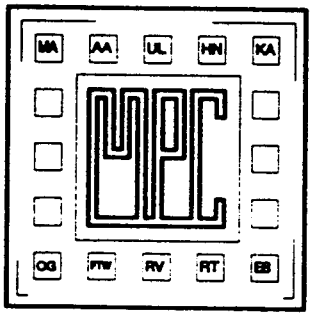
MULTIPROJEKT CHIP-GRUPPE

BADEN - WÜRTTEMBERG

WORKSHOP FEBRUAR 1993

REUTLINGEN



HERAUSGEBER: FACHHOCHSCHULE ULM

Inhaltsverzeichnis

Entwicklung elektronischer Systeme und Schaltkreise mit VHDL

Djamshid Tavangarian, Michael Koch

FernUniversität Hagen, Technische Informatik II
Postfach 940, 5800 Hagen 1
Tel. 02331 987 4412, Fax. 02331 64 373
E-mail: michael.koch@fernuni-hagen.de

1 Einleitung

Bei der Entwicklung digitaler hoch- (VLSI) und höchstintegrierter Schaltkreise (ULSI) haben sich die strukturierten und systematischen Entwurfsmethoden und -verfahren weitgehend durchgesetzt. Dabei wird die im Rahmen einer Schaltkreisentwicklung formulierte Aufgabenstellung mit hoher Komplexität in kleinere Einheiten zerlegt, um einerseits eine hierarchische Sicht über die Gesamtschaltung zu ermöglichen und andererseits die Komponenten überschaubar und einfach zu gestalten. Diese Vorgehensweise wird als Top-Down-Entwurf bezeichnet.

Ein wesentliches Hilfsmittel für den Entwurf sind Hardware-Beschreibungssprachen (HDLs). HDLs sind formale Sprachen, die zur strukturierten Beschreibung der Architektur von Hardware-Systemen verwendet werden. Unter der Architektur wird die Angabe der Ein- und Ausgangsanschlüsse, die Topologie eines Schaltkreises, also die Konnektivitätsbeschreibung der Schaltkreiskomponenten, und die Verhaltensbeschreibung eines Hardware-Systems verstanden.

Die HDLs werden beim Entwurf von Schaltkreisen als Eingabesprache von Simulatoren und Synthese-Programmen eingesetzt. Die strukturierte und präzise Beschreibung eines Hardware-Systems mit einer HDL unterstützt die Dokumentation des Systems und bildet eine Basis für eine effiziente Kommunikation zwischen Schaltungsdesignern.

Eine neue Entwicklung auf diesem Gebiet stellt die Hardware-Beschreibungssprache VHDL dar, die im Auftrag der US-Regierung im Rahmen des Projektes VHSIC (Very High Speed Integrated Circuits) entwickelt wurde [VHD87].

Ursprünglich sollte VHDL als eine gemeinsame Plattform zur Verständigung zwischen den Entwicklern innerhalb großer Entwurfsprojekte dienen. Inzwischen sind aber sowohl von der Industrie als auch von wissenschaftlichen Institutionen so viele Erweiterungen in VHDL eingeflossen, daß sie über fast alle für einen Schaltkreisentwurf notwendigen Merkmale verfügt.

Die erste Version der VHDL wurde 1985 unter der Bezeichnung VHDL 7.2 freigegeben. Eine spätere Version wurde 1987 mit einem völlig neuen Satz von Werkzeugen ergänzt. Im gleichen Jahr wurde die Sprache durch das "Institute of Electrical and Electronics Engineers" in den USA als IEEE-1076 standardisiert.

Der Einsatz von VHDL wird mit vielfältigen Vorteilen beim Entwurf integrierter Schaltkreise verknüpft. Sie unterstützt sowohl eine hierarchische als auch eine bibliotheksbasierte Entwicklung eines Schaltkreissystems. Eine technologieunabhängige Beschreibung eines Schaltkreises ist möglich. Damit können die zu realisierenden Schaltkreise in eine Systembeschreibung eingebettet

und mit den peripheren Schaltungsteilen simuliert und analysiert werden. VHDL stellt Sprachelemente zur Verfügung, die sowohl eine Verhaltens- als auch eine Strukturbeschreibung von Schaltkreisen erlauben. Diese Beschreibungen können in unterschiedlichen Abstraktionsebenen erfolgen. Die in VHDL verwendete Syntax und Semantik kann konsequent über alle Abstraktionsebenen hinweg verwendet werden. Durch die Standardisierung der VHDL können unter Einsatz konsistenter Bibliotheken die in dieser Sprache beschriebenen Systeme zwischen unterschiedlichen Werkzeugen transferiert, ausgetauscht und weiterverarbeitet werden.

Die heute verfügbaren VHDL-Simulatoren unterstützen vorwiegend die Entwicklung von Schaltkreisen auf Gatterebene. Die Basiselemente eines Schaltkreises auf dieser Ebene sind logische Gatter und Verbindungselemente (Leitungen).

2 Abstraktionsebenen

Bei der Beschreibung komplexer Schaltungen können hierarchisch gegliederte Abstraktionen der Schaltungskomponenten vorgenommen werden, die unterschiedliche Sichten einer Schaltung ermöglichen. Die unterschiedlichen Sichten einer Schaltung führen zur Bearbeitung von Schaltungen in unterschiedlichen Ebenen, den sogenannten Abstraktionsebenen. Diese unterscheiden sich in ihren beobachtbaren Objekten (z.B. Ströme, Spannungen, digitale Signale u.ä.) sowie den auf der Ebene verwendeten Modellierungs- und Berechnungsmethoden.

Unterschiedliche Beschreibungsarten können zur formalen Definition einer Schaltung in den Abstraktionsebenen zugrundegelegt werden. Dabei kann eine Schaltung durch eine funktionale, eine strukturelle oder eine physikalische Beschreibung dokumentiert werden.

Die funktionale Beschreibung gibt an, wie Operationen oder Transformationen auf Eingangswerte (Eingangsinformationen) angewendet, wie sie intern in Zwischenwerte überführt und wie sie anschließend auf Ausgangswerte (Ausgangsinformationen) abgebildet werden.

Die strukturelle Beschreibung stellt die Zusammensetzung größerer Strukturen aus kleineren Strukturen und Modulen in einer Schaltung dar. Die einzelnen Elemente können in Abhängigkeit der abstrakten Sicht durch spezielle Verbindungsstrukturen (Ein-/Ausgabetore, Kommunikationskanäle etc.) oder auch direkt durch Leitungen, Busse u.ä. verbunden sein.

Bei der physikalischen Beschreibung können geometrische Aspekte (z.B. die Lage von Schaltungselementen im Layout) oder zeitliche Aspekte (z.B. Datenflüsse) zusammengefaßt und zur Beschreibung zugrundegelegt werden.

In den Abstraktionsebenen werden die Komponenten und die zu verarbeitenden Informationen mit unterschiedlichen Dichten bzw. Abstraktionen erfaßt. Die wichtigsten Abstraktionsebenen digitaler Systeme sind:

- System-Ebene
- Registertransfer-Ebene
- Gatter-Ebene
- Transistor-Ebene
- Layout-Ebene

Die verschiedenen Ebenen sind in ihrer Reihenfolge durch einen immer größeren Detailreichtum und eine geringer werdende Übersichtlichkeit gekennzeichnet.

Für jede Ebene existieren aufgrund unterschiedlicher Aspekte unterschiedliche Beschreibungsarten einer Schaltung. Die unterschiedlichen Beschreibungsarten in unterschiedlichen Abstraktionsebenen lassen sich aus der Abbildung 2-1 in Anlehnung an das Gajsky-Kuhn-Diagramm (oder auch Y-Diagramm genannt) ableiten [RAM89].

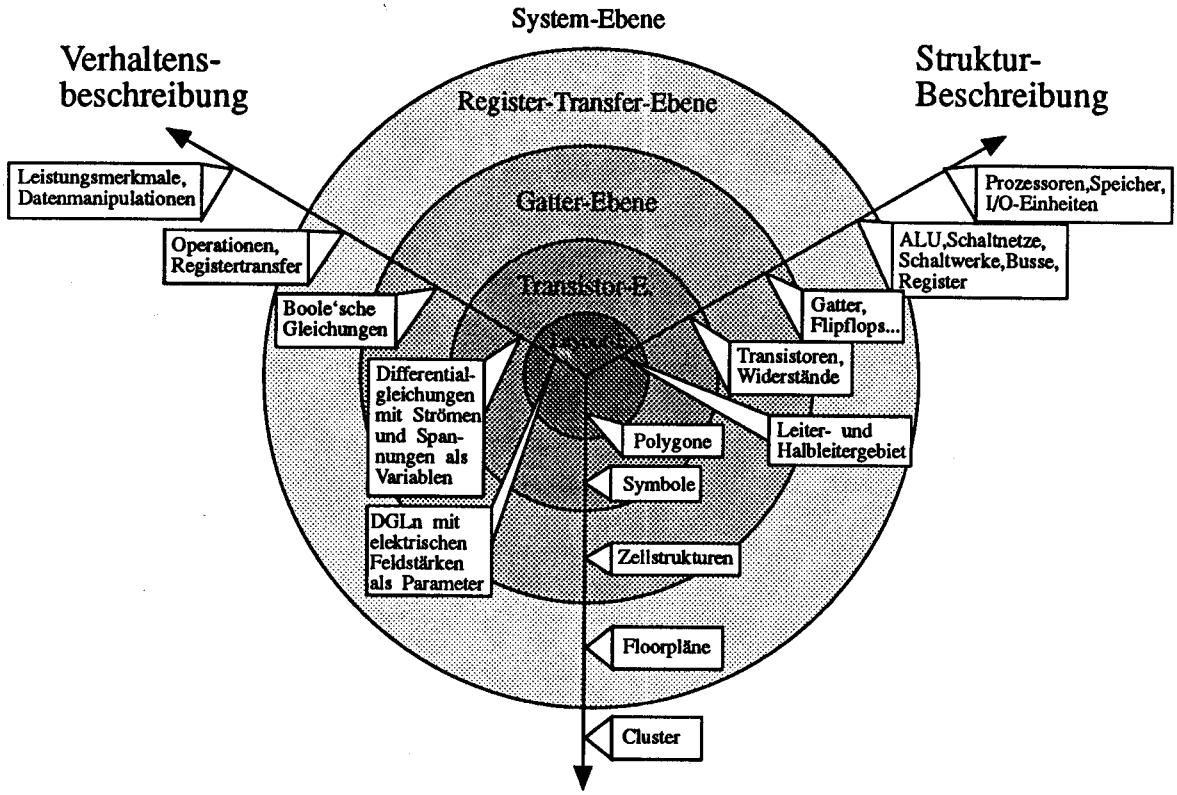


Abbildung: 2-1: Das Y-Diagramm

3 Simulation auf Gatter-Ebene

Mit der fortschreitenden Entwicklung der Technologie und der Verbreitung großintegrierter Schaltungen werden von Designern und Anwendern integrierter Schaltkreise wirklichkeitsgetreue und zuverlässige Simulatoren zur Analyse und Verifizierung von Schaltungen verlangt, da die Prototypherstellung einer Schaltung mit hohen Kosten verbunden ist. Die korrekte Analyse einer Schaltung mit adäquaten Simulationszeiten stellt daher hohe Anforderungen an einen Simulator. Durch die Vielfältigkeit der Eigenschaften der Elemente bzw. der Technologie- und Prozeßparameter werden von einem Simulator genaue Definitions- und Behandlungsmöglichkeiten für Modelle mit den entsprechenden Zeitparametern und physikalischen Größen gefordert. Um möglichst genaue Simulationsergebnisse zu erzielen, muß der Simulator weiterhin über ausgereifte Berechnungsmethoden und Verfahren in Verbindung mit erweiterten Zustandsräumen für digitale Signale verfügen, die die erhöhte Anzahl logischer Signalzustände berücksichtigt.

Der Einsatzbereich der Logiksimulatoren als Entwicklungs- und Entwurfswerkzeuge erstreckt sich auf die Architektur- bzw. Logikentwicklung einer Schaltung (CAE: Computer Aided Engineering), den Layoutentwurf (CAD: Computer Aided Design) und den Schaltungstest (CAT: Computer Aided Test).

Ein Logiksimulator bildet auf Gatter-Ebene die reale Schaltung durch den Einsatz mathematischer und/oder funktioneller Modelle für die Gatter der Schaltung im Rechner ab. Mit Hilfe dieser Modelle werden bei der Berechnung der Schaltung die Ausgangsinformationen der Schaltung (Ausgangssignale) für bestimmte vom Entwickler definierte Eingangsinformationen (Eingangssignale, Stimulies) ermittelt. Die Genauigkeit der Simulationsergebnisse hängt weitgehend von der Genauigkeit der gewählten Modelle sowohl für die Elemente als auch für die Signale entlang der Verbindungen zwischen den Elementen ab. Durch die Erfassung unterschiedlicher Parameter (z.B. Verzögerungszeiten, Anstiegs- und Abfallzeiten, Lastfaktoren u.ä.) wird die Genauigkeit der Modelle stark beeinflußt.

Mit der Simulation einer Schaltung können etwaige Entwurfsfehler entdeckt und beseitigt werden, was zu einem fehlerfreien Entwurf einer Schaltung aus der logischen (architektonischen) Sicht führt. Außerdem können die Entwürfe hinsichtlich unterschiedlicher Schaltungsparameter (z.B. Leistungsbedarf, Signallaufzeiten entlang der Pfade, Empfindlichkeiten u.ä.) optimiert werden.

Der Einsatz von Simulatoren bei der Schaltungsentwicklung ermöglicht eine beträchtliche Zeitersparnis in der Entwurfsphase einer Schaltung. Der aufwendige Aufbau und Test eines Musters wird überflüssig.

Auch die Verifikation einer Schaltung kann durch eine wiederholte Simulation der Schaltung durchgeführt werden.

Bei der Verifiktion wird die Frage gestellt, inwieweit eine entwickelte Schaltung die bei der Aufgabenstellung formulierten Randbedingungen erfüllt. Die Grundlage für die Beantwortung dieser Frage bildet im allgemeinen ein Pflichtenheft. Aus der praktischen Erfahrung kann festgestellt werden, daß hierbei neben Entwurfsfehlern auch Fehler im Pflichtenheft erkannt werden.

Eine digitale Schaltung auf Gatter-Ebene wird als eine Einheit mit Eingängen und Ausgängen betrachtet, die Signale empfängt, sie manipuliert und die Resultate an die Ausgänge weiterleitet. Die Manipulation der Informationen wird in Abhängigkeit der in der Schaltung realisierten Funktion durch entsprechende Komponenten durchgeführt.

4 Modellierung von Gattern

Zur Modellierung eines Gatters sind zu erfassen:

a) Die logische Funktion, die das ideale Verhalten eines Gatters dokumentiert. Hierfür werden Boole'sche Gleichungen eingesetzt.
b) Die physikalischen Eigenschaften aufgrund der verwendeten Technologie, die das reale Verhalten des Gatters in einer Schaltung bestimmen.

1.4

Der Informationsfluß entlang eines Gatters erfolgt mit Zeitverzögerungen, die einerseits durch Wanderungsgeschwindigkeit der Ladungsträger in den Transistoren und anderseits durch Ladung und Entladung von Transistor- und Leitungskapazitäten bestimmt werden. Die Nichtlinearitäten der Ströme und Spannungen der Transistoren und Kapazitäten bewirken, daß unterschiedliche Verzögerungszeiten beim 0-1- bzw. 1-0-Übergang am Ausgang eines Gatters entstehen.

In Abhängigkeit der Dauer der Einschaltzeit t_{on} und Ausschaltzeit t_{off} unterscheidet man zwischen

- einer linearen Verzögerungszeit, wenn t_{on} gleich t_{off} ist, und
- einer nichtlinearen Verzögerungszeit, wenn t_{on} ungleich t_{off} ist.

Diese Zeitabschnitte werden bei der Modellierung eines Gatters als eine Basis zur Berechnung von Gatterverzögerungszeiten zugrunde gelegt. Falls keine Verzögerungszeiten bei der Simulation eines Netzwerkes berücksichtigt werden, entsteht eine verzögerungsfreie Modellierung der Gatter. Sie erlaubt eine zeitoptimale Simulation und Verifikation eines Schaltnetzes hinsichtlich seines logischen Verhaltens. Für Schaltwerke ist die Verzögerungszeit jedoch ein essentielles Attribut, das für eine korrekte Arbeitsweise der Schaltung verantwortlich ist. Hier werden die Signalverzögerungen zur Speicherung von Schaltwerkzuständen eingesetzt.

Eine einfache Modellierung des Zeitverhaltens eines Gatters G (Abb. 4-1a), das durch eine Boole'sche Funktion

$$Y(t) = f(X_1(t), X_2(t),..., X_n(t)) \qquad (4\text{-}1)$$

charakterisiert ist, verwendet ein zweistufiges Modell (Abb. 4-1b), das als "Unit-Delay-Modell" eines Gatters bezeichnet wird und eine lineare Zeitverzögerung

$$\tau = t_{on} = t_{off} \qquad \text{oder} \qquad \tau = (t_{on} + t_{off})/2$$

zugrunde legt. In der Gleichung 4-1 sind X_1 bis X_n die Eingangssignale und Y das Ausgangssignal des Gatters.

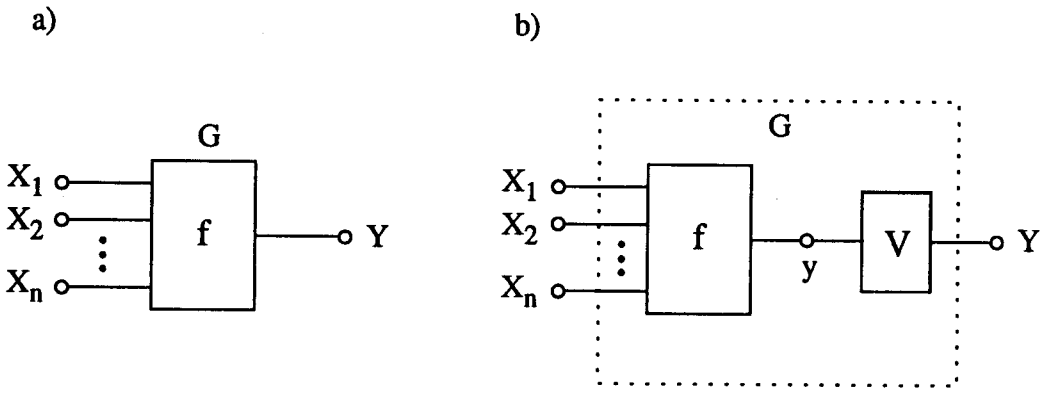


Abbildung 4-1: a) Ein allgemeines Gatter G
b) Unit-Delay-Modell des Gatters G

In der ersten Stufe dieses Modells wird die Boole'sche Funktion des Gatters mit

$$y = f(X_1, X_2, \ldots, X_n) \tag{4-2}$$

erfaßt. Gleichung 4-2 beschreibt das logische Verhalten des Gatters und ermöglicht die Berechnung des logischen Zustandes der eingeführten Zwischenvariable y in Abhängigkeit der Eingangsvariablen X_1 bis X_n.

In einer zweiten Stufe (Stufe V) wird das Zeitverhalten des Gatters als eine lineare Zeitverzögerung τ, also sowohl für den 0-1- als auch für den 1-0-Übergang, berücksichtigt. Das neu eingeführtes Element wird als Verzögerungsglied V (Unit-Delay-Element) bezeichnet. Für das Verzögerungsglied gilt :

$$Y(t) = y(t-\tau) \tag{4-3}$$

Durch Einsetzen der Gleichung 4-2 in die Gleichung 4-3 kann die Modellgleichung für das Gatter ermittelt werden :

$$Y(t) = f\{(X_1(t-\tau), X_2(t-\tau),..., X_n(t-\tau)\} \tag{4-4}$$

Diese Gleichung beschreibt sowohl das logische als auch das zeitliche Verhalten eines Gatters. In diesem Modell bewirkt jedes Eingangssignal die gleiche Zeitverzögerungszeit τ, die durch das Verzögerungsglied V am Ausgang des Gatters repräsentiert wird. Damit wird angenommen, daß alle Signalpfade zwischen den Eingangssignalen und dem Ausgangssignal gleich sind. Diese Modellierungsart wird auch als **"ausgangsseitige Delay-Modellierung"** bezeichnet.

Eine formale Beschreibung des Gatterverhaltens in dieser Modellierungsart kann wie folgt angegeben werden:

IF "eine Änderung an einem der Eingänge vorliegt" THEN
  $Y \Leftarrow (f(X_1, X_2, X_3 \ldots X_n))$ AFTER τ ns
ENDIF

In dieser Formulierung wird der Variable Y der durch die Funktion f errechnete Wert mit einer Verzögerung von τ ns zugewiesen.

Prinzipiell kann aber im Unterschied zu dem oben angegebenen Modell jeweils jedem Eingang ein Verzögerungsglied zugeordnet werden. In diesem Fall spricht man von einem **"eingangsseitigen Modellierungsverfahren"**.

5 Signalzustände und -modelle

Signale sind Informationsträger. Sie stellen zeitlich veränderliche Größen dar, die die auf sie abgebildeten Informationen tragen. In elektronischen Systemen sind Ströme und Spannungen die Informationsträger. Sie sind physikalische Größen, die zeitlich entweder einen kontinuierlichen Verlauf oder einen diskontinuierlichen Verlauf in diskreten Stufen besitzen können. Auch der Informationsparameter, der in den meisten Fällen die Amplitude eines Trägers darstellt, kann einen kontinuierlichen oder einen diskreten Verlauf aufweisen.

1.6

Zur Differenzierung von Signalzuständen und zur Behandlung von Signalen in einem Simulator, der für die Simulation von Schaltkreisen unter Berücksichtigung physikalischer Eigenschaften in unterschiedlichen Technologien eingesetzt werden soll, wird eine Modellierung der Signalzustände durch die Zuordnung von Signalgewichten oder Signalstärken vorgenommen.

Bei der Modellierung der Signalstärke werden im allgemeinen mehrere Stufen zugrundegelegt. Die am häufigsten verwendeten Stärken sind im folgenden angegeben:

F (forcing):

Die Stärke bezeichnet den Signalzustand eines Knotens mit höchstem Gewicht. Hier liegt ein Knoten vor, der seinen Zustand durch eine Verbindung mit verschwindend kleinem Widerstand zu anderen Knoten überträgt. Derartige Knoten in einem Netzwerk sind insbesondere die Betriebsspannungs- und Masse-Knoten. Bei dieser Stärke wird die Verbindung als annähernder Kurzschluß zwischen zwei Knoten betrachtet.

W (weak resistive)

Diese Stärke gilt für einen Knoten, der eine Verbindung mit einem niedrigen Widerstandswert zu einem anderen Knoten aufweist. Damit wird der Knoten relativ stark beeinflußt, falls er von V_{DD}- oder V_{SS}-Knoten getrieben wird.

R (resistive):

Bezeichnet den Signalzustand eines Knotens, der durch einen mittelohmigen Widerstand an einen anderen (aktiven) Knoten im Netzwerk angeschlossen ist.

Z (high impedance):

Dieser Zustand entspricht dem typischen Schwebezustand bei Tri-State-Ausgängen eines Gatters. Eine Beeinflussung des Gatter-Ausgangsknotens durch den V_{DD}- oder durch den V_{SS}-Knoten liegt in einer vernachlässigbar schwachen Form vor.

D (disconnect):

Dieser Zustand stellt die Unterbrechung zwischen zwei Knoten dar, d. h. die Knoten können sich gegenseitig nicht beeinflussen.

Zusätzlich wird ein Zustand U (Undefiniert, Unknown) eingeführt, wenn keine Aussage über die tatsächliche Stärke eines Knotens gemacht werden kann.

Jede der oben beschriebenen Stärken (bis auf D) kann den logischen Zuständen 0, 1 oder X zugeordnet werden, so daß das Schema in Abb. 5-1 zustande kommt.

In der Tabelle sind die Grundzustände der Signale in der obersten Zeile angegeben. Die restlichen Zeilen stellen die möglichen Kombinationen der Grundzustände dar. Insgesamt erhält man aus der Tabelle 46 logische Werte zur Darstellung von Signalen (Summe der Zeilen und Spalten, d. h. Summe aller Grundzustände und ihrer Kombinationen).

Durch die Einführung von Stärken ist man nun in der Lage, wenn mehrere treibende Knoten auf einen gemeinsamen Knoten wirken, den effektiven Zustand des Knotens zu bestimmen.

