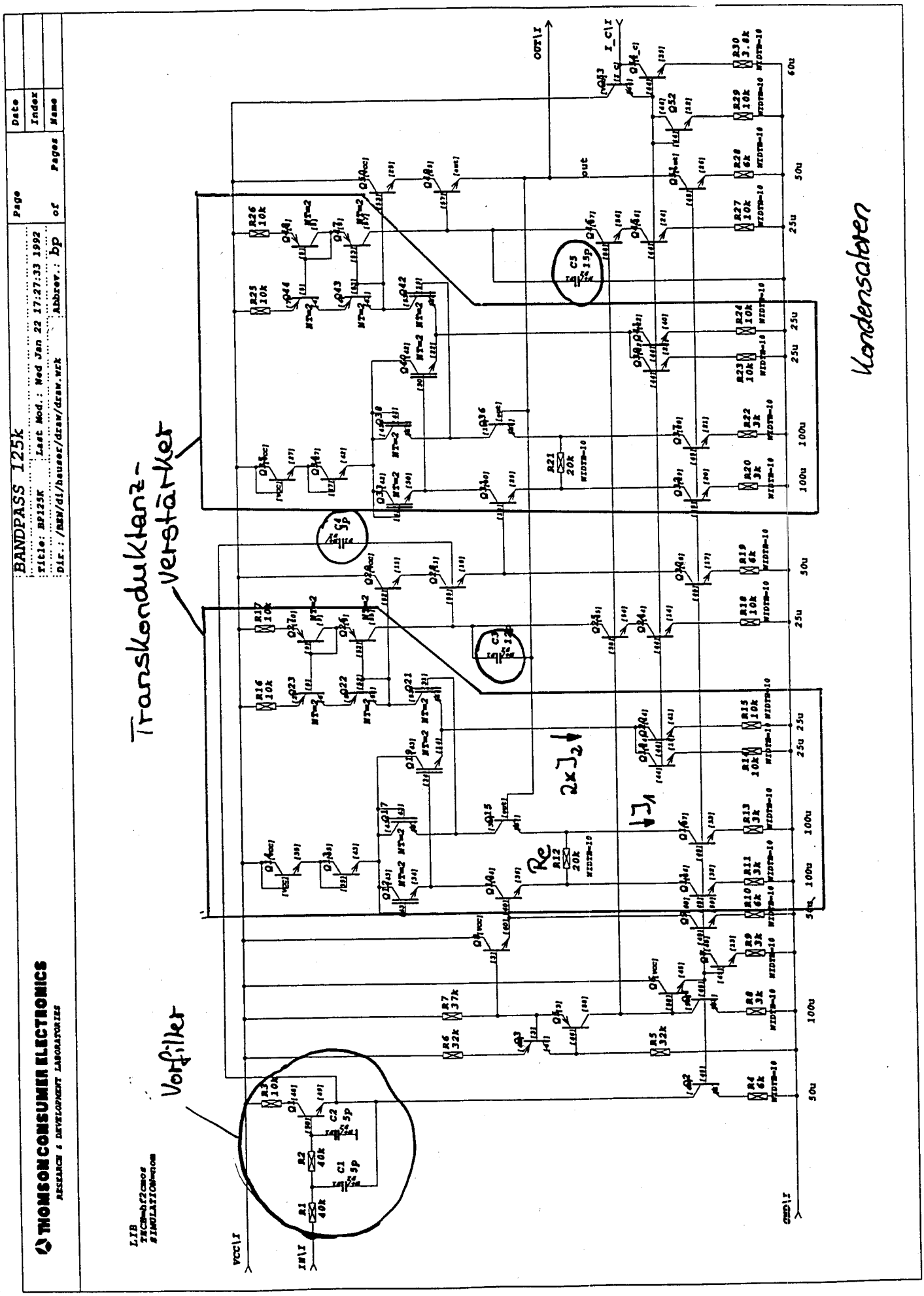


Bild 5: BP-Simulationsergebnis

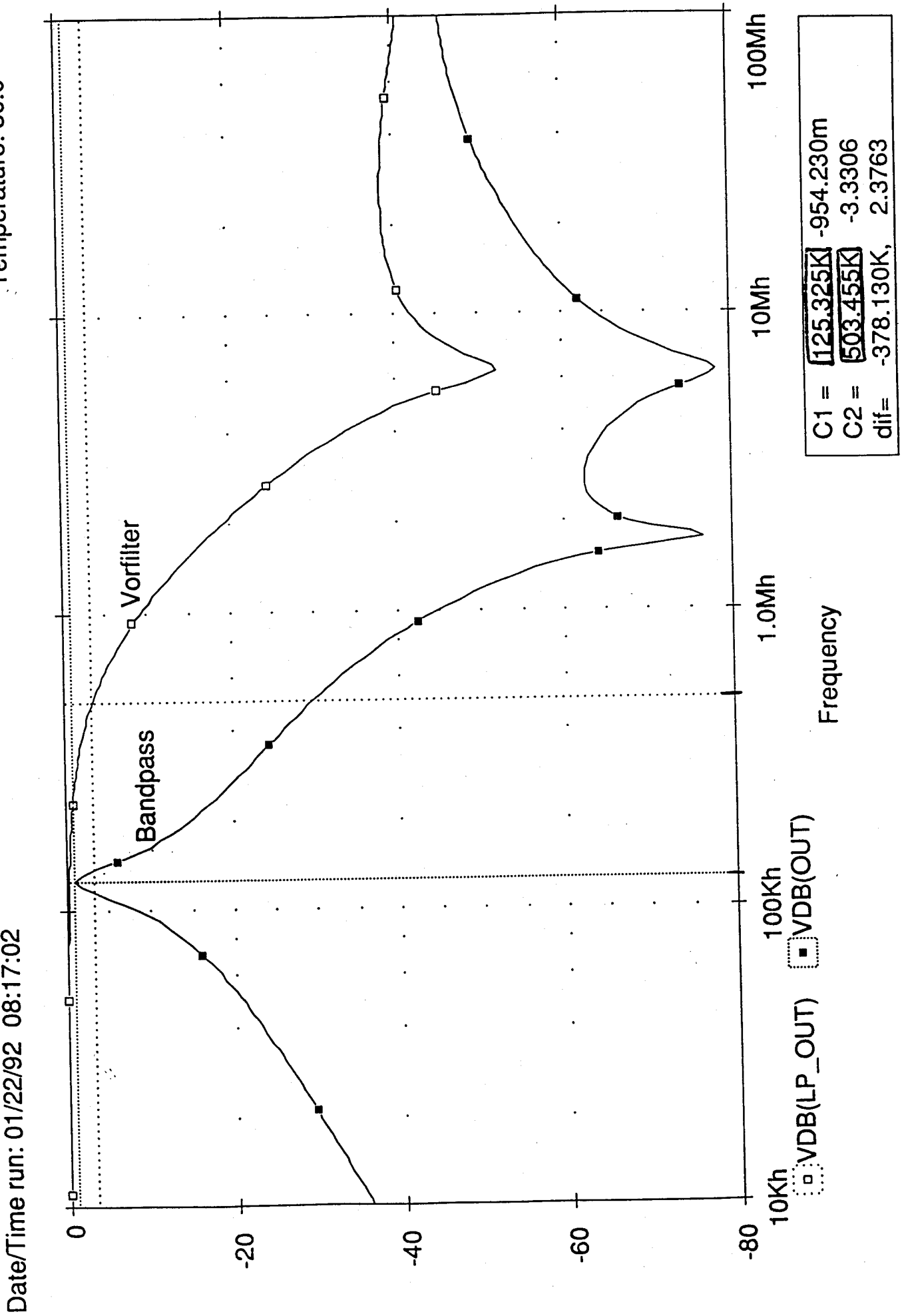
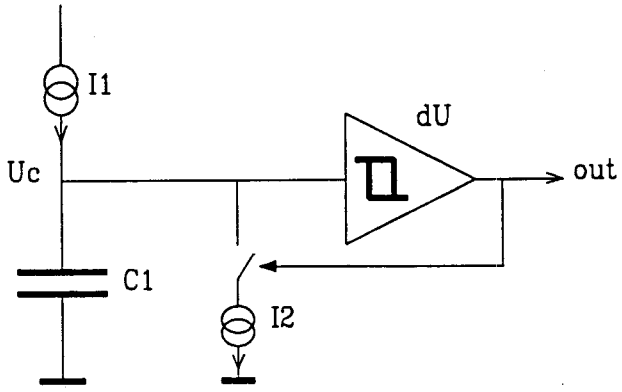
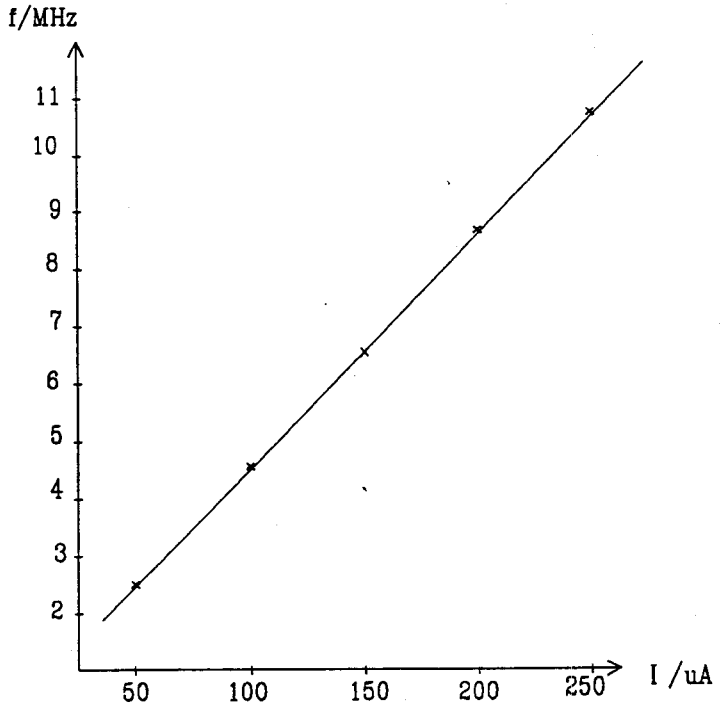


BILD 6: Relaxations-Oszillator und Kennlinie des VCO



$$T = \frac{dU*C1}{I1} + \frac{dU*C1}{I2-I1} = \frac{I2*dU+C1}{I1(I2-I1)}$$



Abhängigkeit der Oszillations-Frequenz des VCO von dem Eingangsstrom

Bild 7: VCO-Logik-Zeichnung

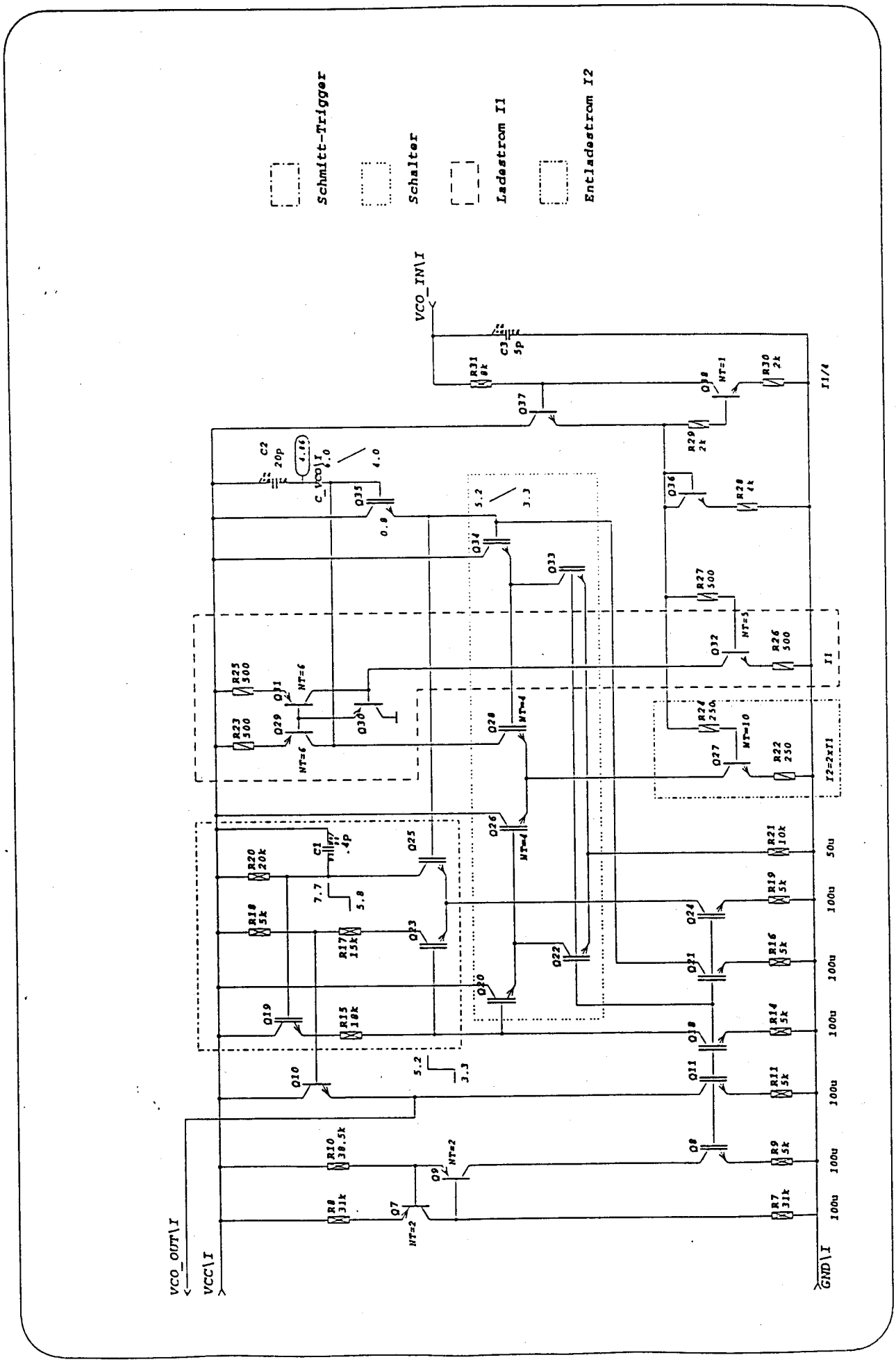
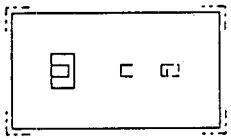
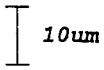


Bild 8: Transistoren im symbolischen Layout

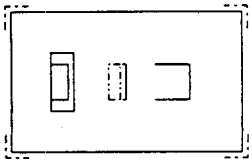
NPN

Emitterflaeche 5*5
Anschluesse C E B

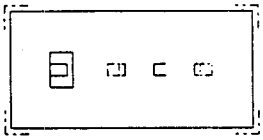




Emitterflaeche 5*5
Anschluesse C B E



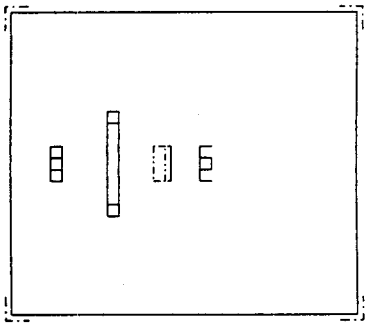
Emitterflaeche 5*5
Anschluesse C B E B



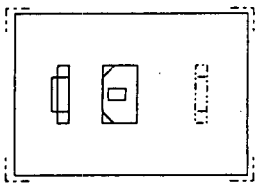
PNP

Isolierter Vertikaler PNP

Anschluesse ISO C B E



Lateraler PNP
Anschluesse C E B



Lateraler PNP
Anschluesse CC E CC B

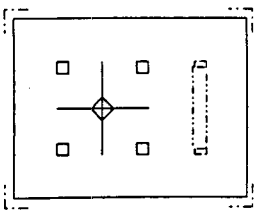


Bild 9: BP-Logik-Zeichnung

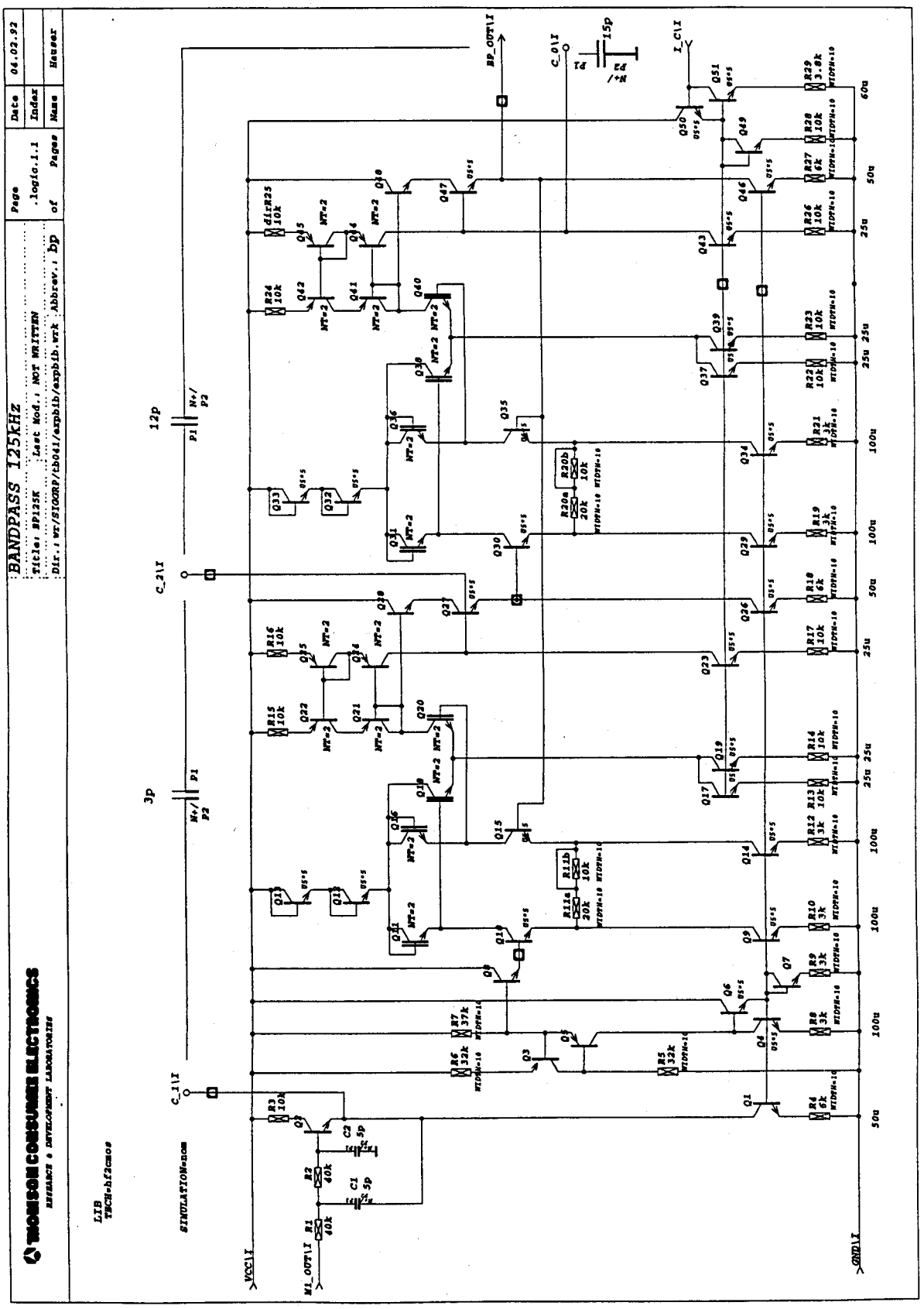


Bild 10: BP-symbolisches Layout

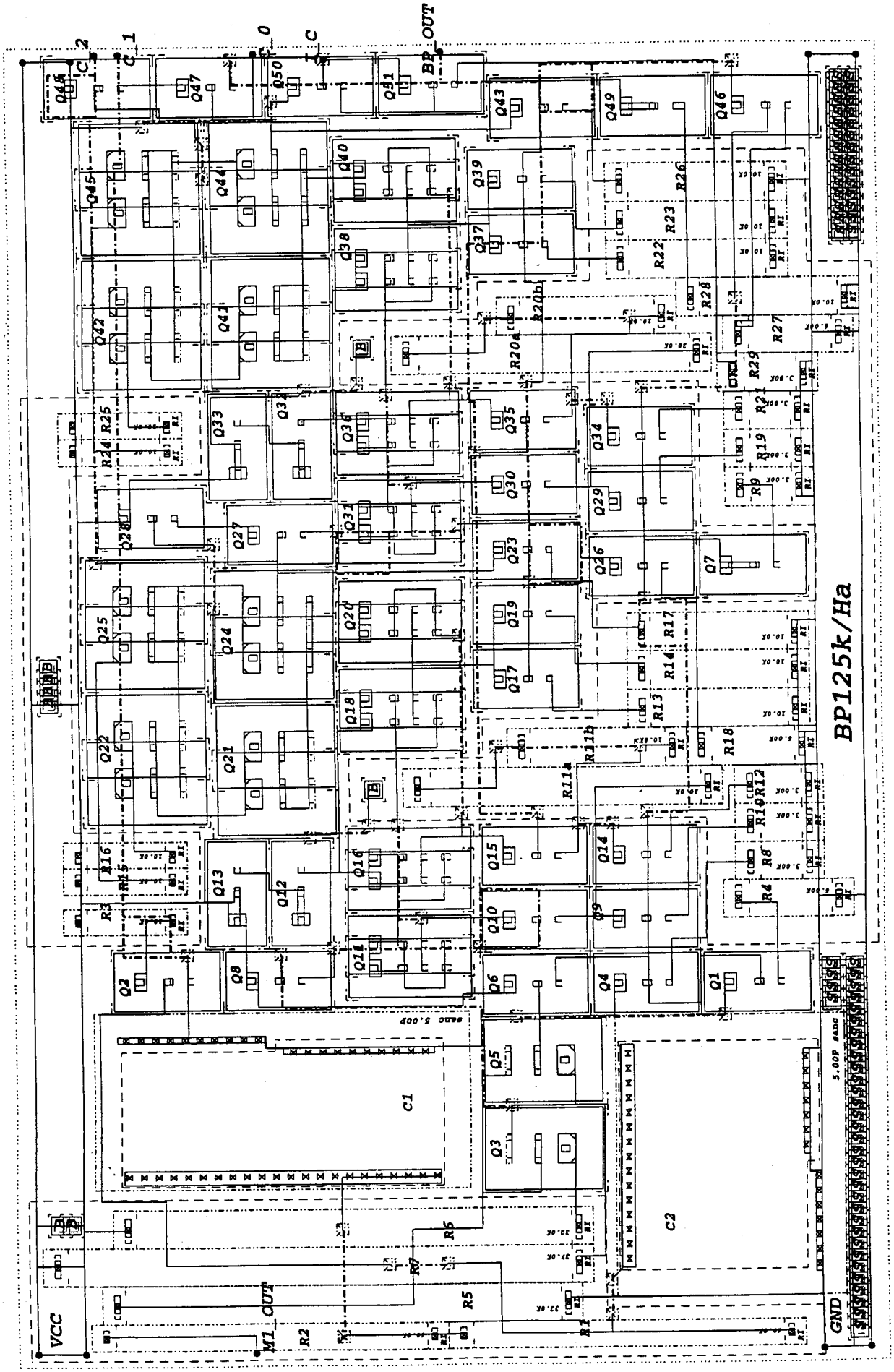


Bild 11: Gesamt-Logik-Zeichnung

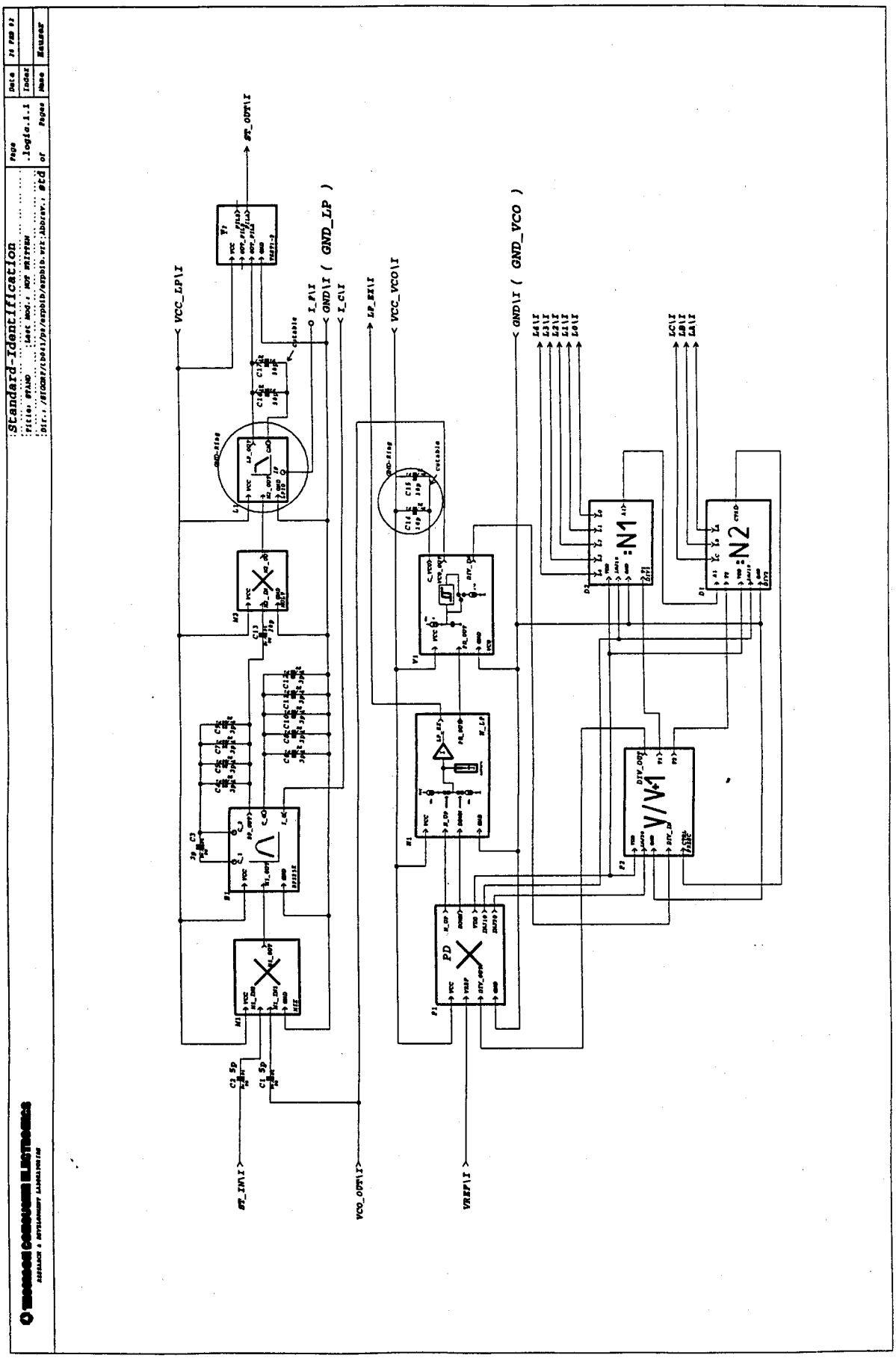
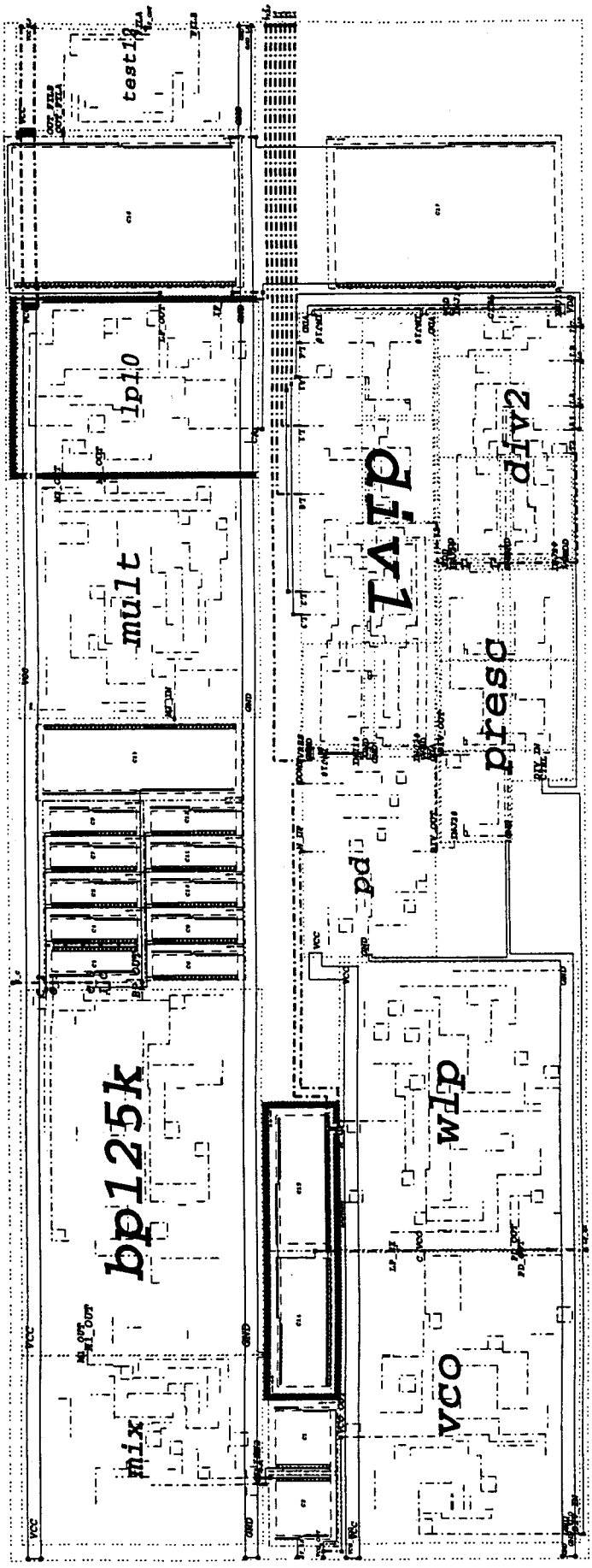


Bild 12:Gesamt-Layout



6. Entwurf und Test eines integrierten Kommunikationsmoduls Teil 1: Entwurf

K. Schmidt, B. Wichert
Fachhochschule Furtwangen

Abstrakt: Vorgestellt wird der Entwurf eines ASIC für einen hardwaregesteuerten Kommunikationsblock, der bei ES2 in 1,5μ-CMOS-Technologie gefertigt wurde. Die zugrundeliegende Buskommunikation arbeitet nach dem CSMA/CD-Verfahren.