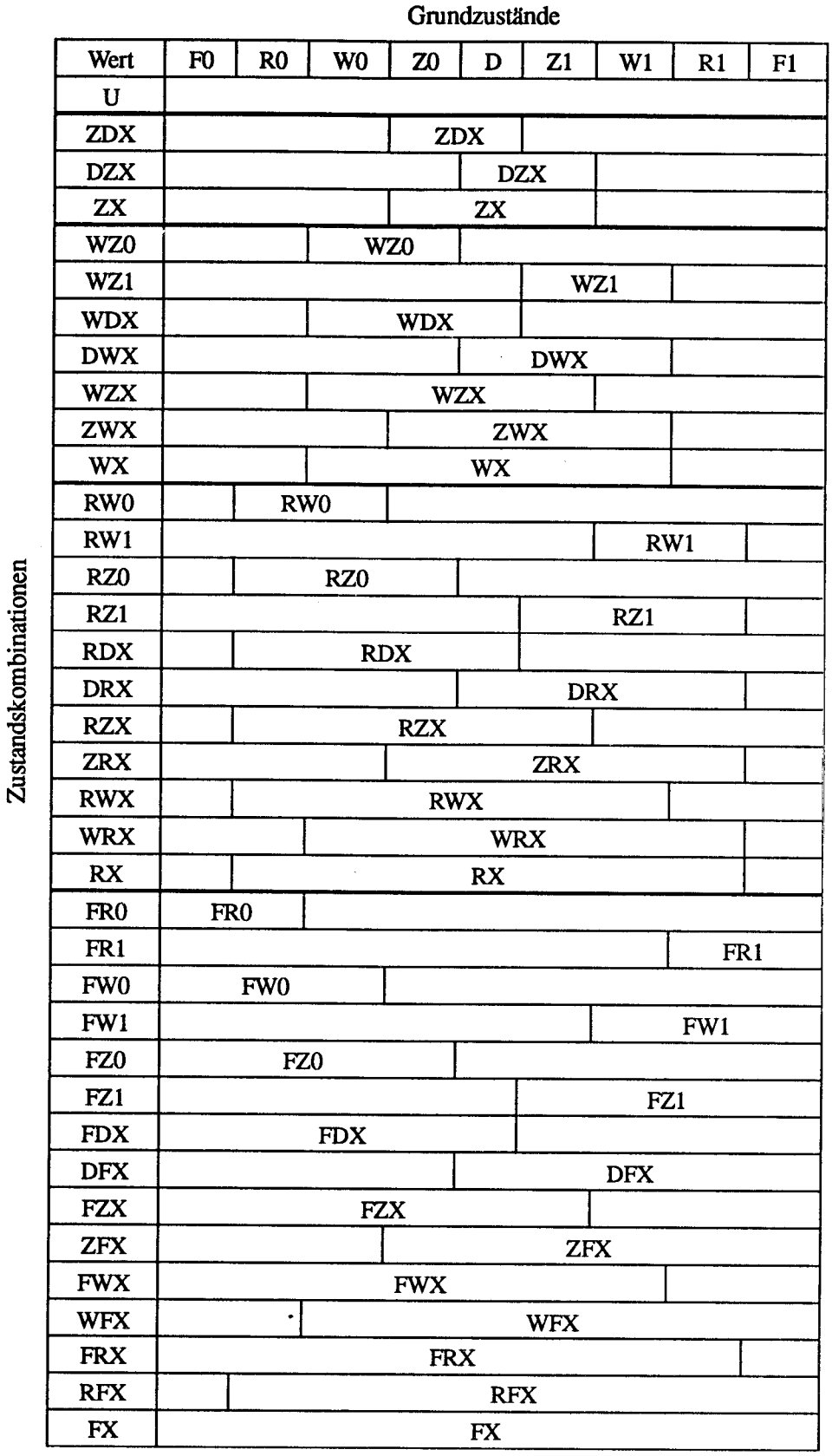


Abbildung 5-1: Gewichtete Standardlogikwerte, Grundwerte (1. Zeile) und ihre Kombinationen (restliche Zeilen).

1.8

6 Simulationsverfahren

Die Verfahren für die Simulation digitaler Schaltungen können in zwei Kategorien, nämlich laufzeitorientierte und aktivitätsorientierte bzw. ereignisorientierte Verfahren, aufgeteilt werden (Abb. 6-1).

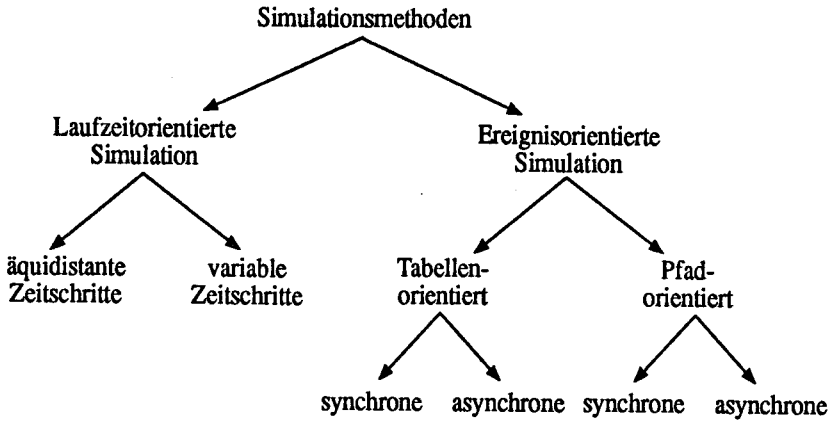


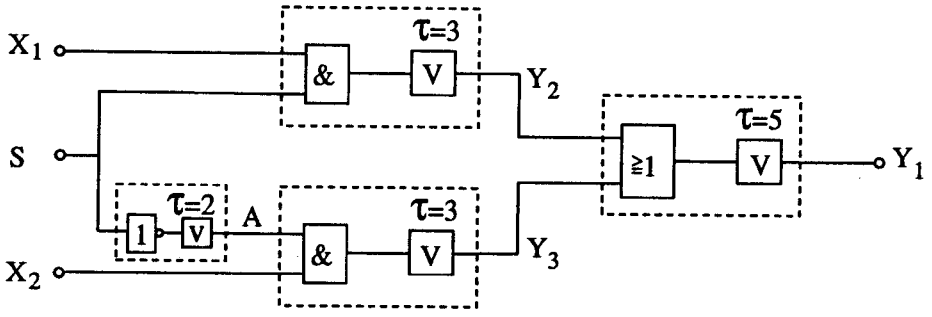
Abbildung 6-1: Strukturierung von Simulationsverfahren.

a) Laufzeitorientierte Simulation

Bei der laufzeitorientierten Methode wird ein Netzwerk vollständig zu den Zeitpunkten eines Zeitrasters berechnet. Zur Bestimmung der Zeitpunkte werden die Gatterlaufzeiten zugrunde gelegt. Die Gatter können unterschiedliche Verzögerungszeiten als lineare und nichtlineare Verzögerungen oder Laufzeitbereiche aufweisen.

Die aufeinanderfolgenden Berechnungszeitpunkte eines Netzwerkes können äquidistante oder variable Abstände aufweisen.

Abb. 6-2 zeigt den Ablauf einer Simulation für eine Schaltung nach diesem Verfahren. In der Abbildung sind neben der Struktur der Schaltung, die Gattermodelle mit ausgangsbezogenen Verzögerungsgliedern enthält, die Modellgleichungen in Abhängigkeit der Signalverzögerungszeiten dargestellt. Eine Simulation wird zum Zeitpunkt t=0 gestartet und in (normierten) Schritten von 1 mit Hilfe der Gleichungen durchgeführt. Das Ergebnis ist als Impulsdiagramm angegeben. Die Eingangssignale X_1, X_2 und S werden für die Simulation definiert. Sie werden als gegeben vorausgesetzt. Eine Signaländerung am Ausgang der Schaltung entsteht, wenn sich die Eingangssignale ändern.



$$Y_2(t)=X_1(t-3)\,S(t-3)$$

$$A(t)=\overline{S}(t-2)$$

$$Y_3(t)=A(t-3)\,X_2(t-3)=\overline{S}(t-5)\,X_2(t-3)$$

$$Y_1(t)=Y_2(t-5)\vee Y_3(t-5)=X_1(t-8)\,S(t-8)\vee\overline{S}(t-10)\,X_2(t-8)$$

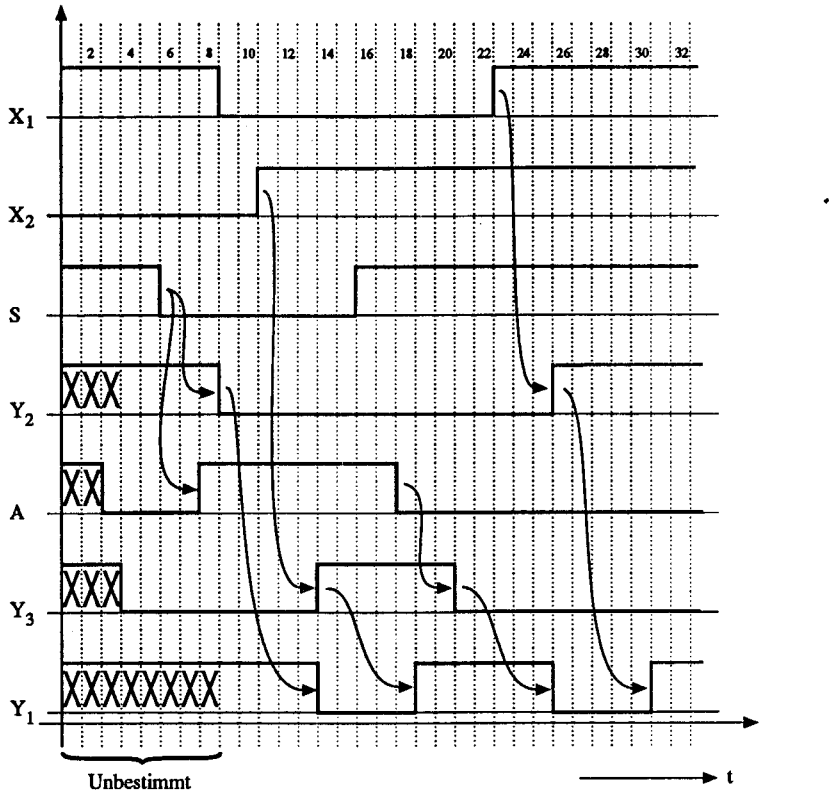


Abbildung 6-2: Beispiel einer laufzeitorientierten Simulation

b) Ereignisorientierte Simulation

Bei einer aktivitäts- oder ereignisorientierten Simulation werden die auftretenden Aktivitäten (Ereignisse) erfaßt und deren Fortpflanzung in den betreffenden Pfaden der Schaltung berücksichtigt. Das Verfahren benötigt eine geringere Simulationszeit als die laufzeitorientierte Simulation, da nur die aktiven Phasen der Signale, d.h. nur wenn Änderungen bei den Signalzuständen vorliegen, betrachtet werden und eine Berechnung der Schaltung vorgenommen wird. Ein Ereignis entsteht also durch eine Signaländerung bei den Eingangssignalen oder an den inneren Knoten eines Netzwerkes.

Prinzipiell unterscheidet man zwischen zwei Verfahren ereignisorientierter Simulationsmethoden (vergl. Abb. 5-1):

- Tabellenorientierte Ereignissimulation
  ("Next Event List Simulation", "Time Mapping Event Scheduling")

- Pfadorientierte Ereignissimulation
  ("Selective Trace Simulation")

1.10

Bei der tabellenorientierten Ereignissimulation wird zur Erfassung der chronologisch auftretenden Ereignisse eine Ereignisliste eingesetzt. Sukzessive werden die Positionen der Tabelle erfaßt und mit Hilfe der zugrunde gelegten Datenstruktur festgestellt, auf welche Elemente sie eine Änderung ausüben. Zusammen mit anderen Eingangszuständen sowie mit der Funktion eines Elementes werden die Ausgangssignale berechnet. Liegt eine Änderung des Ausgangssignals vor, wird die Ereignis-Tabelle aktualisiert, um derartige Änderungen nach der Bearbeitung aller Elemente als neue Ereignisse zu bearbeiten.

Bei der pfadorientierten Ereignissimulation wird nur eine selektive Fortpflanzung der Ereignisse entlang der Pfade berücksichtigt. Bei der Simulation werden also nur die Gatter bearbeitet, die durch ein Ereignis (eine Signaländerung) an den Eingängen aktiviert werden. Alle anderen Gatter werden nicht berücksichtigt.

7 Hardware-Beschreibung in VHDL

Die Basis zur Beschreibung digitaler Komponenten und Systeme in VHDL bilden das funktionale und das zeitliche Verhalten sowie der strukturelle Aufbau eines Systems. Daraus ergeben sich bei der Modellierung digitaler Systeme in VHDL drei sich gegenseitig ergänzende Modelle:

- das Verhaltensmodell (behavioral model),
- das Zeitmodell (timing model) und
- das Strukturmodell (structural model).

a) Das Verhaltensmodell

Eine funktionale Beziehung, die aus einer Folge von Operationen besteht, repräsentiert das Verhalten des Netzwerkes, das durch die Beobachtbarkeit von Auswirkungen an den Ausgängen infolge der Erregungen an den Eingängen (Stimuli) des Netzwerkes charakterisiert wird. Somit bildet die funktionale Beziehung eines Netzwerkes die Grundlage zur Modellierung des logischen Verhaltens des Systems.

Als Beispiel wird ein digitales System, das aus einem Äquivalenz-Gatter besteht, betrachtet. Das Verhalten des Gatters kann mit Hilfe der Gatterfunktion, die die Ausgangsvariable y in Abhängigkeit der Eingangsvariablen i1 und i2 angibt, beschrieben werden.

Die Struktur der Schaltung ist jedoch erst nach der Auflösung der Gleichung durch NICHT-, UND- und ODER-Gatter darstellbar.

Die formale Verhaltensbeschreibung eines Gatters oder einer Einheit eines Systems in VHDL wird mit Hilfe einer Programmstruktur, die als *Prozeß* bezeichnet wird, vorgenommen.

Ein Prozeß umfaßt Datenstrukturen, Instruktionen, Prozeduren etc., ähnlich wie in prozeduralen Programmiersprachen Pascal oder C. So wird das gesamte System durch eine Ansammlung von unabhängigen und konkurrierenden Prozessen beschrieben, die parallel ausgeführt werden. Die zeitliche Folge der parallel ablaufenden Prozesse in einem System dokumentiert das Verhalten des Gesamtsystems. Zur Kommunikation und zur Synchronisation zwischen den Prozessen werden *Signale* eingesetzt. Da die Signale sich entlang von Signalpfaden in einem digitalen System

fortpflanzen können, werden sie auch zur Aktivierung und zur Steuerung von Prozessen eingesetzt. Ein Prozeß generiert ein Signal, das sich entlang eines Pfades auswirkt. Ein oder mehrere Prozesse entlang des Signalpfades, die das Signal empfangen, werden zur Bearbeitung des Signals aktiviert.

Die Kommunikation über Pfade in einem System ist an einen festen Datentyp gebunden. Z.B. kann über einen Pfad vom Typ *Integer*, kein Signal vom Typ *Bit* übertragen werden.

Die Prozesse werden solange ausgeführt, bis sie in einen Warte-Zustand versetzt werden. Aus diesem Warte-Zustand heraus kann der Prozeß wieder aktiviert werden, wenn eine bestimmte Bedingung erfüllt ist. Solche Bedingungen könnten benutzt werden, um den Abstand von Prozessen und so das Verhalten einer Schaltung zu modellieren. Z.B. werden Prozesse benötigt, die nur dann reagieren, wenn sich ein Zustand im System ändert. Diese Zustandsänderung wird durch die Änderung eines Signalwertes repräsentiert, da die Signale die Zustände des Systems repräsentieren. In VHDL heißt das, daß ein Prozeß sensitiv im bezug auf den Wert eines Datenpfades ist. Derartige Datenpfade heißen sensibilisierte Kanäle (*sensitivity channels*). Wenn sich der Wert auf einem dieser Kanäle ändert, wird der zugehörige Prozeß reaktiviert. Um diese Signal-Bedingungen innerhalb eines Prozesses zu implementieren, wird die **wait**-Anweisung benutzt. Wird bei der Ausführung eines Prozesses eine **wait**-Anweisung gefunden, wird die Bedingung, die zur Aktivierung benötigt wird, gespeichert und der Prozeß in den Warte-Zustand versetzt.

Der zu dem Beispiel des Äquivalenzgatters mit zwei Eingängen gehörende Prozeß hat zwei sensitive Eingangskanäle, nämlich die Eingänge i1 und i2. Daher wird der Prozeß, der die Äquivalenzfunktion für den Ausgangswert y berechnet, bei jeder Änderung eines dieser Eingangswerte neu gestartet.

b) Das Zeitmodell in VHDL

Um das reale Verhalten eines Systems abbilden bzw. modellieren zu können, werden in VHDL die Zeit und speziell die Verzögerungszeiten von Komponenten als Parameter in die Modelle aufgenommen.

Das Zeitverhalten der Komponenten im System ist im allgemeinen durch Verzögerungszeiten gegeben, die in ihrer Gesamtheit das zeitliche Verhalten des Systems widerspiegeln. Neben Gatterverzögerungszeiten und Verzögerungszeiten für sonstige Komponenten in einem Netzwerk können in VHDL auch die Aktionen mit Verzögerungszeiten behaftet sein. Beispielsweise können entlang eines Signalpfades mit mehreren Prozessen Verzögerungszeiten für die Aktivierung von Prozessen und für die Ausführung der im Prozeß vorhandenen Aktionen eingesetzt werden. Derartige Verzögerungen werden als *Aktionsverzögerung* bezeichnet. Die Anzahl von Aktionen entlang eines Signalpfades ist beliebig.

Eine Aktionsquelle (z.B. Ausgang eines Gatters), die einen Signalpfad beeinflußt, wird als Signaltreiber (*signal driver*) bezeichnet.

Zu einem Signal kann nur ein Treiber gehören. Ein Treiber kann jedoch mehrere Signalquellen umfassen. Der Treiber wird durch eine geordnete Menge von Zeit/Werte-Paaren, die jeweils den Wert des Signals zu den zugehörigen Zeiten angeben, definiert. Folglich stellt ein Signal eine physikalische Größe dar, die während der Aktivierung und Ausführung von Prozessen ständig aktualisiert wird.

Die Bearbeitung eines Modells in VHDL erfolgt in zwei Phasen, die als *Simulationszyklus* bezeichnet werden. In der ersten Phase werden die Ursachen (Stimuli) erfaßt, die jeweils eine Signal-Änderung erzeugen. In der zweiten Phase werden die daraus folgenden Auswirkungen im Modell bestimmt.

Während der ersten Phase des Simulationszyklus werden die Werte den Signalen zugeordnet. Dieser Schritt ist abgeschlossen, wenn alle Signale die zur augenblicklichen Simulationszeit gehörenden Werte angenommen haben. In der zweiten Phase werden alle die Prozesse, die über sensitive Kanäle neue Werte erhalten haben, aktiviert und ausgeführt, bis sie in den Wartezustand versetzt werden, d.h. bis sie auf eine **wait**-Anweisung stoßen oder vollständig ausgeführt worden sind. Die Phase ist beendet, wenn alle Prozesse jeweils im Wartezustand (*suspend*) sind. Um den Zyklus zu beenden, wird die Simulationszeit auf den nächsten Zeitpunkt gesetzt, zu dem wiederum eine Aktion stattfindet. Dann beginnt der nächste Simulationszyklus.

Die Simulation ist beendet, wenn ein bestimmter (vorgegebener) Zeitpunkt erreicht ist, oder wenn keine Signaländerungen mehr vorliegen, die einen Prozeß neu starten könnten.

Betrachtet man den oben beschriebenen Ablauf, so ist zu erkennen, daß immer eine gewisse Zeit von der Generierung eines Wertes durch einen Prozeß bis hin zum Eintragen in das zugehörige Signal vergeht. Wenn keine explizite Verzögerung bei der Zuweisung eines Wertes an ein Signal angegeben ist, wird mindestens ein sogenanntes *delta-delay* eingesetzt. Das *delta-delay* bewirkt keine Erhöhung der Simulationszeit, es sorgt aber dafür, daß ein Simulationszyklus zum Eintragen der neuen Werte durchlaufen wird (*update*).

Immer wenn sich der neue Wert eines Signals vom vorherigen unterscheidet, bezeichnet man auch dies als ein Ereignis. Ein sensitiver Kanal reagiert also auf die Ereignisse des zugehörigen Signals, so daß mit VHDL das ereignisorientierte Simulationsverfahren einfach zu realisieren ist.

c) Das Strukturmodell

Die in einem digitalen System vorhandenen Ein- und Ausgänge werden in der Schnittstelle der Schaltung (*entity* oder *entity declaration*) definiert.

Die Verbindungen zwischen den einzelnen Einheiten des Systems übernehmen die Kommunikation und den Datenaustausch zwischen den Einheiten. Die Schnittstellen, die zu Verbindungen mit anderen Einheiten führen, werden als *ports* bezeichnet. Die Port-Beschreibungen (*port declaration*) enthalten die Definitionen der am Datenaustausch beteiligten Signale.

Als Beispiel eines Strukturmodells kann das Äquivalenzgatter aus drei Teilsystemen aufgebaut werden. Jedes Teilsystem stellt eine vollständige Einheit dar (AND, OR, NOR). In Abb. 7-3 sind Elemente der Schaltung, die jeweils als getrennte *entities* definiert werden, angegeben.

In jedem Element existiert eine Port-Beschreibung, die die Namen der intern verwendeten Ein- und Ausgänge definiert. Durch den Aufruf dieser Teilsysteme aus der nächst höheren Ebene (Äquivalenz), werden die aktuellen Werte über die Signale an die internen Signale übergeben bzw. die berechneten Werte von den internen Signalen auf die "Verbindungssignale" weitergeleitet. Auf diese Weise kann ein Element, daß sich in einer Bibliothek befindet mehrfach benutzt werden (Instanziierung).

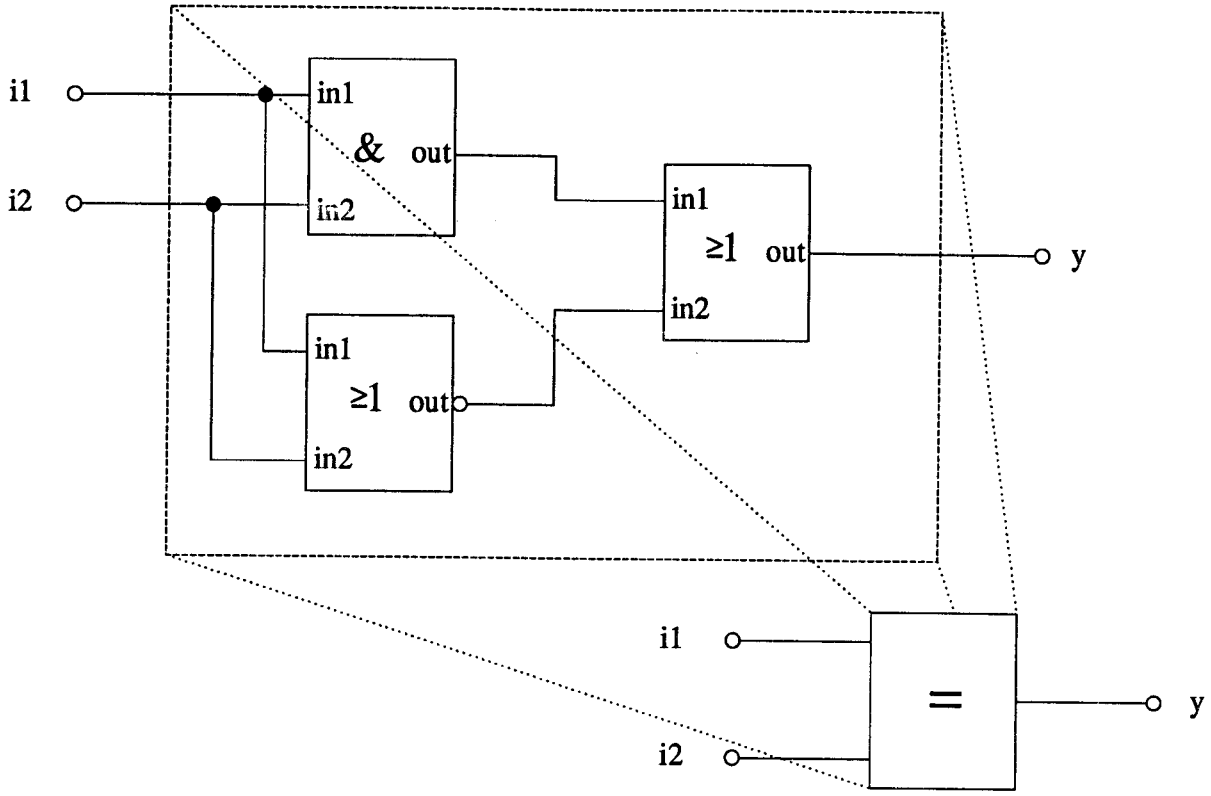


Abbildung 7-3: Strukturmodell des Äquivalenzgatters

8 Struktur eines VHDL-Programms

Ein VHDL-Entwurf - eine VHDL-Entwurfseinheit (*unit*) - setzt sich aus der Bausteindeklaration (*entity declaration*) und mindestens einem Architekturrumpf (*architectural body*) zusammen. Eine VHDL-Entwurfseinheit ist ein abgeschlossener Block von Anweisungen. Sie kann einzeln auf Korrektheit der Syntax überprüft werden. Ist die Syntax korrekt, kann die Einheit als Bibliothekselement für weitere Entwürfe verwendet werden.