1. Einleitung

Die integrierte Schaltung realisiert die Kommunikation von Interrupt-Frontend-Chips im Sicherheitseinsatz mit einer zentralen Steuereinheit, wobei das CSMA/CD-Verfahren angewendet wird (Carrier Sense Multiple Access/Collision Detect). Die Übertragung verwendet den Manchester-Code. Der Entwurf entstand im Rahmen einer Diplomarbeit. Der ASIC-Baustein wurde im Jahre 1991 über einen Herstellungslauf der 'EUROCHIP VLSI Design Action' bei ES2 in 1,5μ-CMOS-Technologie gefertigt (Run 13).

2. Systemanordnung

Bild 1 zeigt die Systemanordnung, in dem das Interrupt-Frontend-Chip (kurz mit IFE-Chip benannt) zum Einsatz kommen soll. Ein Vielzahl solcher Chips kommen in einem verteilten System vor, die alle über einen einzigen Bus mit einer zentralen Steuereinheit MAIN kommunizieren. Die Aufgabe des IFE-Chips ist das Notieren und die sofortige oder spätere Abgabe von Information über von ihm festgestellte Ereignisse.

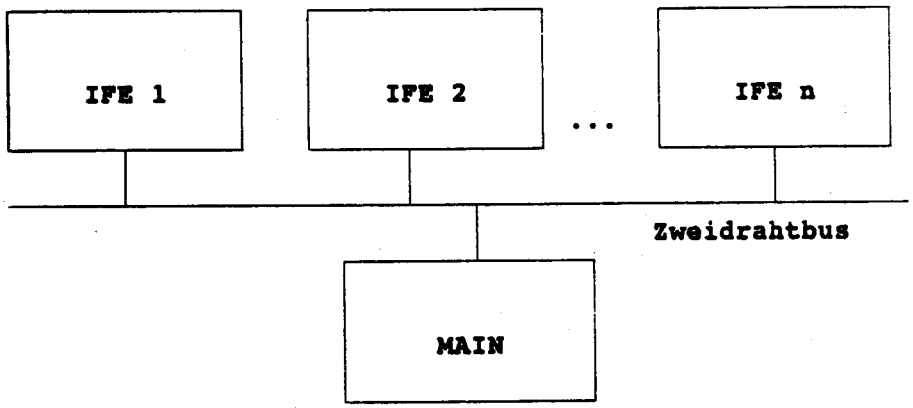


Bild 1 Systemanordnung

3. Übertragung

Die zugrundeliegende Buskommunikation arbeitet mit dem CSMA/CD-Verfahren nach IEEE-Standard 802.3. In Anlehnung an IEEE-Standard 802.4 wird vom IFE-Chip mit dem Manchester-Code wie folgt übertragen, Bild 2.

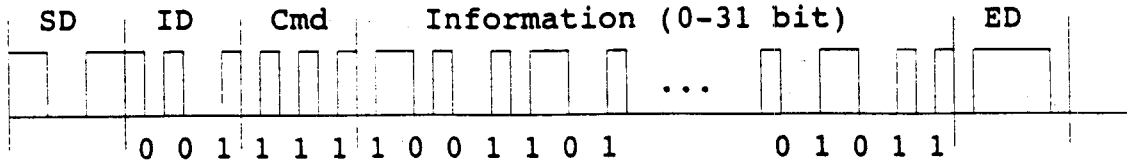


Bild 2 Beispiel für ein Übertragungsframe

Dem Startdelimiter SD folgt die Identifikationsnummer ID des Bausteins. Die Nachricht ist ein Befehl (Command Cmd) mit oder ohne Information. Das Frame schließt mit einem Enddelimiter ED. Der Ablauf zwischen den einzelnen IFE-Kommunikationsblocks und MAIN ist etwa der folgende:

1. IFE-Chip sendet "Mitteilung abrufbar"
2. MAIN quittiert (=vorgemerkt!)
3. MAIN kann andere Dinge noch erledigen
4. MAIN sendet Befehl "Aufforderung zur Mitteilung"
5. IFE-Chip quittiert (=sofort!)
6. IFE-Chip sendet "Mitteilung" gefolgt von der Information
7. MAIN quittiert (=korrekt erhalten!)

etc.

4. Schaltungsentwurf

Die Realisierung soll ausdrücklich nicht durch Software, sondern vollständig durch Hardware erfolgen. Die Steuerung muß das leisten.

Die Ansätze zur Schaltungsentwicklung liegen zunächst in den State-Diagrammen. Zum Beispiel zeigt Bild 3 das State-Diagramm für "Mitteilung abrufbar".

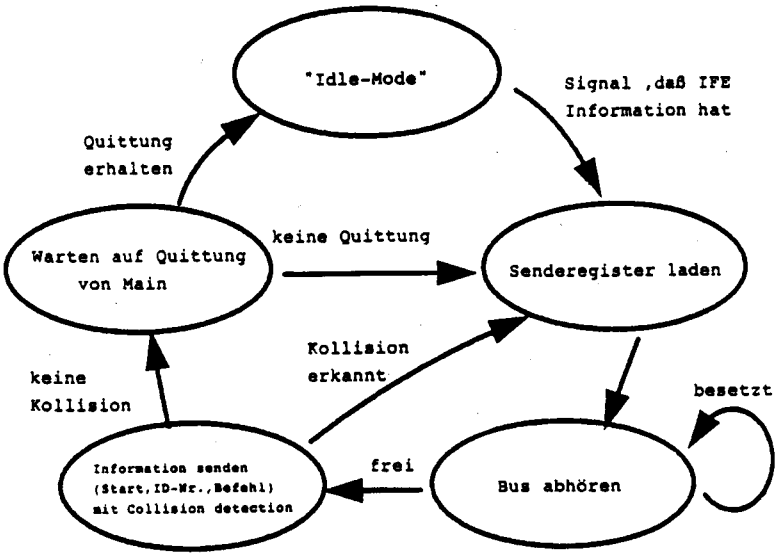


Bild 3 State-Diagramm für "Mitteilung abrufbar"

Es sind weitere State-Diagramme notwendig, die hier nicht dargestellt sind. Aus den State-Diagrammen kommt man mit einem Top-Down-Design folgerichtig zu einem Blockschaltbild des Kommunikationsmoduls, s. Bild 4.

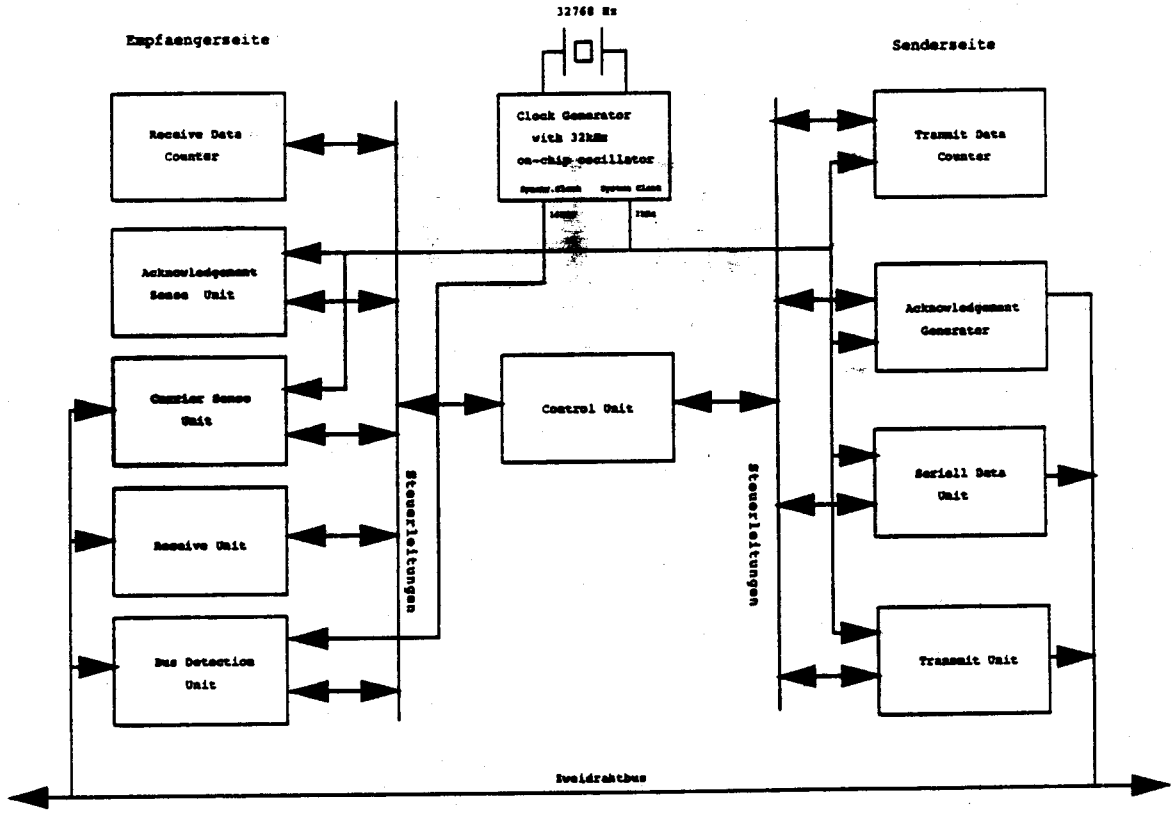


Bild 4 Blockschaltbild des Kommunikationsmoduls

Für die Funktionen der Blöcke lassen sich Hardwareflußdiagramme angeben, die hier nicht im einzelnen dargestellt sind. Diesem Entwurfsschritt folgt ein hierarchischer Aufbau des Designs aus Einzelmodulen (unterste Ebene die Zellen der ES2 Bibliothek) einschließlich des Einbaus der Testmöglichkeiten (Scan-Path-Verfahren), s. Bild 5. Zum Entwurf wurde das System SOLO 1400 auf PC verwendet.

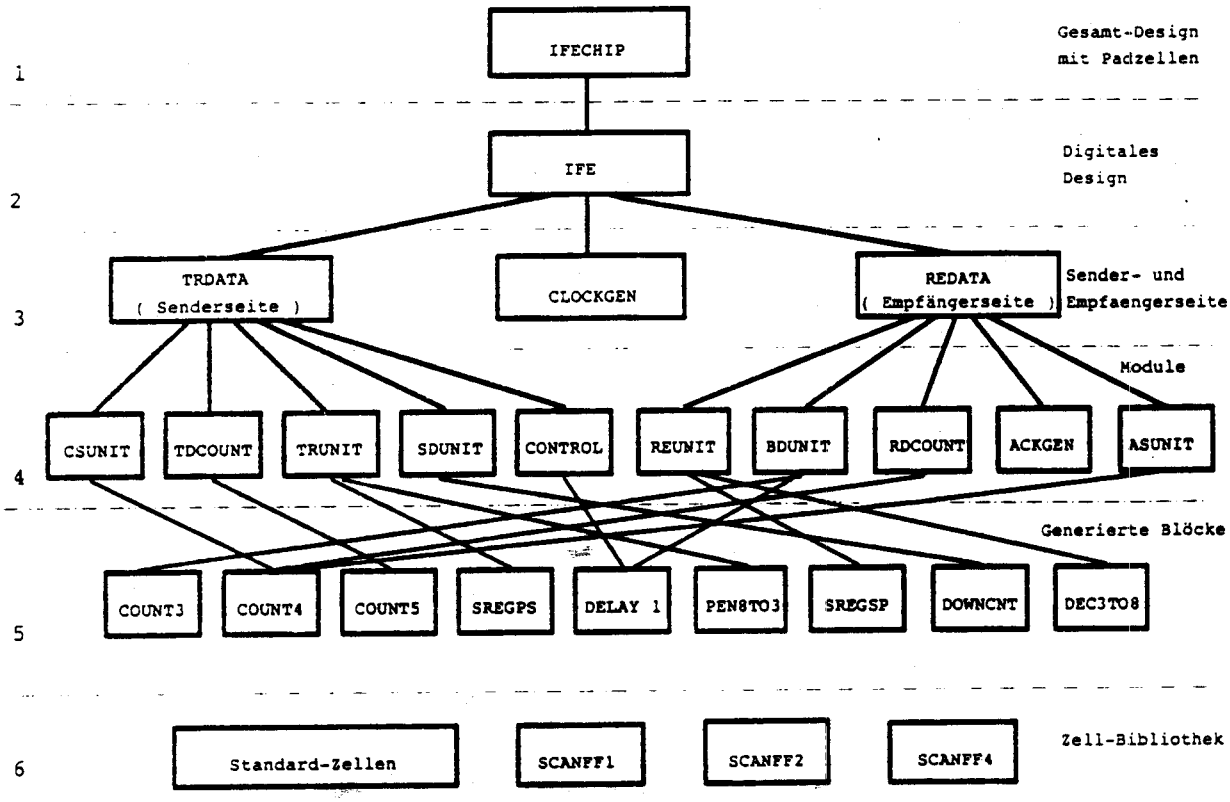


Bild 5 Hierarchische Struktur des Entwurfs

Jedes Macro des Entwurfs wird mit Schematics dargestellt und wird in allen Funktionen simuliert, um sowohl Funktion und Zeitablauf korrekt sicherzustellen.

Dem Zusammenfügen zum Gesamtsystem mit einer Pinbelegung für den geplanten Chip, Bild 6, folgt das automatische Placement und schließlich die Abgabe der Fertigungsdaten.

Der Entwurf beinhaltet 11250 Transistoren (das ist eine FH-Furtwangen-Premiere, über 10^4 Transistoren/Chip !). Die Chipfläche beträgt 17 mm^2. Die Entwurfszeit belief sich auf 345 Stunden.

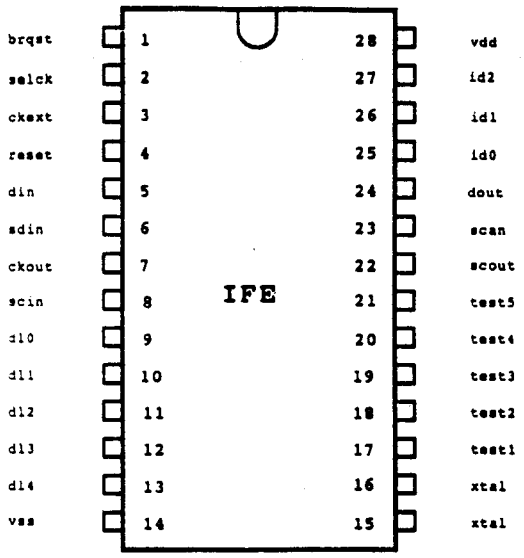


Bild 6 Pinbelegung des ASIC Kommunikationsmoduls

5. Fertigung

Die Turn-around-Zeit betrug etwa 20 Wochen. Nachstehende Fotos zeigen einen der zehn gefertigten Bausteine, Bild 7, ein Mikrofoto des Chips, Bild 8, ein Mikrofoto mit einem Chipausschnitt, Bild 9.

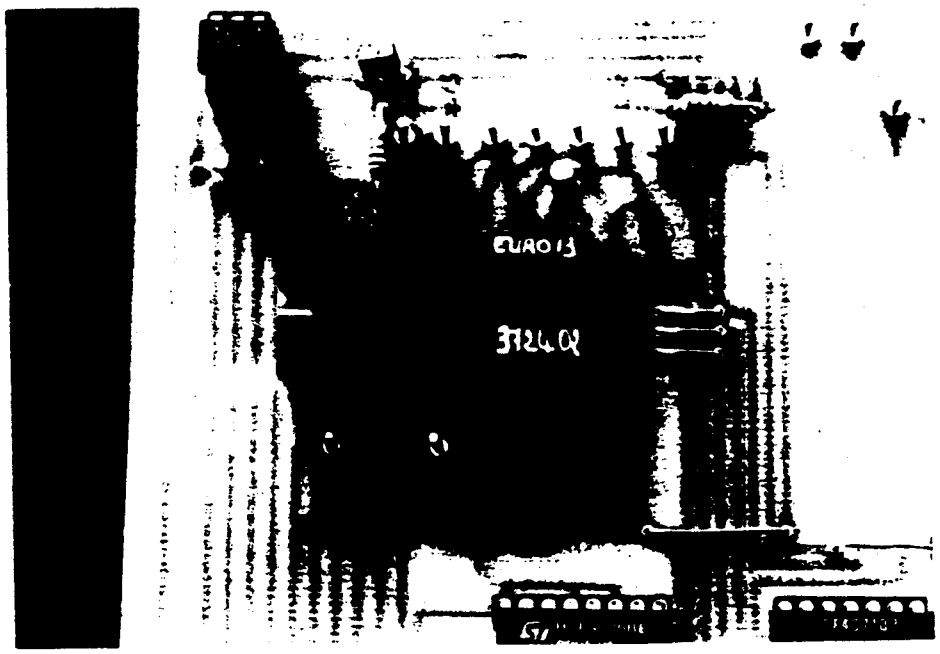


Bild 7 Einer der gefertigten Bausteine EURO13 3724-02 ifechip

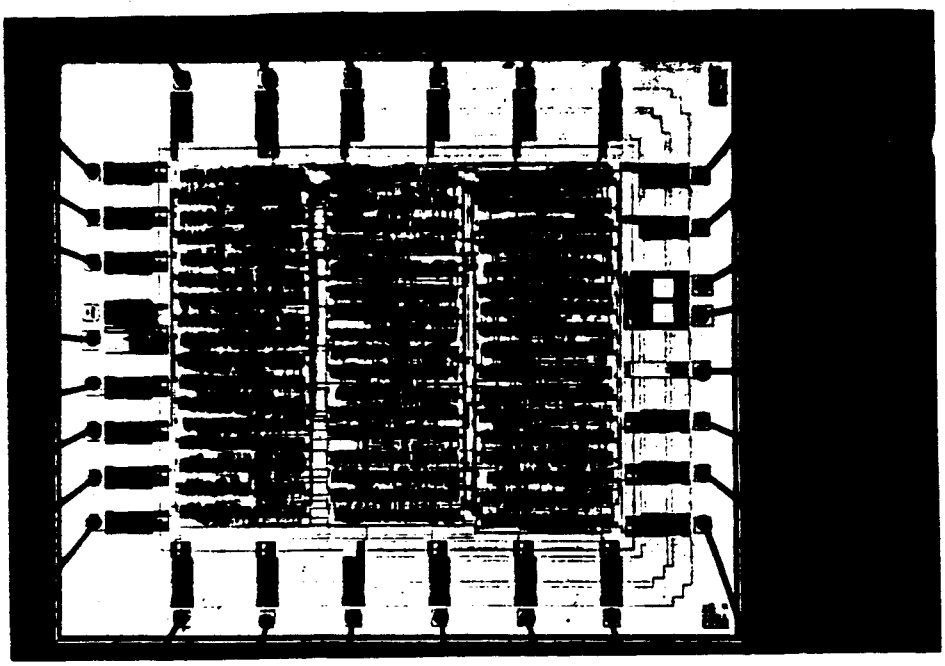


Bild 8 Mikrofoto des Chips EURO13 3724-02 ifechip

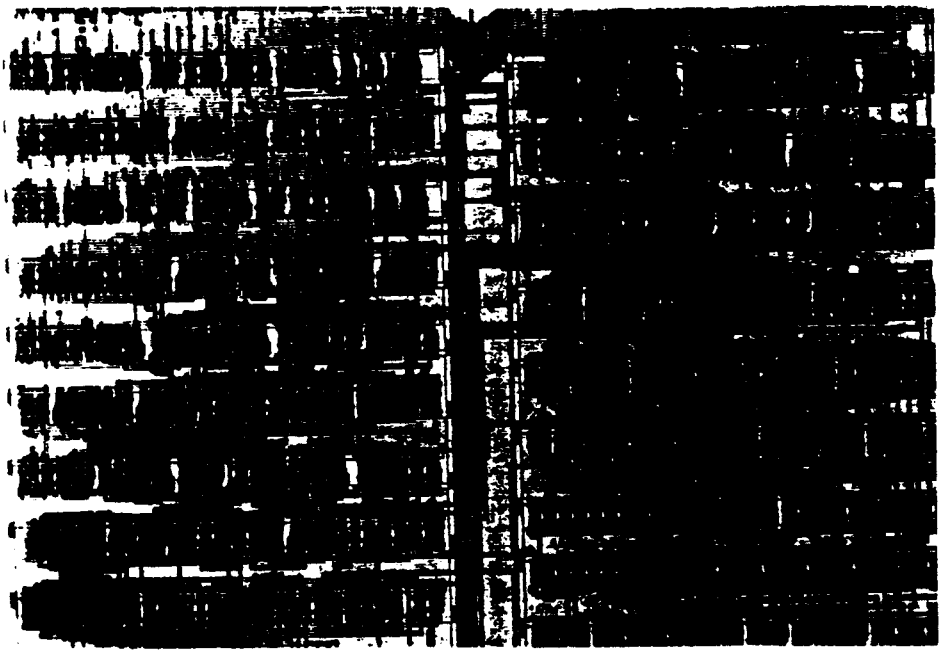


Bild 9 Mikrofoto mit einem Ausschnitt des Chips EURO13 3724-02 ifechip

6. Zusammenfassung

Beschrieben wurde der Entwurf eines ASIC für ein Kommunikationsmodul von ziemlich umfangreicher Schaltungsgröße (>10^4Transistoren). Die Fertigung erfolgte über die 'EUROCHIP VLSI Design Action'.

Literatur

Schmidt K. Entwurf digitaler Systeme, Kohlhammer Verlag

7 Entwurf und Test eines integrierten Kommunikationsmoduls, Teil 2: Test

A. Gauckler, R. Glatthaar, K. Schmidt
Labor für IC-Entwurf, Gerwigstr.11, 7743 Furtwangen

Die Ergebnisse des Tests am 1.5µ CMOS Chip "EURO13 3724.02" werden vorgestellt. Acht von zehn Chips erfüllen die Tests. Zwei Chips zeigen ein fehlerhaftes Verhalten und ihre Stromaufnahme ist etwa 15mal größer im Vergleich mit den fehlerfreien Chips.

Einleitung: Heute weitgehend übliche Abkommen zwischen Herstellern integrierter Schaltungen und deren Designern bzw. Auftraggebern sehen umfangreiche Tests bereits beim Hersteller vor. Für den Hersteller ist dies vorteilhaft, da er damit die Funktion der ausgelieferten Chips garantieren kann und somit keine Information über Ausbeute und Prozessführung nach außen dringt. Für den Auftraggeber ist ein solches Abkommen vorteilhaft, da funktionsfähige Chips garantiert und umfangreiche Tests und somit teueres Testequipment nicht notwendig sind. Ein Nachteil ist jedoch in den höheren (Chip-) Kosten zu sehen. Ein weiterer Nachteil kann sich für den Designer ergeben, denn er muß Testvektoren mit ausreichender Fehleraufdeckung generieren. Dies sollte beim Entwurf immer obligatorischer Bestandteil sein, sodaß dies als geringer Nachteil zu bewerten ist. Im Gegenteil, diese Notwendigkeit stellt eine gute Ausgangsbasis für den Test dar. Das Abkommen für die Fertigung des Chips im Rahmen von EUROCHIP sieht die Auslieferung von 10 ungetesteten Chips vor. Die Testergebnisse der gefertigten Chips sind somit von allgemeinem Interesse, da hier deutlich wird mit welcher Ausbeute bei ungetesteten Chips zu rechnen ist.
Im Anschluß an diese Einleitung wird der Test des Chips "EURO13 3724.02" vorgestellt. Dieser Baustein wird im Weiteren auch als Interrupt Frontend Controller (IFE) bezeichnet. Die Funktion des Bausteins ist bereits im ersten Teil des Berichts [1] beschrieben worden und kann dort nachgelesen werden. Hier wird nur darauf eingegangen, wenn es zum Verständnis notwendig erscheint.
Beim Prüfen der Chips wird nach einer Testcheckliste vorgegangen, die mit dem Numerieren der Chips beginnt. Dann werden die Chips unter dem Lichtmikroskop optisch inspiziert, die Bonddrähte und das Pad-Pin Assignment überprüft.
Im Anschluß folgt die Strommessung. Diese umfaßt sowohl die Messung der Stromaufnahme des gesamten Chips als auch die Eingangsstrommessung der Input-Pins. Dieser Test hat hohe Aussagekraft für den Gesamttest und wird deshalb in Kapitel 1 genauer dargestellt. Da bereits beim Entwurf der Test mit berücksichtigt wurde (design for testability) wird kurz auf die Erfahrungen mit dem SCAN-PATH Design [2] eingegangen. Der Scan-Path Design und Teile des Logiktest wurden mit dem Logic Analyzer DAS9200 und dem Test-Verifications System hp81810S durchgeführt. Zur Überprüfung des funktionellen Verhaltens wurden Testplatinen entworfen, mit den Chips bestückt und im Verbund mit einem Mikroprozessor als System getestet. Dieser Test ist in Kapitel 3 erläutet.

1) **Strommessung:** Die Eingangsstrommessung der Input Pins kontrolliert die elektrischen Verbindungen zwischen PIN und PAD. Diese und weitere Messungen

werden auf dem hp81810S durchgeführt. Für die Eingangsbuffer ohne Pull Up Widerstand waren die Ströme kleiner 4nA und somit innerhalb der Spezifikation (I_{in} < |10μA| bei V_{DD} bzw. V_{SS}). Zusätzlich wurden mit Vin=-0.5V Messungen die Eingangsschutzstrukturen aktiviert und gemessen (I_{in}≈40μA). Bei den Clock-Inputs und den Pull-Up Input Buffern (Rpu=125kΩ) wurden deutlich höhere Ströme gemessen. Insgesamt waren alle Werte innerhalb der Spezifikation und alle Chips passierten diesen Test erfolgreich.

Die nominale Stromaufnahme der Chips bei V_{DD}=5V Versorgungsspannung ist I_{nom}=21μA. Die Messungen der Versorgungs-Ströme sind in Tabelle 1 aufgelistet. Die mittlere Stromaufnahme ohne die Ausreißer Chip #3 bzw. #6 ist 19.7μA und entspricht in etwa der Spezifikation I_{nom}=21μA.
Bei den Chips #3 bzw. #6 wurden etwa 15fach höhere Ströme gemessen. Dies weist auf eine irreguläre niederohmige Verbindung zur Versorgungsspannung hin.

Chip #	Current
1	19.8μA
2	19.8μA
3	385.0μA
4	19.8μA
5	19.3μA
6	248.0μA
7	19.5μA
8	19.9μA
9	19.8μA
10	19.8μA
Mittelwert (ohne #3,6)	19.7μA

Tab.1: Stromaufnahme der IFE-Chips

2) SCAN-PATH Test: Alle Flip-Flops des Designs sind als Scan Flip-Flops ausgelegt (s.Bild 1) und können somit zu einem SCAN-PATH zusammengeschaltet

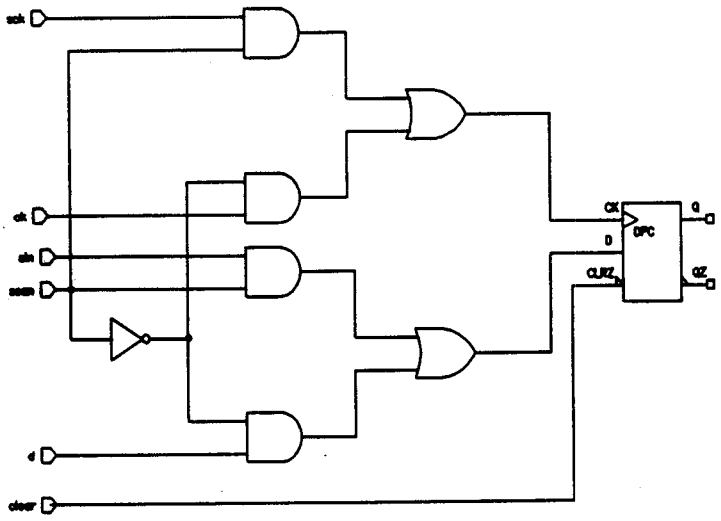


Bild 1: Scan-Path Zelle

werden. Dies erleichtert den Test der Schaltung. Zähler bzw. Teiler können schneller getestet werden und Testmuster werden über den SCAN-PATH an kom-

binatorische Logik herangeführt. Der SCAN-PATH umfaßt bei diesem Entwurf in etwa die Hälfte aller logischen Gatter (Gatecount) des Designs und hat eine Tiefe von 98 Stufen. Es wurden verschiedene Testmuster in den SCAN-PATH über den Patterngenerator des DAS9200 eingebracht und die Reaktion am SCAN OUT des SCAN-PATH beobachtet. Diesen Test durchliefen alle Chips erfolgreich. Zusätzlich wurde die Clockfrequenz variiert um dynamische Eigenschaften zu testen. Der SCAN-PATH Test funktioniert bei 2MHz noch fehlerfrei. Da der Chip bei 32kHz zum Einsatz kommt wurde auf weitergehende dynamische Tests verzichtet.

3) **Der Systemtest:** Für den Funktionstest bzw. Systemtest wurden mehrere identische Testplatinen entworfen und mit den IFE-Chips bestückt. Über DIP-Schalter können Identificationsnummer und Datenlängen eingestelllt werden. Zwei Taster ermöglichen die Auslösung von Busrequest (BRQST) bzw. RESET. Die Taktfrequenz wird wahlweise aus einem 32kHz-Quarz abgeleitet oder von außen zugeführt. Die Ausgabe von Daten ins Netz erfolgt über einen Optokoppler, der Data Input Pin des Chips ist direkt mit dem Zweidraht-Bus verbunden.

Für den Systemtest werden mehrer dieser Platinen über einen Zweidrahtbus mit einem MAIN Controller verbunden (80C166 Prozessor System). Das Prinzip des Systemtests ist in Bild 2 dargestellt. Die Testplatinen werden mit Power-Up-

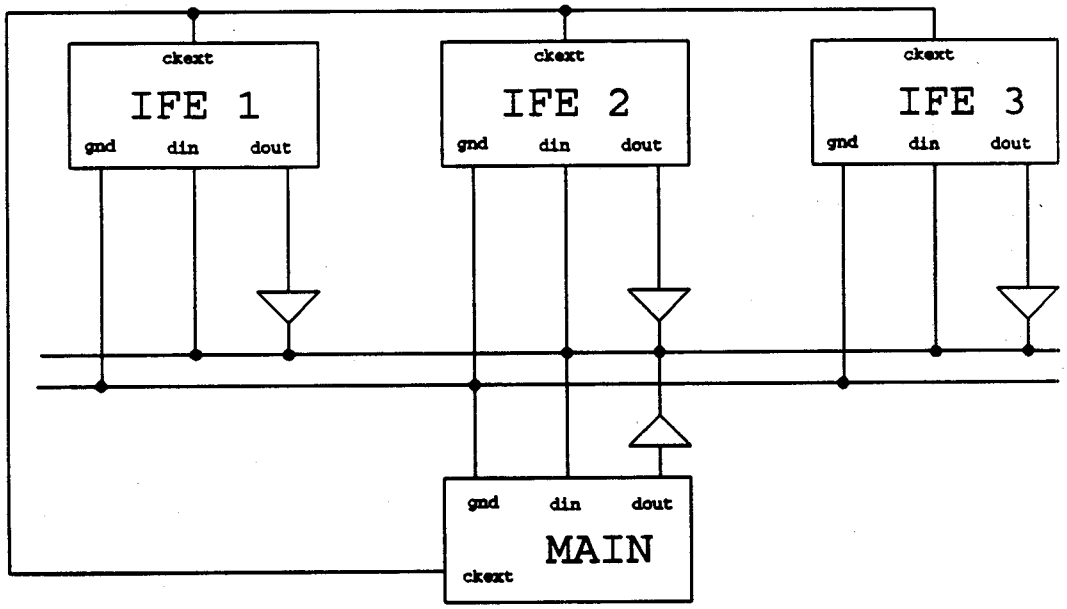


Bild 2: Systemtest

Reset bzw. Reset in einen definierten Zustand versetzt. Dann wird über Bus Request ein Übertragungswunsch einer Karte ausgelöst. Ist der Bus frei, wird mit dem Bus Protokoll begonnen.

Die mit einem Speicheroszilloskop protokollierte Datenübertragung wird in Bild 3 dargestellt. Das Protokoll beginnt mit dem Start Delimiter (SD), der Identifications Nummer (ID) und Command Code (CC). Der SD besteht aus einer logischen 110011 Folge. Die ID-Nummer ist in einer dem Manchester Code ähnlicher Verschlüsselung verpackten Datenfolge abgelegt (Teilnehmer Nr.7 ist durch logisch 01 01 01 codiert). Der Command Code 01 01 10 entspricht dem Befehl #6 und bedeutet "Mitteilung abrufbar".

Das vollständige Busprotokoll mit Mitteilungsaufforderung, Datenübertragung, Quittierung und Simulation von Buskonflikten ist wesentlich umfangreicher als

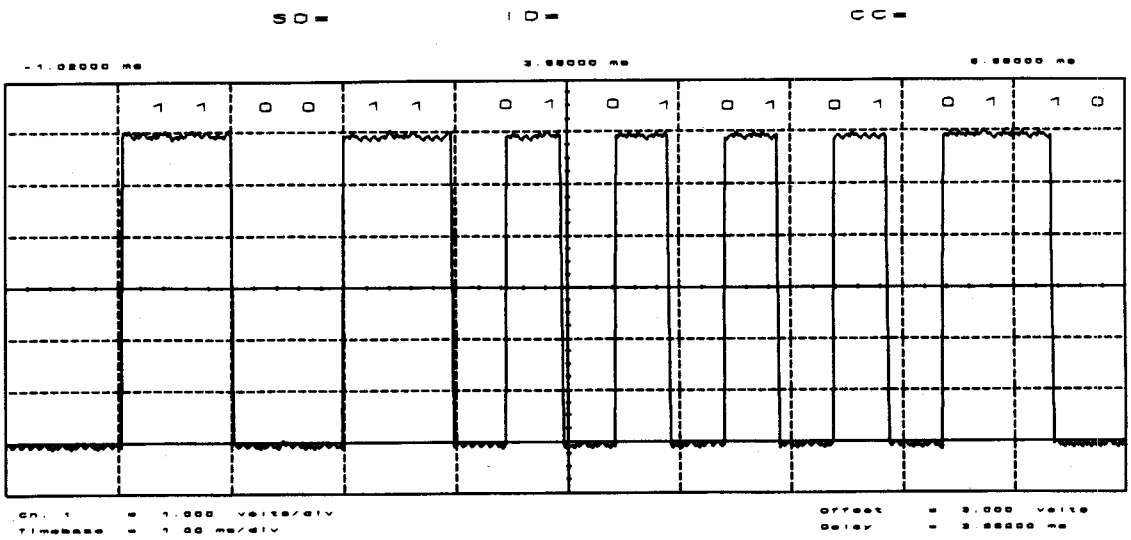


Bild 3: Fehlerfreie Datenübertragung

hier dargestellt und wird vollständig im Systemtest überprüft. Hierbei wurden bei zwei Chips Fehler festgestellt.

Bild 4 zeigt einen Fehler beim Command Code "Mitteilung abrufbar" der bei Chip #6 auftrat. Anstatt einer CC=01 01 10 für den Befehl Mittteilung abrufbar, wird CC = 00 01 10 übertragen. Dies ist ein ungültiger Manchester Code.

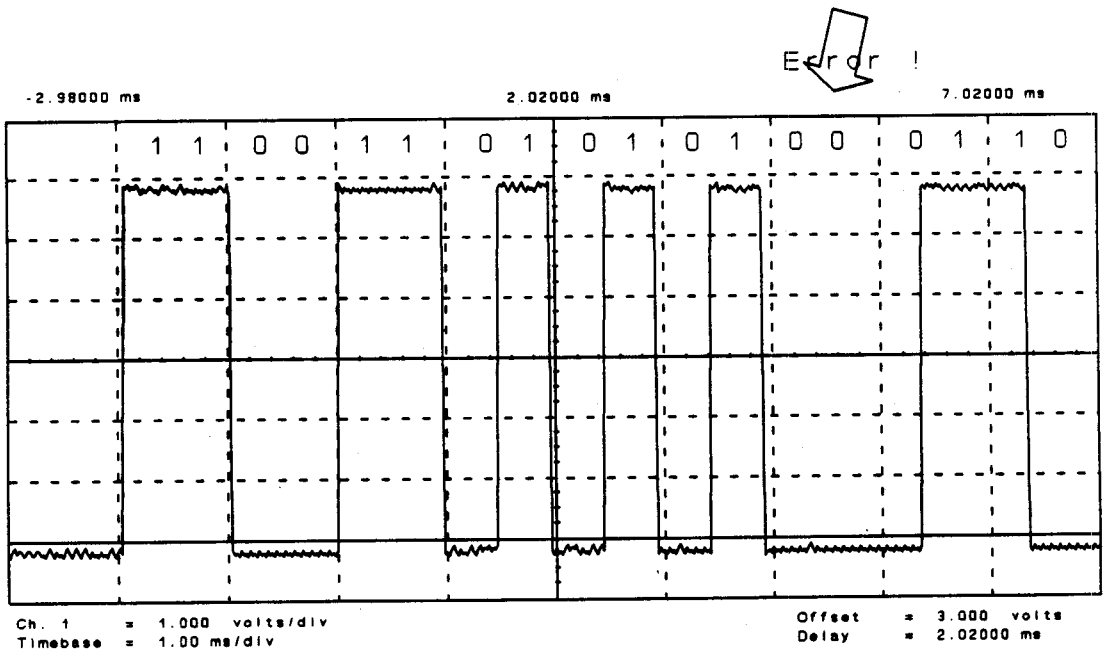


Bild 4: Fehler bei Chip #6

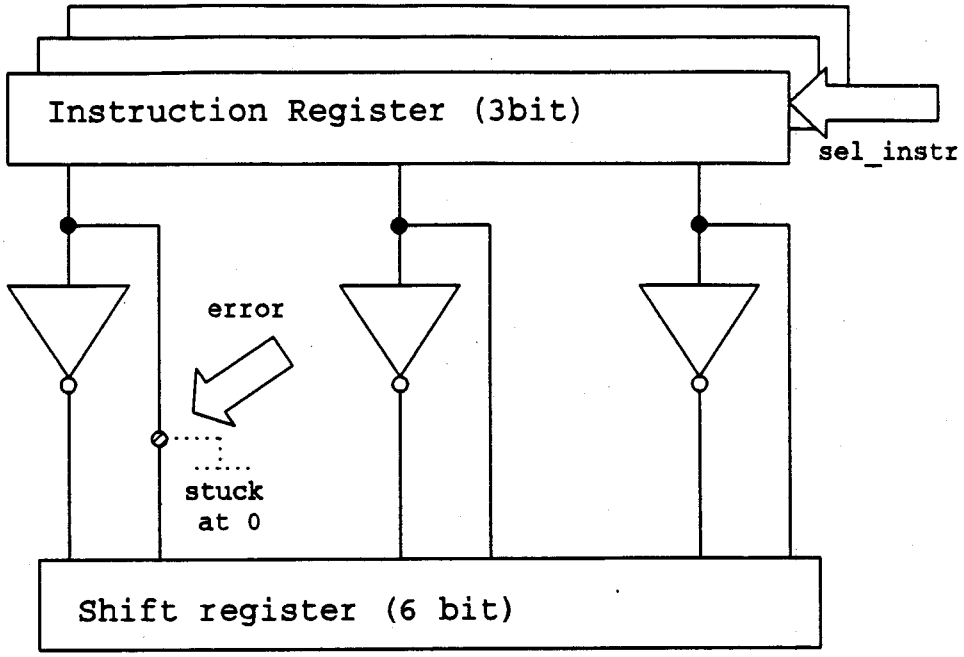


Bild 5: Stuck at Fehler Modell

Der Command Code wird entsprechend Bild 5 aus einem Register ausgewählt und über Inverter in den Manchester Code aufbereitet. Der Fehler im Manchester Code läßt einen Stuck at-Fehler vermuten. Diese Vermutung wird auch durch die bei Chip #6 gemessene hohe Stromaufnahe gestützt.

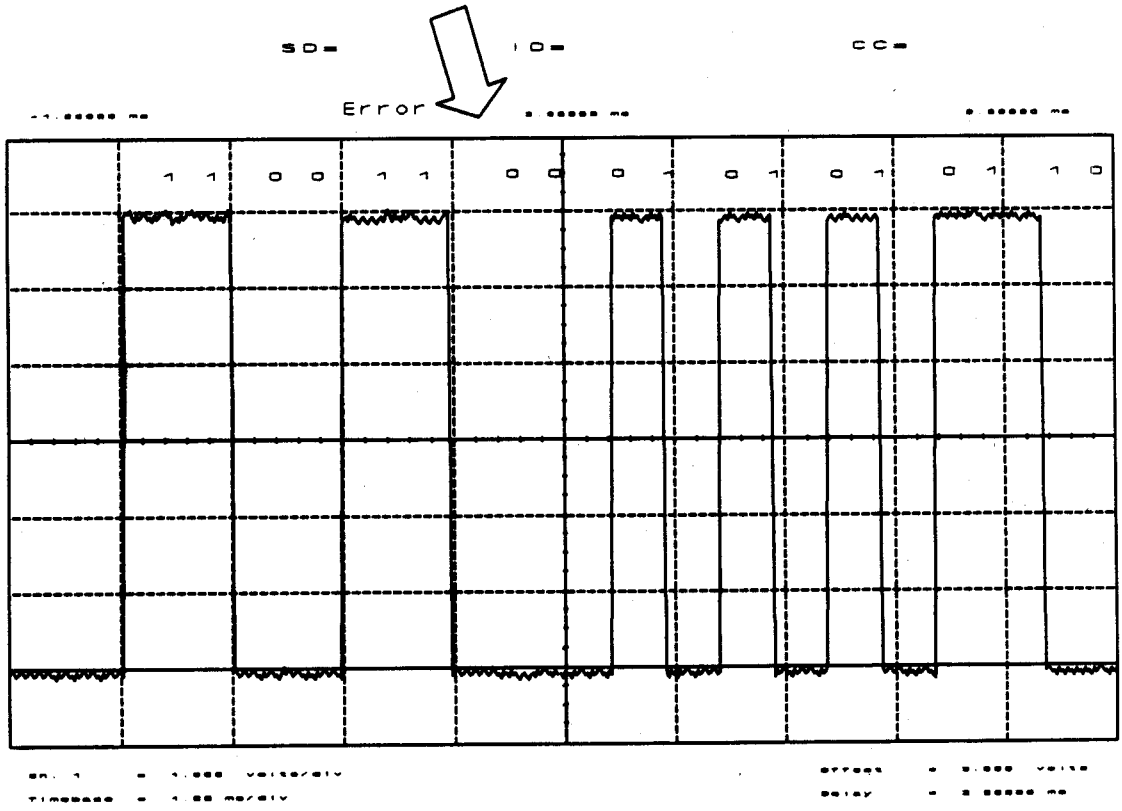


Bild 6: Fehler bei Chip #3

Ein weiterer ähnlich gelagerter Fehler (s. Bild 6) trat bei Chip #3 auf. Tests mit der ID=1 zeigen kein fehlerhaftes Verhalten im Systemtest. Die hohe Stromaufnahme jedoch weist auf einen Fehler hin. Nachdem der ID-Code des 1.ten Bits auf logisch Null gesetzt wird, werden Fehler beobachtet. Anstatt ID= 10 xx xx wird das Datum ID= 00 xx xx im Kommunikationsprotokoll ausgegeben (xx bedeutet Wert ohne Bedeutung). Auch hier ist zu vermuten, daß bei der Generierung des ID-Codes über die Inverterstruktur beim ersten Bit des ersten Codebits ein Stuck at Fehler vorliegt. Bei ID's zwischen 4 und 7 wird korrekt ID=01 xx xx erzeugt und der Systemtest findet keinen Fehler.
Eine genaue Analyse der Fehler wurde bisher noch nicht vorgenommen. Jedoch ist geplant mittels Tests mit Flüssigkristallen die Stelle der Stuck at Fehler zu lokalisieren, da durch Temperaturunterschiede Hot Spots durch Helligkeitsunterschiede zu beobachten sind.

Zusammenfassung und Ausblick: Von den 10 ausgelieferten Chips EURO-13 3724.02 funktionieren 8 Chips, zwei haben fertigungsbedingte Fehler. Die Fehler können durch einfach Strommessungen beobachtet werden und werden durch den Systemtest bestätigt.
Es ist vorgesehen, die Fehler auf den Chips mittels Flüssigkristalltests zu lokalisieren und die Ursache zu dokumentieren.
Die Testbarkeit der Chips wurde durch den Einbau von Scan Path Flip-Flops deutlich erhöht und soll in Zukunft durch Boundary Scan Design noch weiter verbessert werden.

Literatur

[1] K.H.Schmidt, B.Wichert, "Entwurf und Test eines integrierten Kommunikationsmoduls", MPC-Workshop 6.Juli 92
[2] F.Tsui, "LSI/VLSI Testabilty Design", Mc-Graw Hill,1987

Fachhochschule Heilbronn
Fachbereich Elektronik

8. Entwicklung eines hochintegrierten Bildverarbeitungs - Schaltkreises zur Echtzeit - Schwerpunktberechnung

Vogel H., Jäger U., Clauss H.

1. Einleitung

Die Bildverarbeitung nimmt neutzutage einen immer wichtigeren Stellenwert in den Bereichen Steuerung, Regelung, Qualitätssicherung, Überwachung, Robotertechnik, Automatisierung und in noch vielen weiteren Bereichen ein. Durch die große Datenmenge, die bei der Verarbeitung von Bilddaten auftritt, speziell wenn sie in Echtzeit bearbeitet werden sollen, sind normale Rechnersysteme meist überfordert. Zur Unterstützung der Computer gibt es eine Vielzahl zusätzlicher Hardware, die schnell einfache Datenbearbeitungen und Datentransfer durchführen kann und somit den Rechner soweit entlasten, daß dieser Zeit für die Auswertung der eingehenden Informationen hat. Beispiele für eine derartige Zusatzhardware sind unter anderem intelligente Framegrabber, Echtzeitfilter mittels Kernel, Kompensationsspeicher, Sequenzer, Bildmischer und vieles weitere. Je mehr rechenaufwendige Aufgaben man diesen Hardwareteilen überlassen kann, umso umfangreicher kann die Auswertung mittels des Rechners geschehen.

2. Aufgabenstellung

Will man mittels Bildverarbeitung bewegte Objekte verfolgen, so scheitert dies ab einer gewissen Objektgeschwindigkeit, wie oben bereits erwähnt, meist an der zur Verfügung stehenden Rechenleistung. Würde man zumindest die aktuelle Position des Objektes kennen, so könnte man dessen Lageänderung (z.B. Rotation) relativ zur vorhergehenden leicht ermitteln. Eine Positionsbestimmung des Objektes in Echtzeit überfordert jedoch, wie bereits erwähnt, die Rechenleistung eines durchschnittlichen Computers bei weitem. Die Verwendung einer zusätzlichen Hardware, die in Echtzeit den Flächenschwerpunkt des zu bearbeitenden Objektes bestimmen würde und man somit seine aktuelle Position kennen würde, könnte den Rechner soweit entlasten, daß dieser einmal die Lage des Objektes und ab dann nur noch die Lageänderung berechnen müßte. Da bis jetzt noch keine Hardware, die dieses leistet, bekannt ist, wurde im Rahmen einer Diplomarbeit ein derartiger Schaltkreis entwickelt.
An diesen Schaltkreis wurden folgende Anforderungen gestellt:

- Berechnung des Flächenschwerpunktes eines Objektes auf einem Videobild in Echtzeit
- Die Berechnung muß unabhängig von der Pixel - bzw. Taktfrequenz geschehen
- Die Berechnung muß unabhängig von der Bildgröße und des Bildformats geschehen
- Man sollte die Möglichkeit haben, die aktive Taktflanke wählen zu können, da bei manchen Systemen die Daten bei der positiven, bei manchen bei der negativen Taktflanke gültig sind
- Man sollte die Möglichkeit haben, die Hintergrundhelligkeit einstellen zu können
- Zur Anpassung an die Videodaten muß man einstellen können, ob es sich beim Eingangsbild um ein Interlace - Bild (Bild im Zeilensprungverfahren) handelt oder nicht

3. Prinzip der Flächenschwerpunktberechnung

Die Grundidee der Schwerpunktberechnung baut auf eine zweidimensionale Mittelwertbildung der Bildpunktkoordinaten des zu berechnenden Objektes, das durch ein Binärbild dargestellt wird, auf. Hierbei werden die Koordinaten der Bildpunkte, die das Objekt darstellen, in X- und Y- Richtung getrennt aufsummiert und nach Ablauf eines Bildes bzw. eines Halbbildes diese Summen jeweils durch die Anzahl betrachteter Bildpunkte dividiert, als Ergebnis erhält man die Flächenschwerpunktkoordinaten des dargestellten Objektes.

3.1. Mathematische Vorüberlegungen

Wie oben bereits erwähnt, geschieht die Berechnung der Schwerpunktkoordinaten mittels einer zweidimensionalen Mittelwertbildung der Objektkoordinaten, die in Gleichung (3.1) dargestellt ist.

$$KS_{(X,Y)} = \frac{\sum K_{(X,Y)}(N)}{\sum_{N \in P} N} \qquad (3.1)$$

Hierbei stehen die Variablen für folgende Werte:

N:	aktueller Bildpunkt
$K_{(X,Y)}(N)$:	Koordinaten des aktuellen Punktes (jeweils getrennt für X und Y)
$KS_{(X,Y)}$:	Koordinaten des Schwerpunktes (jeweils getrennt für X und Y)
P:	Menge aller Objektpunkte

Da es sich um ein Binärbild handeln muß, weil sonst die Schwerpunktbildung auf diese Weise nicht zu einem befriedigenden Ergebnis führen würde, muß man unterscheiden, ob ein weißes Objekt auf einem schwarzen Hintergrund, oder ein schwarzes Objekt auf einem weißen Hintergrund vorliegt. Zur Berechnung müssen die Koordinaten der Bildpunkte, die das Objekt darstellen, also diejenigen, die die Vordergrundhelligkeit besitzen, in X- und in Y- Richtung getrennt aufsummiert werden. Weiterhin muß die Gesamtzahl der Punkte mit Vordergrundhelligkeit bestimmt werden. Dies kann beim Addieren der Koordinaten geschehen, indem man einen Zähler bei jeder Addition erhöht.

Diese Berechnung ist im folgenden anhand eines Beispielobjektes, das in Bild 3.1 zu sehen ist, erläutert. Zur besseren Übersicht wurde hier nur ein sehr kleines Objekt verwendet und die Bildpunkte so dargestellt, daß sie deutlich einzeln erkennbar sind.

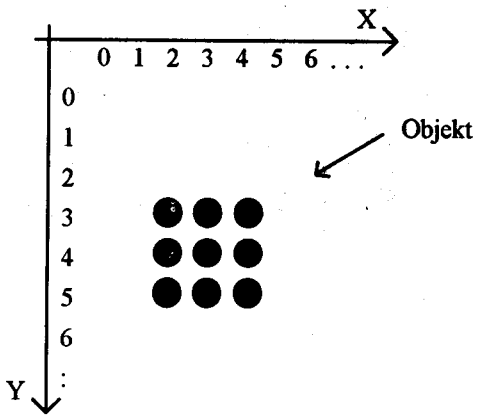


Bild 3.1: Beispielbild zur Schwerpunktberechnung

Die konkrete Berechnung der X - Koordinaten zeigt Gleichung (3.2), die der Y -Koordinaten ist in Gleichung (3.3) zu sehen. Mittels einer Division der Koordinatensummen durch die Gesamtzahl betrachteter Vordergrundbildpunkte erhält man die Koordinaten des Flächenschwerpunktes des dargestellten Objektes.

$$X-Koordinate: \quad X_S = \frac{2+3+4+2+3+4+2+3+4}{9} = 3 \qquad (3.2)$$

$$Y-Koordinate: \quad Y_S = \frac{3+3+3+4+4+4+5+5+5}{9} = 4 \qquad (3.3)$$

Das in Bild 3.1 gezeigt Beispielobjekt hat seinen Flächenschwerpunkt folglich bei X = 3 und Y = 4.

3.2. Anforderungen an die Rechengeschwindigkeit

Die Größe des Videobildes einer normalen CCD - Kamera beträgt 512 x 512 Bildpunkte bei einer Bildwiederholfrequenz von 60 Hz, was einer Pixelfrequenz von etwa 15 MHz entspricht. Zur Berechnung des Schwerpunktes müssen zuerst die Koordinaten des aktuellen Bildpunktes berechnet werden, da die Videodaten pixelseriell ankommen und nur durch die Synchronisationssignale ZIP (Zeilenimpuls) und BIP (Bildimpuls) wieder zu einem Bild rekonstruiert werden können. Die so berechneten Pixelkoordinaten müssen dann aufsummiert werden, falls der aktuelle Bildpunkt zum Vordergrundobjekt gehört, was parallel hierzu ermittelt werden muß. Weiterhin muß die Gesamtzahl der Vordergrundbildpunkte berechnet werden. Nach einem Bild bzw. einem Halbbild müssen die Koordinatensummen in X - und Y - Richtung getrennt jeweils durch die Gesamtzahl der Vordergrundpunkte dividiert werden. Man benötigt also, abgesehen von der Berechnung diverser Bedingungen, zwei Zählvorgänge (X - Koordinate und Gesamtzahl) und drei Additionen (Y - Koordinate sowie X - und Y - Koordinatensummen) pro Bildpunkt sowie zwei Divisionen (Koordinatensummen durch Gesamtzahl in X - und Y - Richtung) pro Bild bzw. pro Halbbild. Bei einer Pixelfrequenz von 15 MHz müssen somit etwa 75 Millionen Additionen und einige Divisionsschritte pro Sekunde durchgeführt werden. Die Verwendung eines Prozessors wird dadurch unmöglich. Bei Verwendung einer geeigneten Hardware, die die einzelnen Bildpunkte im Pipelineverfahren bearbeitet, kann aber die Berechnung des Schwerpunktes bei einer Taktfrequenz von 15 MHz durchaus erfolgen.