Bausteindeklaration (*entity declaration*)

Die Bausteindeklaration ist formal folgendermaßen aufgebaut:

```
entity Bausteinname is
  port (Schnittstellenliste);
  generic (Parameterliste);
  Deklarationen;
  begin
    Anweisungen;
end Bausteinname;
```

- Schnittstellenlisten definieren die Verbindungen eines Bausteins zur Außenwelt. Sie werden durch das reservierte Wort **port** eingeleitet. Die einzelnen Schnittstellenelemente werden

jeweils durch ein Semikolon voneinander getrennt. Jedes Schnittstellenelement kann mehrere Schnittstellenobjekte enthalten, die durch Kommata getrennt werden. Die Schnittstellenliste darf nur Objekte der Objektart **signal** enthalten. Eine Schnittstellenliste definiert folgende Objekteigenschaften:

a) den Datentyp der Schnittstellenobjekte
b) die Richtung des Datenflusses über die Schnittstelle

Für die Festlegung der Richtung des Datenflusses stellt VHDL vier Modi zur Verfügung (**in**, **out**, **inout** und **buffer**). Durch die Angabe der Richtung kann bei der Simulation überprüft werden, ob die Schaltungsstruktur des Entwurfs korrekt ist. Die Bedeutung der Modi ist in Tab. 8-1 dargestellt.

Modus	Art der Leitung
in	Eingangsleitung; nur lesender Zugriff (default)
out	Ausgangsleitung; nur schreibender Zugriff
inout	bidirektionale Leitung; lesender und schreibender Zugriff
buffer	Ausgangsleitung; lesender und schreibender Zugriff

Tabelle 8-1: Modi zur Steuerung des Datenflusses

Der Modus **buffer** sollte verwendet werden, wenn das Signal eigentlich ein Ausgangssignal ist, jedoch auch für interne Berechnungen benötigt wird.

Die allgemeine Form eines Schnittstellenelements ist folgende:

```
Objektart Bezeichnerliste :
                Richtung Datentyp := Standardwert
```

Die minimale Definition eines Schnittstellenelements besteht aus der Bezeichnerliste und der Angabe des Datentyps. Die Objektart ist bei der Schnittstellenliste festgelegt und muß daher nicht angegeben werden.

```
port (Eingang1, Eingang2: in Bit;
      Wert: in Integer; Ausgang: out Bit);
```

- Parameterlisten erlauben die Übergabe von modellabhängigen Parametern an einen Baustein. Z.B. ist es möglich, universelle Modelle zu erstellen und diesen bausteinabhängige Parameter, wie die Verzögerungszeit, über diese Parameterlisten anzugeben. Eine Parameterliste ist eine spezielle Schnittstellenliste, die durch das Schlüsselwort **generic** eingeleitet wird. Für diese Liste ist nur die Objektart **constant** und die Datenflußrichtung **in** zugelassen.

```
generic (VerinLH: Integer := 10 ns;
         VerinHL: Integer := 8 ns);
```

- Deklarationen innerhalb der Bausteindeklaration können eingesetzt werden, um dieselben Objekte tür verschiedene Architekturrümpfe benutzen zu können. Die Deklaration in jedem einzelnen Architekturrumpf kann so entfallen.

- Anweisungen innerhalb der Bausteindeklaration können dazu benutzt werden, einige Aktionen (wie die Überprüfung von Eingangsbedingungen) für alle Architekturrümpfe gemeinsam durchzuführen. Dies vereinfacht wiederum die Architekturrümpfe.

Im folgenden ist die Bausteindeklaration für ein RS-Flip-Flop angegeben:

```
entity RS_FF is
   port (R, S: in Bit; Q, QN: buffer Bit);
   generic (Ver);
   begin
     assert not (S = '1' and R = '1')
      report "Fehler: S und R sind auf '1'"
       severity Error;
end RS_FF;
```

Architekturrumpf (*architectural body*)

Ein Architekturrumpf beschreibt die Struktur und das Verhalten des Bausteins. Zu jeder Bausteindeklaration muß mindestens ein Architekturrumpf existieren. Dies ist der grundlegende Aufbau:

```
architecture Rumpfname of Bausteinname is
   Deklarationen;
begin
   Anweisungen;
end Rumpfname;
```

Der Deklarationsteil enthält die Typ-, Signal- und Konstantendeklarationen. Außerdem werden hier Komponenten deklariert, die schon in VHDL beschrieben worden sind und die im Anweisungsteil instanziiert werden.

Der Anweisungsteil enthält die Struktur- und/oder die Verhaltensbeschreibung des Bausteins. Die Verhaltensbeschreibung besteht aus Prozessen oder parallel zu verarbeitenden Anweisungen (*concurrent statements*). Die Strukturbeschreibung umfaßt die Komponenteninstanziierungen.

Neben Bausteindeklarationen und Architekturrümpfen gibt es Elemente, die die Übersichtlichkeit einer VHDL-Beschreibung verbessern können:

- Unterprogramme (*subprograms*),
- Pakete (*packages*) und
- Bibliotheken (*libraries*).

Insbesondere die Bibliotheken ermöglichen es, getestete VHDL-Entwurfseinheiten ohne erneute Analyse und damit ohne zusätzlichen Zeitaufwand in anderen VHDL-Bausteinen einzusetzen.

9 Zusammenfassung

In diesem Beitrag wird eine Einführung in VHDL gegeben. Es werden die prinzipiellen Modellierungsverfahren digitaler Komponenten und Signale und der Aufbau eines VHDL-Programms dargestellt, sowie die in VHDL verwendeten Modelle, Daten und Objekte. Die Struktur- und Verhaltensbeschreibung digitaler Schaltkreise in VHDL wird erläutert.

Mit zunehmender Steigerung der Komplexität elektronischer Systeme hat die standardisierte Hardware-Beschreibungssprache VHDL ((very high speed integrated circuits) Hardware Description Language) als ein höchst geeignetes Dokumentations- und Entwicklungswerkzeug für den strukturierten Entwurf komplexer Schaltkreise an Bedeutung gewonnen. Die VHDL erlaubt es, ein elektronisches System sowohl aus einer abstrakten Sicht als auch auf Gatter-Ebene zu synthetisieren, sein Verhalten durch Simulations- und Verifikationswerkzeuge zu überprüfen und zur Herstellung vorzubereiten. Bereits entwickelte und getestete Komponenten können wiederholt zur Vereinfachung und Beschleunigung eines Entwurfs herangezogen werden.

Mit ihrem Einsatz wird insbesondere "Time-To-Market" drastisch reduziert, die Verständigung zwischen den Beteiligten in einem Projekt in hohem Maße gesteigert und der strukturierte Entwicklungsablauf gefördert.

Im vergangenen Jahr wurde vom "VHDL Standardization Committee" eine überarbeitete Version der Sprache vorgestellt. Der neue Entwurf umfaßt mehrere sinnvolle Ergänzungen (z.B. Fanout abhängiges Timing, Einsatz globaler Variablen, Varianten-Records, etc.). Damit wird die Verbreitung der Sprache VHDL weiterhin unterstützt.

10 Literaturverzeichnis

[ASH90] Ashenden, P. J.:
"The VHDL Cookbook", Adelaide, 1990

[BEu87] Bechtold, M.; Reus, Th.; Tavangarian, D.:
"Simulation hybrider Schaltungen", 4. Symposium Simulationstechnik, Zürich, Sept. 1987

[COE89] Coelho, D.R.:
"The VHDL Handbook", Kluwer Academic Publishers, Boston, 1989

[HOR85] Horneber, E.-H.:
"Simulation elektrischer Schaltungen auf dem Rechner", Springer, Berlin, 1985

[HuS81] Hachtel, G. D.; Sangiovanni-Vincentelli, A. L.:
"A Survey of Third-Generation Simulation Techniques", Proc. of the IEEE, Vol. 69, No. 10, S. 1264-1280, Oct. 1981

[LIu89] Lipsett, R.; Schaefer, C.;Ussery, C.:
"VHDL: Hardware Description and Design", Kluwer Academic Publishers, Dordrecht, 1989

[RAM89] Rammig, F. J.:
"Systematischer Entwurf digitaler Systeme", Verlag B. G. Teubner, Stuttgart, 1989

[RuD83] Ruehli, A. E.; Ditlow, G. S.:
"Circuit analysis, logic simulation and design verification for VLSI", Proc. IEEE, 71, pp. 34-48, January 1983

[SCH87] Schwarz, A. F.:
"Computer Aided Design of Microelectronic Circuits and Systems", Academic Press, London, 1987

[SPI85] Spiro, H.:
"Simulation integrierter Schaltungen", Oldenbourg Verlag, 1985

[TAu91] Koch, M.; Macke, H.; Tavangarian, D.:
"Struktur eines VHDL-basierten Simulators für die Lehre", ASIM91 7.Symposium Simulationstechnik, Hagen 1991, S.406-410

[TuK92] Koch, M.; Tavangarian, D.:
"Practical Design Experience Using VHDL", Proceedings of the 3.EUROCHIP Workshop on VLSI Design Training, Grenoble 1992, S.124-129

[VHD87] IEEE Standard VHDL Language Reference Manual -Std 1076- 1987, IEEE, New York, 1988.

Entwicklung eines Radschlupfprozessors mit der Beschreibungssprache VHDL

Dipl.Ing. T.Büchner, Roland Granzer
Institut für Mikroelektronik Stuttgart
Prof.Dr.Ing G.Kampe
Fachhochschule für Technik Esslingen

Hier wird über eine Diplomarbeit berichtet, bei der ein Radschlupfprozessor mit der Hardwarebeschreibungssprache VHDL in einem anwenderspezifischen IC implementiert werden soll. Nach einem Überblick über die Synthese mit VHDL wird am Beispiel einer Teileinheit des Prozessors die Umsetzung in Hardware beschrieben.

1. Einleitung

Die Fahrsicherheit eines Automobils hängt in hohem Maße vom Kraftschluß zwischen Reifen und Fahrbahn ab. Eine bestimmende Größe dabei ist der Radschlupf, mit dessen Hilfe Aussagen über den aktuellen Zustand der Bodenhaftung gemacht werden können. Bild Nr.3 zeigt die Eingangsgrößen des Radschlupfprozessors. Die Signale der ABS-Sensoren eines angetriebenen und eines nicht angetriebenen Rades werden zur Schlupfberechnung benötigt. Die ABS-Sensoren bestehen aus einer gelochten Metallplatte und einer Spule, in der während der Fahrt zur Geschwindigkeit proportionale Signale erzeugt werden (siehe Bild 3 unten). Der so ermittelte Schlupfwert wird dann über eine interne serielle Schnittstelle an einen zentralen Kfz-Controller übermittelt, der dann die Auswertung bzw. Darstellung für den Fahrer vornimmt.

2. Einführung in VHDL

In Bild 4 werden die Gründe genannt, warum mit dieser Diplomarbeit vorhandene Prototypen des Radschlupfprozessors in VHDL neu beschrieben werden sollten. Mit die wichtigsten Gründe sind dabei, daß mit VHDL ein großer Teil der Schaltkreis-entwicklung durchführbar ist und die Sprache durch die Softwarehersteller unterstützt wird. Auf Bild 5 ist am Beispiel eines einfachen Flipflops der Unterschied zwischen den beiden wichtigsten Beschreibungsarten von VHDL dargestellt. Die Verhaltensbeschreibung geschieht ähnlich wie in C oder Pascal auf einer sehr hohen Ebene, während die Strukturbeschreibung einer Netzliste oder

2.1

einem Schaltplan entspricht. Bild 6 zeigt den Entwicklungszyklus mit einer Synthese der VHDL-Beschreibung. Die eigentliche Umsetzung (Synthese) von Verhaltensbeschreibung in Netzliste werden durch den VHDL-Compiler bzw. den technologieabhängigen Design-Compiler vorgenommen. Allerdings ist nicht der komplette Umfang der VHDL-Sprache synthetisierbar, sondern nur ein gewisser, nicht allzu komplexer, Ausschnitt davon (z.B. keine Rekursion). Die Vorgehensweise bei der Erstellung einer synthetisierbaren VHDL-Beschreibung ist auf Bild 7 zu sehen. Besonders wichtig bei diesem Top-down Entwurf ist, schon bei der Verhaltensbeschreibung möglichst genaue Vorstellungen über die gewünschte Hardware mit einfließen zu lassen. D.h. man sollte wissen, wo man Zähler, Register, Schaltwerke usw. einsetzen will und diese Einheiten dann auf Verhaltensebene beschreiben. Dadurch ist dann später eine gute Verifizierbarkeit und Änderbarkeit der synthetisierten Schaltung gewährleistet.

3. Der Aufbau des Radschlupfprozessors

Auf Bild 8 sind die Einzelkomponenten des Radschlupfprozessors zu sehen. Das RISC-Controller-Makro wurde durch die Diplomarbeit von Volker Wahl in VHDL beschrieben. Die Entwicklung der Meßeinrichtungen und der RS232-Schnittstelle sowie die Verbindung der Komponenten waren Ziel dieser Arbeit. Die einzelnen Einheiten sind durch einen 8-Bit Datenbus und entsprechenden Steuerleitungen mit dem internen RISC-Controller verbunden. Die Steuerung der Abläufe sowie die eigentliche Berechnung des Schlupfwertes sind die Aufgaben dieses RISC-Controllers. Die von den Meßeinrichtungen gelieferten Meßwerte sowie der zu übertragende Schlupfwert haben eine Breite von 16 Bit. Da für einen internen 16 Bit Bus jedoch nicht genügend Ports am Controller-Makro vorhanden sind, wurde hier der Weg über einen 8-Bit Bus mit zwei Buszyklen gewählt.

4. Realisierung der RS232-Schnittstelle

Die Aufgaben der Schnittstelle und ihre hierarchische Aufteilung in funktionale Ebenen sind in Bild 9 dargestellt. Ein wichtiges Ziel dabei war, die Schnittstelle mit einem integrierten Selbsttest und veränderbaren Übertragungsparametern auszustatten. Außerdem ist auf Bild 9 (mitte) der Datenfluß innerhalb der Schnittstelle zu erkennen. Im Normalbetrieb werden die Daten in jeder Ebene verändert und dann weitergereicht. Während des Selbsttests werden in der Testebene spezielle Daten erzeugt und durch die weiteren geschickt. Die Auswertung der Testdaten erfolgt dann wieder in der Testebene. Auf dem unteren Teil des Bildes sind außerdem zwei der Probleme die bei der Umsetzung in VHDL autraten genannt. Die Probleme

waren hauptsächlich im Bereich der Testbarkeit bzw. dem Selbsttest zu finden. Da in der Verhaltensbeschreibung noch keine genauen Aussagen über eine Realisierung durch den Synthetisierer gemacht werden können, sind Testergebnisse auf Verhaltensebene zum Teil nicht ermittelbar. Bild 10 zeigt die Schnittstelle in ihrem endgültigem Aufbau. Die Einheiten im grau unterlegten Teil des Bildes werden aussschließlich für die Testbarkeit benötigt. Durch eine Mehrfachausnutzung einzelner Register (z.B. dem Signaturregister) für den Selbsttest und für den normalen Betrieb konnte der Testaufwand hier auf etwa 30% gesenkt werden. Dieser Wert ist jedoch trotzdem noch relativ hoch.

5. Zusammenfassung

Bild 11 faßt die Ergebnisse dieser Arbeit in Bezug auf VHDL bzw. den WSP noch einmal zusammen. Besonders wichtig ist, daß die Schaltungen nach der Synthese bezüglich des Zeitverhaltens genau überprüft werden, da hier keine Fehlerfreiheit garantiert wird. Die kommende Aufgabe liegt nun im Test und in der Verifikation eines fertigen Musters des RISC-Controllers.

2.4

Diplomarbeit

Entwicklung eines Radschlupfprozessors mit der Beschreibungssprache VHDL

Roland Granzer

Institut für Mikroelektronik Stuttgart

Fachhochschule für Technik Esslingen

Betreuer am IMS:

Dipl.-Ing. T.Büchner

Betreuer an der FHTE:

Prof.Dr.Ing. G.Kampe

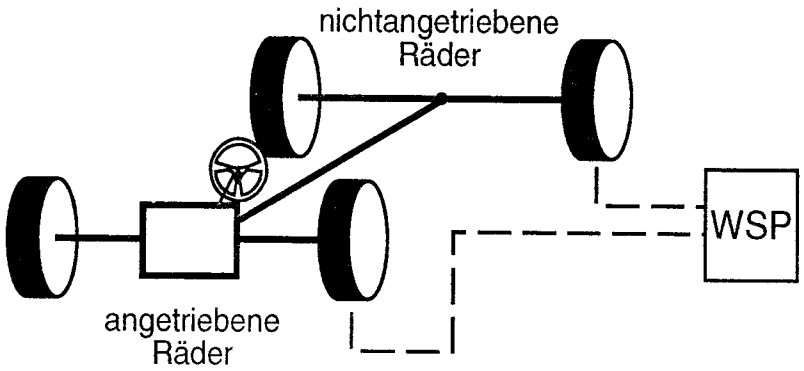
Übersicht

Aufgaben des Radschlupfprozessors WSP

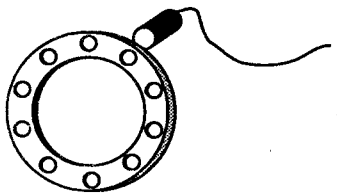
Vermeidung von Extremsituationen bei Kraftfahrzeugen:

- präzise Messung des Radschlupfs
- Messung in festen Zeitintervallen
- Datentransfer zu zentralem Kfz-Controller

Der WSP am Fahrzeug:



Der ABS-Sensor:



Gründe für eine Entwicklung in VHDL

- Standisierung durch den IEEE
  - → genormter Sprachumfang
  - → Unterstützung durch Softwarehersteller
  - → Portabilität zwischen unterschiedlichen Systemen

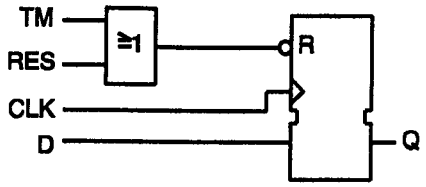
- Umfangreiche Möglichkeiten mit VHDL
  - → Verhaltensbeschreibungen
  - → Netzlisten (Strukturbeschreibungen)
  - → Testumgebung (Stimuli, Auswertung)

  alles in einer Sprache

- Schaltungsspezifikationen in VHDL
  - → VHDL-Spezifikation ist simulierbar

Beschreibungsarten in VHDL

Beispiel : D-FF mit asynchr. Reset



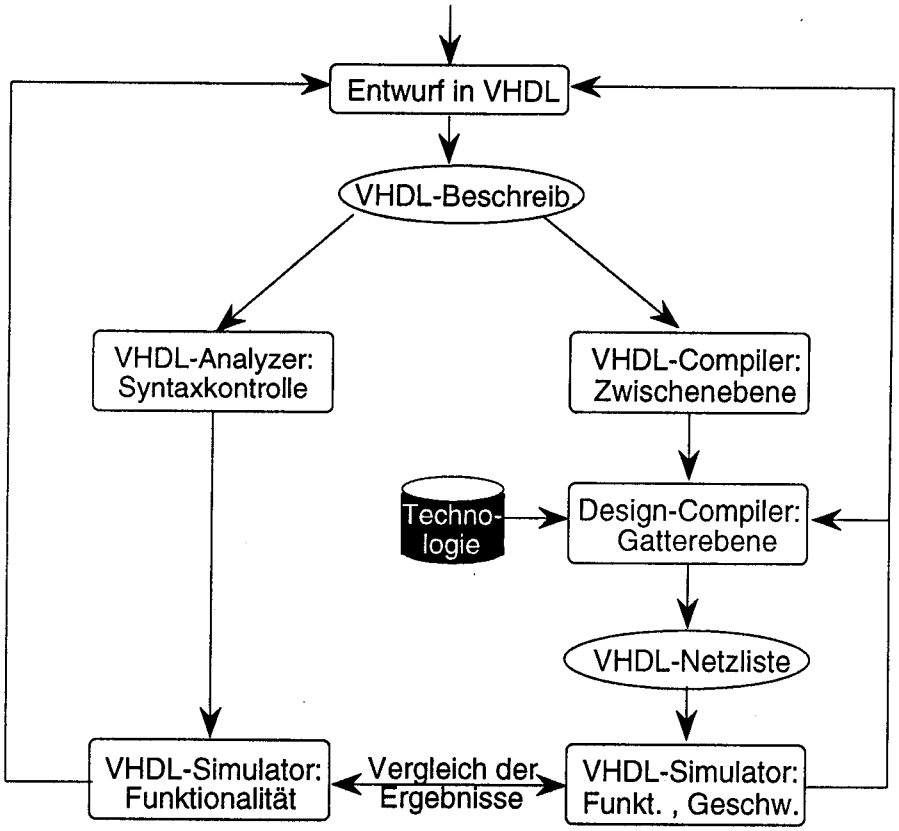
1. Die Verhaltensbeschreibung

```
                :
FF1:
process(CLK,RES,TM)
begin
    if (RES='0' and TM='0')
       then Q <= '0';
    elsif (CLK'event and CLK='1')
       then Q <= D;
    end if;
end process;
                :
```

2. Die Strukturbeschreibung (Netzliste)

```
                :
component or_gate
    port(A1,A2:in Bit; Z:out Bit);
end component;
component D_ff
    port(R,CLK,D:in Bit;
                 Z:out Bit);
end component;
signal or_Z: Bit;
begin
    OR:or_gate
       port map(TM,RES,or_Z);
    FF:D_ff
       port map(or_Z,CLK,Q);
end Structural;
```


Schaltkreisentwicklung mit VHDL



Erstellen synthetisierbarer VHDL-Beschreibungen

Pflichtenheft, Spezifikationen

Grobe Vorstellungen über die gewünschte Hardware

Partitionierung, Hierarchische Aufteilung in möglichst kleine Einheiten

synthetisierbare VHDL-Beschreibung der Einzelkomponenten

Synthese der kompletten Beschreibung

Achtung: Fehler beim Synthetisieren sind möglich !