4. Architektur des Systems

Zur Umsetzung des Algorithmus zur Flächenschwerpunktbestimmung in Hardware sind einige mathematische Grundfunktionen nötig. Als wichtigste Grundfunktion werden hierbei Paralleladdierer benötigt, die innerhalb einer Taktperiode eine vollständige Addition der benötigten Breite durchführen können. Durch die Pipelineverarbeitung können zuerst die Koordinaten berechnet werden, die dann im nächsten Takt addiert werden, während die Koordinatenzähler bzw. - addierer bereits mit dem nächsten Bildpunkt beschäftigt sind. Die prinzipielle Architektur des Schwerpunktberechners ist in Bild 4.1 zu sehen. Die Berechnung der X - Koordinaten kann mittels eines Zählers geschehen, der bei jedem Bildpunkt erhöht wird und nach Ablauf jeder Zeile zurückzusetzen ist. Im Gegensatz hierzu wird bei den Y - Koordinaten ein Addierer benötigt, da abhängig von der Bildart (Interlace - oder NON - Interlace - Bild) die Koordinaten in Einer - bzw. Zweier - Schritten erhöht werden müssen. Die so berechneten Koordinaten werden, gehört der jeweilige Bildpunkt zum Objekt, in weiteren Addierern aufsummiert.

Nach Ablauf dieser Koordinatensummierung über ein Bild bzw. ein Halbbild, müssen die Koordinatensummen durch die Gesamtzahl der Vordergrundpunkte dividiert werden, wofür man jedoch fast die Zeit eines Bildes bzw. Halbbildes aufwenden kann. Dies ist bei einer Hardwarerealisierung auch nötig, da es eine Dividierschaltung nicht gibt und somit die Division algorithmisch ablaufen muß. Die Division wird nach Ende eines Bildes bzw. Halbbildes gestartet und läuft automatisch weiter, bis die Schwerpunktkoordinaten feststehen und zeigt dieses Ende der Berechnung auch selbständig an. Alle anderen Schaltungsteile werden, wie aus Bild 4.1 ersichtlich ist, durch eine Ablaufsteuerung beeinflußt, die den Ablauf der Berechnung an die Bilddaten anpaßt.

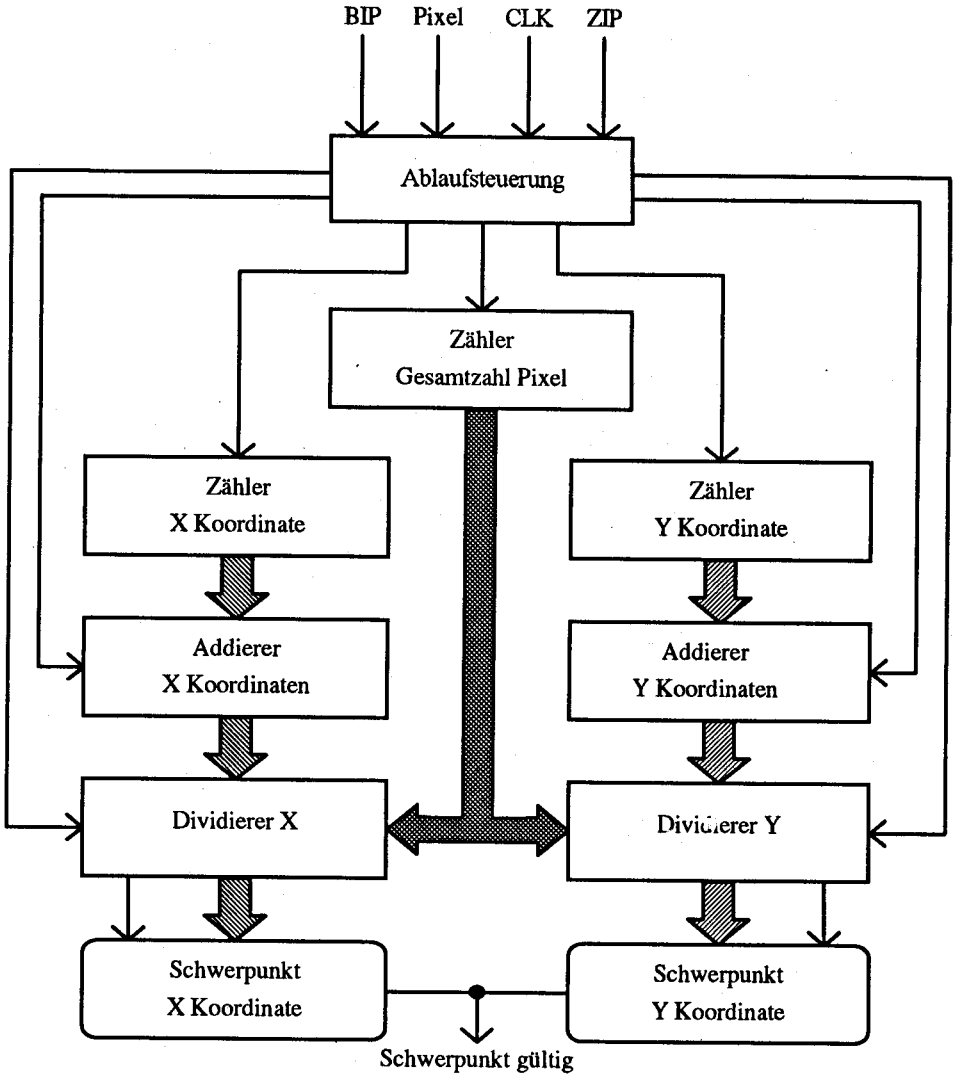


Bild 4.1: Architektur des Flächenschwerpunktberechners

Bisher wurden die Breiten der Datenbusse nicht beachtet, speziell bei der Summierung der Koordinaten kann dieses aber zu Problemen führen. Betrachtet man ein normales Interlace - Bild, so sind für die verschiedenen Busse die Breiten wie sie die Tabelle 4.1 zeigt nötigt bzw. wurden tatsächlich verwendet. Durch die hohe Anforderung an die Rechengeschwindigkeit

wurde ein schneller 6 Bit Paralleladdierer entwickelt, der zur Realisierung aller Zähler und Addierer verwendet wurde. Demzufolge wurde versucht die Busbreiten in einer Breite, die ein Vielfaches von 6 ist, zu realisieren, was allerdings bei den Koordinatenaddierern nicht gelang, weshalb ein 1 Bit Addierer hinzugefügt werden mußte.

	geforderter Zahlenbereich	minimale Busbreite	realisierte Busbreite	erreichter Zahlenbereich
Koordinatenzähler	0 - 511	9 Bit	12 Bit	0 - 4095
Summenzähler	0 - 131071	17 Bit	18 Bit	0 - 262143
Koordinatensumme X	0 - 33488896	25 Bit	25 Bit	0 - 33554431
Koordinatensumme Y	0 - 33423360	25 Bit	25 Bit	0 - 33554431

Tabelle 4.1: Verwendete Busbreiten und erreichte Zahlenbereiche

Man sieht, daß außer bei den Koordinatensummen immer ein gewisser Sicherheitsbereich vorhanden ist. Speziell bei den Koordinatenzählern ist dies auch nötig, da jederzeit auch größere Bildformate zur Berechnung anstehen können. Man würde zwar in diesem Fall bei Objekten, die nahezu den ganzen Bildschirmbereich ausfüllen auch an die Grenzen der Addierer und des Summenzählers stoßen, da dies aber kaum der Fall sein wird, die Koordinaten aber in jedem Fall den gesamtem Bildbereich durchlaufen müssen, ist dies der wichtigste Sicherheitsbereich.

5. Schwierigkeiten und Grenzen

Die Umsetzung des Schaltungskonzepts auf das CMOS Sea-Of-Gate - Array des IMS gestaltete sich leider etwas schwierig, da durch die Komplexität der Schaltung sowie durch die hohen Anforderungen an die Rechengeschwindigkeit nahezu an alle Grenzen der Entwicklung gestoßen wurde. Die erste erreichte Grenze lag in der Bearbeitungsgeschwindigkeit der Addierer, die aufgrund der großen Busbreite und der kurzen Taktzeiten durch normale Addierergrundschaltungen nicht erreichbar war. Nur mittels einer stärkeren Parallelverarbeitung, die natürlich einen erheblich größeren Platzbedarf hat, konnten diese Addierer realisiert werden. Ähnliches war auch bei den Zählern nötig, da diese zum einen schnell und zum anderen auch testbar sein mußten, was durch die Möglichkeit, die Zähler parallel laden zu können, erreicht wurde. Die hohe Anzahl von nötigen Addierern und Zählern sowie einiger Ablauflogik führte jedoch dazu, daß die Chipfläche des größten verfügbaren Arrays zur Realisierung der Schaltung nicht ausreichte.
Dieses Problem konnte nur durch eine Partitionierung der Schaltung, wie sie in Bild 5.1 dargestellt ist, gelöst werden, welche allerdings zu weiteren Problemen führte. Die einzige Möglichkeit einer sinnvollen Trennung der Schaltung bestand bei der Übernahme der Koordinatensummen in den Dividierer, da sich hier die Daten nicht mit der Pixelfrequenz sondern mit der wesentlich niedrigeren Bildwiederholfrequenz ändern und somit eine sichere

Übertragung der Daten über die Leiterplatte zu realisieren ist. Die Breite der Busse an dieser Stelle führt jedoch dazu, daß zusätzlich 74 I/O - Pins benötigt wurden, welche bei den bisherigen Gehäusegrößen nur durch Einsparungen an anderer Stelle freizumachen wären. Als Alternative zur parallelen Übertragung der Daten bot sich die Übertragung mittels Multiplexern an. Der zusätzliche Platzbedarf der Multiplexer, der die zur Verfügung stehende Fläche auch nach der Partitionierung überschreiten würde, sowie die Möglichkeit das benötigte Array in ein Gehäuse mit der benötigten Anzahl I/O - Pins einzubauen, hatte zur Folge, daß die parallele Datenübertragung als bessere Alternative ausgewählt wurde.

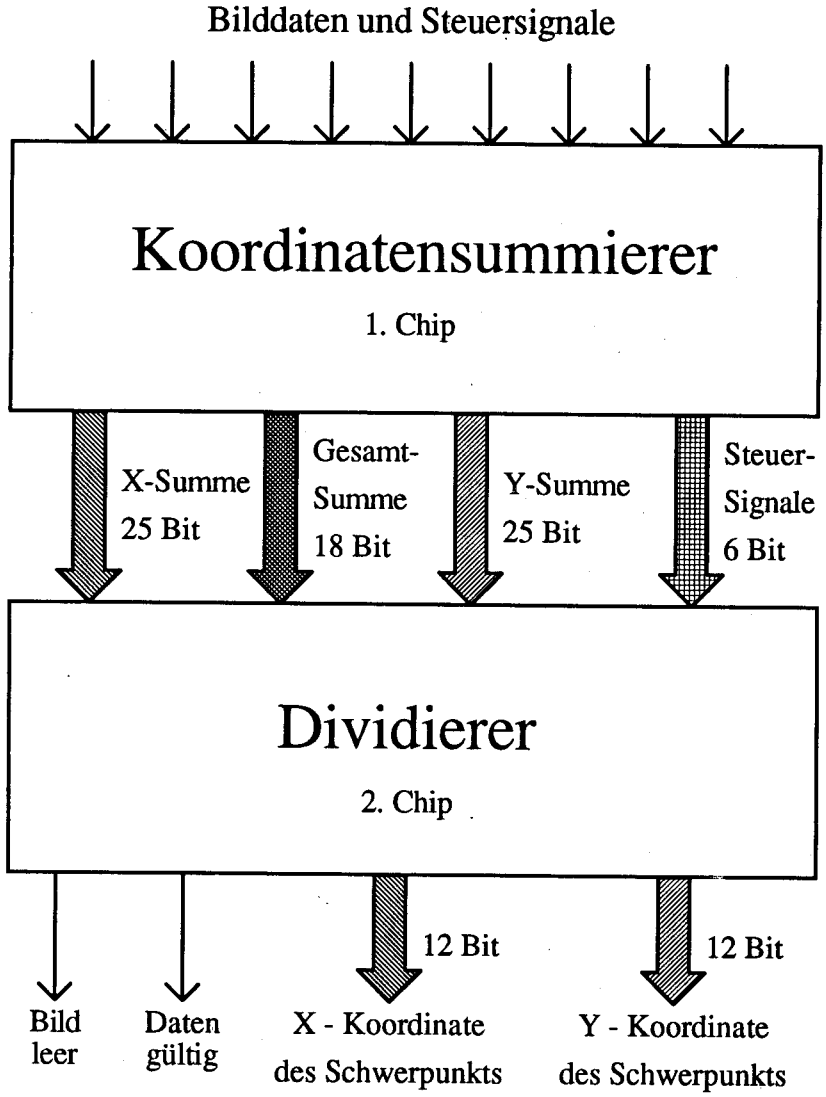


Bild 5.1: Partitionierung der Schaltung

Weitere Probleme stellten sich durch die Komplexität der Schaltung im Bereich der Testbarkeit, wodurch die Integration zusätzlicher Teststrukturen unumgänglich wurde. Zum einen wurde die Testbarkeit der Schaltung durch Hinzufügen zusätzlicher Testpins, deren Anzahl durch die wenigen freien I/O - Pins stark eingeschränkt war, zum anderen durch die Implementierung eines Scan - Path verbessert.
Die Komplexität der Schaltung warf noch weitere Probleme auf, speziell bei den Treiberleistungen (Fan Out) der einzelnen Grundelemente, die ohne zusätzliche Treiberelemente teilweise um den Faktor 50 überschritten worden wären. Um dieses zu verhindern wurden bei sehr vielen Signalen baumartige Treiberstrukturen implementiert, die teilweise eine unterschiedliche Anzahl von Stufen erhielten, um den maximalen Strom der im Umschaltaugenblick fließt zu mindern.
Obwohl der Schaltungsentwurf all diese Grenzen erreichte, konnte die Entwicklung bis zur fertigbaren Schaltung erfolgen.

6. Ablauf der Schwerpunktberechnung

Der Entwurf der benötigten Schaltung geschah mittels des Programms NetEd, wobei die Möglichkeit eines hierarchischen Schaltungsentwurfes genutzt wurde. Da die Erläuterung der einzelnen Schaltungsteile zu weit führen würde, soll an dieser Stelle nur der prinzipielle Aufbau und der Ablauf der beiden Schaltkreise gezeigt werden. Dies ist zum einen der Koordinatensummierer und zum anderen der Dividierer. Insgesamt besteht die Schaltung aus ca. 3200 Bibliothekselementen auf 31 verschiedenen Schaltplänen, die durch Mehrfachverwendung zu insgesamt 72 eingebundenen Schaltplänen führten.

6.1. Ablauf des Koordinatensummierens

Die Pipelinestruktur des Koordinatensummierers ist im Blockschaltbild, das in Bild 6.2 zu sehen ist, deutlich zu erkennen. Die eingehenden Bildpunkte werden von der Ablaufsteuerung in Steuersignale für die einzelnen Schaltungsteile umgesetzt. Bedingt durch die Pipelineverarbeitung, deren Ablauf in Bild 6.1 zu sehen ist, muß der zeitliche Zusammenhang der Steuersignale zu dem Zeitpunkt, an dem die Daten die entsprechenden Schaltungsteile erreichen, beachtet werden. Im ersten Schritt werden die aktuellen Bildpunktkoordinaten berechnet und parallel dazu ermittelt, ob der Bildpunkt zum Objekt oder zum Hintergrund gehört. Ist es ein Vordergrundpunkt, so werden die gerade berechneten Koordinaten an die Addierer angelegt und im nächsten Schritt zu der bisherigen Summe addiert. Gleichzeitig werden die Steuersignale zum Zählen der Gesamtzahl aller Vordergrundpunkte generiert, diese werden allerdings zeitlich so korrigiert, bis die entsprechenden Summen ebenfalls diese Stufe der Berechnung erreicht haben. Während der Addition der Koordinaten werden bereits die Koordinaten des nächsten Bildpunktes berechnet.

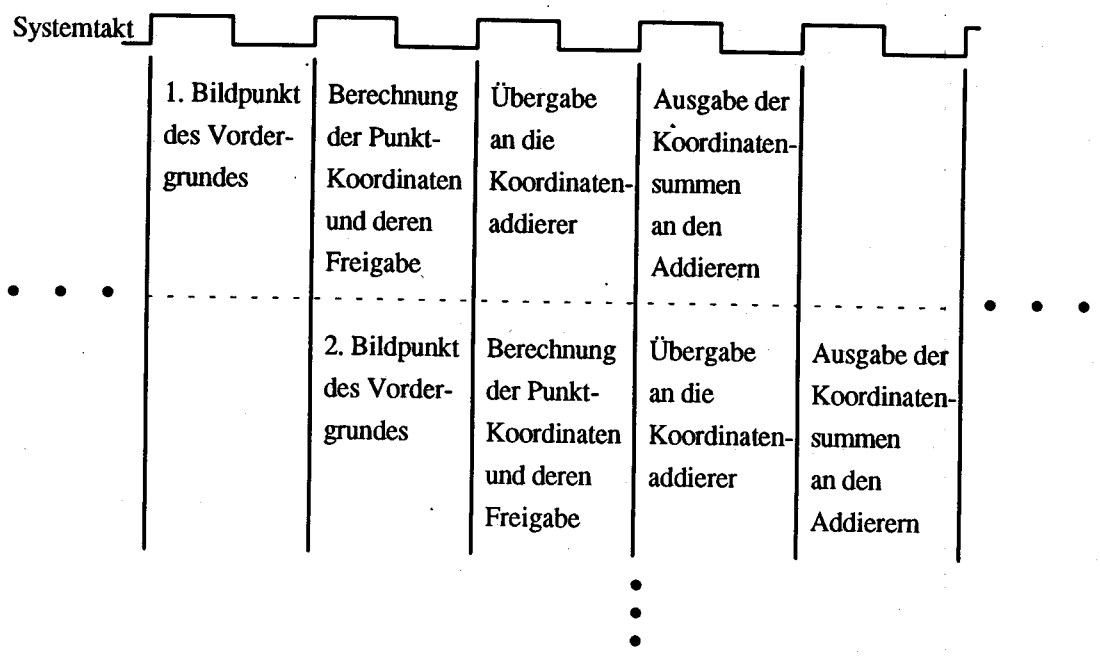


Bild 6.1: Ablauf der Koordinatenaddition in Pipelineverarbeitung

Zur Anpassung an die Videodaten wird von der Ablaufsteuerung während des Zeilenimpulses der X - Koordinatenzähler auf null gehalten und erst ein Takt nach Beenden dieses Zeilenrücklaufes der Zähler wieder freigegeben. Bei den Y - Koordinaten wird abhängig von der Art des Eingangsbildes (Interlace - Bild oder nicht) nach jedem Zeilenrücklauf der Zähler um eins oder zwei erhöht. Hierbei wird kein Unterschied zwischen dem ersten und zweiten Halbbild beim Interlace - Verfahren gemacht, dieses wird nach der Division vor Ausgabe der Schwerpunktkoordinaten aber korrigiert. Nach Ende eines Bildes bzw. eines Halbbildes, das durch den Bildimpuls gekennzeichnet wird, werden alle Zähler und Addierer zurückgesetzt und die berechneten Summen dem zweiten Schaltkreis übergeben.

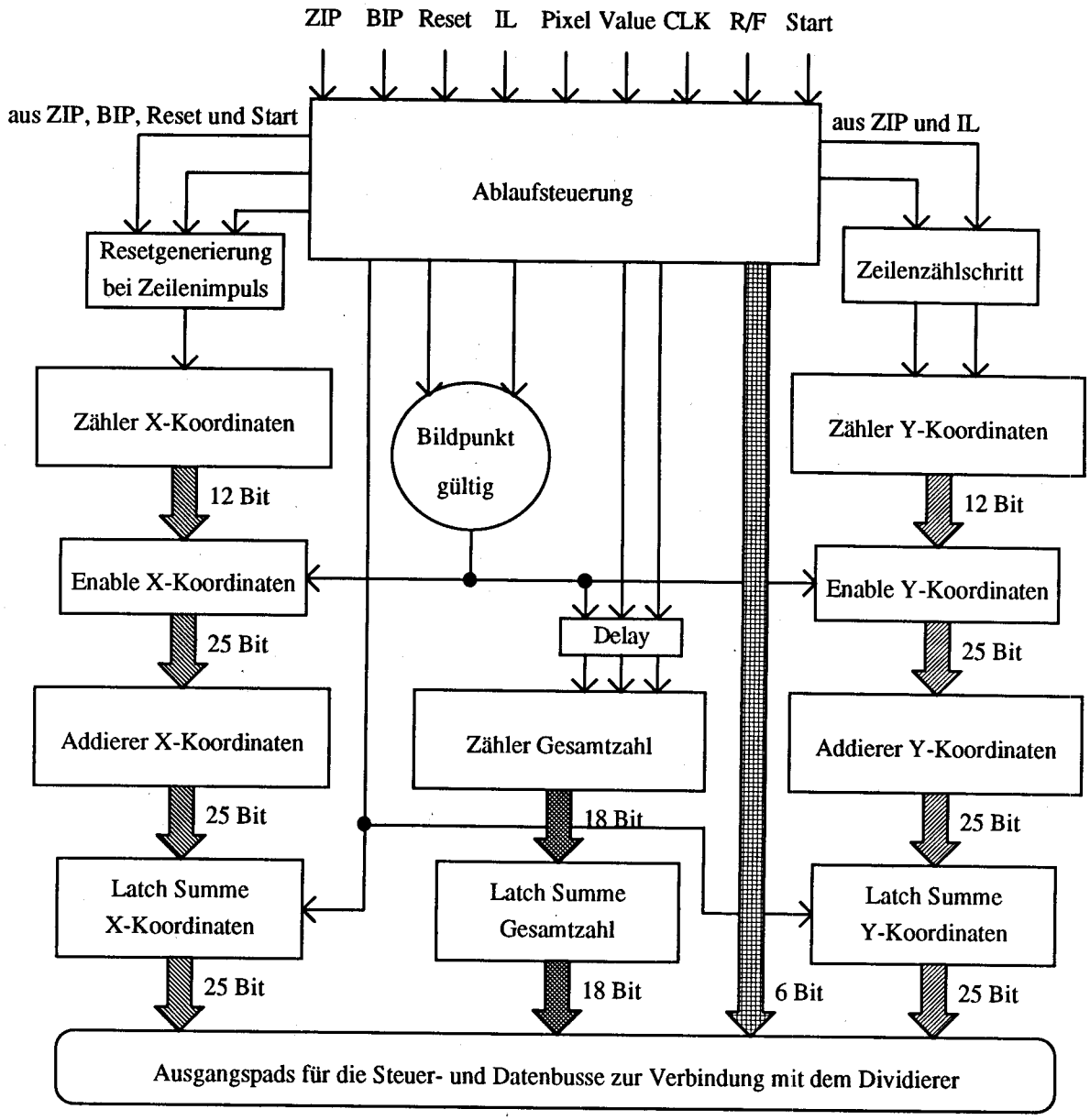


Bild 6.2: Blockschaltbild des Koordinatensummierers

6.2. Ablauf des Dividierens

Wie oben bereits erwähnt existiert keine Dividiererschaltung, die Division der Koordinatensummen durch die Gesamtzahl Vordergrundbildpunkte muß somit algorithmisch durchgeführt werden. Beim verwendeten Algorithmus wird mittels Addition des Zweierkomplements die Gesamtzahl der Punkte von den Koordinatensummen abgezogen. Die Anzahl der hierbei benötigten Subtraktionen entspricht dem Ergebnis der Division, wobei eine Genauigkeit von 1 Pixel erreicht wird, welche völlig ausreichend ist.
Das Blockschaltbild der zu diesem Zweck entwickelten Schaltung ist im Bild 6.5 zu sehen. Der Ablaufsteuerung wird vom Koordinatensummierer das Ende der Summenberechnung mitgeteilt, worauf diese zuerst die Bildung des Zweierkomplements der Gesamtzahl abwartet und dann, falls kein leeres Bild vorliegt, mittels den beiden Multiplexern zuerst die Koordinatensummen und dann das Zweierkomplement der Punktgesamtzahl in die Addierer lädt. Der Start dieser Division ist schematisch in Bild 6.3 zu sehen. Nach der ersten Addition werden die beiden Zähler, die die Anzahl nötiger Subtraktionen ermitteln sollen, freigegeben. Dies darf erst bei der zweiten Subtraktion geschehen, da die erste Subtraktion den Schwerpunktkoordinaten 0 entspricht.

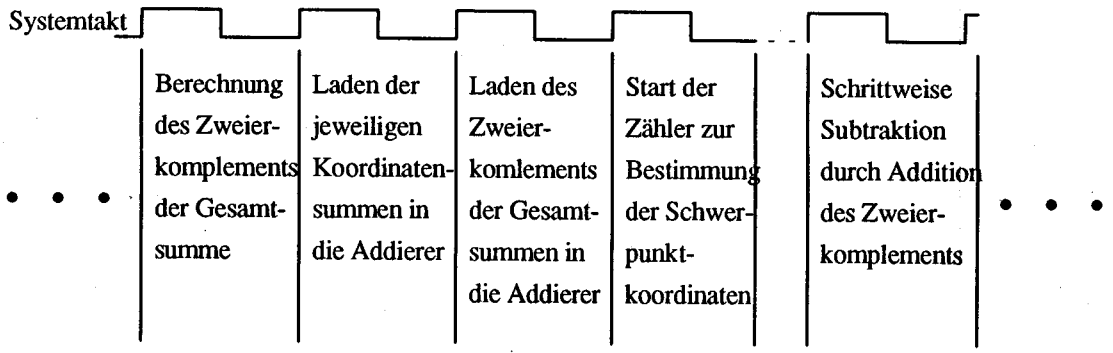


Bild 6.3: Start der Division

Das Ende der einzelnen Divisionen (siehe Bild 6.4) wird durch den Addiererüberlauf, der nur beim Übergang in den negativen Zahlenbereich nicht gesetzt wird, für X - und Y - Koordinaten getrennt detektiert. Diese Unterscheidung ist nötig, da die beiden Divisionen eine unterschiedliche Anzahl Takte zur Berechnung benötigen können. Falls es sich um Bilder im NON - Interlace - Verfahren oder aber um das erste Halbbild eines Interlace - Bildes handelt, enthalten die Zähler nach Beenden beider Divisionen die Schwerpunktkoordinaten. Beim zweiten Halbbild muß jedoch die Y - Koordinate um einen Bildpunkt nach unten verschoben werden, da bei der Koordinatenaddition kein Unterschied zwischen den beiden Halbbildern gemacht wurde.

Ist diese Koordinatenkorrektur erfolgt, so können die Schwerpunktkoordinaten in die Ausgangslatch übernommen werden und das Signal zur Kennzeichnung gültiger Koordinaten kann gesetzt werden.

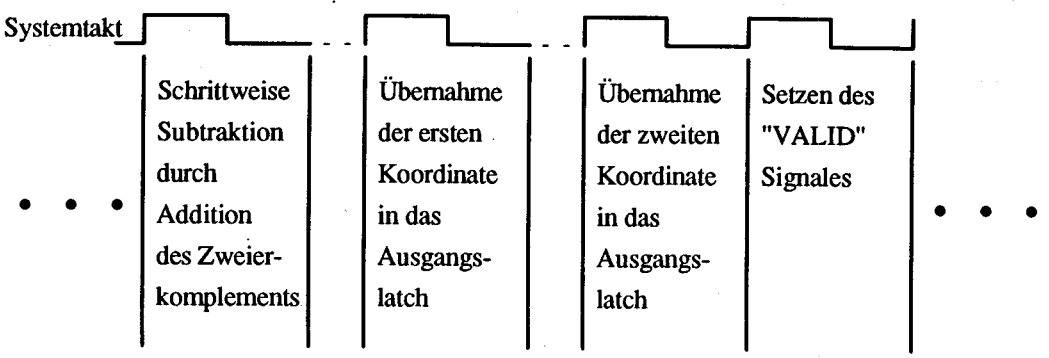


Bild 6.4: Ende der Division

Die prinzipielle Struktur der so entwickelten Dividiererhardware ist im folgenden Blockschaltbild dargestellt.