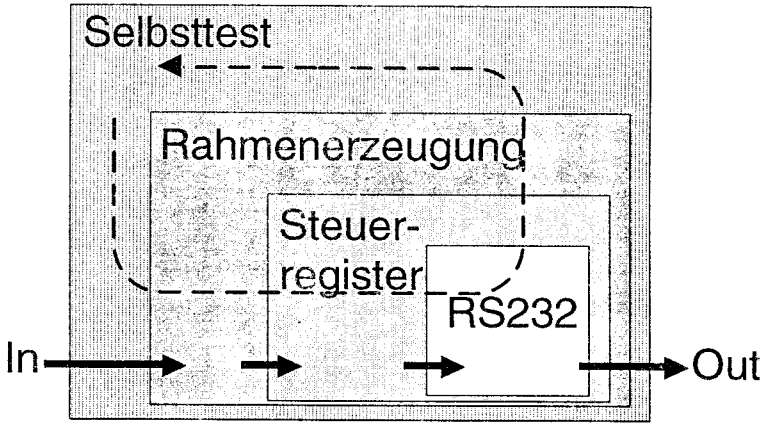
Der Aufbau des Wheel-Slip-Processor

Realisierung der RS232-Schnittstelle

Ziel: Übertragung des 16-Bit Schlupfwertes in 4 Rahmen

- Erzeugung der Rahmenbits
- Einstellbare Parameter (Parität,Stop-,Datenbits,Handshake,Baudrate)
- Selbsttest

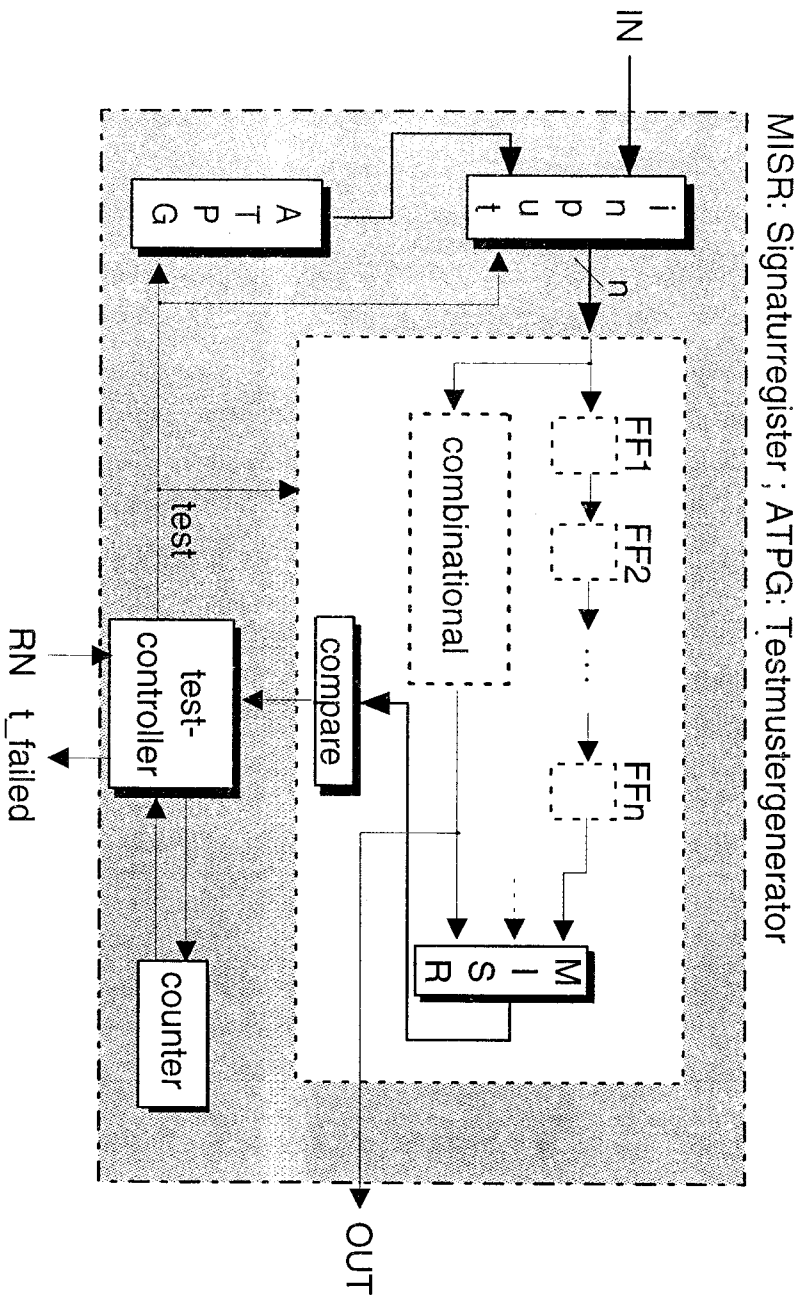
Schnittstelle:



Probleme:

- Testbarkeit muß von Anfang an berücksichtigt werden
- In VHDL vor der Synthese kein Zugriff auf Register von Schaltwerken (FSM's)

Realisierung der RS232-Schnittstelle



Zusammenfassung und Ausblick

Zu VHDL:

- Schaltungsentwicklung in VHDL mit anschl. Synthese vereinfacht die Umsetzung in Netzlisten
- Syntheseergebnisse müssen genau überprüft werden
- Änderungen sind in VHDL schneller durchführbar
- Testbarkeit von Anfang an berücksichtigen
- Beispiel für eine selbsttestende Schaltung in VHDL

Zum WSP:

- Ein einziger Chip für Messung, Berechnung und Ausgabe des Schlupfwertes
- Vielseitig einsetzbar durch progr. Mikrocontroller
- Alle Einheiten des Chips als VHDL-Beschreibung vorhanden → weiterverwendbar
- kommende Aufgabe: Test + Verifikation des Controllers an einem fertigen Muster

Entwurf und Realisierung einer ASIC-basierenden opto-elektronischen Busschnittstelle

Dipl.-Ing. Heinz Berl
Prof. Dr.-Ing. N. Höptner
Prof. Dr. G. Krieg
Dipl.-Ing. A. Larsch

IIT der Fachhochschule Karlsruhe

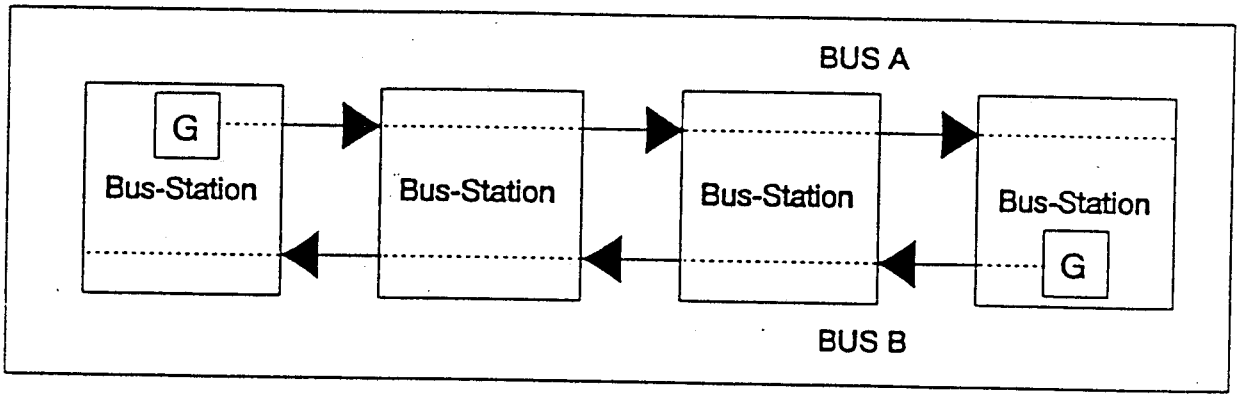
Dieses Projekt wurde im Zeitraum 1.1.91 bis 31.3.93 bearbeitet.
Es wurde mit Personalmitteln für 3 Mannjahre aus dem Schwerpunktprogramm gefördert. Weiterhin wurde die ASIC-Herstellung im Rahmen des Budgets der MPC-Gruppe durchgeführt.

Projektziele:

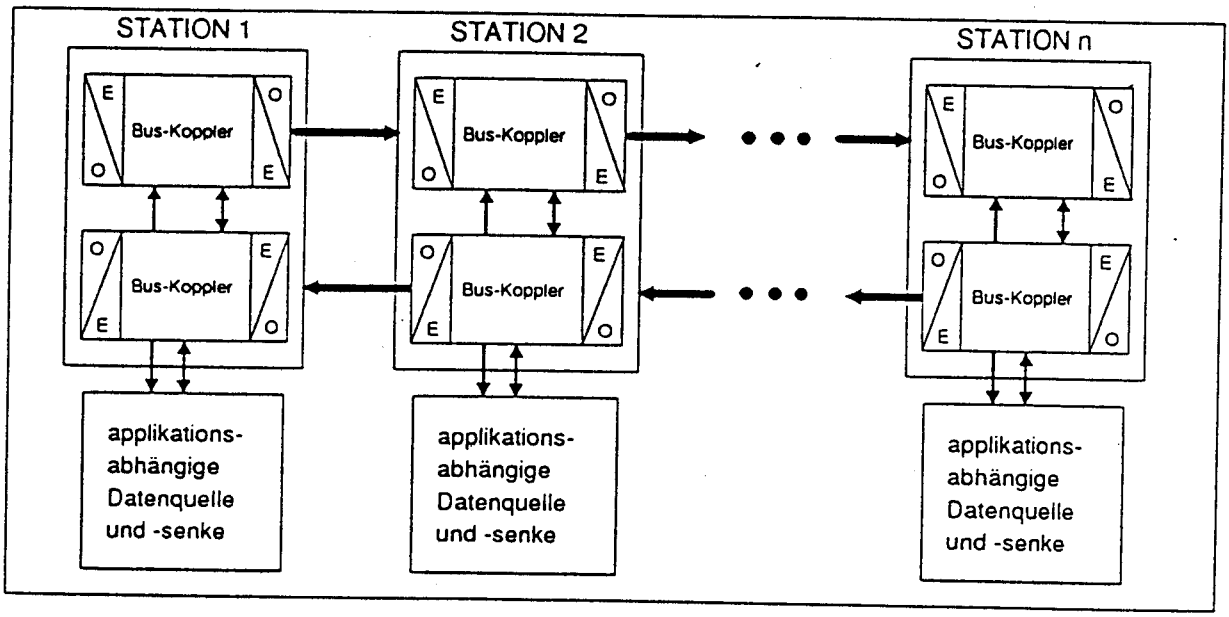
- für den Anwender einfache Busschnittstelle mit möglichst hoher Datenrate
- Verwendbarkeit von Lichtwellenleitern und Kupferleitern
- adressierbare Busteilnehmer (mindestens 256)
- faires Buszuteilungs-Konzept: Echtzeitfähigkeit
- flexible Austauschbarkeit von Kosten gegen Übertragungsrate ohne Änderung der Kernhardware
- integrierte Kernhardware
- Vergleich der Leistungsfähigkeit ASIC - LCA

Systemkonzept

Zur Anwendung gelangt das DQDB - Zugriffsverfahren (DQDB - Distributed Queue Dual Bus).



Bussystem - Übersicht:



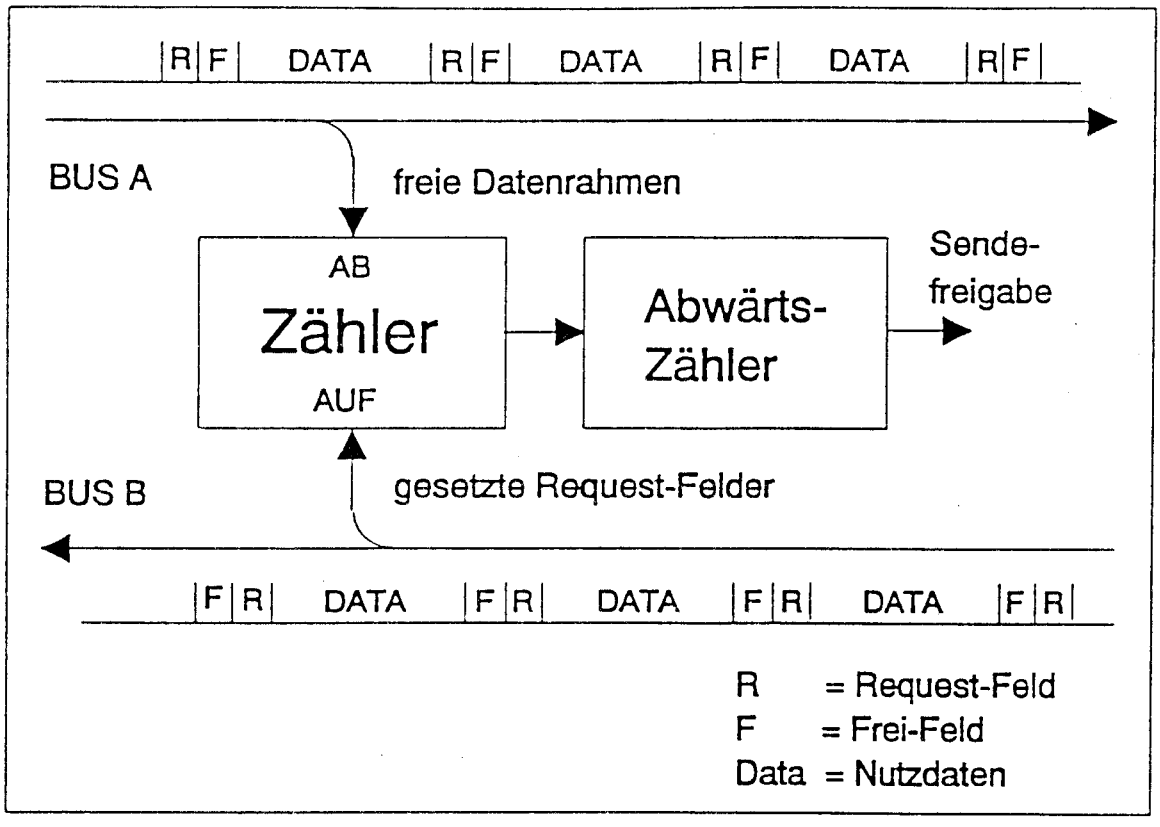
Bus-Zugriffsverfahren

Wenn eine Station senden will, so setzt sie im Datenstrom der entgegengesetzten Übertragungsrichtung ein freies Requestfeld und signalisiert somit allen vor ihr liegenden Stationen ihren Sendewunsch.

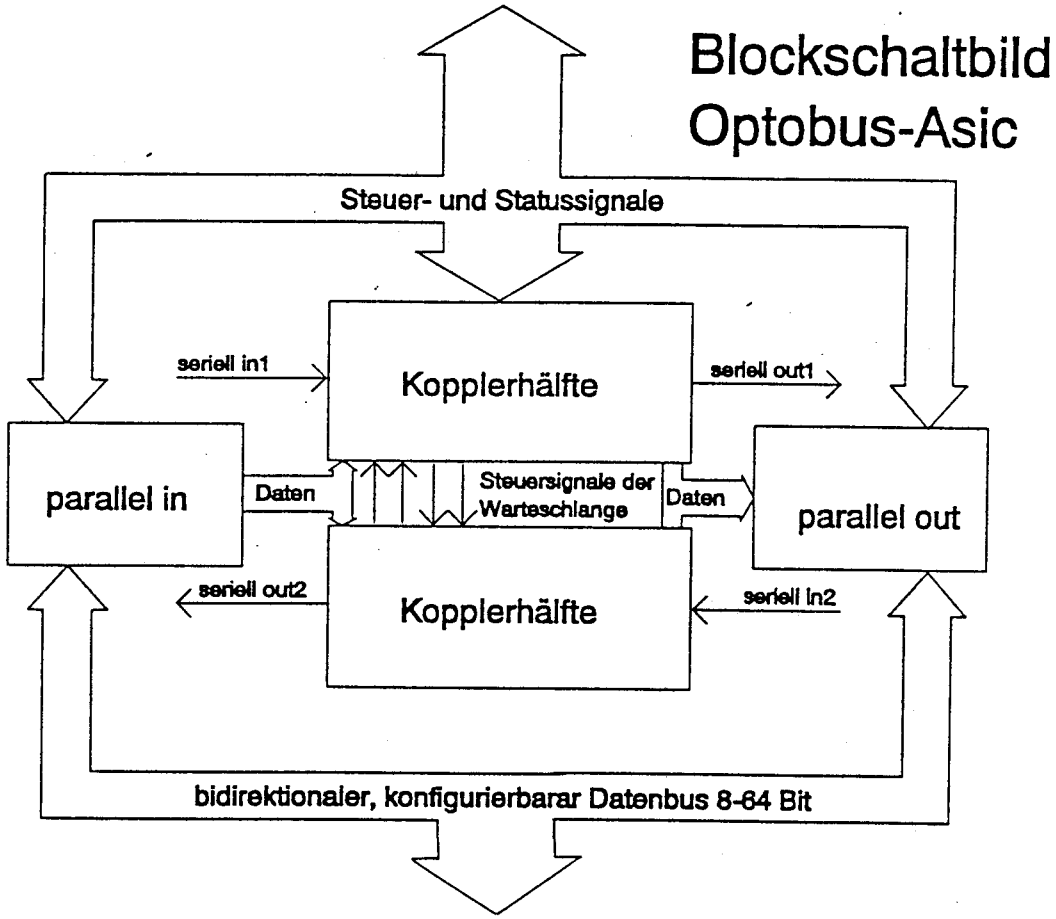
Anhand des Zählerstandes in der eigenen Station ist sie über die Anzahl der Sendewünsche aller nach ihr liegenden Stationen informiert.

Bevor sie nun einen freien Datenrahmen belegt, läßt sie zuvor so viele freie Datenrahmen passieren, wie zum Zeitpunkt des Zugriffwunsches im Zähler gespeichert waren.

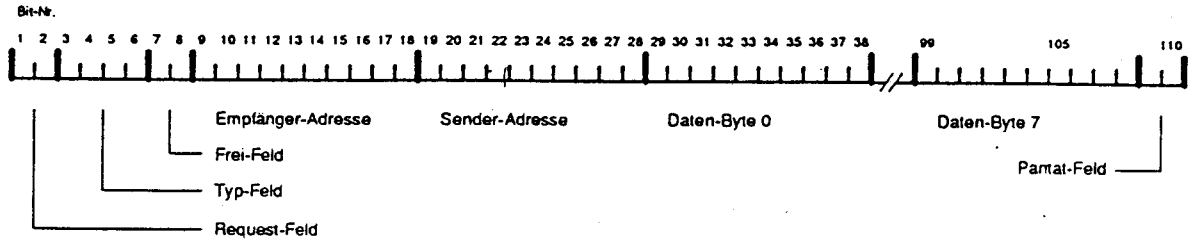
Damit ist gewährleistet, daß ohne Arbiter eine gleichmäßige Verteilung der Bandbreite an alle senden wollende Stationen erfolgen kann.



Das folgende Bild zeigt die Struktur einer dafür realisierten Bus-schnittstelle, wie sie in einem ASIC nun implementiert werden muß.



Der Datenrahmen hat folgenden Aufbau:



Damit ergibt sich ein Nutzdatenraten-Anteil von
64 bit / 110 bit - 58% .
(Beipiel: Bei 50 Mbit/s ----> 29 Mbit/s)

Weiteres Vorgehen:

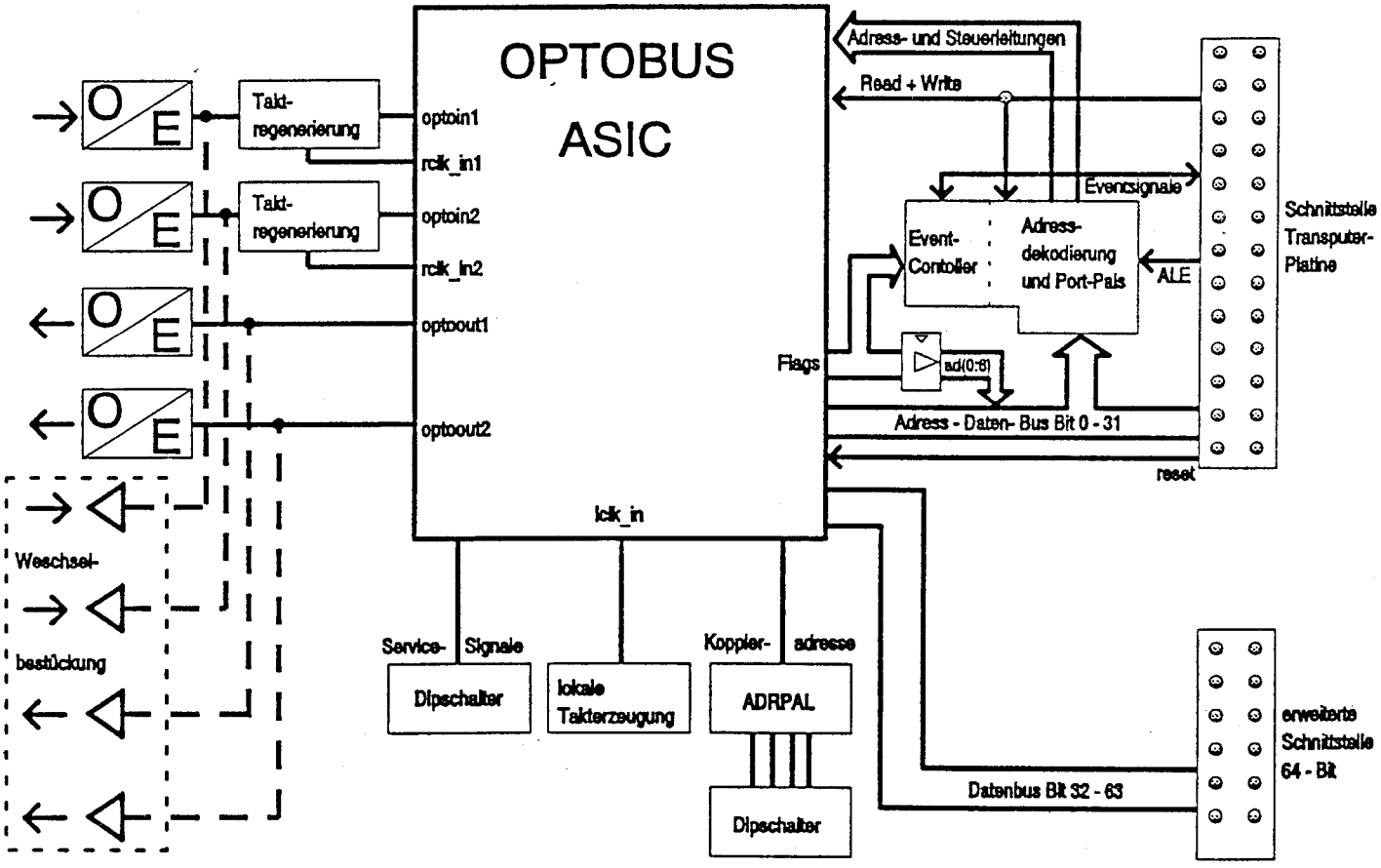
Zur Zeit wird eine Demonstrationssystem realisert, daß

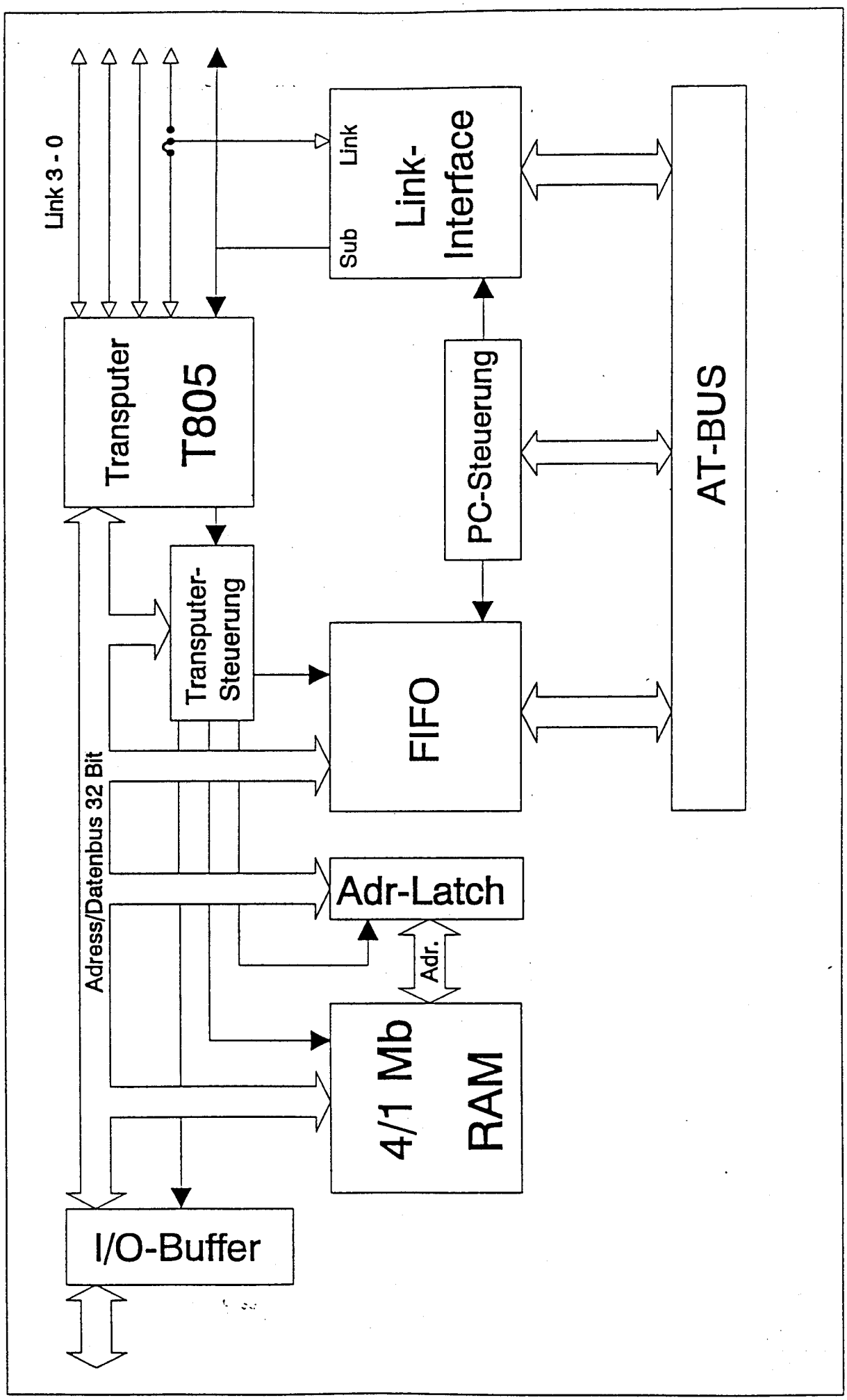
- die Adressierbarkeit und
- die effektive Datenrate

unter Beweis stellen soll. Natürlich wird damit ebenfalls die volle System-Funktionsfähigkeit nachgewiesen.

Gleichzeitig geben wir damit potentiellen Anwendern eine Vorstellung von der Komplexität einer anwenderspezifischen Schaltung.

Dieses Demonstrationssystem soll von einem PC zu mehreren anderen PCs adressierbar auswählbare Bilder übertragen.




Link 3 - 0
Link
Sub
Link-
Interface
AT-BUS
Transputer
T805
PC-Steuerung
Transputer-
Steuerung
FIFO
Adress/Datenbus 32 Bit
Adr-Latch
Adr.
4/1 Mb
RAM
I/O-Buffer

System-Merkmale:

- 50 MBit/s Brutto- 29 MBit/s Netto-Übertragungsrate
- freie Adressierbarkeit
- 2 - 256 Stationen
- gleiche Bandbreitenverteilung auf alle Stationen (DQDB-Verfahren)
- echtzeitfähig
- parallele Schnittstelle, 8 - 64 Bit breit konfigurierbar
- automatische Initialisierung und Einsynchronisation aller Stationen nach dem Einschalten
- automatische Fehlererkennung und Auftrennung des Bussystems an der Fehlerstelle
- automatische Neuinitialisierung nach Beseitigung des Fehlers
- 8/10-Codierung für ausbalancierten, seriellen Datenstrom
- Service und Diagnoseschnittstelle
- optionale elektrische serielle Übertragung

DQDB Funktionsprinzip

Das Funktionsprinzip des opto-elektronischen Bussystems wurde aus dem DQDB-Zugriffsverfahren abgeleitet. Dieses Verfahren befindet sich zur Zeit in der Normung und ist für den Einsatz in MAN´s (Metropolitan Area Networks) gedacht. DQDB steht für "Distributed Queue Dual Bus" (Verteilte Warteschlange, Doppel-Bus). Es handelt sich dabei um ein deterministisches Zugriffsverfahren, wie es beispielsweise für Echtzeitsysteme benötigt wird.

In Bild 1 ist der prinzipielle Aufbau eines solchen Systems dargestellt. Bei diesem Verfahren werden zwei entgegengesetzt gerichtete Übertragungsstrecken, hier als BUS A und BUS B bezeichnet, verwendet. Die Busstationen, die sich am Anfang der Übertragungsstrecken befinden, erzeugen über einen hier mit G bezeichneten Generator einen Datenstrom, der aus kontinuierlich aufeinanderfolgenden Datenrahmen besteht (siehe Bild 3).

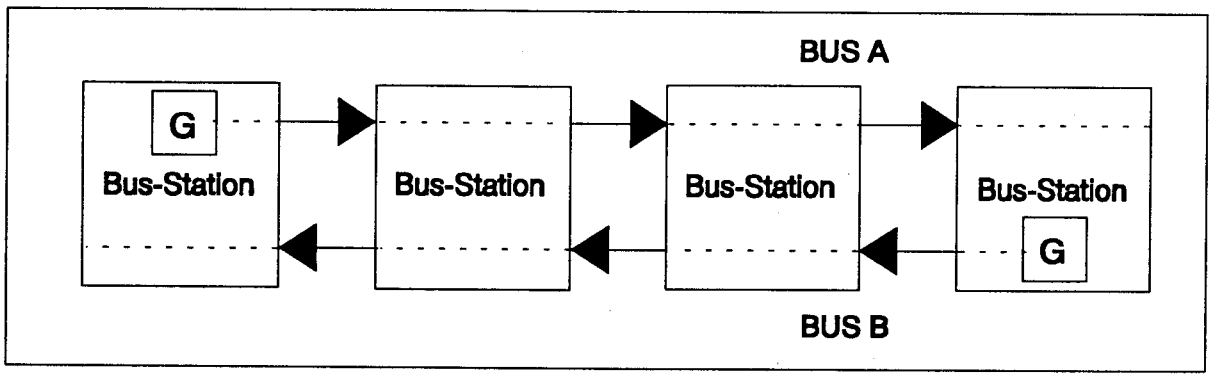


Bild 1 : Bustopologie

Jeder Datenrahmen enthält neben den eigentlichen Nutzdaten Steuerinformationen, über die der Zugriff geregelt wird. So gibt es ein Feld über das signalisiert wird, ob der Datenrahmen frei ist oder ob er gültige Daten enthält. Weiterhin enthält jeder Rahmen ein sogenanntes Request-Feld, durch das eine Datenübertragung initialisiert werden kann.
Jeder Teilnehmer am Bus (Busstation) ist mit beiden Übertragungsrichtungen verbunden und hört diese Datenströme ständig ab. In jeder dieser Busstationen existiert nun für jede Übertragungsrichtung jeweils ein Zähler, der durch passierende freie Datenrahmen dekrementiert und durch gesetzte Request-Felder im Datenstrom der Gegenrichtung inkrementiert wird (siehe Bild2).

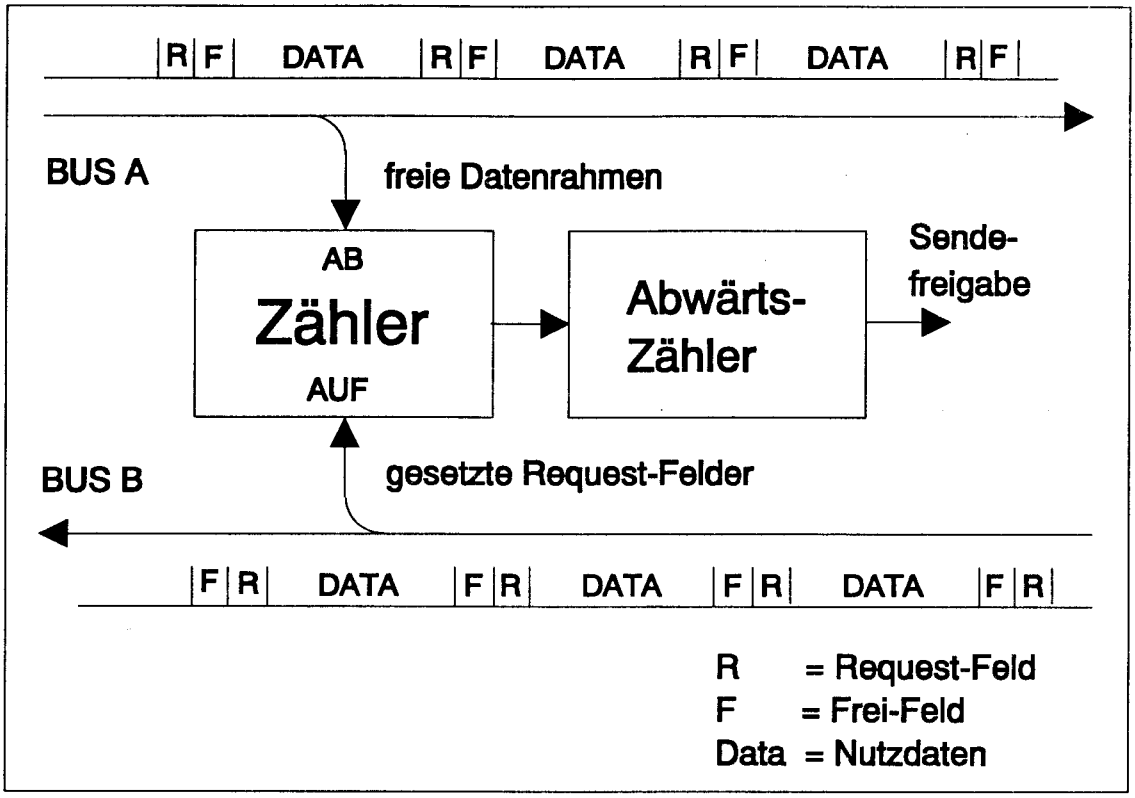


Bild 2 : Die Zähler der verteilten Warteschlange

Wenn eine Station senden will, so setzt sie im Datenstrom der entgegengesetzten Übertragungsrichtung ein freies Request-Feld und signalisiert somit allen vor ihr liegenden Stationen ihren Sendewunsch. Anhand des Zählerstandes in der eigenen Station ist sie über die Anzahl an Sendewünsche der nach ihr folgenden Stationen informiert.
Bevor sie einen freien Datenrahmen belegt, läßt sie deshalb soviele freie Datenrahmen passieren, wie zum Zeitpunkt des Requestsetzens durch den Zähler angezeigt wurden. Hierzu ist ein Abwärtszähler vorhanden, der mit dem Inhalt des Zählers zum Zeitpunkt des Setzens des Request-Feldes geladen wird und dann durch jeden passierenden freien Datenrahmen dekrementiert wird. Auf diese Art und Weise ist gewährleistet, daß die zur Verfügung stehende Bandbreite gleichmäßig auf alle sendewilligen Stationen verteilt wird.

Das für das opto-elektronische Bussystem festgelegte Format des Datenrahmens ist in Bild 3 dargestellt. Ein Datenrahmen besteht aus 110 Bit und beinhaltet neben Kontrollinformationen die Empfänger- und Sender-Adresse sowie 8 Datenbytes. Hieraus ergibt sich eine Nutzdatenrate von 58 %. Neben den bereits erwähnten Frei- und Request-Feldern ist in den Kontrollinformationen ein Paritäts-Feld enthalten, mit dem es möglich ist, Ein-Bit Fehler zu erkennen.

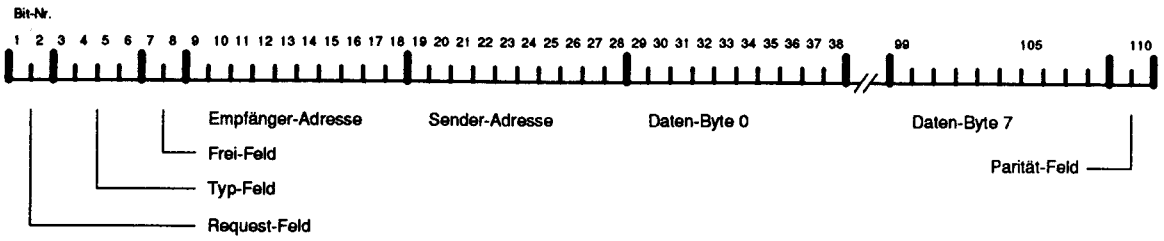


Bild 3 : Format des Datenrahmens

Ein weiteres sogenanntes Typ-Feld bietet außerdem die Möglichkeit, einzelne Datenrahmen von dem beschriebenen Zugriffsverfahren auszunehmen und für Sonderaufgaben zu verwenden. Das Typ-Feld kann 6 verschiedene Zustände annehmen, so daß bis zu 5 solcher besonderen Buszuteilungsmöglichkeiten realisiert werden können.
Eine mögliche Sonderaufgabe ist es zum Beispiel, daß eine Station einen speziell gekennzeichneten Datenrahmen (Typ + eigene Stationsadresse) immer belegen darf. Je nachdem wie oft solche Rahmen von der Kopfstation generiert werden, kann einer oder mehreren Stationen daduch eine feste Übertragungsbandbreite zugeordnet werden. Die verbleibende Bandbreite verteilt sich dann wieder gleichmäßig auf die verbleibenden Stationen.

Von der als Kopfstation arbeitenden Station werden zusätzlich zu den Datenrahmen zyklisch Synchronisationszeichen ausgesendet, anhand derer die folgenden Stationen Übertragungsfehler erkennen können und sich dann neu auf den Datenstrom synchronisieren.
Im Fall einer Unterbrechung in der Übertragungsstrecke übernimmt die Station hinter der Unterbrechungsstelle die Funktion der Kopfstation. Der Bus zerfällt dadurch in zwei Teil-Busse, die jedoch für sich jeweils voll funktionsfähig sind.

Das Bussystem besitzt eine Schnittstelle zur anwenderspezifischen Datensenke- oder Quelle, die sich in ihrer Busbreite flexibel konfigurieren läßt. Es sind Busbreiten von 8, 16, 32 und 64 Bit einstellbar.
Bild 4 gibt eine Bussystem-Übersicht an.

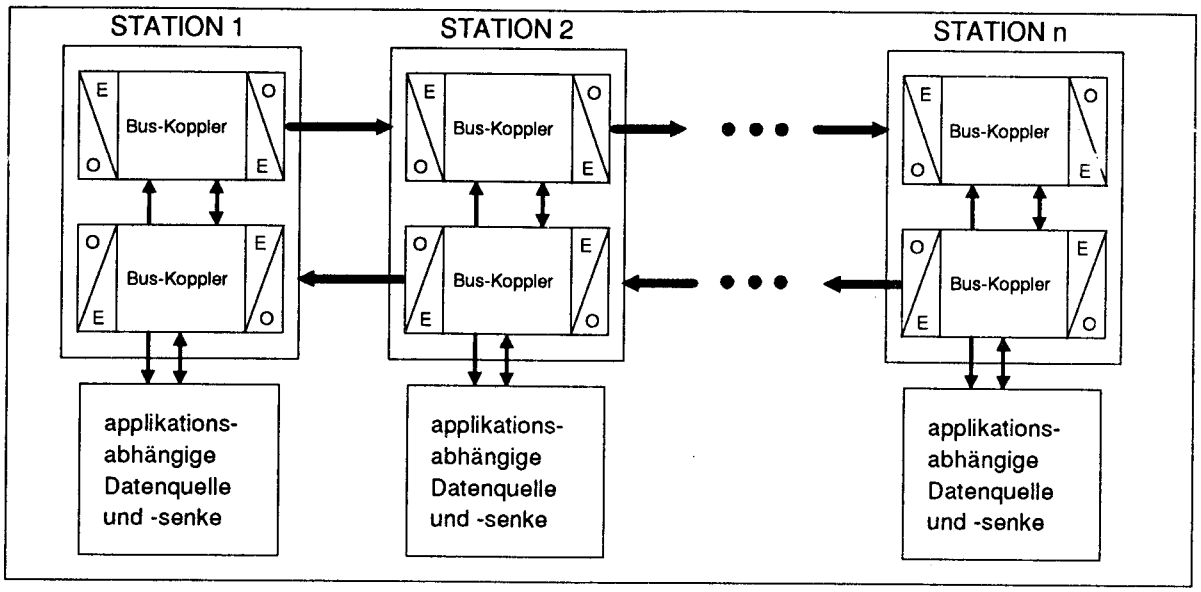


Bild 4 : Systemübersicht

Wie vorher erwähnt, kann der Datenübertragungsrahmen eine Empfänger- und eine Sender-Adresse enthalten. Werden diese mit je 8 Bit codiert, so lassen sich bis zu 256 Busteilnehmer adressieren.

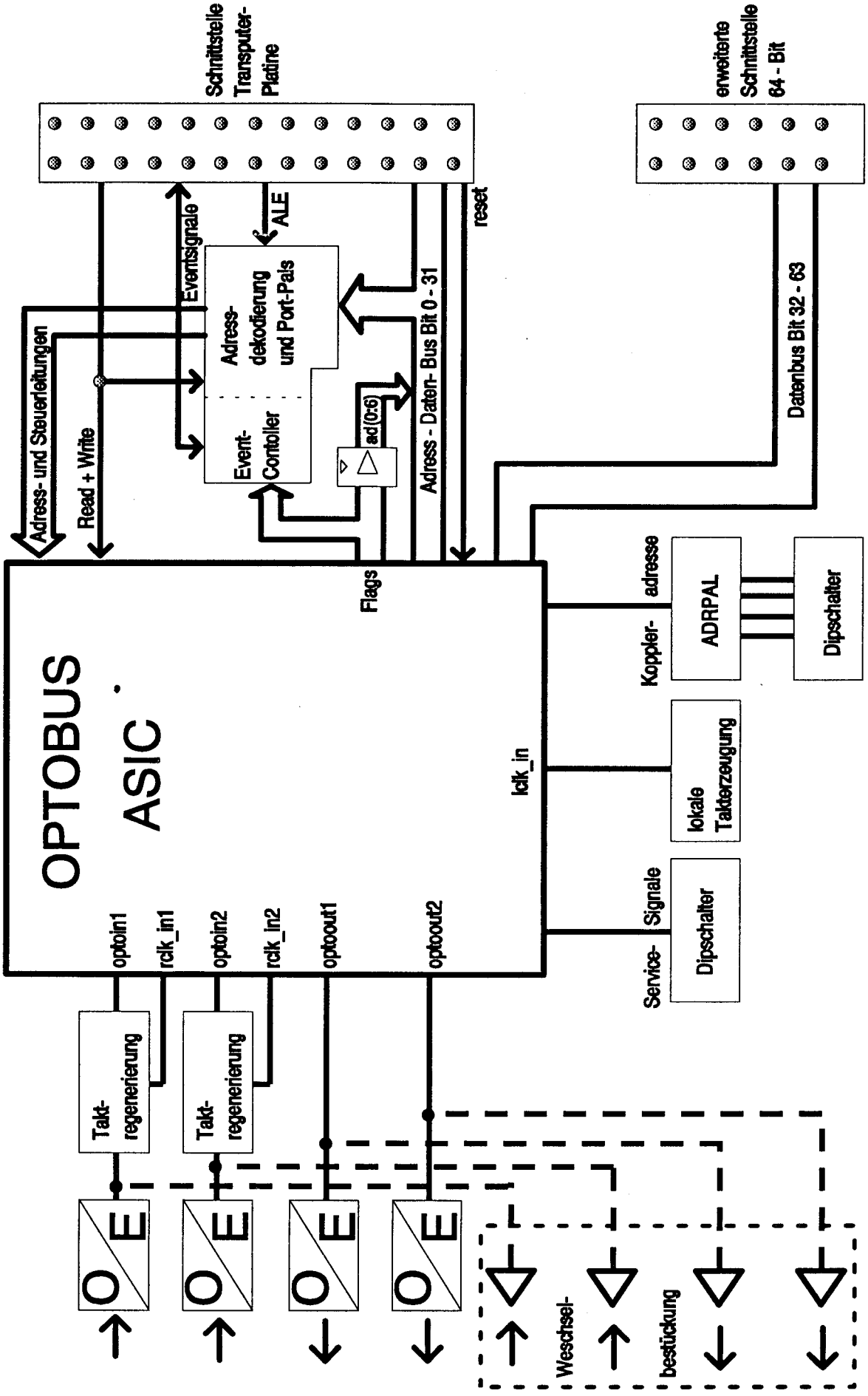
Das Übertragungsmedium muß nicht notwendigerweise ein Lichtwellenleiter sein. Bei Einsatz des selben Verfahrens und auch der gleichen Bausteine lassen sich kürzere Übertragungsstrecken bzw. solche mit geringerer Datenrate auch kostengünstiger mit Kupferkabel realisieren.

Kurzbeschreibung Optobus-Leiterplatte

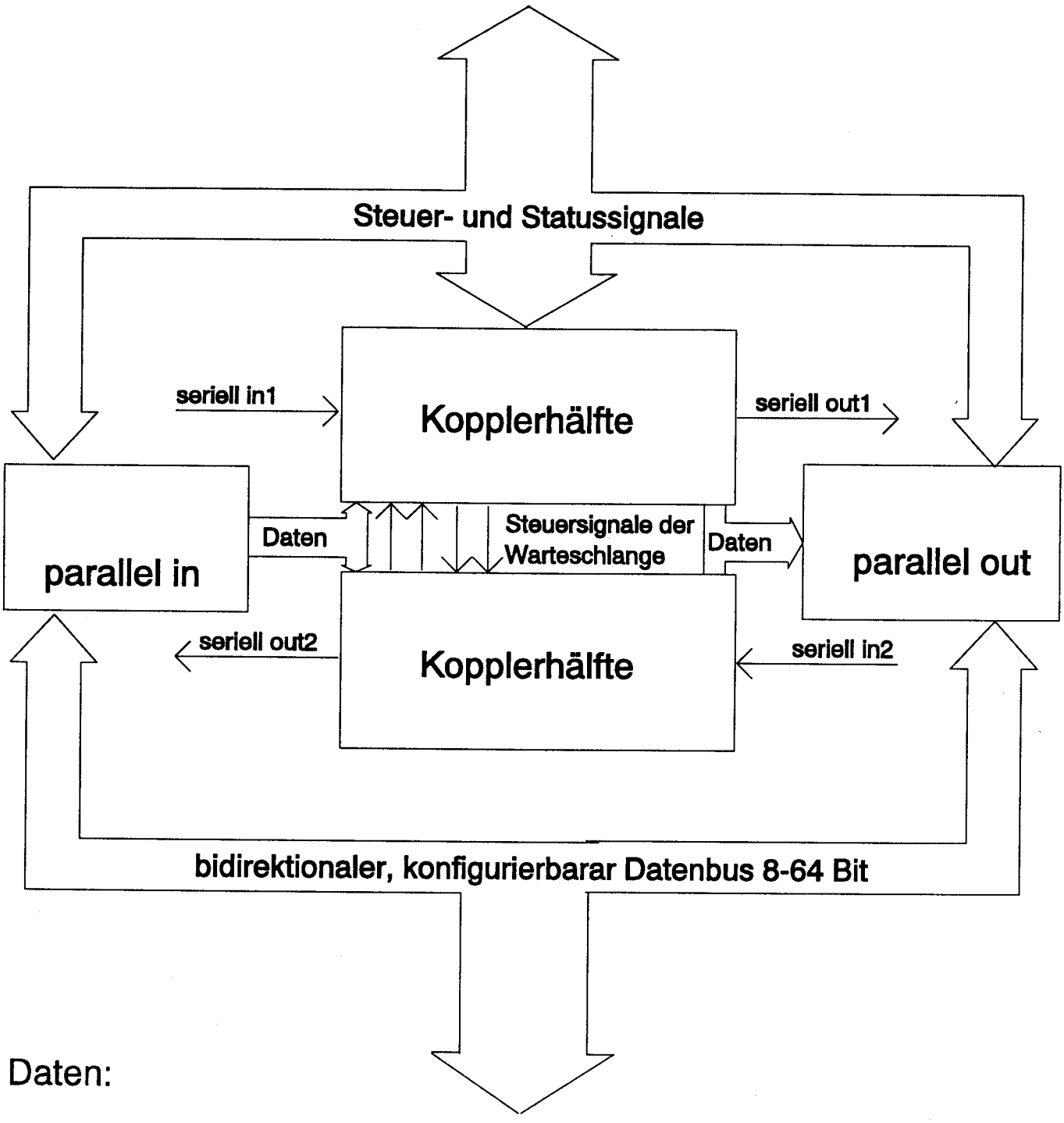
Auf der seriellen Seite haben wir bei optischem Medium vier Wandler, für jede Richtung einen Empfänger und einen Sender. Als Wechselbestückung kann man stattdessen Bustreiber einsetzen, und über Koaxleitung übertragen. Die eingehenden Daten werden über einen Taktregenerator geführt, der daraus den Systemtakt zurückgewinnt, und die Daten wieder auffrischt. Über Schalter einstellbar sind Servicesignale, die den Buskoppler in einen bestimmten Modus zwingen, und zur Diagnose dienen. Für den Betrieb als Generator, wird der Systemtakt von einem Quarzoszillator erzeugt. Über einen weiteren Schalter wird die 8Bit Adresse jedes Kopplers eingestellt, die während der Initialisierung seriell eingetaktet wird. Als Steuerndes System, auf dem auch unser Demo laufen wird, haben wir ein B008 Kompatibles Transputerboard, mit PC-Schnittstelle entwickelt. Die Optobus-Platine besitzt eine Schnittstelle zu diesem Transputerboard. Vom A/D-Bus dieses 32Bit-Systems dekodieren wir alle Steuersignale aus, und können den Optobus ansprechen, als wäre er ein Speicherbereich des Transputers. Außerdem können vom Optobus aus Events am Transputer ausgelöst werden, und vom Transputer aus kann man die Flags, die den Zustand des Asic anzeigen, auslesen.