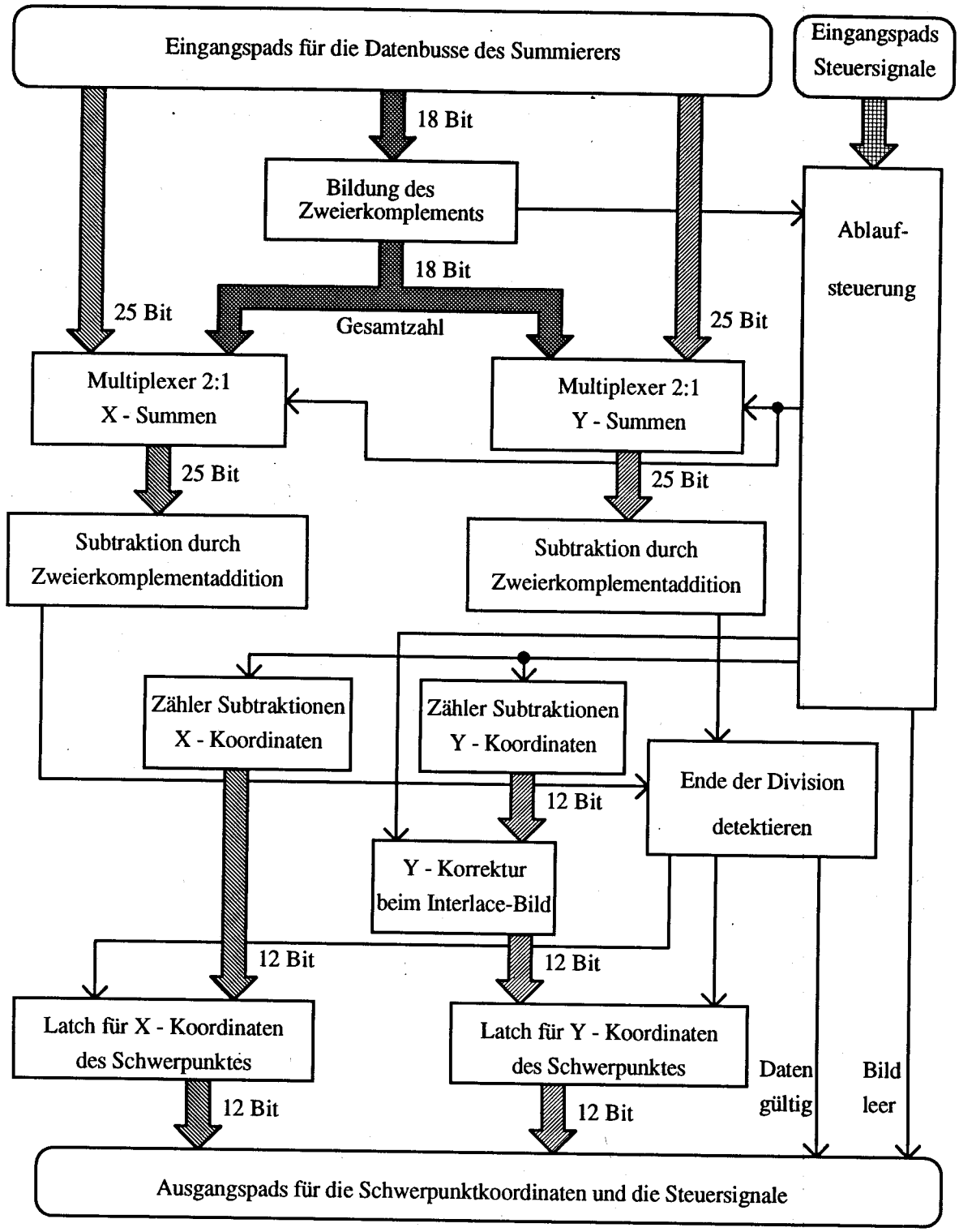


Bild 6.5: Blockschaltbild des Dividierers

7. Erfahrungen bei der Entwicklung

Der Ablauf der Schaltungsentwicklung mit Hilfe des Programms NetEd lief dank der guten Möglichkeiten beim hierarchischen Entwurf reibungslos ab. Wesentlich größere Probleme entstanden durch die Bibliothek des IMS. Nach Abschluß etwa der Hälfte der Entwicklung mit der 2µm Technologie wurde uns mitgeteilt, daß diese auslaufen würde und an ihrer Stelle die neue 1,2µm Technologie treten werde, für welche jedoch noch keine fertige Bibliothek existiert. Durch den Erhalt einer Vorversion dieser Bibliothek konnten die bereits erstellten Schaltungsteile auf die neue Technologie umgearbeitet werden. Da sich in der neuen Bibliothek nicht nur das Aussehen und die Größe der Elemente, sondern auch deren Namen, die Pinbelegung sowie die Polarität einiger Signale (low - oder highaktiv) geändert hat, mußten alle Schaltungsteile umgezeichnet, manche sogar völlig neu entworfen werden.
Die Überprüfung der Schaltung um Verletzungen der "FanOut" - Leistungen zu erkennen scheiterte ebenfalls, da bei allen Bibliothekselementen falsche Treiberfähigkeiten eingetragen waren und der verfügbare Checker auch nach Anpassung unsererseits an die neue Bibliothek jedes "FanIn" eines Bauteiles, das nicht 1 war, ignorierte. Ein funktionstüchtiger Checker soll jedoch in Bälde verfügbar sein.
Die Realisierung der Schaltung scheint ebenfalls schwierig, da das IMS momentan noch Schwierigkeiten mit den Datenmengen des benötigten Arrays hat und die Schnittstelle für Testdaten noch nicht verfügbar ist.
Durch die Komplexität der Schaltung ergaben sich auch bei der Schaltungssimulation Probleme. Zwar konnten die einzelnen Schaltungsteile für sich leicht simuliert werden, die vollständige Simulation der Addierer sowie der Gesamtschaltung scheiterte jedoch an der Rechenzeit und der Speicherkapazität. Wollte man z.B. mit dem 25 Bit Addierer einen vollständigen Funktionstest durchführen, indem man nur eine Ebene tief jede Zahl zu jeder möglichen addiert, würde man eine Simulationszeit von ca. 60 Millionen Jahren benötigen. Zur Verkürzung dieser doch recht langen Zeit könnte man z.B. die 6 Bit Addierergrundeinheit, aus der alle Zähler und Addierer aufgebaut sind, auf die gleiche Weise testen. Dieses scheiterte nicht an der Simulationszeit, die etwa zwei Tage betragen würde, sondern am zur Verfügung stehenden Speicherplatz. Nach ca. einem Tag war der freie Plattenplatz gefüllt, zur vollständigen Simulation würden mindestens 200 MByte benötigt, die auf keinem Rechner freizumachen waren. Ähnliche Probleme wie bei der Simulation stellten sich bei der Testbarkeit. Trotz des integrierten Scan - Path wären die Testzeiten sowie die Menge hierzu benötigter Testpattern für einen vollständigen "Stuck - at" - Test bei weitem zu groß. Als Alternative hierzu blieb uns nur ein umfangreicher Funktionstest, der durch zusätzliche Testpins sowie den bereits erwähnten Scan - Path eine recht hohe Fehlerüberdeckung erreicht. Näheres zu Testability, Fehlermodelle und Testzeiten sind dem Vortrag des Herrn Dr. Denner, dessen Beitrag ebenfalls in diesem Band zu finden ist, zu entnehmen.

8. Anwendungsbeispiel

Außer der einfachen Berechnung des Flächenschwerpunktes kann mit Hilfe der entwickelten Schaltkreise und eines Steuerrechners eine Echtzeit - Objektverfolgung realisiert werden. Die Hardware berechnet die aktuelle Position des zu verfolgenden Objektes über dessen Flächenschwerpunkt in Echtzeit, ohne den Steuerrechner zu belasten. Dieser muß nur noch die Lage des Objektes berechnen (bei Drehung des Objektes), was leicht in Echtzeit erfolgen kann. Durch zusätzliche Einschränkung des zu bearbeitenden Bereiches kann auch eines von mehreren Objekten verfolgt werden. Hierbei muß vom Steuerrechner eine "area of interest" (interessanter Bildbereich) aus dem gesamten Bild ermittelt werden, in welcher sich das Objekt befindet. Dieser Bereich kann dann relativ zur Bewegung des Flächenschwerpunktes geändert und der Hardware zur erneuten Berechnung nur noch dieser interessante Teil übergeben werden. Die nötige Bewegung des relevanten Bildausschnitts liefern ebenfalls die Schaltkreise über die Bewegung des Flächenschwerpunktes, es wird kein zusätzlicher Rechenaufwand benötigt. Problematisch wird die Objektverfolgung, wenn zwei oder mehrere Objekte nicht mehr über eine "area of interest" getrennt werden können, in diesem Fall scheitert das erläuterte Verfahren.
Durch die Echtzeitbearbeitung bieten sich auch weitere Anwendungsmöglichkeiten an, bei denen bewegte Objekte bearbeitet werden müssen. Da dem Bildformat, dem Bildinhalt sowie den Steuersignalen so wenig Einschränkungen wie möglich gemacht wurden, können die entwickelten Schaltkreise leicht in nahezu jedem Bildverarbeitungssysteme eingesetzt werden.

9. Zusammenfassung

Abschließend läßt sich über die Entwicklung sagen, daß diese im Großen und Ganzen ohne Probleme ablief, abgesehen von den oben erwähnten Schwierigkeiten. Ob die Herstellung den gleichen Erfolg haben wird bleibt noch abzuwarten, da bisher nur mit einem Probelauf begonnen wurde, um die Bearbeitung dieser Datenmenge auszutesten. Nach Herstellung der Schaltkreise werden diese im Bildverarbeitungssystem "Zyklop" der Fachhochschule Heilbronn eingesetzt, durch die universellen Anpassungsmöglichkeiten bieten sich jedoch auch Verwendungen in anderen Systemen an.

9. Entwurf einer Schaltung zur Suche von Zeichensequenzen

H.Kreutzer, W.Ludescher

FH Reutlingen, FH Ravensburg-Weingarten

Kurzfassung:

Der Entwurf einer Schaltung zur ON-LINE Suche nach Zeichensequenzen in Datenströmen wird dargestellt.
Diese Darstellung ist eine Ergänzung des Beitrages " ASIC unterstützt Sequenzsuche, Frequenz- und Phasenmessung" des Workshops der Multiprojekt Chip-Gruppe im Februar 1992 in Karlsruhe.
In diesem vorliegenden Beitrag wird die Aufgabenstellung und Problemlösung anhand von Beispielen illustriert.

Einleitung:

- Durchführung eines Gate-Array-Entwurfs zu Übungszwecken

- Test der Geschwindigkeitsgrenze für eine ON - LINE Zeichensequenzsuche

- Entwurfsdarstellung der verwendeten Schaltung

Problem:

- Zeichensequenzsuche in einem Datenstrom zur Selektion von Daten

- z.B.: In Datenstromanalysatoren für ISDN, SDH

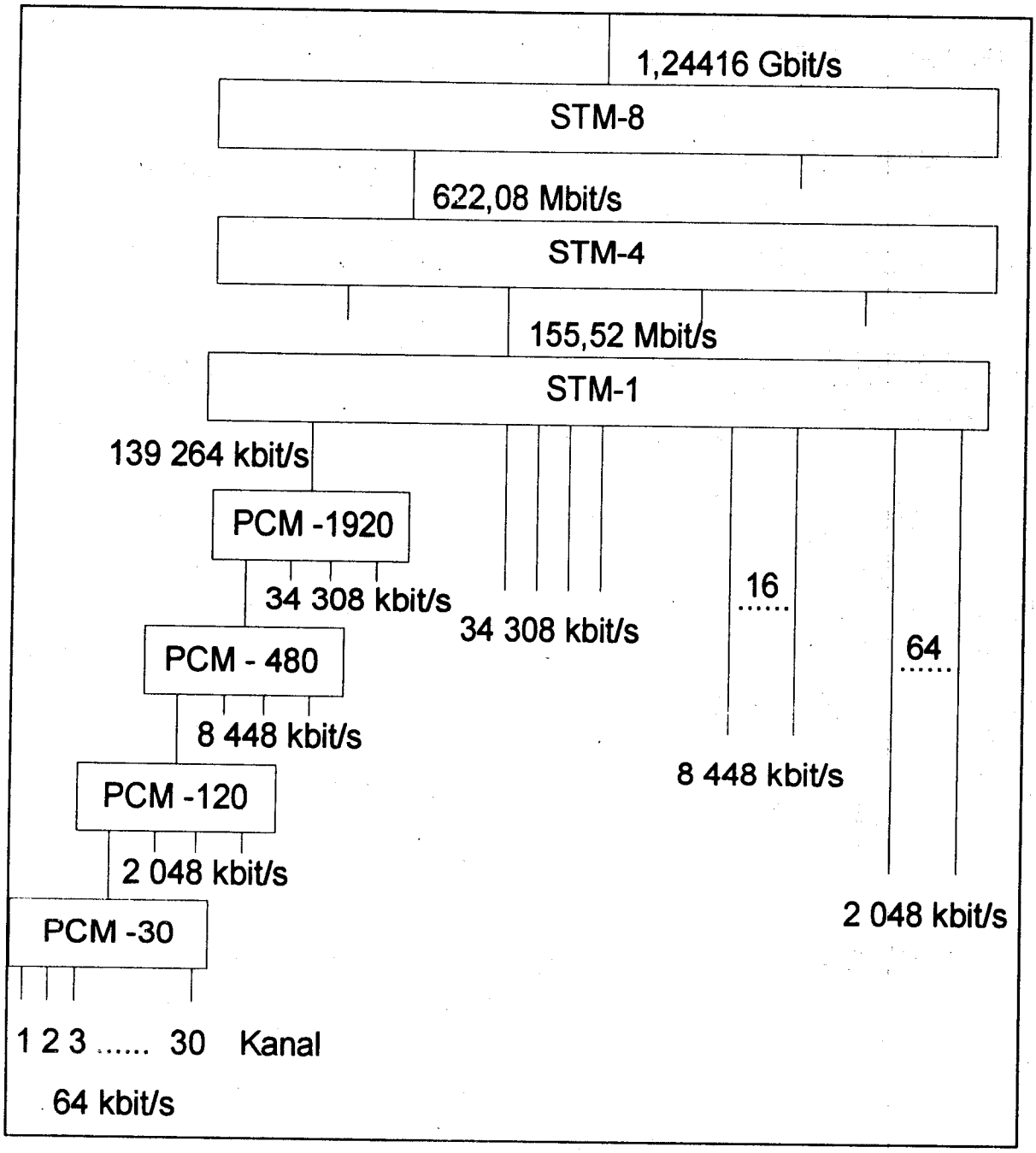


Bild 1: Synchrone digitale Hierarchie (SDH)

Aufgabe der Sequenzsuchschaltung:

- Max. Länge der Suchsequenz = 64 Zeichen (a' 8bit, 7bit, 6bit)
- Länge und Muster beliebig einstellbar
- Suche mehrerer Suchsequenzen gleichzeitig
- Datenrate 2 Mbit/s (500 Mbit/s)
- Erkennung ineinandergeschachtelter Zeichen-sequenzen

- Beispiel:

{A,B,C,...,Z} ∊ {Menge der EBCDIC-Zeichen}

```
                  A
Suchsequenz:   ABZCAGZ   (G überstrichen)
                  D

                  |------------|
Datenstrom:   ABAB ABZAABZCAFZEFG
                         |------------|
```

Realisierungskonzepte:

- Hardwarerealisierung
- Direkter Vergleich der Suchsequenz mit dem Datenstrom

  Zu aufwendig!
- Schaltwerkrealisierung (klassisch)

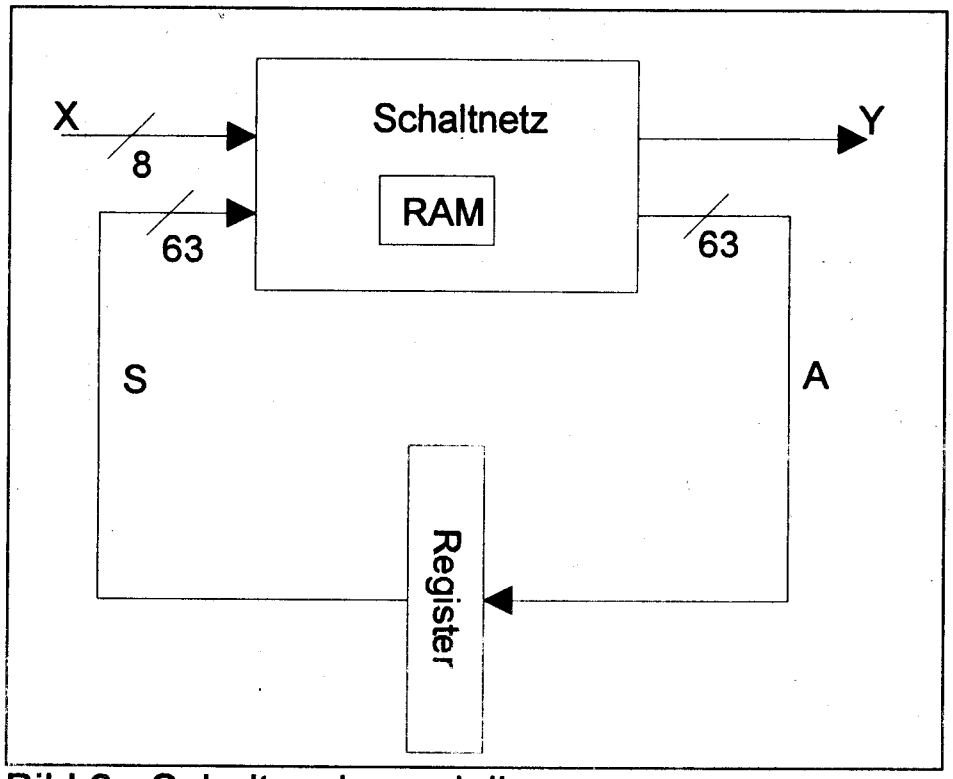


Bild 2: Schaltwerksmodell

Aufwand: 63 FF
71 Adressleitungen

Zu aufwendig!

- Schaltwerkrealisierung (aufgabenangepaßt)

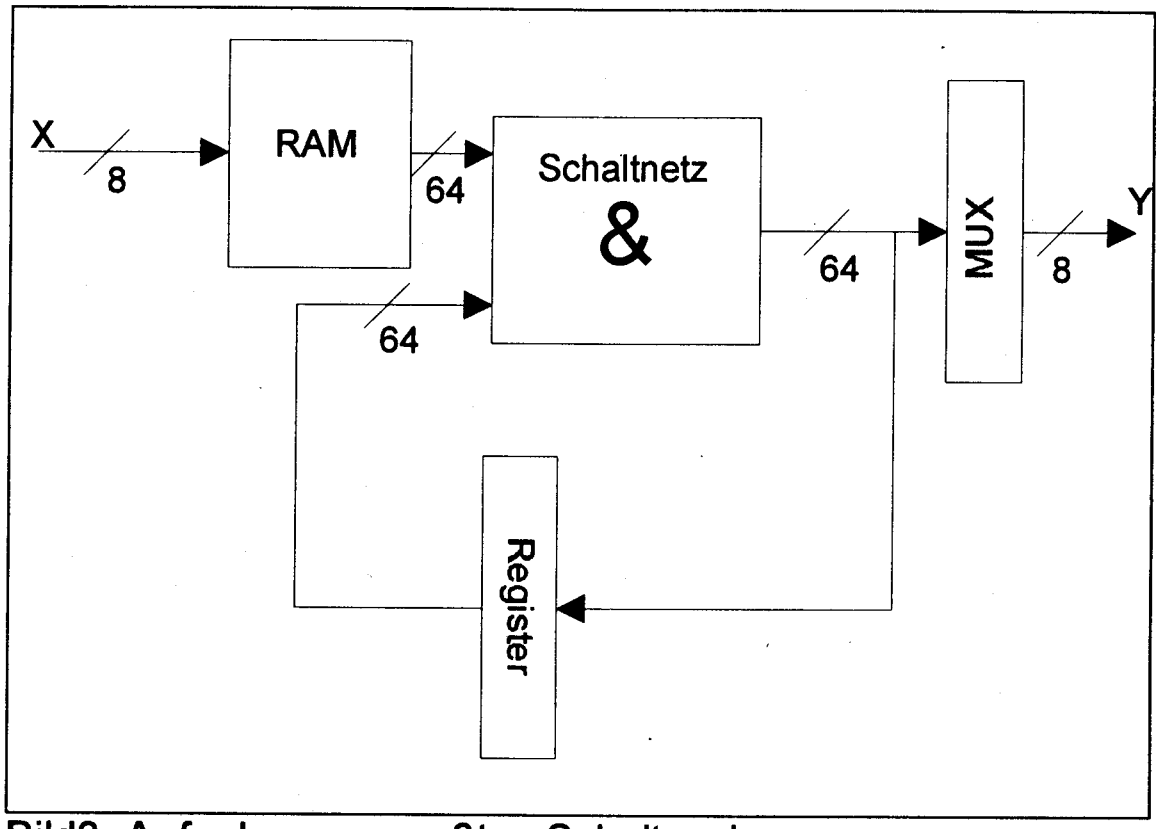


Bild3: Aufgabenangepaßtes Schaltwerk

Modell der Sequenzsuchstrategie:

(Formale Beschreibung der Sequenzsuche)

- Einteilung in Zustände und Ebenen
- Ebenen: Zeichenpositionen in der Suchsequenz
- Zustände: Berücksichtigung der relevanten Positionen der Suchsequenz

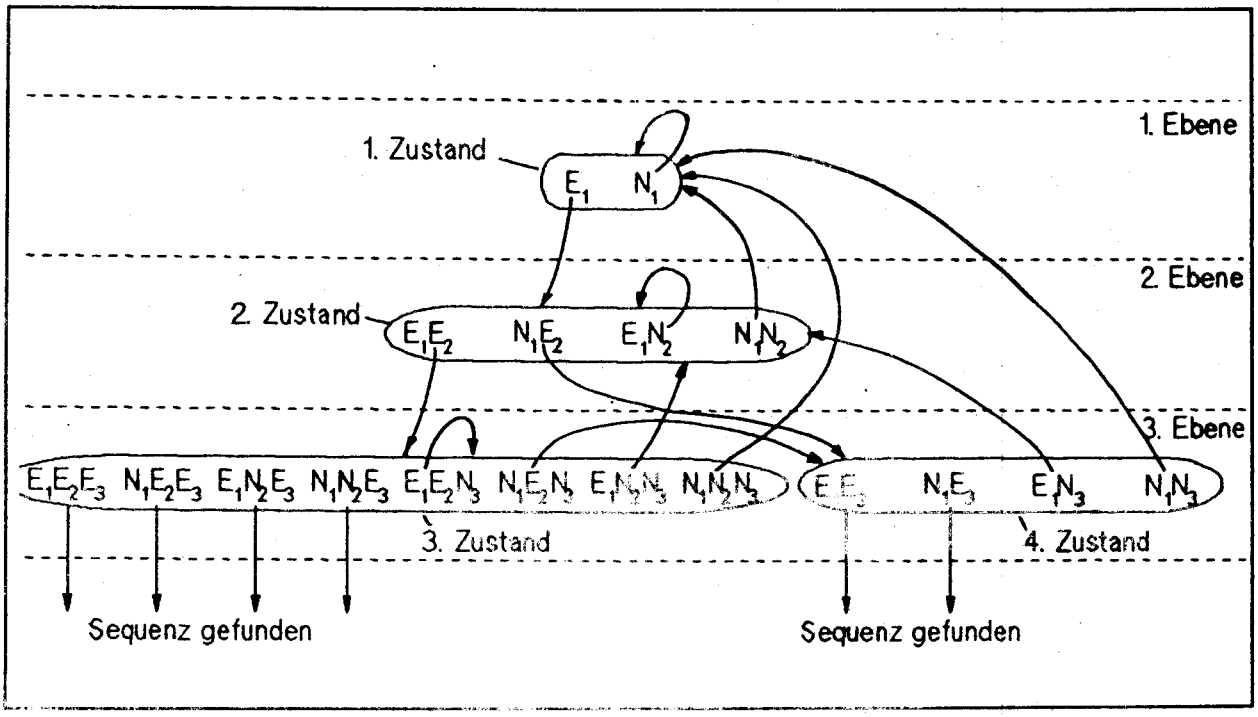


Bild 4: Zustandsdiagramm für eine 3-Zeichen Suchsequenz

E_i = {Zeichen, die in der i-ten Suchsequenzposition erlaubt sind}

N_i = {Zeichen, die in der i-ten Suchsequenzposition nicht erlaubt sind}

Beispiel: Suchsequenz: $\overset{A}{ABC}$

$E_1 = \{A\}$
$N_1 = \{B,C,D,...\} = \{\bar{A}\}$

$E_2 = \{A,B\}$
$N_2 = \{C,D,E,...\}$

$E_3 = \{C\}$

$N_3 = \{\bar{C}\}$

Datenstrom: BCABAAABAACCC
|-----|

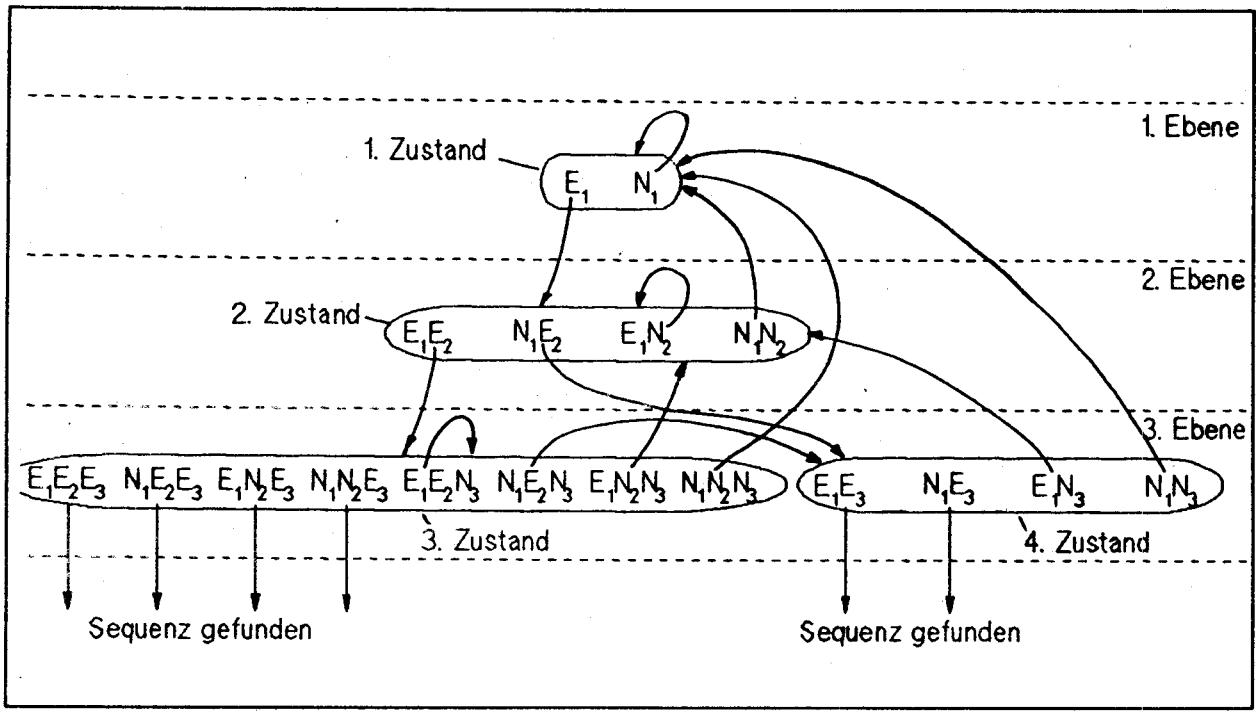


Bild 4: Zustandsdiagramm für eine 3-Zeichen Suchsequenz

Systematik der Zustände:

- Eindeutige Kennzeichnung der Zustände durch Indizes (relevante Suchsequenzpositionen)
- Zustandskennzeichnung durch eine Maske
- Die Zustandsanordnung einer Ebene ergibt sich aus der Zustandsanordnung der darüberliegenden Ebene

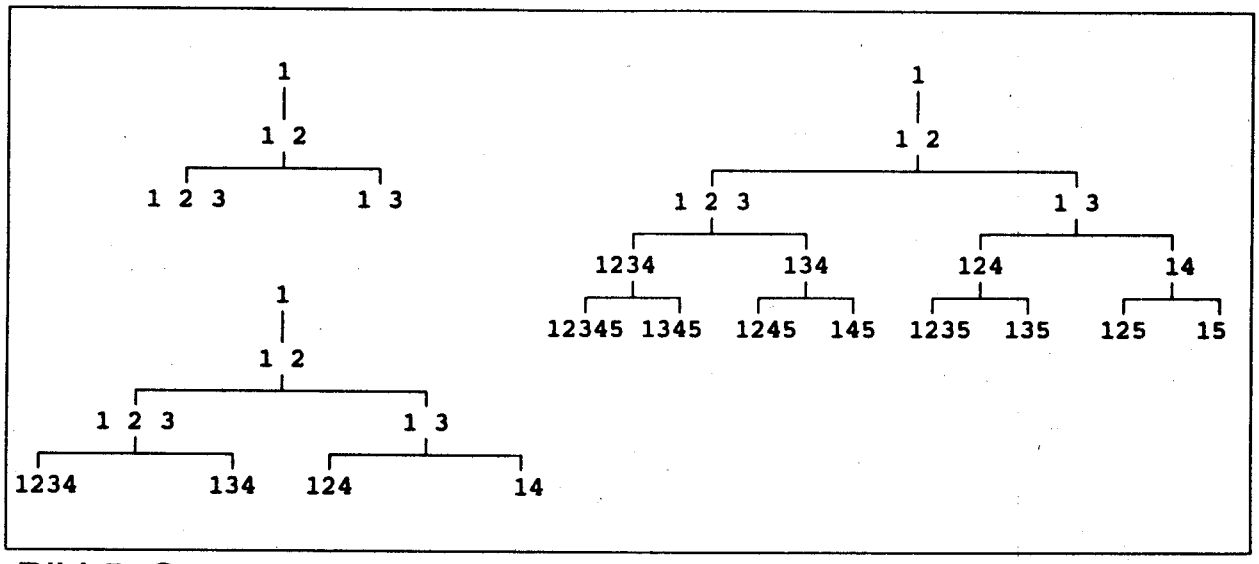


Bild 5: Systematik der Zustände für Suchsequenzfolgen mit 3, 4 und 5 Zeichen

- Zustandsanzahl = $2^{(\text{Zeichenanzahl} - 1)}$

Systematik der Übergänge:

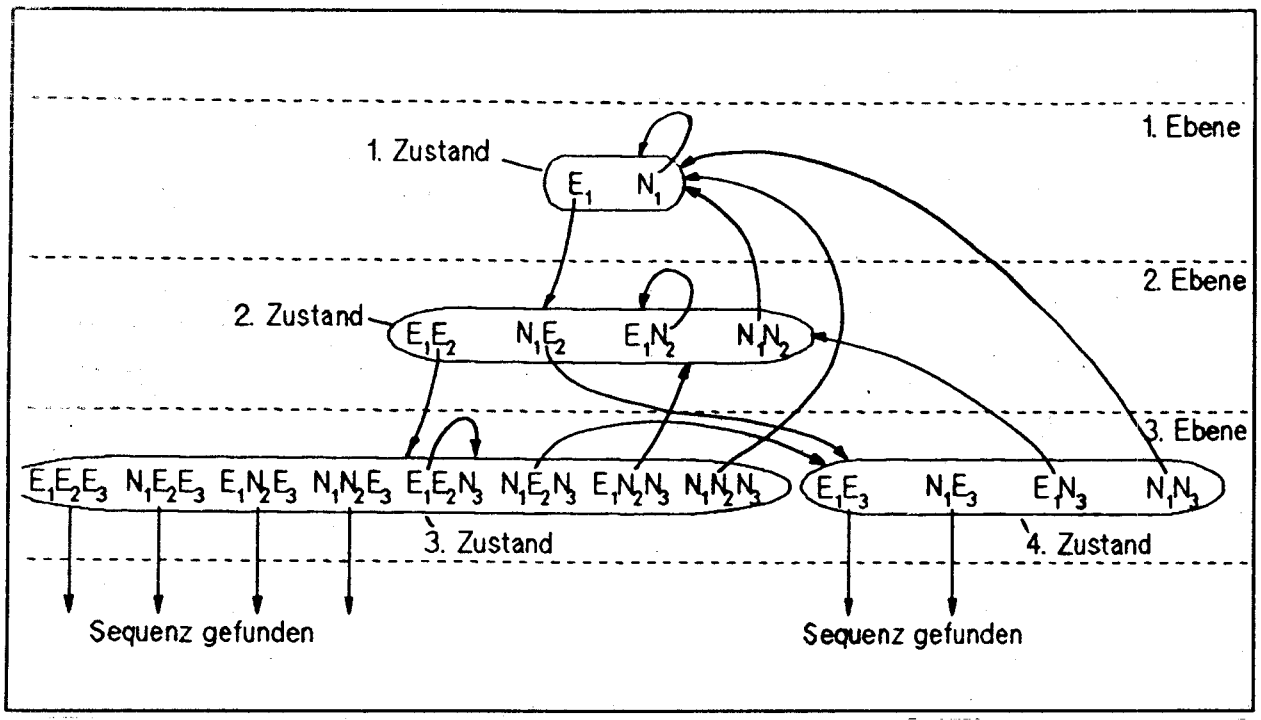


Bild 4: Zustandsdiagramm für eine 3-Zeichen Suchsequenz

- Sprungziele sind identisch, bei Betrachtung der E_i - Positionen

- Festlegung des Zielzustandes (Maske):
  - Die erste Suchsequenzposition muß immer untersucht werden
  - Die um eins inkrementierten mit E_i gekennzeichneten Positionen sind die relevanten Suchsequenzpositionen des Zielzustandes

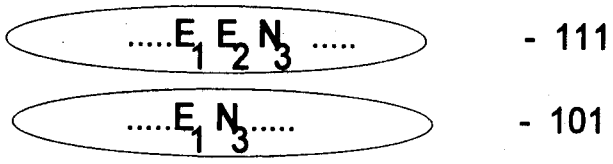
Realisierung:

- Kennung an welcher Suchsequenzposition ein Zeichen zugelassen ist

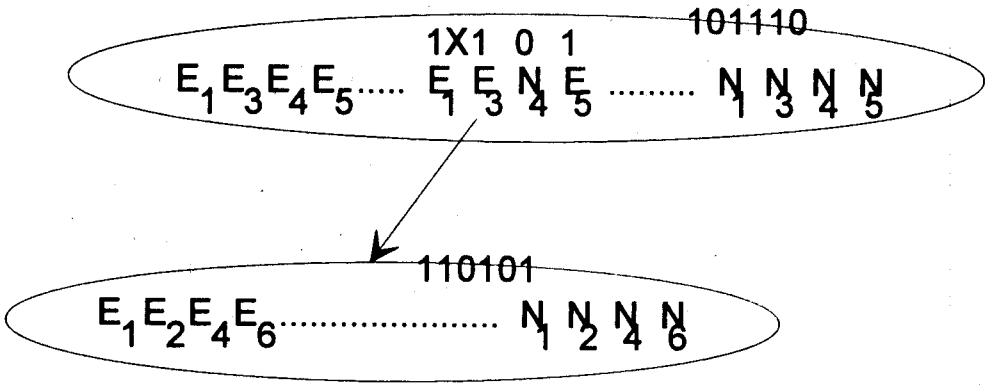
  Z.B.: N_i - 0
  E_i - 1

- Maske zur Zustandskennzeichnung

  Z.B.: 3 relevante Stellen (3. Ebene)



- Beispiel: 6-Zeichen Suchsequenz



Schaltung:

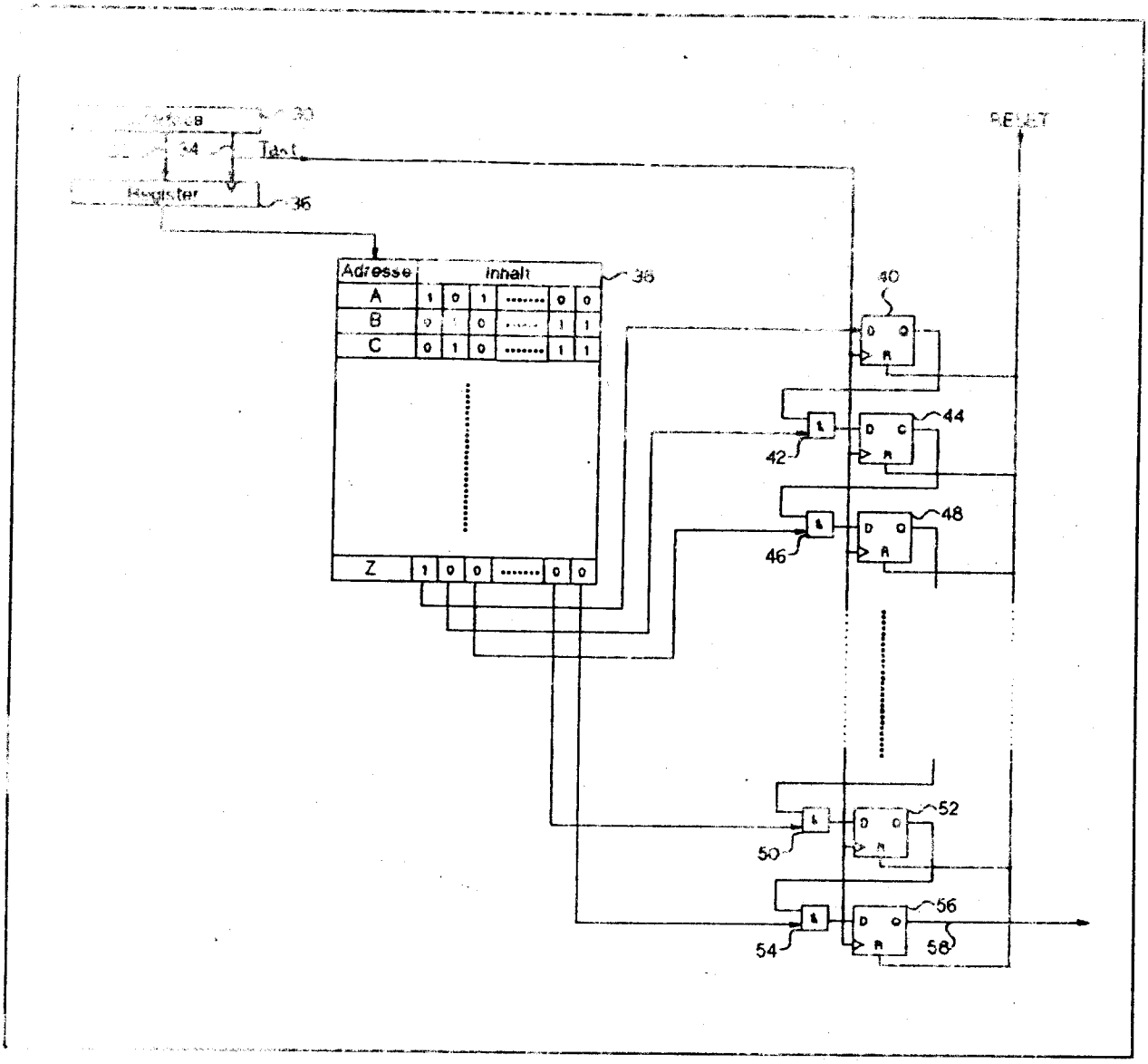


Bild 6: Prinzipschaltung zur Suche nach beliebig langen Zeichensequenzen

Beispiel: 5-Zeichen-Suchsequenz

Suchsequenz:

B A
A C D D D
D

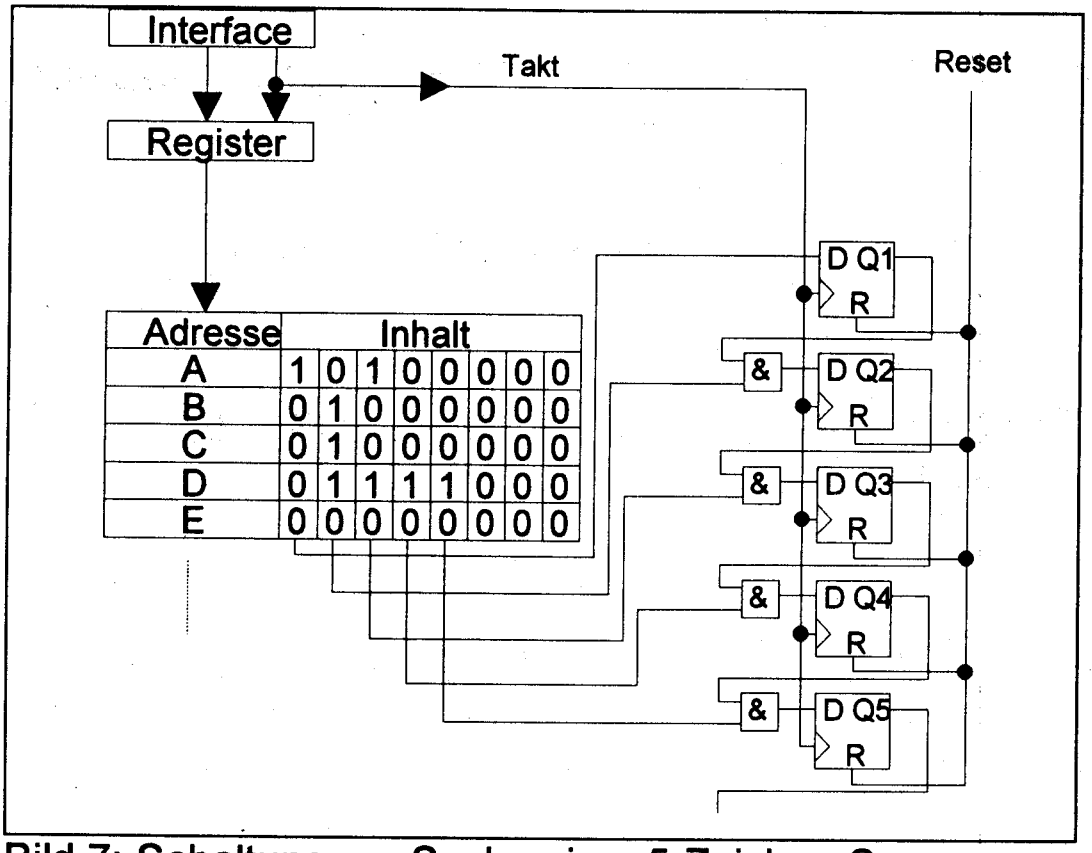


Bild 7: Schaltung zur Suche einer 5-Zeichen-Sequenz

Datenstrom:

	A	B	A	D	A	C	A	D	A	B	A	D	D	D	D	D
Q1:	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0
Q2:	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0
Q3:	0	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0
Q4:	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0
Q5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
Q6:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q7:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q8:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Eigenschaften der Schaltung:

- Geschwindigkeit: 0.5 Gbit/s
  (unabhängig von der Suchsequenzlänge)

- Beliebige Zeichen pro Zeichensequenzposition

- Zeichensequenzlänge einstellbar (MUX)

- Parallele Suche nach mehreren Zeichen-sequenzen möglich

- Testschaltung für 8-Zeichen-Suchsequenz

10. Entwicklung eines elektronischen Würfels und dessen Umsetzung in einen anwenderspezifischen Schaltkreis (ASIC)

Dipl. Ing. (FH) Bernd Reinke
Prof. Dr. Ing. Dirk Jansen

Mitteilung aus dem IIT der Fachhochschule Offenburg

Die Fachhochschule Offenburg bietet den Studenten des Fachbereichs Nachrichtentechnik seit Ende 1990 das Wahlfach "Entwicklung integrierter Anwenderschaltkreise (ASIC)" an. Ziel des Wahlfachs ist es, den Studenten Grundkenntnisse im Entwurf eines ASIC's zu vermitteln, und wie im folgenden Beitrag aufgezeigt, die Möglichkeit zu bieten, den gesamten Entwurfszyklus von der Schaltungsentwicklung bis hin zur Fertigungsmaske zu durchlaufen.

Einführung

Im Idealfall entspricht ein elektronischer Würfel genau der statistischen Zufallsgenauigkeit eines mathematischen Würfels, also einer Gleichverteilung der aufgetretenen Würfelaugen. Da dies in einem digitalen System nur annähernd unter hohem schaltungstechnischen Aufwand möglich ist und der Schwerpunkt hier auf der Entwurfsmethodik liegt, wird hier auf eine Gewährleistung dieser gewünschten Eigenschaft verzichtet und ein einfacher Entwurf vorgestellt. Er soll im Rahmen einer praktischen Seminarsarbeit für die Studenten nachvollziehbar sein. Ziel ist die Umsetzung einer entsprechenden Schaltung in ein Gate - Forest der Firma IMS (Institut für Mikroelektronik in Stuttgart).

Schaltungsentwurf

Der elektronische Würfel besteht aus einem sequentiellen Schaltwerk, das für das Zählen bzw. Ausdekodieren der Würfelaugen zuständig ist, einem vier Bit breiten Zufallsgenerator, der ein quasi "zufälliges Ausrollen" des Würfels verursacht, einem 16 Bit breiten synchronen Zähler als Taktteiler und einer Auswahllogik als Bindeglied zwischen Taktteiler, Zufallsgenerator und Schaltwerk (Bild 1). Die Schaltung soll so ausgelegt sein, daß alle Komponenten für sich prüfbar sind, und so die Testbarkeit der Gesamtschaltung gewährleistet ist. Auf eine Maßnahme zur Verbesserung der Prüfbarkeit mittels Scan-Path oder ähnlichen schaltungstechnischen Maßnahmen wird daher verzichtet.

Die Ansteuerung der Würfelaugen ist in Bild 2 wiedergegeben.

Die Herleitung der Übergangs- Ergebnisgleichungen des Schaltwerks erfolgt mittels Zustandsdiagramm, Übergangs- Ergebnistabelle für D-Flip-Flops, sowie Minimierung der Logikgleichungen im KV-Diagramm. Zu beachten ist die Umsetzung der Ergebnistabelle in negativer Logik, da die Ausgangstreiber des ASIC nur für den logisch "0" Pegel spezifiziert sind.

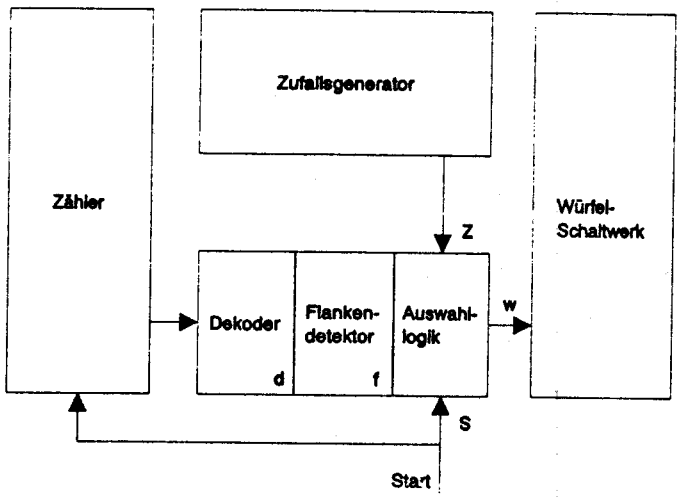
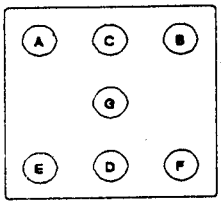


Bild 1: Blockschaltbild des elektronischen Würfels



	1	2	3	4	5	6
A				ein	ein	ein
F				ein	ein	ein
B		ein	ein	ein	ein	ein
E		ein	ein	ein	ein	ein
C						ein
D						ein
G	ein		ein		ein	

Bild 2: Ansteuerung der Würfelaugen

Wie man dem Zustandsdiagramm des Würfelschaltwerks entnehmen kann, zählt der Würfel bei Freigabe zyklisch von eins bis sechs. Um nun zu verhindern, daß jeder Zählimpuls des Taktgenerators den Würfel hochzählt, und damit die Würfelzahl nicht abschätzbar wird, unterdrückt ein über die Auswahllogik geschalteter Zufallsgenerator etwa die Hälfte aller Zählimpulse.
Der Zufallsgenerator besteht aus einem 4 Bit breiten rückgekoppelten Schieberegister, das seriell ausgelesen wird. Die Anordnung der Rückkopplung ist in diesem Anwendungsfall von untergeordneter Bedeutung und wird beliebig festgelegt. Der Zustand, bei dem alle Flip-Flop Ausgänge des Schieberegisters den logischen Wert 0 einnehmen ist verboten, und wird in einen definierten Zustand überführt.

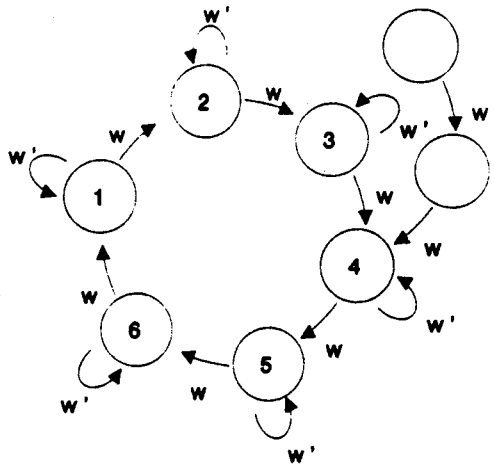


Bild 3: Entwurf des Würfelschaltwerks

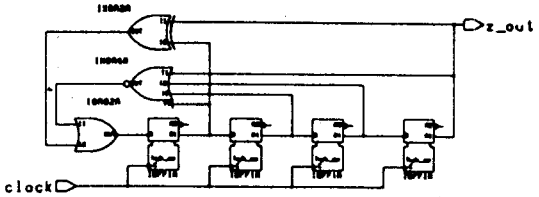


Bild 4: Schaltung des Zufallsgenerators

Eingänge	D3	0000000011111111
	D2	0000111100001111
	D1	0011001100110011
	D0	0101010101010101
Dekoderausgang	d	0101001001000101

Tabelle 1: Dekodierung des Zählers

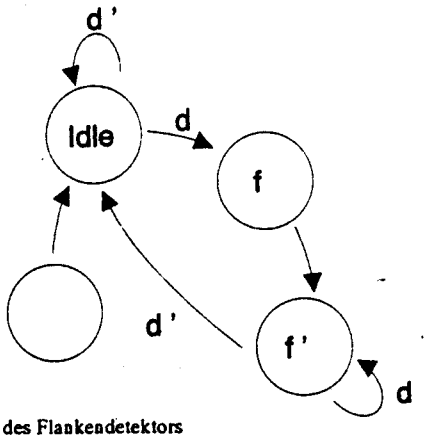


Bild 5: Entwurf des Flankendetektors

Die Auswahllogik erzeugt das Freigabesignal für den Würfel. Gibt der Benutzer ein Startsignal z.B. über eine Taste, so zählt der Würfel während des Tastendrucks abhängig vom Zustand des Zufallsgenerators zyklisch von eins bis sechs. Wird das Startsignal zurückgenommen, so zählt der Würfel abhängig vom Zählerstand eines übergeordneten Zählers weiter. Hierbei werden vier Bit des Zählers entsprechend folgender Tabelle ausgewertet.

Die Aufbereitung des Freigabesignals für den Würfel erfolgt duch einen nachgeschalteten Flankendetektor und den Zufallsgenerator. Es soll somit ein quasi zufälliges Ausrollen des Würfels erzeugt werden.

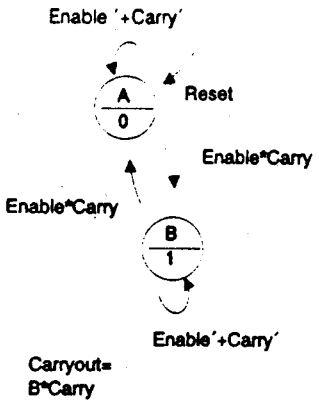


Bild 6: Entwurf des Zählers (1bit)

Bezeichnung	Nutzbare Gatter	Prozeß (um)	Chip-Fläche (qmm)	Bemerkung
GF9	500	2	16	
GF4	1500	2	36	
GF2	4000	2	80	
GF1	10000	2	180	
PGF4	1200	2	36	Sonderfunktionen
PGF6	750	2	24	Sonderfunktionen
AGF6	500	1,2	24	Sonderfunktionen
GF2.9	2000	1,2	16	
GF2.6	3000	1,2	24	
GF2.4	6000	1,2	36	
GF2.2	16000	1,2	80	Boundery-Scan
GF2.1	40000	1,2	180	Boundery-Scan

Tabelle 2: Gate Forest Familie

Der Zähler ist als 16 Bit breiter synchroner Binärzähler mit Rücksetz- und Freigabeeingang ausgeführt. Er unterteilt den Systemtakt für den Ausrollvorgang und ermöglicht den Betrieb des Würfels mit einem Standard-Uhrenquarz (ca. 32kHz).
Für den Oszillator ist ein frei verfügbarer Inverter mit Treiberstufe vorgesehen, so daß der Systemtakt mit wenig externen Komponenten erzeugt werden kann.
Für die Ansteuerung der Würfelaugen in Form von LED's werden Padzellen mit hoher Treiberfähigkeit von bis zu 12mA eingesetzt. Die übrigen Ein Ausgangszellen werden mit normalen Treibereigenschaften gewählt. Alle Ausgangszellen müssen zusätzlich über Padbuffer angesteuert werden.

Entwicklungsumgebung

Die Entwicklungsumgebung besteht aus einer Workstation vom Typ Apollo DN3500 mit Entwicklungssoftware Version 7.0 der Firma Mentor Graphics und Ausgabemöglichkeit auf Postscript Laserdrucker bzw. HPGL - Plotter. Die Schaltung wird auf einem IMS Gate Forest in 2um Technologie plaziert und geroutet. Dazu steht eine Zellbibliothek mit statischen Zellen zur Verfügung, sowie ein vom Institut für Mikroelektronik an der Fachhochschule Ulm aufbereiteter Entwicklungskit.

Chipentwurf

Der Entwurf erfolgt anhand des dargestellten Zyklusdiagramms. Die Schaltung wird in logische Blöcke unterteilt, und als Schaltplan (NETED) eingegeben. Die logische Simulation der Einzelblöcke erfolgt mit dem Digitalsimulator (Quicksim). Nach erfolgreichem Test der Einzelblöcke wird die gesamte Schaltung auf oberster Hierarchieebene simuliert, wobei hier schon lastabhängige Verzögerungszeiten der Bibliothekszellen und geschätzte Verzögerungen von Verbindungsleitungen berücksichtigt werden können. Vor dem eigentlichen Layout der Schaltung ist dies jedoch nur dann sinnvoll, wenn der Entwurf schwer einschätzbare laufzeitkritische Pfade enthält.

Dieser Teil des Entwurfszyklus nimmt ca. 80% der Gesamtentwicklungszeit in Anspruch, da hier die Umsetzung der Logik zum physikalischen Layout ohne besondere Anforderungen zu bewerkstelligen ist.

Die Anzahl verwendeter Logikzellen und der daraus resultierenden etwaigen Anzahl logischer Gatterequivalente bzw. Anzahl benötigter physikalischer Ein-Ausgänge führt zu einer Vorauswahl des Gate-Arrays. Die hier vorgestellte Schaltung benötigt 40 Ein-Ausgänge und 973 Sites - dies entspricht ca. 400 Gatterequivalenten, bezogen auf vergleichbare CMOS Prozesse (Tabelle 3) - .

Nach erfolgreichem Abschluß der simulierten Netzliste, die nun frei von Logikfehlern sein sollte, wird die Schaltung auf einem Chip vom Typ Gate Forest Gfxx9 mit 1977 nutzbaren Sites zu 100% automatisch plaziert und geroutet. Die Plazierung und das Routing erfolgt mit den Programmen "Gateplace" und "Gateroute". Laufzeitänderungen, die routingbedingt auftreten, werden mit dem Programm Add_Delay erfaßt und bei einer Nachsimulation berücksichtigt.