Kurzbeschreibung Optobus-Asic

Das Blockschaltbild zeigt die Aufteilung der Funktionen im Optobus-Asic.
Wir haben eine parallele Schnittstelle für von außen eingehende Daten, und eine für nach außen gehende Daten. Diese Schnittstellen greifen beide auf den bidirektionalen Bus zu, der durch von außen angelegte read/write-Signale in der Richtung umgeschaltet wird. Außerdem haben wir eine Reihe von Signalen, die von außen die Funktionen des Asic steuern, sowie Statussignale, die z.B. den Kopplermodus oder Datenempfang anzeigen. Für jede Richtung ist eine Kopplerhälfte vorhanden, die unabhängig und asynchron zur gegenüberliegenden läuft. Die Steuerung der Warteschlange wird über Signale im Handshake-Verfahren geregelt.
Aufgrund der Busbreite von 64Bit mußten wir ein 144 Pingrid-Array-Gehäuse wählen, das wir auch bis auf den letzten Pin ausgereizt haben. Der GF2G1-Master der von uns verwendet wurde stellt ca. 13500 Gatterfunktionen zur Vefügung, wovon wir 9600 Gatterfunktionen verwendeten. Das entspricht 53000 aktiven Transistoren, und einer Ausnutzung von 76%. Damit liegen wir noch 4% unter den Vom IMS vorgeschlagenen 80%, und hätten also noch Platz gehabt die eine oder andere Idee zu verwirklichen, was aus Zeitgründen aber leider unmöglich war.


Schnittstelle Transputer-Platine
erweiterte Schnittstelle 64 - Bit
Eventsignale
ALE
reset
Adress- und Steuerleitungen
Read + Write
Adress-dekodierung und Port-Pals
Event-Contoller
ad(0:6)
Adress - Daten- Bus Bit 0 - 31
Datenbus Bit 32 - 63
Flags
OPTOBUS ASIC
adresse
Koppler-
ADRPAL
Dipschalter
clk_in
lokale Taktterzeugung
Signale
Service-
Dipschalter
optoin1
rclk_in1
optoin2
rclk_in2
optoout1
optoout2
Takt-regenerierung
Takt-regenerierung
O/E
O/E
O/E
O/E
Wechsel-bestückung

Blockschaltbild Optobus-Asic



Daten:

144 PGA-Gehäuse
9600 Gatterfunktionen
53000 aktive Transistoren
76% Ausnutzung

Produktion eines ASIC beim Institut für Mikroelektronik Stuttgart

IIT

-Erzeugung einer EDIF-Netzliste, eines Pin-Files, eines Supersynchronous-Tracefiles und eines At-Speed-Tracefiles nach den Richtlinien des IMS

IMS

-Übersetzung der EDIF-Netzliste in das Format des IMS
-Erstellung der statistischen Daten (Ausnutzung etc:)
-Prüfung der Netzliste
-Übersetzung der Tracefiles und deren Prüfung
-Extraktion von Stimulifiles aus den Tracefiles
-(Fehlersimulation)
-> Simulation der Schaltung beim IMS -> Erzeugung eines Tracefiles -> Vergleich mit unserem Tracefile

Freigabe der Netzliste

-Platzieren und Routen der Schaltung
-Extraktion einer Netzliste mit realen Kapazitäten
-Vergleich dieser, mit unserer Netzliste (Logiktest)
-Design-Rule-Check
-Postroutesimulation und erneuter Vergleich mit unseren Tracefiles

Layout-Freigabe und Produktion

-Wafertest: Stromaufnahme, Anlegen der Stimuli
-> Vergleich mit unseren Simulationsergebnissen
-Gut-Schlecht-Auswahl aller Chips
-Verpackung der guten Chips
-Test der IC bei Betriebsfrequenz

Auslieferung der ASIC

Baustein zur Datenverschlüsselung

Bearbeiter : **Peter Jonski**
Fachbereich : **Industrieelektronik**
Betreuer : **Prof. A. Führer**
Fachhochschule : **Ulm**

4.1

TABLE OF CONTENTS

LIST OF FIGURES

1. Vorwort

1.1 Thema der Diplomarbeit

Das Thema der Diplomarbeit war der Entwurf eines integrierten Bausteins zur Datenverschlüsselung nach einem kryptografischen Verfahren.
Dieser Chip wurde als Peripheriebaustein konzipiert und arbeitet in einem IBM PC AT oder Kompatiblen mit der Mikroprozessorfamilie 80x86 zusammen. Der Data Encryption Processor (DEP) kann Daten sowohl ver- als auch entschlüsseln. Er ist in der Lage seine Daten entweder mit dem Hostprozessor auszutauschen, kann jedoch auch mit dem Plattform DMA-Controller zusammenarbeiten. Der Prozessor verarbeitet seine Daten im Binärformat und unterscheidet keine anderen Formate. Die Wortbreite des Prozessors beträgt 8 Bit.

2. Einfuehrung zum verwendeten Chiffrieralgorithmus

2.1 Historie

Bei dem verwendeten Chiffrieralgorithmus handelt es sich um ein sogennantes Blockchiffrierverfahren.
Im Gegensatz zu Bit-stream-Verfahren, bei denen ein kontinuierlicher Bitstrom kodiert wird, werden bei diesem Verfahren immer mehrere Bits parallel chiffriert.
Das Verfahren gehört zu den klassischen Chiffriermethoden.

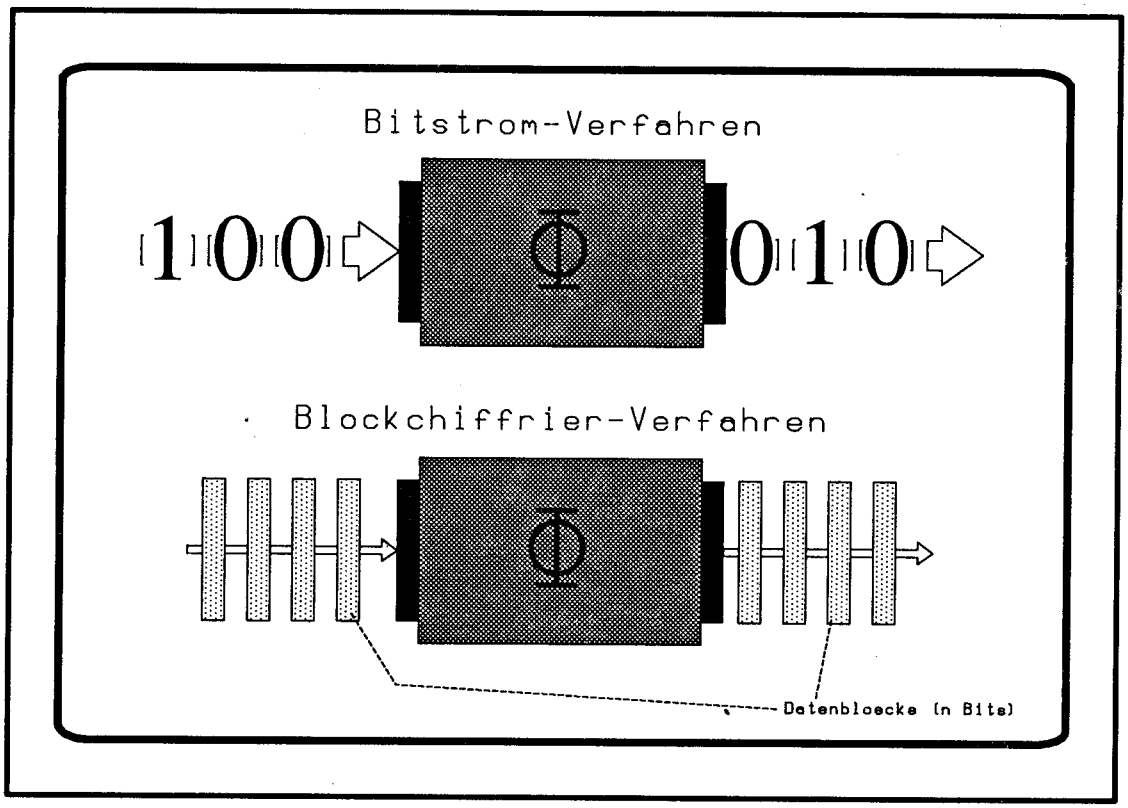


Abb. 1 Vergleich Bit-stream-Ciphering und Block-Ciphering

4,4

Vor zwei Jahren wurde der Algorithmus von der kryptografischen Abteilung der Universität Ulm unter der Leitung von Prof. Dr. Trautner in seinem Geschwindigkeitsverhalten entscheidend verbessert. Die Abteilung entwickelte ein allgemeines Designkonzept unter dem Namen ' Schnelles Blockchiffrierverfahren oder RNK-Design', das sich durch eine hohe Flexibilität bei der Umsetzung in Hardware auszeichnet. Dieses Verfahren ist bereits zum Patent angemeldet.
Durch die Wahl der Parameter R, N und K kann der Entwickler entscheiden, ob ein Design zeitoptimales oder hardwareoptimales Verhalten zeigen soll.
Für den von mir entworfenen Chip wurde die hardwareoptimale Lösung ausgewählt.

2.2 Schnelle Blockchiffrierverfahren

2.2.1 Ablauf einer Verschluesselung

Prinzipiell ist der Ablauf einer Verschlüsselung beim zeitoptimalen und beim hardwareoptimalen Design gleich.
Bei der Chiffrierung durchläuft der Klartext T mehrere Runden R. In jeder Runde wird eine eindeutige Funktion ϕ aus N-Funktionsblöcken insgesamt K-mal auf den Text angewendet. Daraus folgt die gesamte Funktion nach R Runden:

$$\Phi = \phi^{K_1} * \phi^{K_2} * \phi^{K_3} * \ldots * \phi^{K_R}$$

wie das folgende Bild verdeutlicht:

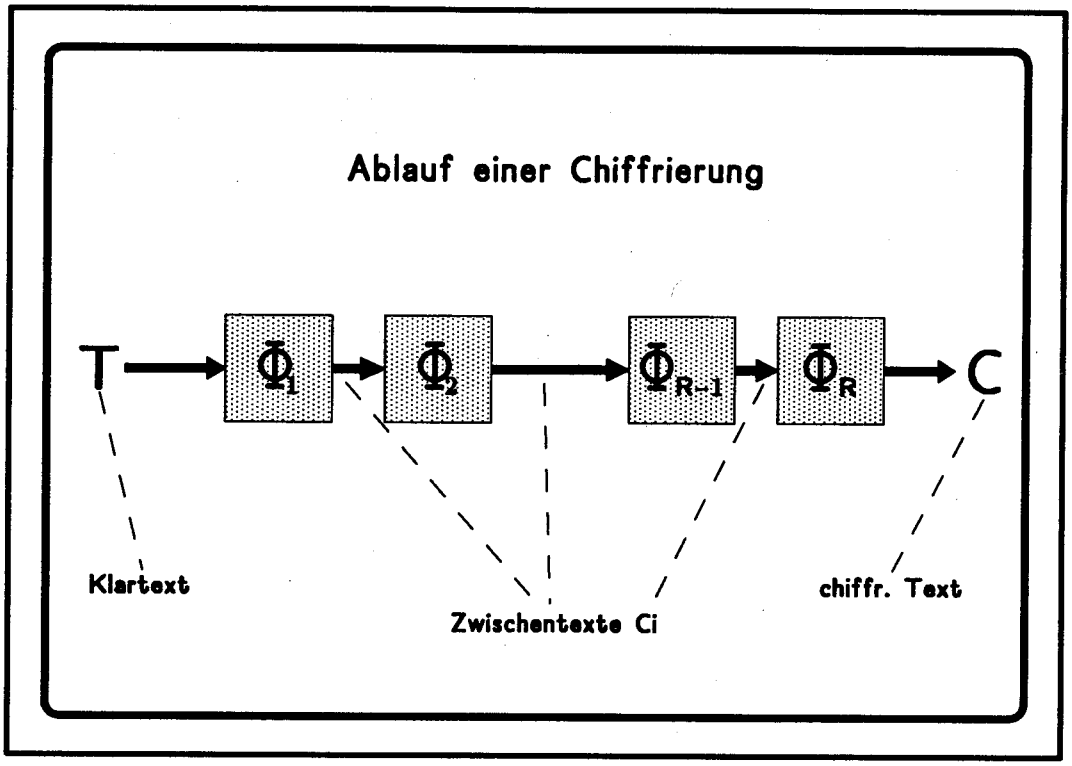


Abb. 2 Prinzipieller Ablauf einer Chiffrierung mit K=1

Jede Funktion ϕ_i führt eine äußerst einfache Operation mit den Daten durch.

Trotzdem ergibt ihre Verkettung eine sehr komplizierte Funktion.
In der ersten Runde erzeugt eine Funktion aus dem Klartext T einen Zwischentext C_1.

$$C_1 = T*\phi^K{}_1$$

In jeder weiteren Runde wird aus dem Zwischentext C_i der Zwischentext C_{i+1} und in der letzten Runde schließlich der chiffrierte Text C erzeugt.

$$C = C_{R-1} * \phi^K{}_R$$

Die Zwischentexte C_i sind von außen nicht beobachtbar.
Jede Verschlüsselungsfunktion besteht aus einer Anzahl verschiedenartiger Tauschfunktionen. Im vorliegenden Fall wird sie aus einer Abbildung und einer Transposition gebildet:

$$\phi_i = d_i * t_i \quad d_i = \text{Abbildung}, \; t_i = \text{Transposition}$$

Um einen kleinen Harwareaufwand zu erzielen, sollte R und N möglichst klein gewählt werden. Die Anzahl der Wiederholungen K muß dann allerdings genügend groß sein, um eine ausreichende Permutation der Daten zu gewährleisten.
Aus diesen Gründen entschieden wir uns für die Werte R=1, N=2 und K=16.

2.2.2 Einsatz eines Schluessels

Die Auswahl der Permutationsfunktionen in einer Runde geschieht mit Hilfe eines Schlüssels, der von dem Benutzer vor der Ver- bzw. Entschlüsselung angegeben wird. Anstelle eines statischen Sclüssel verwendet das RNK-Design einen zeitabhängigen Schlüssel. Nach jeder Wiederholung in einer Runde wird dieser Schlüssel, wie der Text, mittels einer Funktion modifiziert.
Die Permutation des Schlüssels geschieht mit Hilfe einer Tauschfunktion. Ob der verschlüsselte Text oder der Klartext jeweils zur Auswahl einer Permutation benutzt werden soll, bleibt dem Designer überlassen. In diesem Design wurde zur Modifizierung des Schlüssels der erzeugte Zwischentext verwendet.
Durch die Verknüpfung von Text und Schlüssel erreicht man, daß sowohl Text als auch Schlüssel voneinander abhängig sind.
Die erforderliche Länge des Schlüssels ist von der Anzahl der Wiederholungen pro Runde und von der Anzahl der verschiedenen Tauschfunktionen abhängig. Im vorliegenden Fall stehen zwei unterschiedliche Funktionen zur Auswahl. Eine Abbildung und eine Transposition. Bei beiden Funktionsarten ist die Auswahl zwischen vier verschiedenen Abbildungen und Transpositionen möglich. Deshalb benötigt man zur Auswahl der Tauschoperationen insgesamt 4 Bit. Da in unserem Entwurf jede Runde 16 mal durchlaufen wird, ergibt sich die Schlüssellänge zu 16*4 Bit = 64 Bit.

2.2.3 Erhoehung der Datensicherheit durch Weisses Rauschen

Da die Schlüsselinformation mit dem chiffrierten Text kodiert wird und nicht explizit zur Verfügung steht, muß ein Unbefugter den Startschlüssel herausfinden um den Klartext sichtbar zu machen. Eine Möglichkeit den Schlüssel herauszubekommen wäre, das System wiederholt mit der Eingabe von unterschiedlichen Schlüsseln zu

starten, und die Anfangsblöcke des verschlüsselten Textes zu entschlüsseln. Dieser Versuch könnte solange wiederholt werden, bis der richtige Schlüssel gefunden worden ist.
Um diese Angriffsmöglichkeit zu vereiteln verfährt man wie folgt:
Beim Start verschlüsselt der DEP erst eine Reihe von Zufallszahlen, die er intern erzeugt. Erst danach verschlüsselt er Daten, die vom Benutzer geliefert werden. Die verschlüsselten Zufallszahlen werden vom Prozessor nicht verworfen, sondern vor den verschlüsselten Daten abgelegt, da sie später zur Entschlüsselung benötigt werden.
Bei der Datenentschlüsselung dechiffriert der Prozessor zuerst die, vor der eigentlichen Information abgelegten, chiffrierten Zufallszahlen und verwirft sie danach. Anschließend dechiffriert er die verschlüsselten Nutzdaten.
Mit dieser Methode wird das erste Datum eines Benutzers immer mit einem bereits durch Zufallszahlen permutiertem Schlüssel chiffriert. Da die Zufallsmuster dem Benutzer nie als Klartext sichtbar werden, kann der Schlüssel nicht durch Angriffe mit unterschiedlichen Texten rekonstruiert werden.
Mit dieser Methode wird ein Angriffsversuch noch weiter erschwert.

2.2.4 Die Chiffrierfunktionen

Im folgenden Kapitel soll der Aufbau einer Runde anhand von Bild 3 genauer erläutert werden.

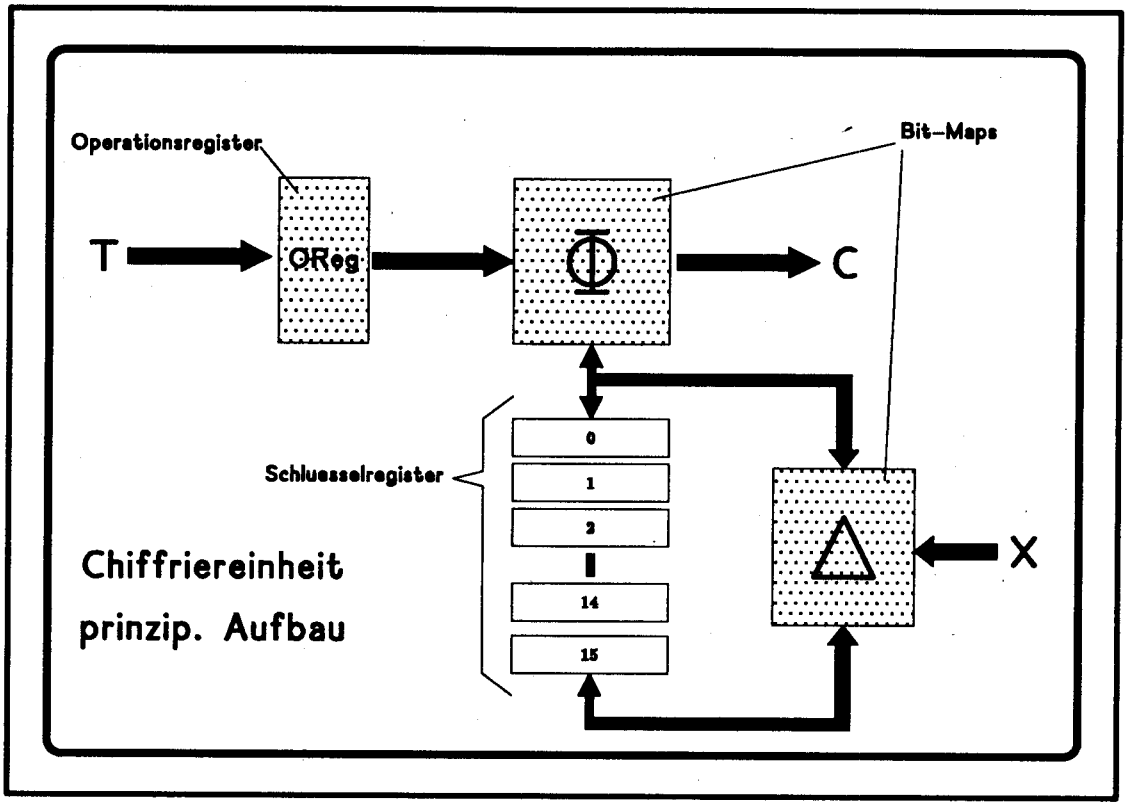


Abb. 3 Aufbau einer Runde mit Schluessel und Schluesselmanipulation

Jede Runde besteht aus einem Operationsregister und einer Baugruppe, in der das Datum manipuliert wird. Diese Einheit enthält die Chiffrierfunktion.
Zu Beginn jeder Runde wird das Datum in das Operationregister geladen. Mit jedem

Rechenschritt durchläuft der Inhalt des Operationsregisters die Manipulationseinheit (Abb. 4), in der die Tauschfunktionen ablaufen. Das Ergebniss von jedem Rechenschritt wird anschließend zurück in das Operationsregister geschrieben. Dabei entscheidet der momentane Schlüssel darüber welche Kombination der verschiedenen Tauschfunktionen ausgewählt wird. Dieser Vorgang wird nun K mal wiederholt.
Mit jeder Wiederholung rotiert der Schlüssel um ein Schlüsselsegment weiter und das vorherige Segment wird ebenfalls durch eine Funktion modifiziert. Für den Aufbau dieser Funktion kann im Prinzip der gleiche, wie für die Datenmanipulation gewählt werden. Sie kann aber auch einfacher oder komplizierter sein. Der Ablauf der Schlüsselchiffrierung entspricht dem Ablauf der Datenmanipulation. Die Auswahl der Tauschfunktion für den Schlüssel geschieht in diesem Fall durch Datenbits.

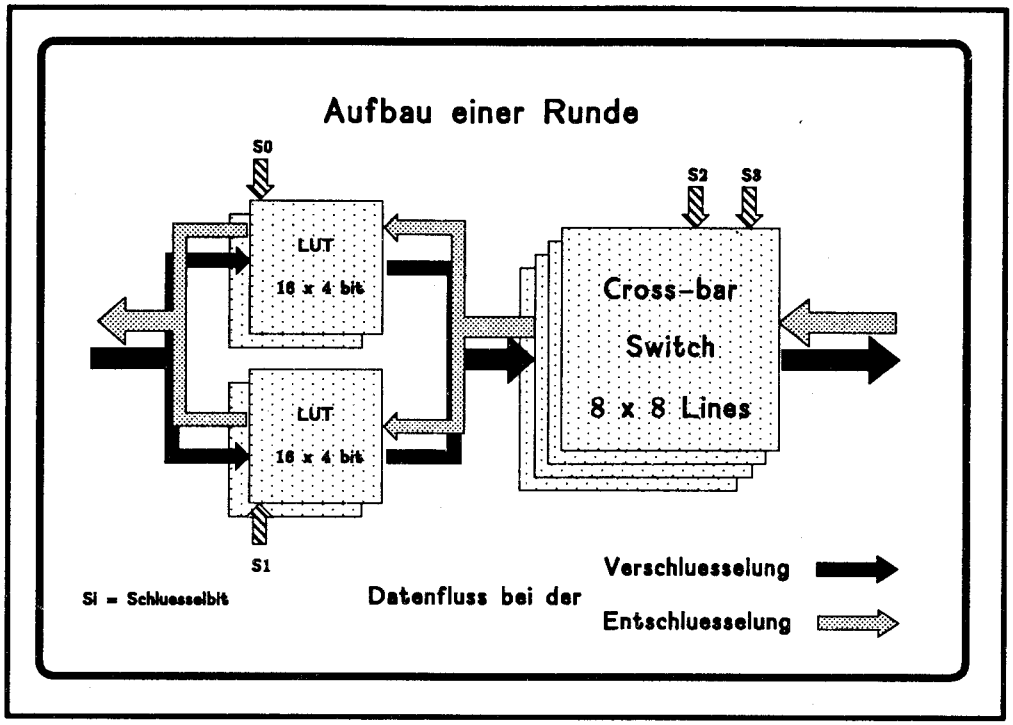


Abb. 4 Abbildung und Transposition

Die Tauschfunktion besteht aus einer Abbildung und einer Transposition. Innerhalb einer Verschlüsselungsrunde durchlaufen die Daten zuerst die Abbildung, und dann die Transposition. Dabei wählt ein Teil des Schlüssels je eine Abbildung und eine Transposition aus einer Anzahl von Abbildungen und Transpositionen aus.

Da es sich bei der Abbildung und Transposition um sehr einfache Funktionen handelt, werden sie auch Elementaroperationen genannt. Ihre Aufgabe ist die Manipulation eines Datenwortes. Alle Datenworte sind Elemente des Zahlenraumes Z_2E. Dabei ist E die Wortbreite in Bits. Man fordert nun, daß eine Elementaroperation jedes Datum wieder in den Raum Z_2E abbildet. In diesem Fall entspricht E = 8, da die Wortbreite des Prozessors 8 Bit beträgt.