Benennung	Anzahl	Gatterequivalente
Buffer (einfach)	7	2
Buffer (vierfach)	23	3
Inverter	2	1
2-input Mux	19	3
2-input Nand	7	1
3-input Nand	3	2
4-input Nand	2	2
2-input Nor	1	2
4-input Nor	1	2
2-input Or	3	2
2-input And	32	2
2-input Xor	16	3
D Flip Flop	26	6
		437

Tabelle 3: Verwendete Zellen und Chipausnutzung im Vergleich zu einem HITACHI CMOS Gate Array HG62E

Einsatzgebiete für dynamische Zellen im Gate Forest sind umfangreiche kombinatorische Logikschaltungen, wie mehrstufige Addierer und Multiplizierer. Deren geschickte Plazierung und Verdrahtung bringt eine bessere Flächenausnutzung auf dem Chip als auch eine merkliche Laufzeitverbesserung an kritischen Pfaden. Der Gewinn an Fläche und Laufzeit steht dann auch im Verhältnis zum Mehraufwand bei der Aufbereitung und Handhabung des Systemtaktes.

Durch den Einsatz primitiver Standardzellen wie And, Or usw. kommt es im Gate Forest zu der für Gate-Arrays typischen Reihenanordnung der Logikzellen und der zwischen den Reihen liegenden Verdrahtungskanäle. Der Übergang zum frei plazierten Standarzellendesign vollzieht sich sichtbar erst bei

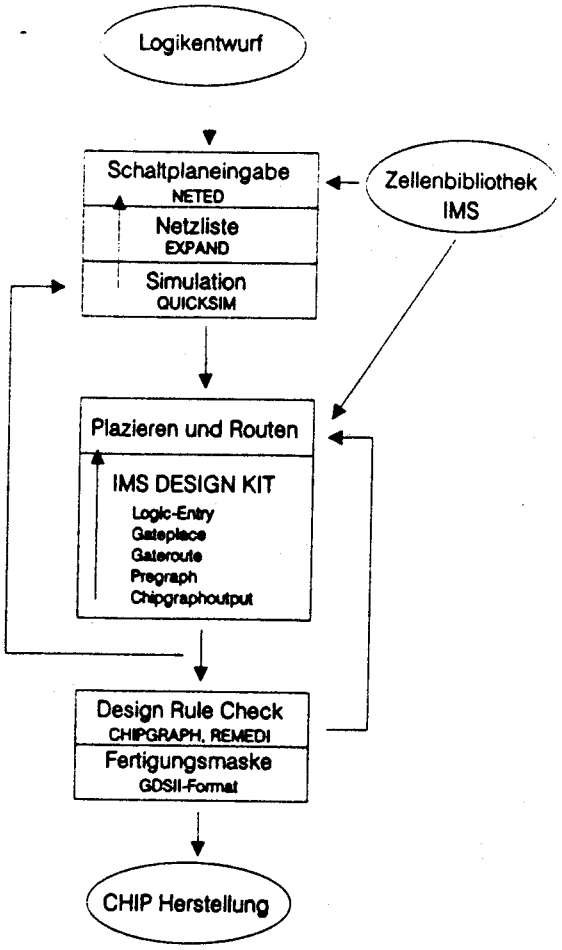


Bild 7:
Entwurfszyklus

Obwohl der Gate Forest als Einsatzgebiet für gemischt statisch/dynamische Standardzellen vorgesehen ist, werden hier nur Zellen aus der statischen Zellbibliothek eingesetzt.

höherkomplexen Zellen, die auf die "Site"-Architektur des Gate-Forest abgestimmt sind.

Änderungen am Layout werden anhand der Programme "Gategraph" oder "Chipgraph" vorgenommen. "Gategraph" ist eine grafische Oberfläche, die eine

interaktive Bearbeitung des Layouts ermöglicht. "Chipgraph" ist ein grafischer Polygoneditor, der für die Bearbeitung von Full-Custom-Schaltungen auf Maskenebene eingesetzt wird. Da "Chipgraph" alle Freiheitsgrade zur Abänderung des plazierten und gerouteten Designs bietet, sind Änderungen am Layout hier nur unter sehr guter Kenntnis des Chipaufbaus zu empfehlen.

Unter der Chipgraph-Oberfläche wird ein Design-Rule-Check mit dem Programm "Remedi" durchgeführt, der den Entwurf auf die vom Fertigungsprozeß vorgegebenen Längen- und Breitenmaße überprüft.

Abschließend werden die geprüften Layoutmasken im GDSII-Format an die Firma IMS zur Fertigung weitergegeben.

Inbetriebnahme und Test

Die Lieferung der gefertigten und gehäusten Chips erfolgt ca. 4 Monate nach Abgabe der Fertigungsmaske und des Bondplans.
Die Inbetriebnahme erfolgt blockweise entsprechend dem Blockschaltbild der Gesamtschaltung auf einem dafür aufgebauten Testboard. Ein erster Testdurchlauf des im CLCC - Gehäuse gelieferten IC's liefert die für jeden Block gewünschte Funktion. Die Eingabe von Testpattern ist im einfachen Testaufbau nicht vorgesehen, da die Funktion der Logikblöcke leicht nachvollziehbar ist.

Zusammenfassung

Anhand eines einfachen Schaltungsbeispiels wurde der Entwurf eines anwenderspezifischen Schaltkreises vom Logikentwurf bis zur Fertigungsmaske aufgezeigt. Der Entwurf der Schaltung erfolgt nach Entwurfsmethoden der digitalen Schaltungstechnik. Der Entwurf des ASIC erfolgt mit Softwaretools auf Workstation- Ebene über die simulierte Netzliste hinaus bis hin zur eigentlichen Fertigungsmaske für einen Gate Forest. Eine erfolgreiche Funktionsprüfung des gefertigten Chips schließt den Entwurfszyklus ab. Die Durchlaufzeit bis zum gefertigten IC beträgt ca. 5 Monate einschließlich der Entwurfszeit. Die Arbeit ist von einem Studenten ab dem 5. Semester im Rahmen einer Seminarsarbeit durchführbar.

Literaturhinweise:

Scriptum Schaltungstechnik
Herr Prof. Dr. Ing. Dirk Jansen

Herstellerunterlagen:
Institut für Mikroelektronik Stuttgart
Mentor Graphics Europe GmbH

Die Arbeiten erfolgten im Rahmen des ASIC-Labors unter Leitung von Herrn Prof. Dr. Ing. Dirk Jansen. Teile der Schaltungsauslegung erfolgten in einer von Herrn Schmidtke durchgeführten Studienarbeit. Die Fertigung des Chips wurde aus dem Schwerpunktprogramm Mikroelektronik Baden Würtemberg gefördert.

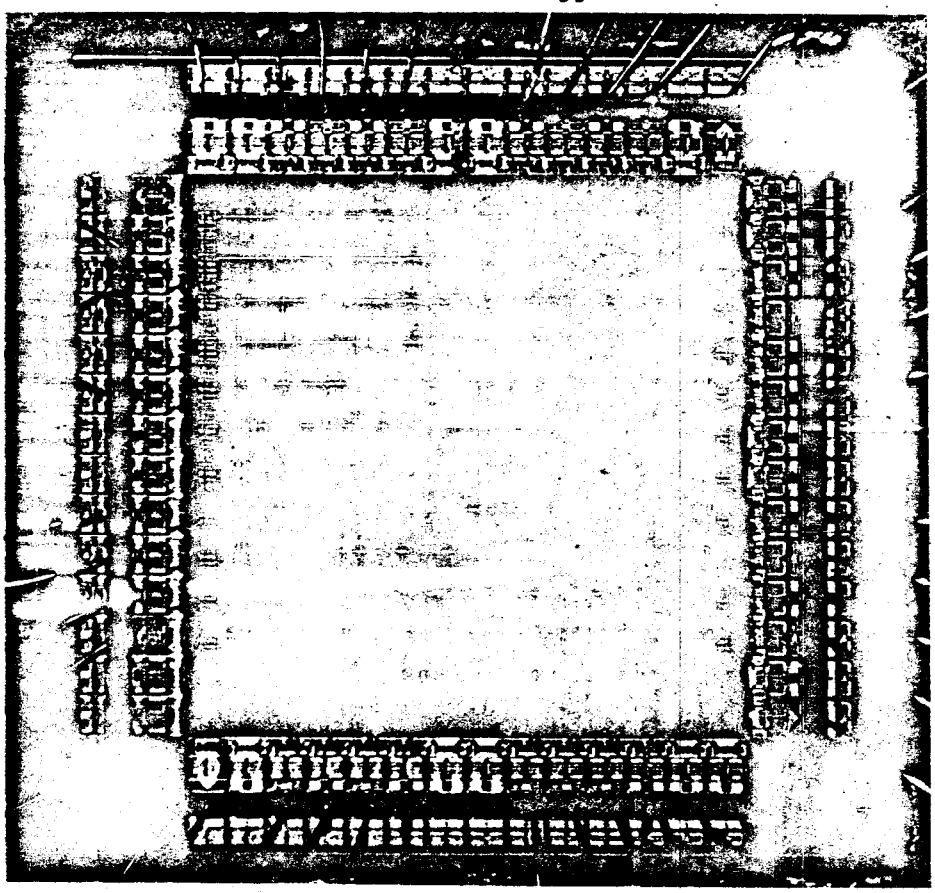


Bild 8:
Elektronischer Würfel
auf dem Gate Forest GF9

11. Digitaler Phasenregelkreis mit numerisch gesteuertem Oszillator als LCA - Microcontroller Kombination

Prof.Dr.-Ing. Dirk Jansen, Dipl.-Ing.(FH) Bernd Reinke, Cand. Dipl.-Ing.(FH) Andreas Rendler

Am Beispiel einer Schrittmotor - Indexerschaltung wird der effektive Einsatz von konfigurierbaren Logic Cell Arrays in Zusammenwirkung mit einem Mikrokontroller demonstriert, wobei die hohe Arbeitsgeschwindigkeit des LCAs den Bereich der Schaltung mit hohen Taktfrequenzen abdeckt, während der Controller die Steuerung und Überwachung der Schnittstelle nach außen übernimmt und im Regelkreis die arithmetrische Berechnung durchführt. Die Konfiguration des LCA aus dem EPROM des Controllers führt zu einer ungewöhnlichen Flexibiltät des Entwurfs und ermöglicht zahlreiche andere Anwendungen mit dieser Architektur.

1. Einführung

Für die Ansteuerung von Schrittmotoren, die an ihrer Welle mit größeren Trägheitsmomenten belastet sind, werden Indexer benötigt, die ein sanftes Erhöhen der Schrittmotoransteuerfrequenz bis zum Schnellgang und beim Abbremsen ebenfalls eine definierte Frequenzerniedrigung bis zum Stillstand realisieren. Die Anstiegs- und Abfallrampe soll dabei je nach Motor, Lastmoment und Anwendungsfall exponentiellen oder linearen Charakter aufweisen (Abb1-1). Die Zeitkonstanten müssen parametrisch einstellbar sein.

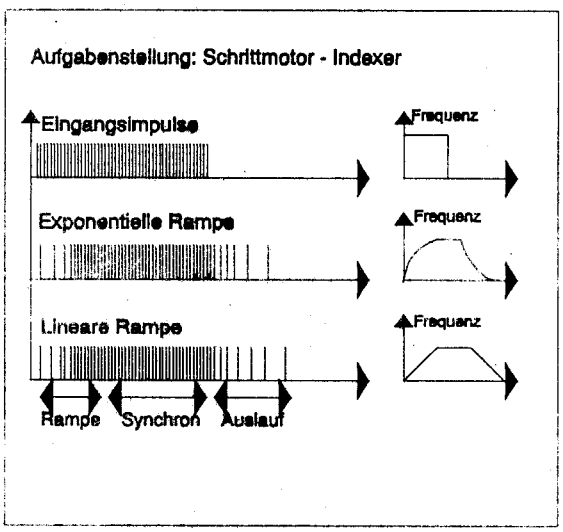


Abb 1-1: Impulsverläufe beim Schrittmotor - Indexer

Im vorliegenden Fall soll der Indexer mit der Motoransteuerung einen gemeinsamen Modul bilden. Die Ansteuerung erfolgt von einer übergeordneten Steuerung durch einen Impulszug konstanter Frequenz. Der Indexer muß nun diesen Impulszug aufnehmen und einen verzögerten Rampenanstieg generieren, wobei die Endfrequenz der Rampe der Eingangsfrequenz entsprechen muß. Beim Abschalten der Eingangsfrequenz soll eine entsprechende Abfallrampe generiert werden. Als wichtige Nebenbedingung muß die Zahl der Eingangs- und Ausgangsimpulse exakt identisch sein. Darüberhinaus wird ein Synchronlauf im eingerasteten Zustand erwartet, d.h. bei Änderung der Eingangsfrequenz muß sich auch die Ausgangsfrequenz entsprechend mitändern. Weitere Forderungen kommen hinzu:

- Anzahl der Impulse muß gleich sein,
- Synchronlauf,
- Rampe exponentiell /linear,
- Rampe parametrisch einstellbar,
- RS 232 Schnittstelle,
- Zusatzsignale wie Endschalter, Richtung,
- Handbetrieb langsam/schnell.
- Frequenzbereich bis 100 kHz.

2. Architektur und Funktion

Die Realisierung aller dieser Funktionen mit einem Microcontroller ist wegen der hohen Betriebsfrequenz unmöglich, andererseits ist ein ASIC durch die Forderung nach Parametrisierbarkeit, RS 232 - Schnittstelle und Funktionskomplexität überfordert. Als Basis für die Architektur des Systems wurde deshalb eine Kombination aus ASIC, realisert durch einen programmierbaren Logic Cell Array (LCA) der Fa XILINX, und einem Micro-

controller, dem weitverbreiteten 87C51 der Fa INTEL, gewählt (Abb 2-1).

Hierbei übernimmt der LCA die Funtionsbereiche hoher Taktfrequenz, enthält damit auch den steuerbaren Oszillator NCO, während der Controller im wesentlichen die Schnittstelle bedient und Regelfunktionen im LCA durchführt.

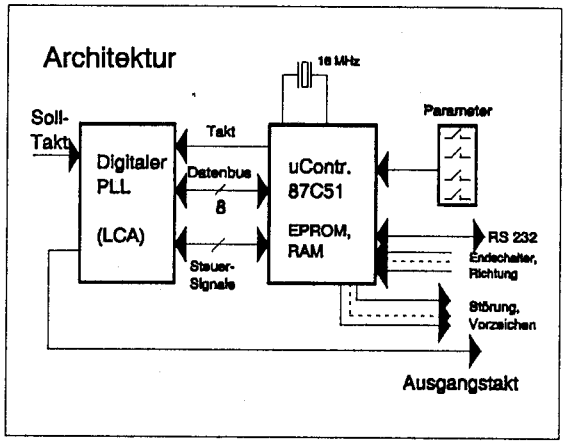


Abb 2-1: Architektur des Indexers

Die Aufgabenstellung der Taktnachführung und Synchronisierung kann am besten durch einen Phasenregelkreis (PLL) gelöst werden, der aus den Funktionsblöcken

- Phasendetektor
- Schleifenfilter (Regler)
- gesteuertem Oszillator

besteht Abb 2-2.

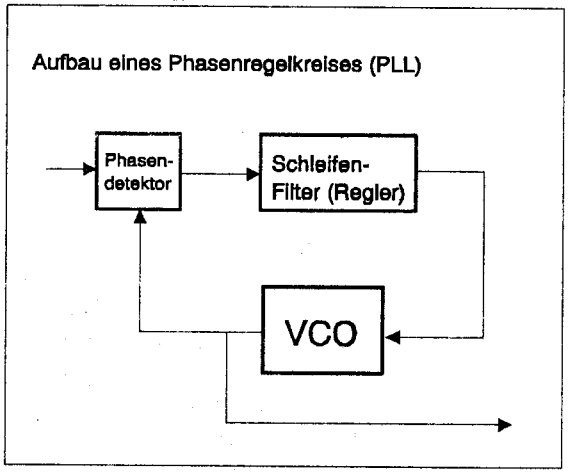


Abb 2-2: Aufbau eines Phasenregelkreises (PLL)

Aufgabe des Phasedetektor ist es hierbei, die Soll - Eingangsfrequenz mit der Motortaktfrequenz zu vergleichen. Ist die Motorfrequenz niedriger, ist dies anzuzeigen, sodaß über den Regler der interne Oszillator in seiner Freuenz erhöht wird, ist sie niedriger, ist die Oszillatorfrequenz abzusenken. Bei etwa gleicher Frequenz führt dies zum Einrasten der Phase, sodaß Eingangs- und Ausgangsfrequenz miteinander synchron sind.

Eine Besonderheit ist dabei, daß der Phasendetektor in der Lage sein muß, auch große Phasenverschiebungen, die ein Vielfaches der Taktperiode betragen, zu speichern und dem Regler zugänglich zu machen.

Die PLL - Funktionen müssen, um sie in einem rein digital arbeitenden LCA implementieren zu können, alle digital realisert werden. Für den Phasedetektor wird deshalb das Konzept eines Zustandschaltwerk in Zusammenwirken mit einem Auf/Ab - Zähler verwendet, der Oszillator wird als NCO (Numerical Controlled Oscillator) realisiert. Der Regler erfordert komplizierte arithmetrische Funktionen, für die Exponentialrampe z.B. eine Mutiplikation, und muß deshalb im Microcontroller als Programmmodul realisiert werden. Abb 2-3 zeigt ein Blockschaltbild der im LCA realisierten Teile des PLL.

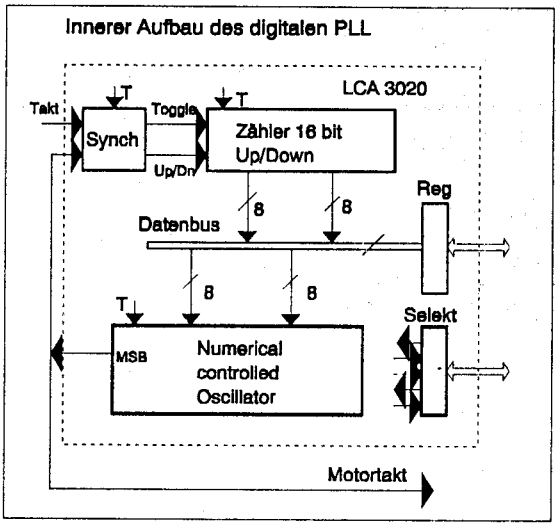


Abb 2-3: Blockschaltbild des LCA

Das Ergebnis des Phasenvergleichs der Eingangs- und Ausgangstaktfrequenz spiegelt sich im Zählerstand des Up/Dn - Zählers wieder, der unmittelbar die Differenz der Anzahl der Flankenübergänge enthält. Der Zähler ist als 16 - Bit Zähler ausge-

führt und kann deshalb bis zu 65536 Takte Vorlauf von Eingangstakt gegenüber Ausgangstakt speichern. Damit können auch sehr große Hochlauf - Zeitkonstanten von mehreren Sekunden Anlaufzeit, wie sie z.B. bei Aufwickeltrommeln auftreten können, problemlos überbrückt werden. Sollte es trotzdem zu einem Überlauf des Phasenzählers kommen, wird dies durch die Überwachung des Zähler - Carry - Ausgangs, der zu einer Fehlermeldung führt, angezeigt Abb 2-4.

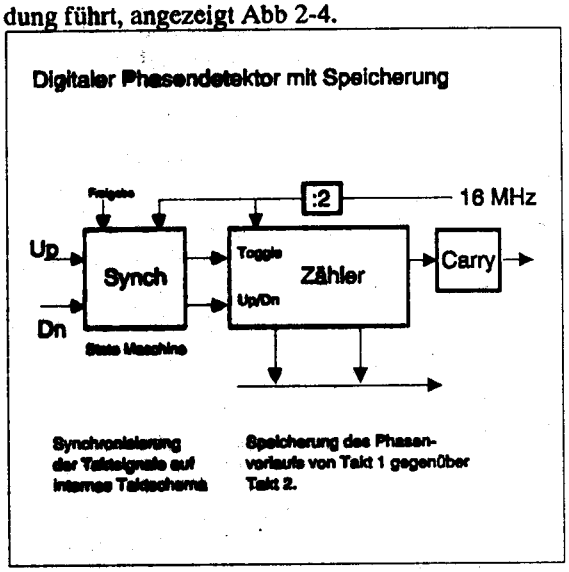


Abb 2-4: Digitaler Phasendetektor mit Speicherung

Da mit steigendem Zählerstand die Ausgangsfrequenz des NCOs erhöht wird, stellt sich irgendwann ein Gleichgewicht ein, dessen Höhe vom Faktor zwischen Zählerstand und NCO - Frequenzansteuerung abhängt. Je niedriger dieser Faktor ist, desto länger dauert es, bis der Gleichgewichtsstand erreicht ist, was einer großen Anlaufzeitkonstante entspricht. Eine einfache Multiplikation des Inhalts des Phasendifferenzzählers mit einem konstanten Faktor führt damit zur exponentiellen Rampe.

Im Gleichgewichtszustand stellt sich durch Phasenvergleich der Eingangssignale ein Synchronlauf ein.

Bei Abbruch des Eingangssignals verringert sich der Zählerstand des Phasendifferenzzählers sofort durch die weiter einlaufenden Abwärts - Zählimpulse des Motors, wodurch sich auch die Frequenz des NCO erniedrigt, bis schließlich bei Zählerstand Null die Ausgangsfrequenz ebenfalls Null beträgt. Es ist leicht zu zeigen, daß die Gesamtzahl der Eingangsimpulse und der Motoransteuerimpulse exakt gleich sind, was der oben dargestellten Forderung an einen Indexer entspricht. Abb 2 - 5 zeigt das Blockschaltbild des digitalen PLL.

Im Falle der linearen Rampe ist die Steuerung des NCO komplizierter. Die Erhöhung der NCO - Frequenz erfolgt durch zeitgesteuerte incrementale Erhöhung um einen voreingestellten Wert, bis die Sollfrequenz erreicht ist. Dies wird durch Vorberechnen des Phasendifferenzwertes und Kontrolle des Phasendifferenzzählers erreicht. Die Phasensynchronsierung erfolgt ähnlich wie bei der exponentiellen Rampe. Die Abwärtsrampe wird ähnlich wie die Aufwärtsrampe durch kontinuierliche Subtraktion eines Increments linear gesteuert. Bei Null des Phasendifferenzzählers wird die Ausgangsfrequenz angehalten. Die Steuerung ist insgesamt komplizierter als im exponentiellen Fall, der Vorgang ist jedoch wegen der zeitoptimalen Schrittmotorbewegung von besonderem Interesse in der Anwendung.

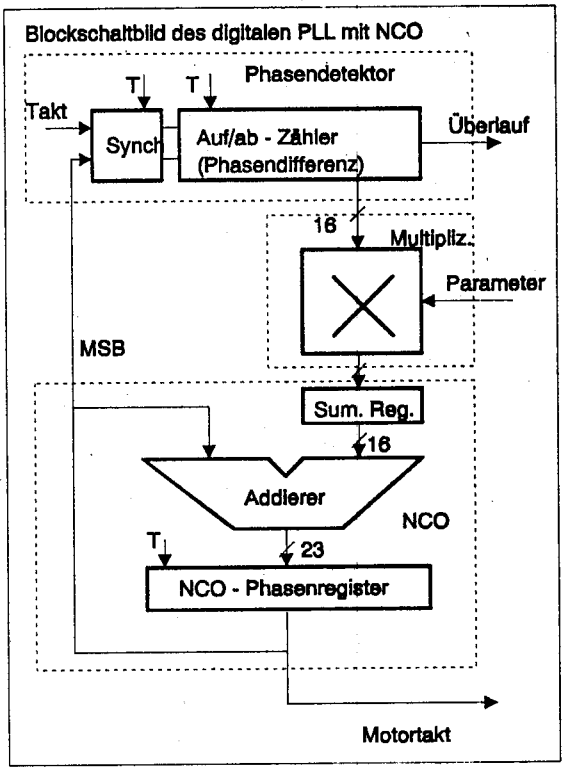


Abb 2-5: Blockschaltbild des PLL mit Exponential - Rampen - Regler

Anstelle der intern rückgeführten NCO - Signale kann natürlich auch ein am Schrittmotor angebauter incrementaler Wegaufnehmer verwendet werden, sodaß eine Closed - Loop - Steuerung des Motors möglich wird. Bei einer kurzzeitigen

Blockade des Motors kommt es damit nicht zu Schrittverlusten.

3. Einzelheiten der Realisierung

Der Phasendetektor besteht aus einem Zähler und einem sequentiellen Schaltwerk (State - Maschine), welches die Synchronisierung der externen Takte auf den internen Takt von 16 MHz bzw hier 8 MHz bewirkt. Der Up/Dn - Zähler ist ein synchroner, mit 8 MHz getakter Zähler mit den Steuereingängen Richtung (Up/Dn) und Zählen (Toggle). Der primäre 16 MHz Takt konnte nicht verwendet werden, da die Durchlaufzeit des Zählers mit 93 nsec zu groß ist. Die 8 MHz Taktfrequenz ist gegenüber der höchsten geforderten Eingangsfrequenz von 100 kHz völlig ausreichend. Der Zähler ist über 2 tristate - Verbindungen von je 8 Bit auf den internen Datenbus des LCA schaltbar und damit vom Microcontroller jederzeit auslesbar.

Der NCO mußte in einer Form realisiert werden, bei der die Ausgangsfrequenz linear durch ein digitales Wort einstellbar ist. Die entsprechende Struktur zeigt Abb 3-1:

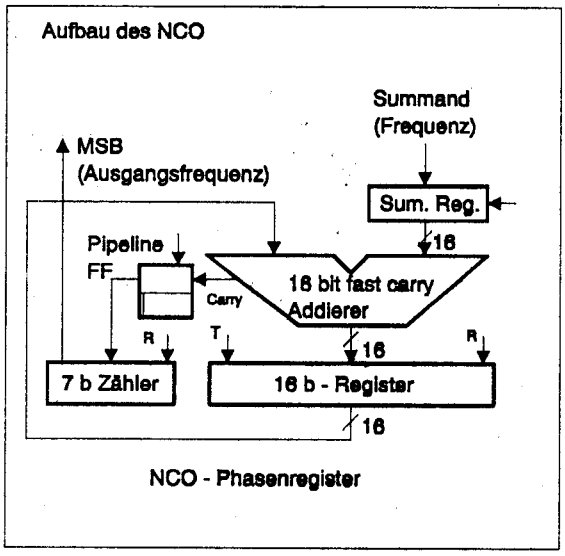


Abb 3-1: Architektur des numerisch gesteuerten Oszillators (NCO)

Die einzustellende Frequenz wird in das Summanden-Register mit 16 bit Breite in 2 Taktzyklen über den Datenbus geschrieben. Ein 16 bit Fast Carry - Addierer addiert diesen Summanden laufend zum Inhalt eines Akkumulators der Breite 23 bit hinzu. Damit ergibt sich ein incrementaler, rampenförmiger Anstieg des Inhalts des Akkumulators Abb 3-2.

Ist die Summe größer als 2^{23}, läuft der Akkumulator über und die Rampe beginnt von neuem. Greift man nun das MSB (Most Significant Bit) des Akkumulators ab, so zeigt dieses eine Frequenz, die direkt proportional der Größe des Increments, d.h. des Summanden ist. Der Accumulator wird auch als Phasen - Akkumulator oder Phasenregister bezeichnet, da der Inhalt der Phase des so generierten Taktsignals entspricht.
Bei einer primären Taktfrequenz von 16 MHz ergibt sich hieraus folgende Ausgangsfrequenz:

$$f_{out} = \frac{16\ MHz}{2^{23}} * Increment$$

was einer Frequenz von 2 Hz ..... 125 kHz bei einem 16 bit Summanden entspricht.

In der Schaltung kann die geringere Eingangsbreite des Summanden von 16 bit gegenüber dem Akkumulator von 23 Bit vorteilhaft genutzt werden. So muß der Addierer nur noch 16 bit breit sein, ebenfalls der eigentliche Akkumulator, die oberen 7 Bit - Stellen können durch Ausführung des Akkumulators als 7- stufiger Zähler , der die aus dem unteren Bit - Bereich kommenden Überträge zählt , realisiert werden.

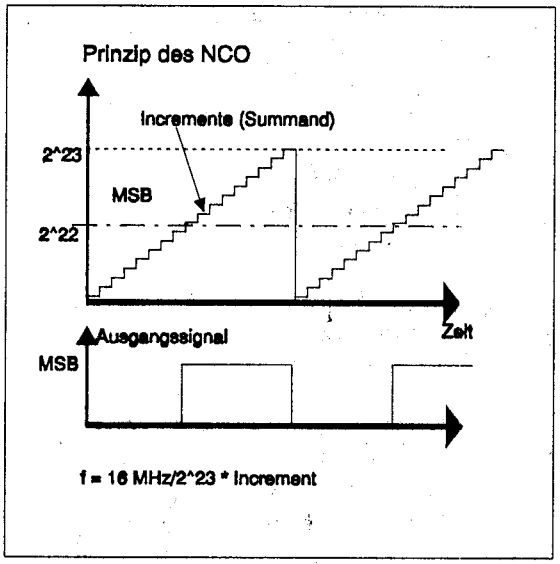


Abb 3-2: Funktionsprinzip des NCO

Die genaue Analyse der Durchlaufzeiten ergab für das Carry des 16 bit Addierers trotz Fast - Carry - Struktur noch 53,3 nsec, was bei einer Taktzykluszeit von 62,5 nsec zu lang war und insbesondere zur Steuerung des Carry - Zählers nicht mehr gereicht hätte. Es wurde deshalb ein Pipeline - FlipFlop in die Carry - Leitung eingefügt, was zu einer Latenzzeit des Ausgangs von einem 16 MHz - Taktzyklus führt, die Laufzeitproblematik jedoch vollständig löst. Da der Inhalt des Flipflops nicht auf die Addition zurückgeführt wird, bleibt auch die obere Taktgrenze von 125 MHz erhalten.

4. Aufgaben des Microcontrollers

Der Controller bearbeitet zahlreiche wichtige Funktionen, die hier nur stichpunktartig aufgeführt werden können:

Regler in der PLL - Funktion. Der Microcontroller ist als Regler wie oben beschrieben in die PLL Funktion integriert. Dies ist möglich, da die Veränderung der NCO - Frequenz relativ langsam erfolgt, was der Arbeitsgeschwindigkeit des Controllers entgegenkommt.

Initialisierung und Konfigurierung des LCAs über den Datenbus. Die Konfigurationsdaten sind dazu im EPROM des Controllers abgelegt. Die Konfiguration erfolgt nach dem Reset des Controllers, z.B. beim Einschalten der Versorgungsspannung.

Berechnung der Rampenparameter aus den Einstellwerten. Die Einstellung der Zeitkonstanten für Hoch- und Auslauframpe erfolgt über Hex - Drehschalter, die bei der Initialisierung einmalig abgefragt werden. Diese Werte sind in einer leicht verständlichen Form kodiert. Sie werden durch das Programm in jene Werte umgewandelt, die zur Durchführung der Regelung erforderlich sind. Über die Schalter werden zudem die Betriebsmodi (Exponential- oder Linear- Rampe) und die zu verwendende Schnittstelle eingegeben.

Bedienung der RS 232 Schnittstelle. Anstelle der Hex - Schalter kann die Parametereingabe auch über eine RS 232 - Schnittstelle erfolgen, was die Kontrolle des Moduls durch einen übergeordneten Leitrechner ermöglicht.

Überwachung der Peripherie. Der Controller überwacht laufend Signale wie Endschalter, Umschaltung auf Handbetrieb, Richtungsumkehr u.s.w., die zu entsprechenden Reaktionen führen. So kann z.B. im Handbetrieb eine Schleichgang - Frequenz oder eine Schnellgang - Frequenz mit Hochlauf kommandiert werden. Für den Testbetrieb ist zudem ein separater Oszillator, realisert durch einen Standard - Timer, auf der Karte implementiert, dessen Frequenz ebenfalls eingestellt werden kann und der eine einfache Integration des Moduls in eine Standardanwendung ermöglicht.

Fehlermeldung und Fehleranzeige. Die Eingabe unsinniger Werte, Zählerüberlauf und andere Fehlfunktionen führen zu einer Fehleranzeige durch LEDs und elektrische Signale, die von einem Leitrechner weiterverarbeitet werden können.

LCA und Controller sind einerseits durch den Datenbus von 8 bit Breite, andererseits durch die Steuer- und Handshakesignale verbunden. Der LCA hat damit die Bedeutung eines Standard - Peripherie - Bausteins wie eines Timers oder Speicherbausteins und benötigt ähnliche Steuersignale wie Adresse , Read /Write und Takt. Controller und LCA arbeiten miteinander synchron.

Auf der Platine Abb 4-2 sind weiterhin vorhanden:

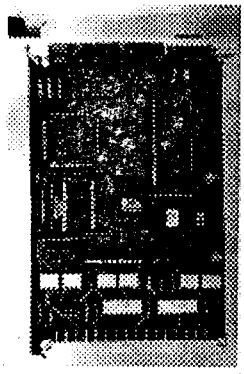


Abb 4-2 : Foto der realisierten Baugruppe

Optokoppler zur galvanischen Entkopplung von Ein- und Ausgängen. Die Forderung nach 24 V - Ausgängen erforderte hier einen besonderen Aufwand.

Watch - Dog für die Kontrolle des Microcontrollers, bei Ausfall der Ansteuerung des Watch - Dogs wird automatisch ein Reset generiert.

Pegelumsetzung für die RS 232 - Schnittstelle. Ein Baustein erzeugt die standardisierten Pegel der RS 232 Schnittstelle.

Testfrequenzgenerator zur Erzeugung von Testfrequenzen und zur Vereifachung der Integration in einfache Schrittmotorsteuerungen.

Anzeige- und Bedienelemente in Form von LEDs und Hex - Schaltern zur Warnanzeige und zur Parametereingabe.

Die Baugruppe ist komplett auf einer Europakarte aufgebaut.

5. Entwicklungsgang

Der Indexer - Modul wurde unter Zuhilfenahme zahlreicher CAE - Programme entwickelt. Nach einer ersten Simulation der Grundfunktion des PLL mit exponentieller und linearer Rampe auf PASCAL - Ebene erfolgte zunächst die Entwicklung des als zeitkritisch angesehenen LCAs. Die entsprechenden Schaltungen wurden deshalb auf MENTOR - Workstations mit dem Programm NETED eingegeben, wobei bereits die XILINX - Bibliothek verwendet wurde.

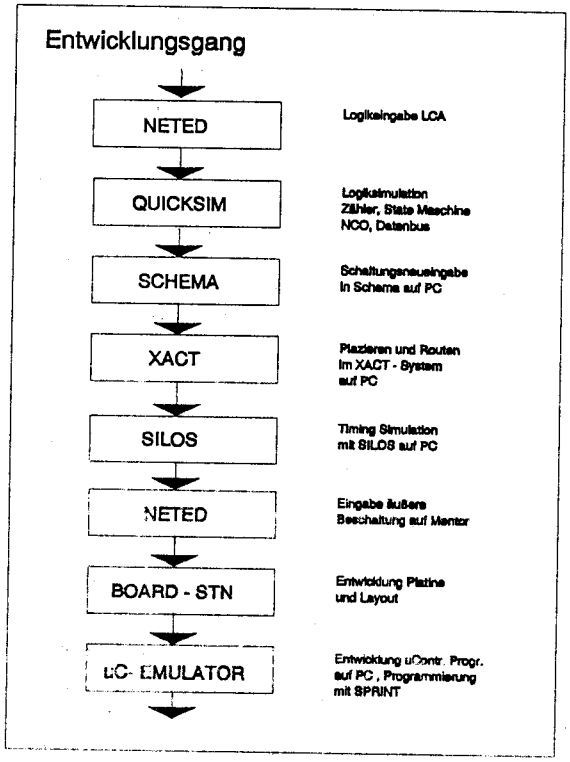


Abb 5-1: Entwicklungsgang des Schrittmotor - Indexers.

Anschließend erfolgte die digitale Simulation mit Quicksim, wobei allerdings die Microcontroller-funktion vereinfacht angenommen werden mußte, da ein Simulationsmodell für den Controller nicht zur Verfügung stand.

Nach Festlegung der LCA - Schaltung erfolgte eine Neueingabe in SCHEMA auf PC, da hier das XILINX - Entwicklungssystem XACT installiert war und keine Netzlistenkompatibilität gegeben war. In XACT erfolgte auch das Plazieren und Routen, wobei von den Möglichkeiten der Vorplazierung und der Festlegung zeitlich kritischer Verbindungen starker Gebrauch gemacht wurde.

Mit Hilfe des zum XACT - System gehörenden Digitalsimulators SILOS wurden dann ausgewählte Signale nach der Plazierung auf Laufzeiten überprüft, was diverse kleinere Änderungen und Vorplazierungen in SCHEMA erforderte. Dieser Zyklus mußte mehrfach durchlaufen werden, bis befriedigende Ergebnisse erzielt werden konnten. Die LCA - Konfiguration konnte damit erzeugt werden.

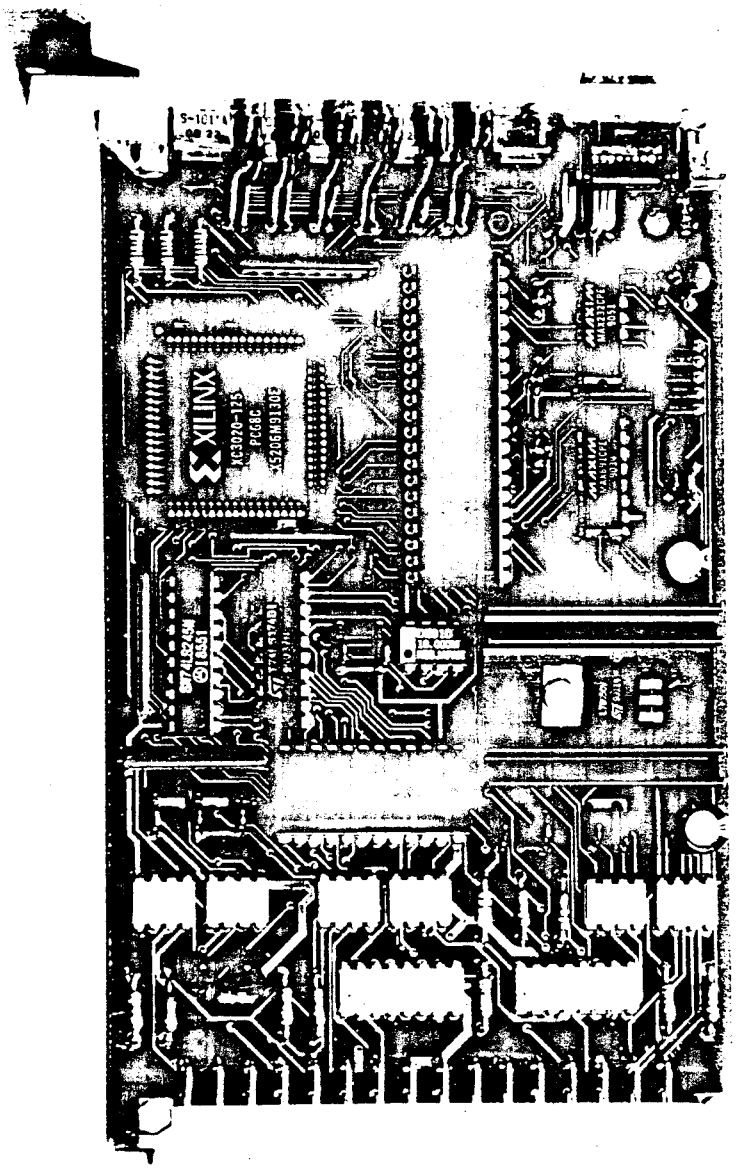
Mit Hilfe von selbst erstellten Programmen wurde schließlich aus den Jedec - Konfigurationsdaten des LCA ein File generiert, der sich zum Einbinden in das Programm des Microcontrollers eignete. Die LCA - Entwicklung war damit abgeschlossen.

Zur Entwicklung der Platine wurde die äußere Beschaltung von Microcontroller, LCA und aller Zusatzbausteine mit NETED auf MENTOR - Workstations eingegeben und das Layout der Platine mit Hilfe der BOARD - STATION - Software entwickelt .

Die Entwicklung der Microcontroller - Programme erfolgte dazu parallel auf einem Microcontroller - Emulator bzw Simulator auf PC, die Programmierung des Controllers schließlich auf SPRINT.

Inbetriebnahme und Erprobung erfolgte mit Labormitteln, u.a. einem hochwertigen digitalen Speicheroszillographen und einem Logic - Analysator. Die Schaltung erfüllt die erhobenen Forderungen.

Nachwort:
Die Entwicklung erfolgte im Labor Schaltungstechnik der FH - Offenburg unter Leitung des Autors. Wesentliche Teile der Schaltung und des Programms wurden in der Diplomarbeit des Herrn Andreas Rendler, betreut von Herrn Dipl-Ing.(FH) Bernd Reinke, erarbeitet. Die Aufgabenstellung kam von der Fa Eble, Offenburg, diie den Modul in Zukunft vertreiben wird. Allen Beteiligten sei hiermit für Einsatz und Unterstützung gedankt.


XILINX

12. Entwurf eines Fernsteuer-IC's als Studienaufgabe

Harald Töpfer
Fachhochschule für Technik Esslingen/Aussenstelle Göppingen
Fachbereich EM

Der Entwurf des Fernsteuer-Sender/Empfänger war Bestandteil der Vorlesung 'Chipdesign' im 7.Semester (SS92) des Fachbereichs EM.
Dazu wurde das Semester in Prakikumsgruppen zu je 3-4 Studenten aufgeteilt. Jede dieser Gruppen hatte jeweils eine Schaltung für Sender und Empfänger zu erstellen und mit dem Valid-Rapidsim zu simulieren.
Da zum Zeitpunkt des Semesterbeginns noch keine geeignete Library auf dem Valid-System vorhanden war, wurde auf der Grundlage der 'sim'-Elementare eine eigene 'CLIB' erstellt, die in reduzierter Form die wichtigsten logischen Grundelemente sowie Ein/Ausgabezellen enthält.

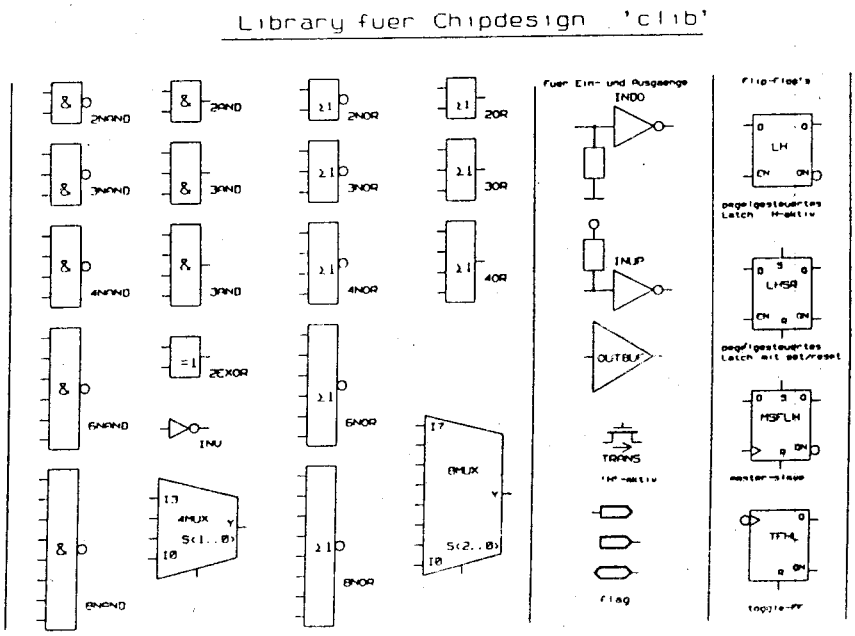


Bild 1: für Übung benutzte 'CLIB'

Der zu entwerfende Fernsteuer-Sender/Empfänger sollte folgende Eigenschaften haben:

- Sender und Empfänger mit jeweils 8 Adreseingängen (Bild 2)
- Sender generiert Adresse als serielles Telegramm mit Pulsphasenmodulation und Startbit

- nach jedem Starimpuls werden mindestens 2 Telegramme gesendet

- stimmen bei 2 aufeinanderfolgenden Telegrammen (Datensicherheit) Sender- und Empfängeradresse überein, wird am Empfänger der Ausgang Match=1

- die Adresseingänge des Senders sollen zum Zweck der Stromersparnis nur mit kurzen Impulsen abgefragt und gelatcht werden

Sender und Empfänger werden mit einem 32,768kHz-Taktgenerator betrieben. Um die Studenten für die Problematik 'Test' zu sensibilisieren, sollten zusätzlich für dem Sender Testpattern mit einer Länge von maximal 200 XT-Takten geschrieben werden, die gleichzeitig aber auch einen 100%-tigen Toggle-coverage gewähr-leisten. Zur Testbeschleunogung konnten deshalb beliebig viele Testeingänge verwendet werden.

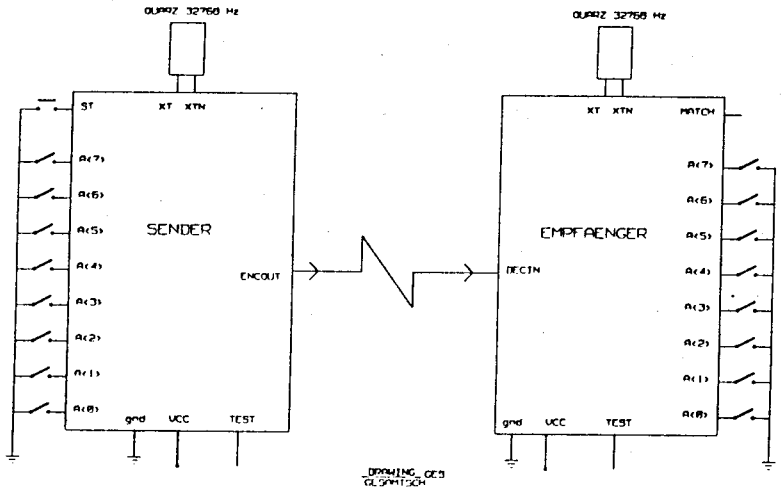


Bild 2: Schaltung Sender-Empfänger

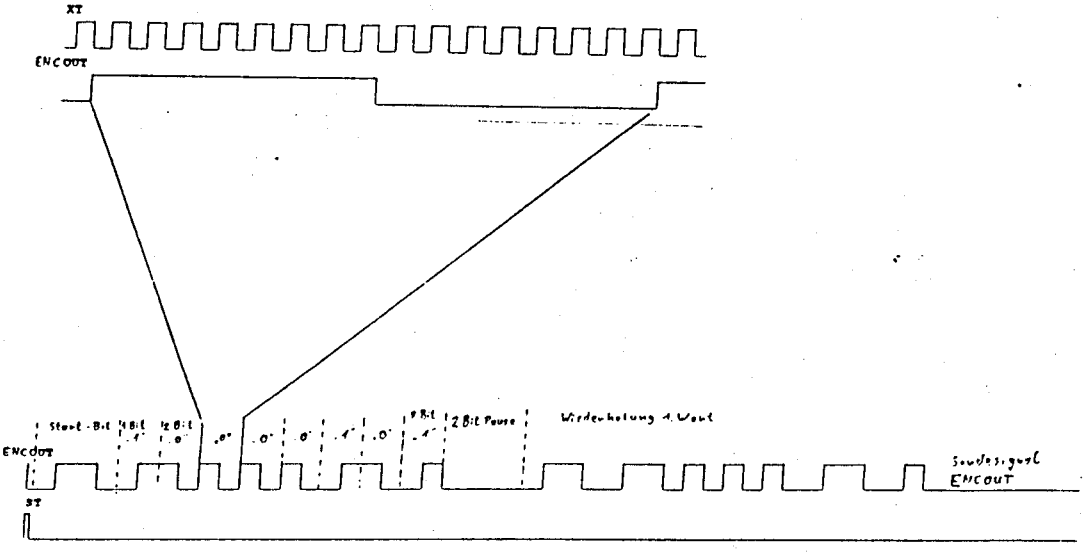


Bild 3: Ausgangsimpulse Sender

Die Funktion der Schaltung ist leicht verifizierbar: es braucht bei der Simulation nur auf das 'Match'-Signal geachtet zu werden.
Da die Studenten im Rahmen dieser Praktikumsübung erstmals mit einem Logiksimulator in Berührung kamen, gab es in den ersten Wochen zum Teil erhebliche Probleme mit dem 'Rapidsim'.
Es zeigte sich auch, daß es, trotz der 2-semestigen Logikvorlesung, bei einigen Studenten noch Defizite im Umgang mit Logikschaltungen gab.
Bewährt hat sich die Projektarbeit in 3er Gruppen, da ein einzellner Student mit dieser Aufgabe überfordert wäre.
Alle Gruppen haben eine Schaltung für Sender und Empfänger abgeliefert, wenn auch von sehr unterschiedlicher Qualität. Einige Gruppen konnten den Entwurf nur unter großen Anstrengungen bis zum Semesterende fertigstellen.
Generell hat sich diese Form der Projektarbeit bewährt, da hier der eigenschöpferische Anteil wesentlich größer ist als bei der sonst üblichen Simulation von Zählerschaltungen u.ä.
Allerdings sind die meisten Entwürfe nicht sofort integrationsfähig, da sie zum Teil noch Hazards oder Wettlaufbedingungen enthalten. Bild 4 zeigt ein Beispiel eines Studentenentwurfes.

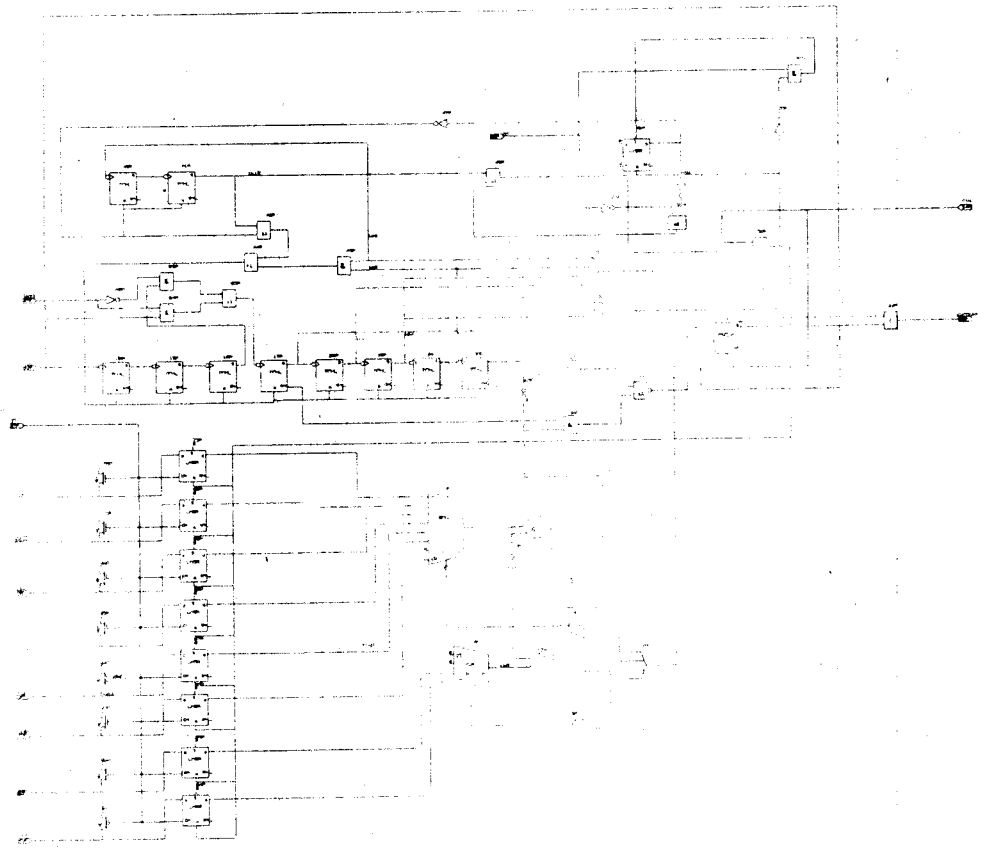


Bild 4: Studentenentwurf für den Fernsteuersender

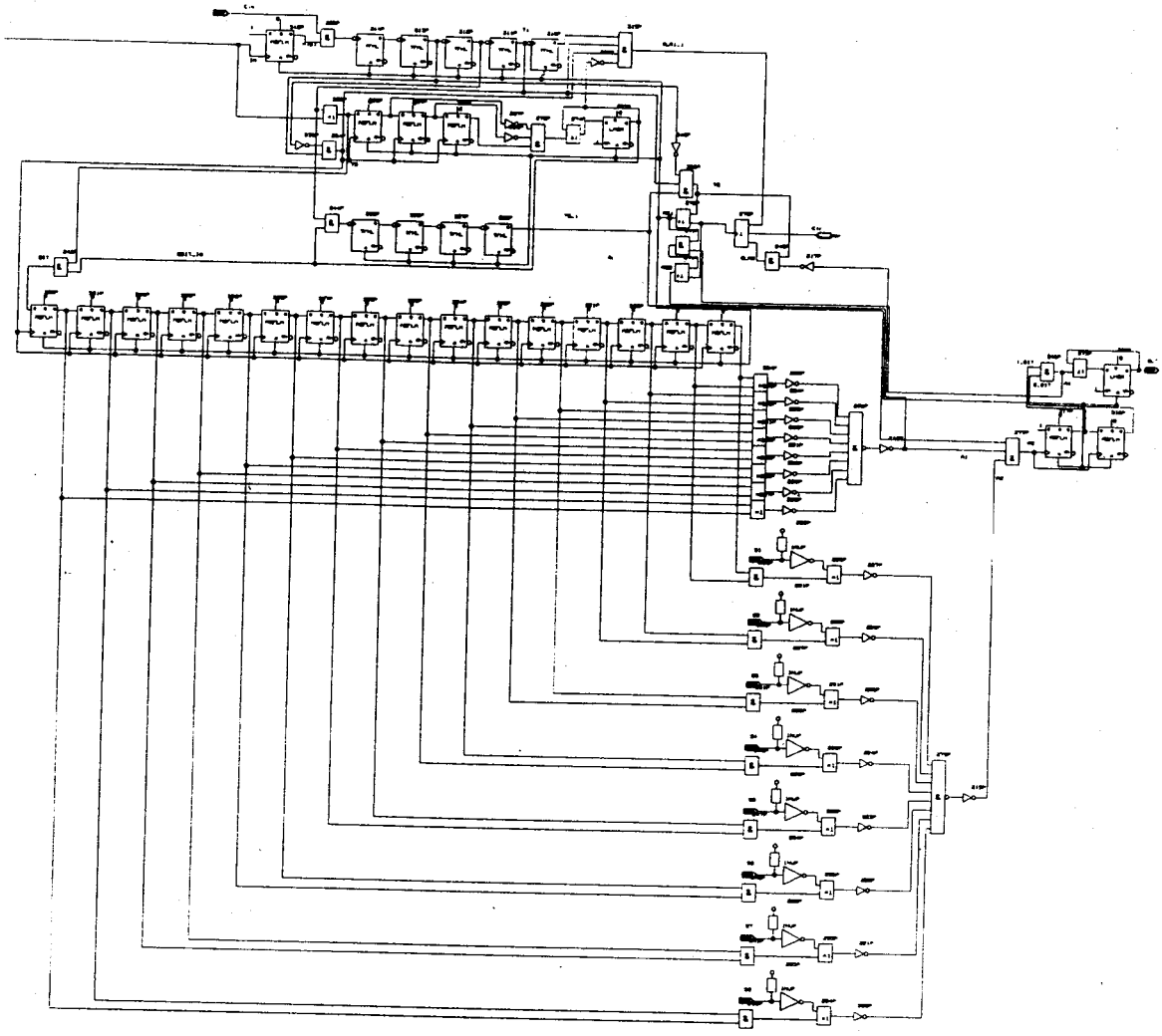


Bild 5: Studentenentwurf für Fernsteuerempfänger