Die e-Bit-Abbildung:
Der Datenblock wird in E/e e-Bit-Blöcke aufgeteilt. Mit jedem dieser Blöcke werden m Abbildung durchgeführt (m=2). Dies geschieht mit Hilfe von

4.8

Permutationstabellen.Von der Abbildung wird gefordert, daß sie eindeutig ist, d.h. jeder Eingangskombination darf nur eine Ausgangskombination zugewiesen werden. Außerdem muß sie bijektiv sein.
Damit die Permutationstabellen nicht zu groß werden, ist im vorliegenden Fall e=4 gewählt worden. So erhält man Tabellen mit 2^4=16 Werten zu 4 Bit. Für jeden e-Bit-Block wird durch ein Schlüsselbit ausgewählt, welche Abbildung weiter verwendet wird. Somit sind für die e-Bit-Abbildung im vorliegenden Fall 2 Schlüsselbit erforderlich.

Die E-Bit-Transposition:
Eine Bittranstransposition ändert einen ganzen E-Bit-Block. Mit einer Transposition wird die Reihenfolge der Bits innerhalb eines Blocks vertauscht.
Die Transposition muß ebenfalls eindeutig sein. Es dürfen nicht mehrere Eingangsbits einem Ausgangsbit zugeordnet werden oder umgekehrt. Sonst ist eine Rekonstruktion des chiffrierten Datums nicht mehr möglich. Im vorliegenden Fall werden 4 Transpositionen angeboten, zwischen denen mit 2 Schlüsselbit gewählt wird. Insgesamt sind zur Steuerung einer Tauschfunktion somit 4 Schlüsselbit erforderlich.

Nach Möglichkeit sollten für die Ver- bzw. Entschlüsselung jeweils die gleichen Abbildungstabellen benutzt werden können. Deshalb fordert man, daß diese Tabellen selbstinvers sind.D.h., wenn durch die Verschlüsselung z.B. eine 4 in eine 5 chiffriert wird, muß beim Entschlüsseln die 5 wieder in eine 4 verwandelt werden.
Dies gilt nicht für die Transpositionen. Da das jeweils i-te Bit dem t_i-ten Bit zugeordnet ist und umgekeht und die Transposition bidirektional ist, ergibt sich die richtige Zuordnung bei der Ver- bzw. Entschlüsselung automatisch.

2.2.5 Vorgehen bei der Entschluesselung

Um die Daten wieder lesbar zu machen, muß der gesamte Verschlüsselungsvorgang rückwärts durchgeführt werden. Dazu muß die Reihenfolge von Transposition und Abbildung vertauscht werden.
Zur besseren Erläuterung soll die Datenverschlüsselung noch einmal betrachtet werden:
Man definiert die Verschlüsselungsfunktion nach R Runden:
$\phi = \phi^{K_1} * \phi^{K_2} * ... * \phi^{K_R}$
Dann gilt beim Entschlüsseln die Umkehrfunktion von ϕ_i :
$\phi^{-1} = \phi^{-K_R} * \phi^{-K_{R-1}} * .... * \phi^{-K_1}$
Die Verschlüsselungsfunktion in einer Runde setzt sich aus einer Abbildung und einer Transposition zusammen :
$\phi_i = d_i * t_i$ d_i = Bitmap, t_i =Transformation
dann folgt für den Entschlüsselungsvorgang:
$\phi^{-1}{}_i = t^{-1}{}_i * d^{-1}{}_i$
Den Datenfluß in einer Runde zeigt die Abbildung 5. Wie man sieht kann diesselbe Hardware zum Ver- als auch Entschlüsseln benutzt werden. Es ist lediglich nötig, die Reihenfolge der Bit-maps und der Transposition zu vertauschen

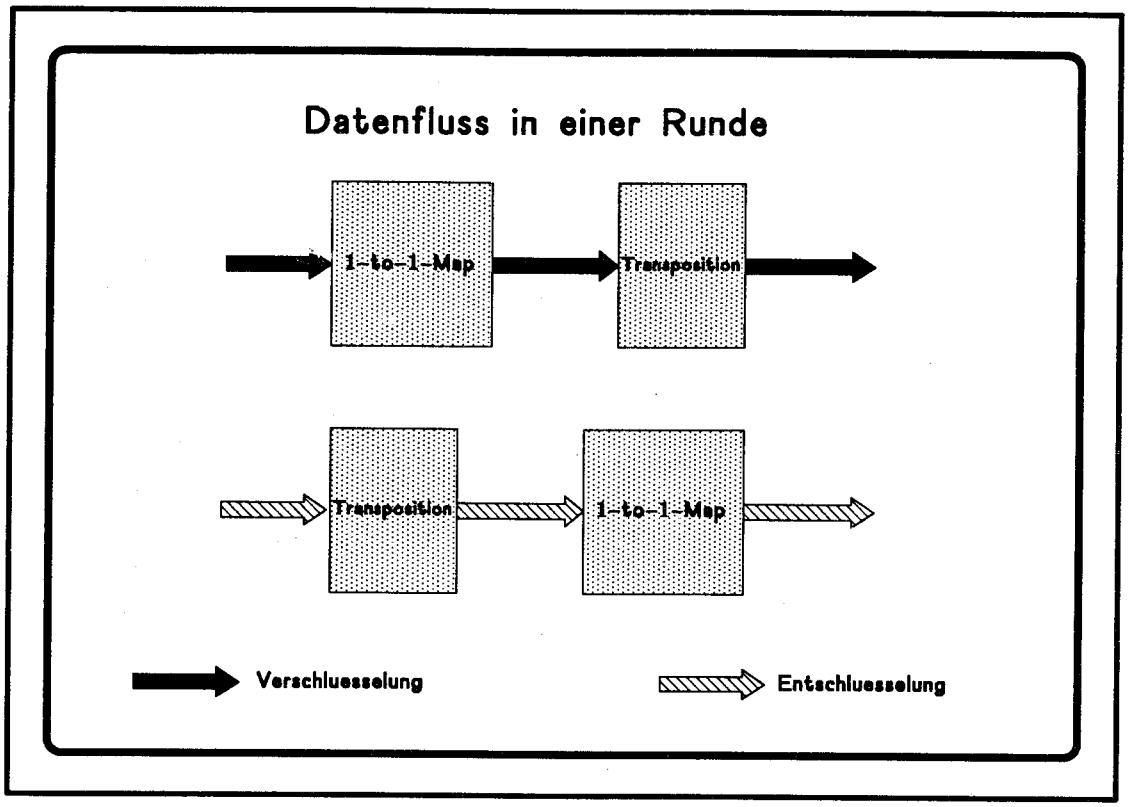


Abb. 5 Reihenfolge der Chiffrierfunktionen

2.3 Anwendungsgebiete

Ursprünglich war dieses Verfahren für die On-line Datenchiffrierung z.B. Telefon, Fax, ISDN usw. gedacht. Bei einem solchen Einsatz muß jedoch unter anderem folgender wichtiger Aspekt beachtet werden. Bei einer Übertragung in Netzwerken (Ethernet usw.) o. ä. werden die Daten meistens in einem bestimmten Rahmen übertragen. Der Header eines Rahmens darf nun auf gar keinen Fall chiffriert werden, da sonst der Empfänger die Daten nicht mehr richtig verarbeiten kann. Eine entsprechende Chiffrierhardware müßte eine by-pass-Funktion besitzen um den Übertragungsrahmen unverschlüsselt durchzulassen. Die Implementierung einer by-pass-Funktion hätte einen hohen Aufwand für den Schaltungsentwurf bedeutet. Deshalb wurde der Chip für die Off-line Datenchiffrierung entwickelt. Die Daten werden vor dem Senden durch den Chip verschlüsselt und dann vom Hostprozessor über eine Schnittstelle verschickt. Diese Methode macht die Verschlüsselung unabhängig von existierenden Daten- oder Übertragungsformaten.

4 10

3. Umsetzung des Algorithmus in einen Integrierten Schaltkreis

Bevor an einer konkreten Hardwarerealisierung gearbeitet wird, muß zuerst die Arbeitsweise und das Arbeitsumfeld des Prozessors definiert werden.

- o Der Prozessor soll in einem PC AT oder Kompatiblen zusammen mit einem Mikroprozessor 80x86 arbeiten. Daher benötigt er eine Bussteuerung, die sich am Protokoll des ISA-Bus im PC AT orientiert.

- o In einem Computersystem soll der Prozessor als reiner 'slave' arbeiten. Das heißt er kann nicht selbstständig Daten transferieren, sondern benötigt dazu immer einen Hostprozessor.

- o Der Prozessor benötigt für DMA Unterstützung eine geeignete Schnittstelle.

- o Damit der Prozessor vom Benutzer programmiert werden kann, braucht er ein Steuer- bzw. Statusregister.

- o Der Prozessor erhält für den Datenaustausch je ein Eingangsdaten- bzw. ein Ausgangsdatenregister.

3.1 Aufteilung in funktionale Gruppen

Zur besseren Übersicht der Aufgaben, die der Prozessor zu bewältigen hat, gliedert man den Baustein in mehrere Funktionsgruppen auf.

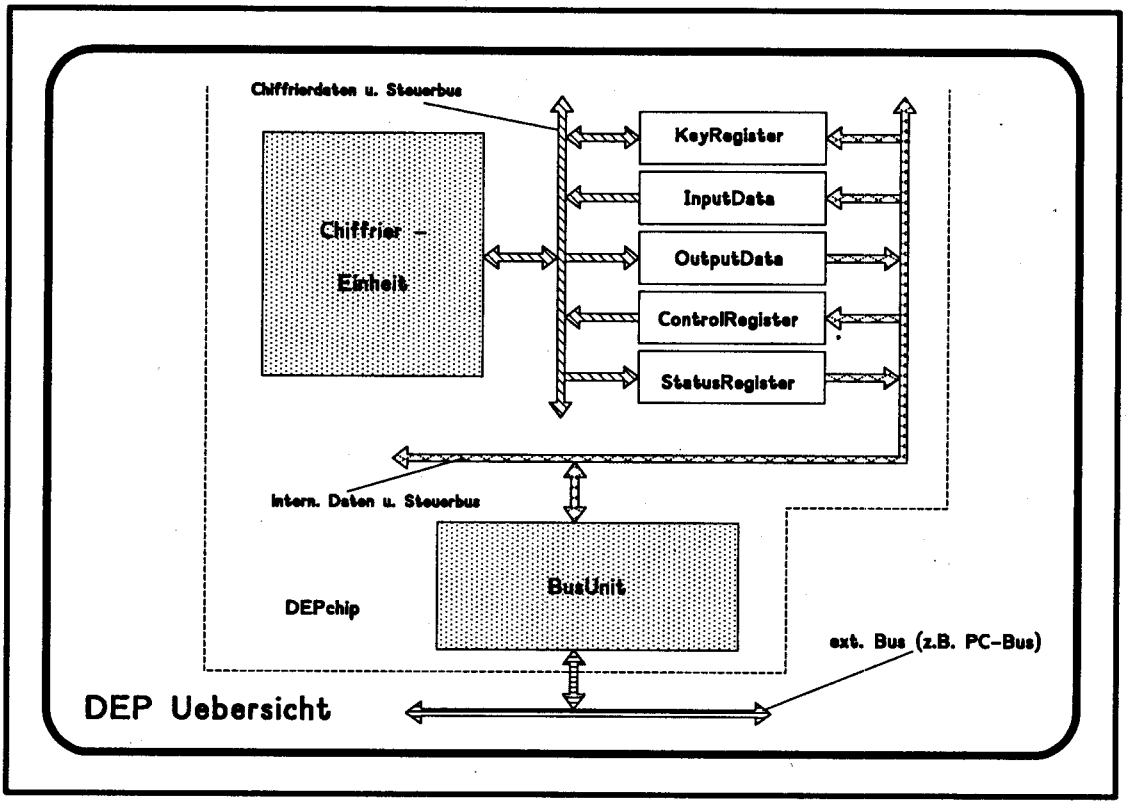


Abb. 6 Blockdarstellung des DEP

4.11

Dabei kann der Chip grob in drei Hauptbestandteile eingeteilt werden:

- Bussteuerung und Registerkontrolle
- Daten- und Steuerregister
- Chiffriereinheit

In der Abbildung 6 sieht man eine Blockdarstellung des Prozessors mit den Funktionsbaugruppen und den Steuer- bzw. Datenbussen

3.2 Aufbau und Wirkungsweise der Funktionseinheiten

In den folgenden Kapiteln soll die Funktion der einzelnen Baugruppen näher erläutert werden:

3.2.1 Das Businterface

Aufgabe des Businterface ist es, den Datenverkehr mit dem Hostprozessor oder dem DMA-Controller zu regeln. Daneben steuert es den internen Datenbus zwischen den Registern.
Über das Kontrollregister kann der Prozessor in zwei Betriebsarten gesetzt werden:

a) Der Prozessor arbeitet in der Betriebsart SingleByteTransfer. In diesem Modus entscheidet der Hostprozessor ob Daten in den DEP geschrieben oder ausgelesen werden sollen.
Ein typischer Datenaustausch läuft z.B. für die Verschlüsselung nach folgendem Schema ab:
Zuerst selektiert der Host den Chip über eine Adresse im I/O-Adreßraum des PC. Dann schreibt er ein Datum in ein Prozessorregister, oder liest ein Prozessorregister aus. Informationen, die der DEP bereit hält, entnimmt der Host in dieser Betriebsart immer den Registern des Verschlüsselungsbausteins. Der DEP kann in dieser Betriebsart keine Datenanforderungen an das System stellen.

b) Der Prozessor arbeitet im BlockTransfer-Modus. In dieser Betriebsart erfolgt der Datenaustausch über den DMA-Controller.
Vor dem Datenaustausch muß der Host den DMA-Controller mit der Blockgröße programmieren, die transferiert werden soll. Dann müssen noch die DMA-Kanäle gesetzt werden. Anschliessend setzt der Host das Startflag im DEP. Nun stellt der DEP eine DMA-Anforderung an den Controller und der Transfer beginnt. Am Ende des Transfers sendet der Controller eine Endkennung an den DEP. Dieser löst nun eine Systemanforderung aus und hält diese bis der Host sie quittiert. Jetzt kann der Benutzer die Chiffrierung beenden oder den DAM-Controller neu einstellen.

3.2.2 Die Register

Datenregister
Zu den Datenregistern gehört das Ein- bzw. Ausgangsdatenregister, sowie das Operationsregister. Diese Register enthalten die zu chiffrierenden Daten. Der Benutzer hat auf das Operationsregister keinen Zugriff. Deshalb ist das Register in Abbildung 5 nicht eingezeichnet. Auf die beiden anderen Register hat der Benutzer nur jeweils Schreib- bzw. Leserechte. Alle Register sind 8-Bit breit.

Steuerregister
Aufgabe dieser Register ist die Einstellung des DEP, sowie die Rückmeldung von internen Ereignissen an den DEP selbst oder an das System. Das Kontrollregister kann vom Benutzer nur beschrieben, das Statusregister nur gelesen werden.
Schlüsselregister
Dieses Register enthält den Schlüssel für die Datenchiffrierung. Dem Benutzer ist es nicht möglich das Register auszulesen. Die Registerbreite beträgt 64 Bit. Das Register selbst ist in 16 Blöcke zu je 4 Bit unterteilt. Von jedem 4-Bitblock dienen 2 Bit zur Auswahl zwischen vier Abbildungen und 2 Bit zur Auswahl zwischen vier Transpositionen. Vor jeder Verschlüsselung trägt der Benutzer seinen Schlüssel ein. Diese Programmierung des Schlüsselregisters durch den Benutzer, erfolgt durch 8-maliges Beschreiben der Registeradresse mit einem Byte.
Im Programmiermodus beträgt die Datenbreite für den Benutzer nur 8 Bit. Im Betriebsmodus beträgt die Registerbreite des Schlüsselregisters für die Chiffriereinheit nur 4 Bit. In dieser Betriebsart kann das Register jeweils um 4 Bit vorwärts bzw. rückwärts geschoben werden.

3.2.3 Die Chiffriereinheit

Kernstücke der Chiffriereinheit sind ein Zufallszahlengenerator, die eigentliche Bitmanipulationsbaugruppe, sowie ein Steuerwerk für den Ablauf der Ver- bzw. Entschlüsselung mit einer Schreib/Lese-Logik für den Datentransfer zwischen dem Operationsregister und den Datenregistern. Abbildung 7 zeigt den schematischen Aufbau der Chiffriereinheit:

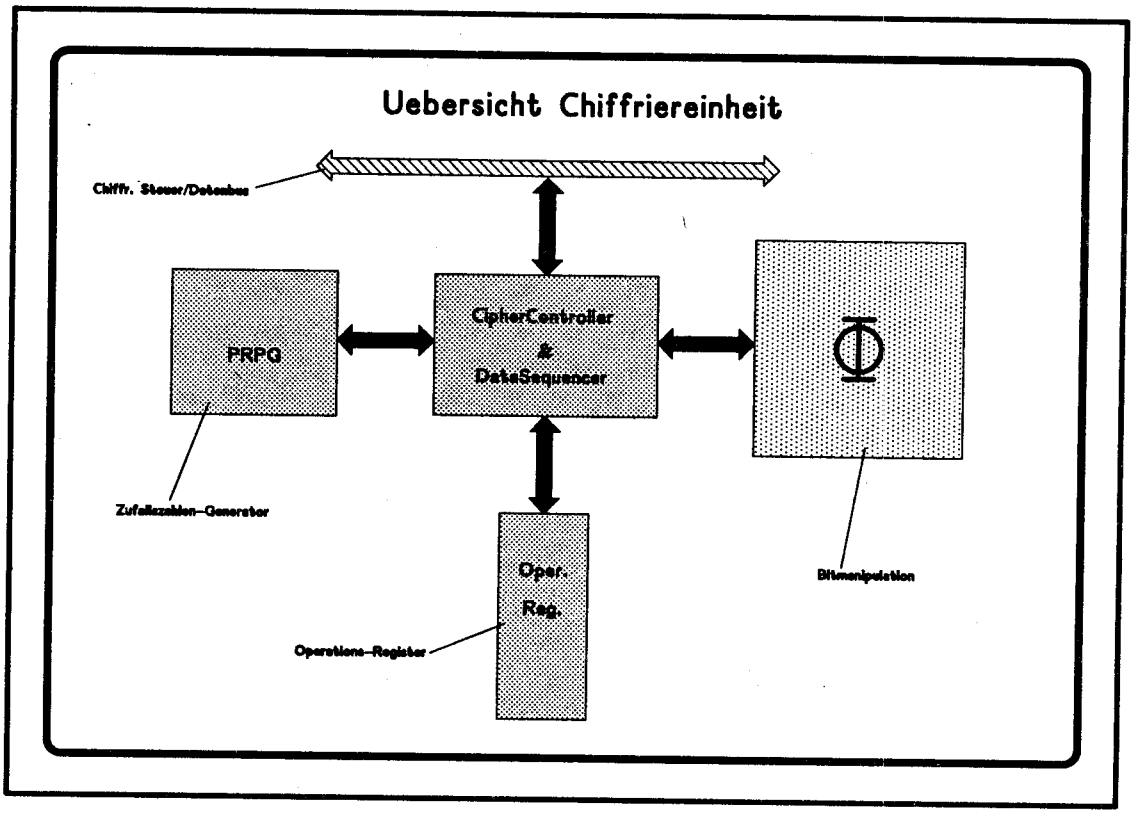


Abb. 7 Aufbau der Chiffriereinheit

Aufgabe der Chiffriereinheit ist die Ver- und Entschlüsselung der Daten. Insgesamt kennt die Chiffriereinheit zwei Betriebsarten:

- Verschlüsseln:
Dieser Modus wird wieder in zwei Betriebsarten eingeteilt. Und zwar müssen einmal die Zufallszahlen verschlüsselt werden und dann die Daten.
Im Modus 'Zufallszahlenverschlüsseln' liest die Chiffriereinheit die Daten aus einem Pseudo Random Pattern Generator (PRPG) aus und schreibt sie dann in das Ausgangdatenregister. In dieser Phase muß der Host oder der DMA-Controller nur lesen.
Im Modus 'Datenverschlüsseln' liest die Chiffriereinheit ihre Daten aus dem Eingangsdatenregister. Jetzt müssen Host oder DAM-Controller sowohl lesen als auch schreiben können.

- Entschlüsseln:
Diese Betriebsart unterscheidet sich nur im Modus 'Zufallszahlenentschlüsseln' von der Betriebsart Verschlüsseln. Hier müssen Host wie DMA-Controller nur auf Schreiben eingestellt sein, da die Chiffriereinheit die entschlüsselten Zufallszahlen nicht wieder hinausschreibt.
Im Modus 'Datenentschlüsseln' müssen Host oder DMA-Controller Daten lesen und schreiben können.

Der Betriebsmodus wird über das DE-Flag im Kontrollregister eingestellt. Defaultmäßig ist die Betriebsart Entschlüsseln gesetzt.

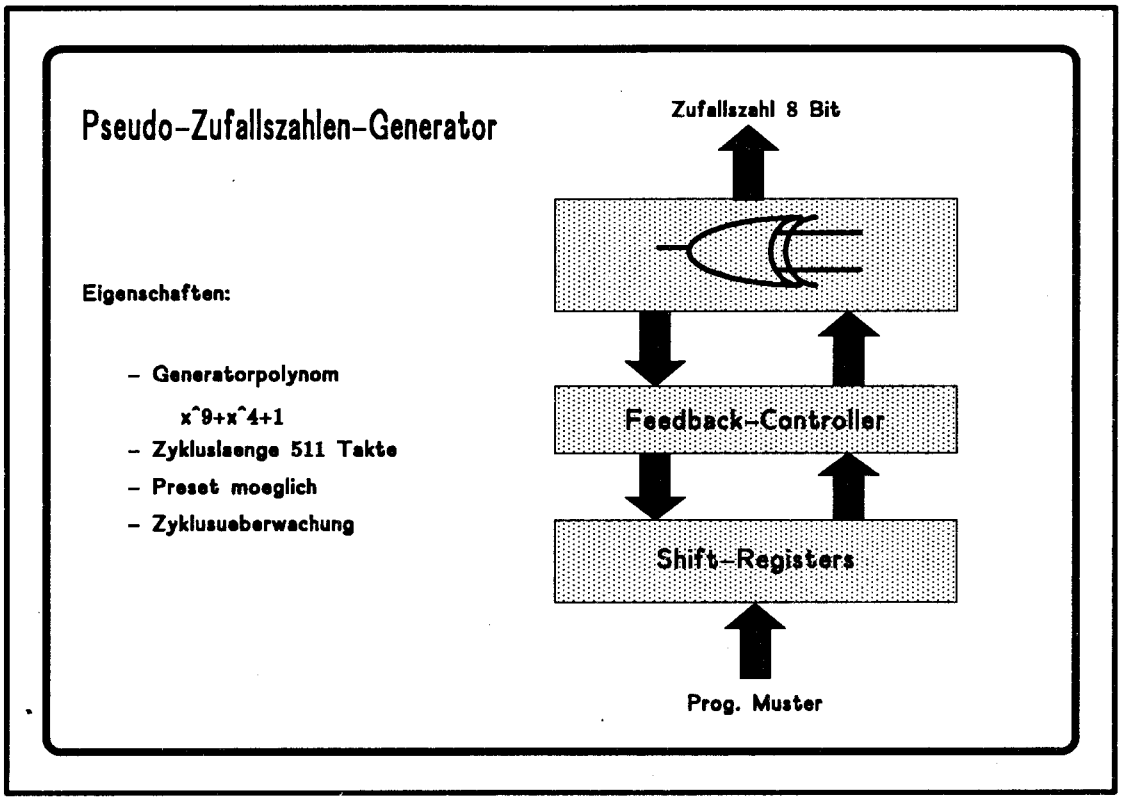


Abb. 8 Pseudo-Zufallszahlen-Generator

Der Zufallszahlengenerator (Abb. 8) besteht aus einem Linear Feedback Shift Register (LFSR) mit 9 Flip-Flops und erzeugt sog. Pseudozufallszahlen. Das Generatorpolynom für die Erzeugung der Zahlensequenz lautet x^9+x^4+1. Dieses Polynom minimiert den Hardwareaufwand für die Feed-Back-Logik und erzeugt gleichzeitig die längste Zeichensequenz, die mit 9 Flip-Flops möglich ist.
Um zu verhindern, daß der Generator nach dem Reset immer den gleichen Anfangswert besitzt, wird er mit einem Teil des Schlüssels geladen. Der Generator funktioniert aber nur dann, wenn mindestens eines der Flip-Flops ungleich Null ist. Deshalb sorgt ein Wächter dafür, daß der Generator nicht mit verbotenen Werten geladen werden kann.

Die Realisierung der Abbildungsfunktionen erfolgte durch Tabellen. Da wegen Softwareproblemen keine ROMs erzeugt werden konnten, wurden die Tabellen mit PLAs realisiert.

Für den Aufbau der Bittransposition wurde ein Kreuzschienenverteiler (Abb. 9) verwendet. Der Verteiler ist bidirektional und besteht aus einem Feld von insgesamt 8x8 Schaltern. Die Schalterkombinationen sind in einem PLA abgelegt. Für die Schalter wurden Tri-state-Treiber ausgewählt.

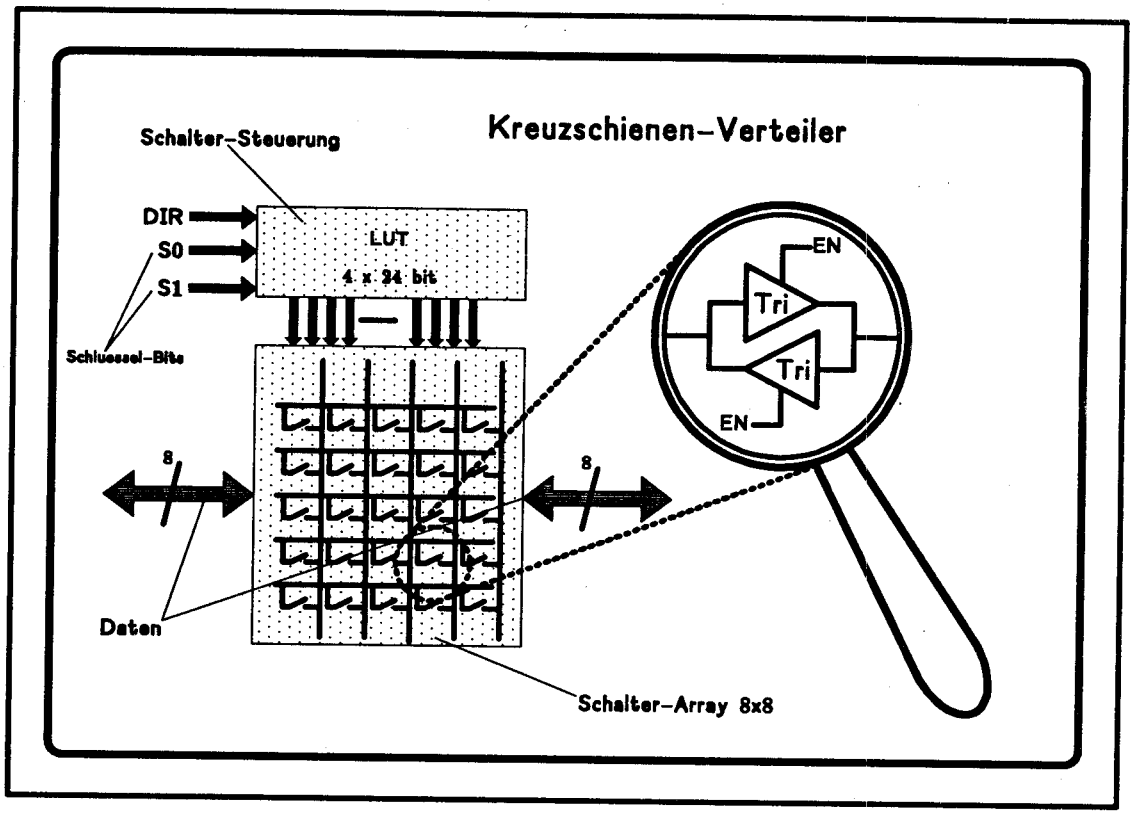


Abb. 9 Kreuzschienen-Verteiler 8x8 Leitungen

4. Realisierung der Schaltung

4.1 Technologie

Für die Realisierung der Schaltung standen zwei unterschiedliche Technologien zur Verfügung. Ein Möglickkeit bestand darin, den Schaltkreis mit Hilfe der IMS Zellenbibliothek auf einem GateForest-Gate Array zu realisieren. Die andere Möglichkeit die Verwendung des ES2 1.5micron Standardzellenprozesses.
Die Wahl fiel zugunsten der ES2 1.5um Bibliothek aus, da in dem Baustein ROMs bzw. PLAs verwendet werden sollten. Die Realisierung dieser Baugruppen auf einem Gate Array wäre wegen der schlechten Flächenausnutzung zu ungünstig gewesen.

4.2 Das CADENCE Entwicklungssystem

Die Umsetzung des Verfahrens in Hardware erfordert ROM. Aus diesem Grund sollte ein Werkzeug eingesetzt werden, das die automatische Erzeugung solcher Strukturen ermöglicht. Der von EUROCHIP gelieferte MENTOR- Entwicklungsbaukasten für ES2 schied aus, da er keine solchen Generatoren zur Verfügung stellt. Deshalb fiel die Wahl auf das CADENCE Entwicklungssystem.
Dieses Software-Paket enthält alle notwendigen Entwurfswerkzeuge von der Schaltplan eingabe bis zum Plazieren und Routen der Schaltung. Außerdem stellt es eine Reihe von Compilern zur automatischen Generierung von Bauteilen wie ROMs, RAMs, PLAs, FIFOs usw. bereit. Allerdings erzeugen diese Compiler nur eine Schaltung deren Aufbau dem Benutzer verborgen bleibt, d.h. der Zellenhintergrund bleibt unsichtbar.
Bedauerlicherweise traten bei der Arbeit mit CADENCE eine Reihe von Problemen mit den gelieferten Compilern auf:
Die Erzeugung von ROMs war nicht möglich. Der Compiler erstellte zwar das Symbol der generierten Schaltung, lieferte jedoch keine Beschreibung für die Simulation. Zudem trat während des Programmlaufs des Generators ein Fehler auf, der den korrekten Ablauf der Compilation verhinderte. Da nur mit dem PLA-Generator gearbeitet werden konnte, wurden die nötigen ROMs in der Schaltung durch PLAs ersetzt.

4.3 Testmoeglichkeiten des Bausteins

Um den Baustein nach der Herstellung auf einwandfreie Funktion überprüfen zu können, muß eine Testmöglichkeit in der Harware implementiert werden. Dabei sollte neben einem seriellen Testpfad auch ein Build-In-Self-Test (BIST) in die Schaltung eingebaut werden. Der eingebaute Selbsttest sollte eine Kontrolle der ROMs und PLAs durch Signaturprüfung ermöglichen.
Alle Compiler des CADENCE-Werkzeugskastens bieten die Möglichkeit die generierte Schaltung mit einem kombinierten Selbsttest auszustatten. Dieser Selbsttest generiert dann ein schaltungsspezifisches Prüfmuster, das von einer externen Testlogik ausgewertet werden kann.
Da die BIST-Option bei dem PLA-Compiler nicht fehlerfrei arbeitete, wurde die Schaltung lediglich mit einem seriellen Testpfad ausgestattet.

4.4 Simulation

Mit Hilfe der Simulation soll die Funktionsfähigkeit der Schaltung gezeigt werden. Aufgrund des hierarchischen Aufbaus, wurde die Schaltung Block für Block durchsimuliert. Dabei erfolgte nach jedem Designschritt ein Simulationsschritt.

4.4.1 Erstellen von Simulationsdateien mit SKILL

Der CADENCE-Entwicklungsbaukasten stellt für den Schaltungstest sowie die Simulation ein gutes Hilfsmittel zur Verfügung. Es handelt sich um die Programmiersprache SKILL, eine etwas vereinfachte Form von LISP. Mit dieser Programmiersprache können sehr komplexe Programme erstellt werden. Deshalb lag es nahe, SKILL für die Erzeugung von Simulationsvektoren zu verwenden.
Mit fortschreitender Größe und Komplexität einer Schaltung wird es für den Entwickler immer schwieriger den passenden Erwartungswert an den Ausgängen einer Schaltung nach einem Simulationsschritt vorauszubestimmen. Vorallem bei Automaten oder Dekodern ist die Bestimmung der Erwartungswerte bei einem großen Schaltungsumfang sehr zeitraubend.
Aus diesem Grund wurde von jedem Modul der Schaltung ein SKILL-Softwaremodell erstellt, das das geplante Verhalten der Hardware emuliert. Dabei wurde jedes Bauelement mit einer Prozedur beschrieben. Somit wurde bei dem Softwaremodell ebenfalls wie bei der Hardware ein strikt hierarchischer Aufbau verwirklicht.

4.4.2 Ablauf der Simulation

Der Ablauf der Simulation gliedert sich in mehrere Stufen. Zuerst wird die eigentliche Simulationsdatei erstellt. In dieser Datei werden die Vektoren eingetragen mit denen das Verhalten der Hardware überprüft werden soll.
In der Regel ist bekannt, mit welchen Vektoren eine Schaltung stimuliert werden soll. Bei der Simulation des Prozessors entstand jedoch ein Problem. Erstens muß der Zufallsgenerator simuliert und zweitens muß für die Entschlüsselung der Daten, der Prozessor mit den richtigen Werten geladen werden, da sonst nicht geprüft werden kann, ob die Chiffrierung korrekt arbeitet. Die Vorausberechnung der verschlüsselten Daten per Hand wäre auf jeden Fall mit hohem zeitlichen Aufwand verbunden gewesen. Deshalb wurde das Softwaremodell dazu benutzt, einen Teil der Stimulivektoren zu erzeugen.
Dazu wird das Modell in die Simulationsdatei eingebunden. Anschließend wird mit einem Compiler, aus dieser Datei, die eigentliche Vektorendatei für den Simulator generiert. Während der Übersetzung der Ausgangsdatei erzeugt das Softwaremodell Werte, die der Compiler in die eigentliche Simulationsdatei für den Simulator einträgt. Mit dieser erzeugten Vektorendatei wird die Hardware vom Simulator stimuliert.
Nach Abschluß der Simulation muß das Simulationsergebnis auf eventuelle Fehler hin überprüft werden. Dazu liest ein Programm die ursprüngliche Ausgangsdatei erneut ein. Dieses Programm ordnet für jeden Simulationsschritt einem Ausgangspin einen bestimmten Erwartungswert zu. Stimmt der Erwartungswert nicht mit dem Simulationsergebnis überein, so liegt ein Fehler vor. Die Überprüfung läuft nach folgendem Schema ab:
Zuerst berechnet das Softwaremodell vor jedem Prüfschritt den entsprechenden Erwartungswert. Dann führt das Prüfprogramm einen Schritt aus. Es vergleicht den Wert aus der Simulation und den berechneten Wert des Softwaremodells. Stimmen

die Werte nicht überein, trägt das Programm den Zeitpunkt des Fehlers in eine Ergebnisdatei ein. Diese Schritte werden solange wiederholt, bis alle Simulationszeitpunkte erfaßt wurden.
Ein Vorteil dieser Prüfmethode ist, daß sowohl die Hardware als auch das Softwaremodell auf Korrektheit überprüft werden können. Der prinzipielle Ablauf der Simulation ist in Abbildung 10 dargestellt.

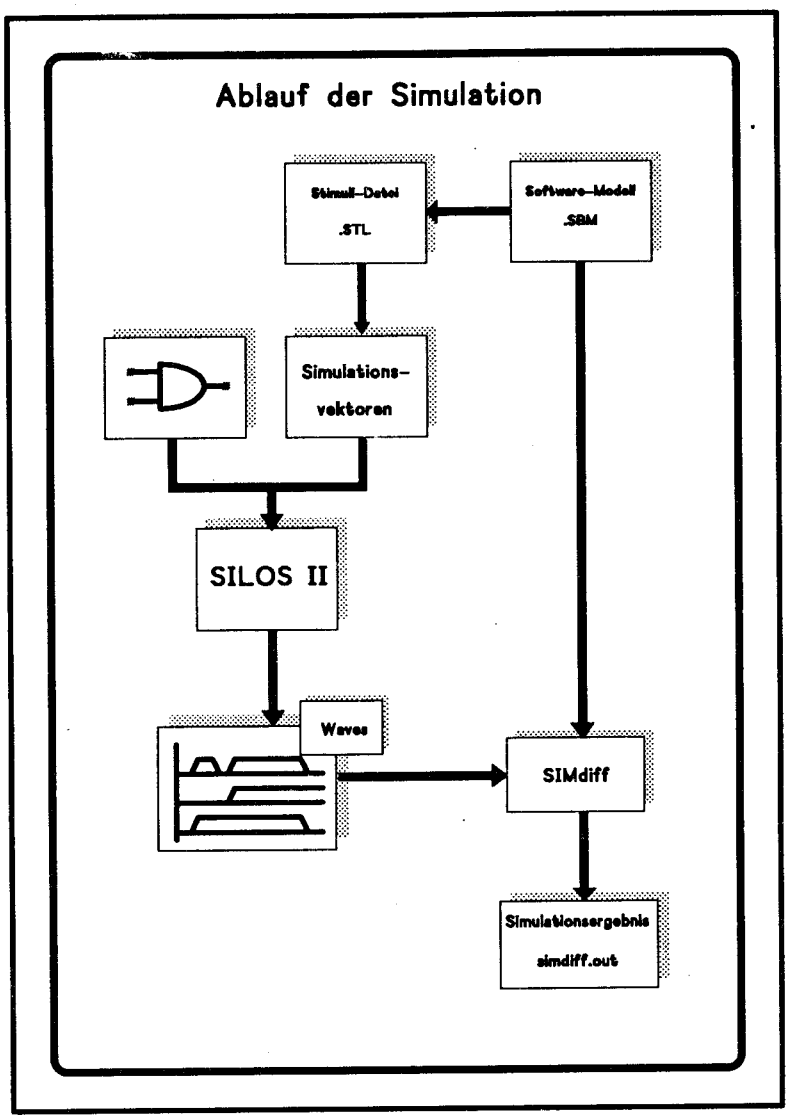


Abb. 10 Ablauf der Simulation

4.5 Zusammenfassung

Das Design wurde bis zur Netzliste fertiggestellt. Die technische Realisierung und die Fehlersimulation mit Erzeugung von Testvektoren stehen noch an.
Augenblicklich leistet der Prozessor eine Datentransferrate von etwa 4MBit/sec. Diese Datenrate wäre für einen Einsatz mit einer seriellen Schnittstelle oder in einem Telfonnetz vollkommen ausreichend. Eine weitere Anwendungsmöglichkeit ist z.B. die Verschlüsselung von Daten auf einer Festplatte oder Diskette, um sie vor unbefugtem Zugriff zu schützen.

Die Datensicherheit ist für den unteren Sicherheitsbereich genügend hoch. Mit einer Schlüssellänge von 64 Bit kann der Benutzer insgesamt 2^{64} verschiedene Schlüssel eingeben. Umgekehrt muß ein Angreifer diese Anzahl von Schlüsseln ausprobieren um eine Nachricht zu dechiffrieren, da bereits ein unterschiedliches Schlüsselbit zu einem falschen Ergebnis führt.
Bei der Datenübertragung dürfen keine Fehler auftreten, da bei verfälschten Datenbits eine Rekonstruktion des Schlüssels, und des chiffrierten Textes, nicht mehr möglich ist. Solche Bitfehler lassen sich mit modernen Übertragungsverfahren jedoch nahezu vollständig vermeiden.

Vortrag zum MPC - Workshop an der FH Reutlingen
am 1.Februar 1993

Entwicklung eines integrierten Temperatur - Spannungsumsetzers mit einstellbarer nichtlinearer Kennlinie in Bipolar - Technik

von

Stefan Christ
Kleingartenstr. 3
6520 Worms 22

Fachhochschule für Technik in Mannheim
Fachbereich Nachrichtentechnik
Institut für den Entwurf integrierter Schaltungen

Betreuer : Prof. Dr. G. Albert

Inhaltsverzeichnis

(Das Kapitel 6 fehlt hier, da diese Kurzzusammenfassung größtenteils aus meiner Diplomarbeit übernommen wurde. Dieses Kapitel ist hier ohne Bedeutung.)

1 Einleitung

1.1 Aufgabenstellung

In dieser Diplomarbeit sollte eine Schaltung entwickelt werden, die eine temperaturabhängige Kennlinie nach der Vorgabe erzeugt, welche in der untenstehenden Abbildung dargestellt ist. Die Kurve ist einer Quarzkennlinie nachgebildet (nähere Hinweise im Kap. 1.2).

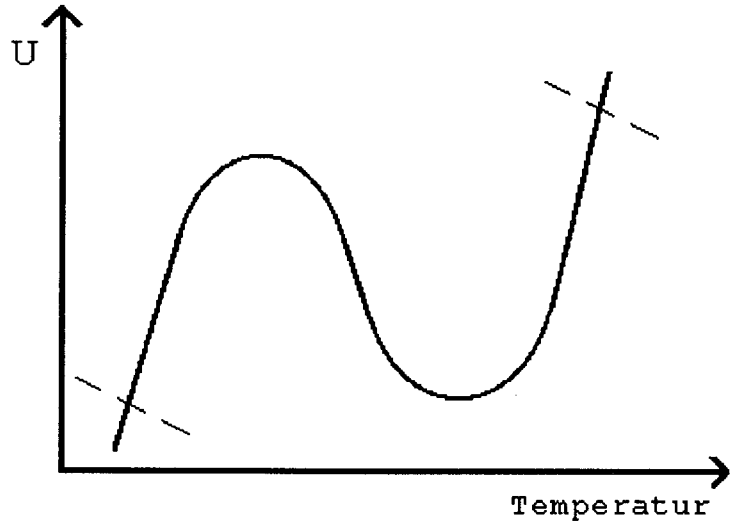


Abb. 1.1 angestrebter Verlauf der Ausgangskennlinie

Hauptanforderungen waren hierbei, daß die beiden Umkehrpunkte der Kurve in der Höhe einstellbar und die ganze Kennlinie zusätzlich über den Temperaturbereich, der von -50°C bis +100°C reicht, verschiebbar sein sollte. Der zweite wichtige Bestandteil der Arbeit war die anschließende Integration der entwickelten Schaltung auf einem analogen Transistor-Array des Typs B500 A das von der Firma AEG hergestellt wird.
Als ein weiterer wichtiger Punkt sind eine möglichst niedrige Betriebsspannung und eine damit verbundene kleine Verlustleistung erstrebenswert.

Die Qualitäten der Schaltung wurden somit zu einem großen Teil von den Eigenschaften der auf dem Chip vorhandenen Widerstände und Transistoren bestimmt, an denen man natürlich nichts mehr ändern kann, da ihr Herstellungsprozeß schon abgeschlossen ist.

5.4

Nach dem Umsetzen der Schaltungsidee und der sich daran anschließenden ausführlichen Schaltungssimulation, mußte das komplette Layout für den Chip entworfen werden. Für eine spätere Herstellung bei einer Halbleiterfirma muß die Leiterbahnebene natürlich vollständig und ohne Fehler vorliegen. Als Softwareunterstützung standen das Simulationsprogramm "Pspice 5.2" für MS-DOS Rechner und ein leistungsfähiger Grafik-Editor für den Leiterbahnentwurf auf einer HP-Apollo Workstation zur Verfügung.

1.2 Spätere Anwendungsmöglichkeiten

Nach einer erfolgreichen Realisierung der Schaltung ergeben sich einige Anwendungsmöglichkeiten. Der in der Aufgabenstellung gezeigte Kurvenverlauf ist im Prinzip aus der Temperaturabhängigkeit einer Quarzkennlinie abgeleitet.
Ein Quarz besitzt seine Nennfrequenz bei einer ganz bestimmten Temperatur. Wird es kälter, so steigt die Frequenz langsam und fällt irgendwann wieder ab. Wenn es wärmer wird , so fällt die Frequenz und steigt dann langsam wieder an. Auf diese Weise entsteht eine Frequenz-Temperaturkennlinie mit zwei Umkehrpunkten (Gleichung 3.Grades = kubische Parabel), ähnlich wie sie in der Einleitung dargestellt ist.
Die Frequenzabhängigkeit von der Temperatur mit dem Quarzschnittwinkel als Parameter soll die folgende Abbildung verdeutlichen.

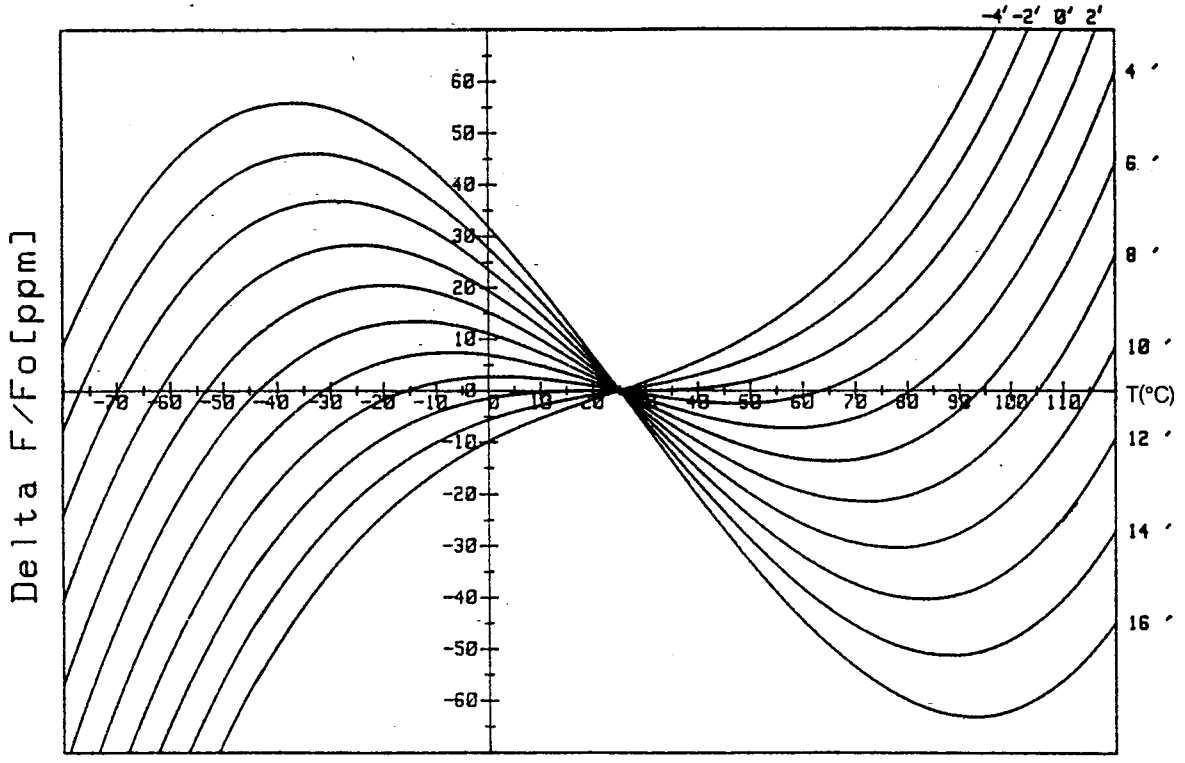


Abb. 1.2 Abhängigkeit der Quarzkennlinie vom Schnittwinkel

Diese Kurvenschar entsteht für einen einzelnen Quarz durch die Variation des Schnittwinkels bei der Herstellung. Der hier verwendete Schnitt wird als AT-Schnitt bezeichnet und hat eine geringere Temperaturabhängigkeit als andere Schnittarten. Die mit der Schaltung erzeugte Kennlinie könnte so einmal dazu dienen, um eine solche Quarzkennlinie zu simulieren und vielleicht die Temperaturabhängigkeit des Quarzes zu kompensieren.

Ein weiteres großes Anwendungsfeld wäre die Erzeugung (in gewissen Grenzen) beliebiger Kurvenverläufe. Dadurch könnten genau definierte temperaturabhängige Spannungen erzeugt werden, die als Eingangssignale für weitere Schaltungen dienen, bei denen die Verwendung anderer, vorhandener Abhängigkeiten wie der Transistorkennlinie oder PTC bzw. NTC Widerstände nicht ausreichen.
Ein dritter Anwendungsfall ist die Kompensierung nichtlinearer Kurven von unterschiedlichsten Bauelementen d.h. eine Zusammenschaltung des jeweiligen Bauelements und der entwickelten Schaltung ergibt eine lineare Abhängigkeit von der Temperatur.

2 Beschreibung des Transistor-Arrays B500 A

2.1 Allgemeines

Von der Firma AEG wurde ein analoges Transistor-Array entworfen um dem Schaltungsentwickler die Möglichkeit zu geben eigene integrierte Schaltungen zu realisieren, ohne jedoch den langen, aufwendigen und sehr teueren Prozeß eines vollständigen Chipentwurfs anwenden zu müssen. Bei dem vorhandenen Array handelt es sich um einen sogenannten Halbkundenschaltkreis, bei dem schon Transistoren, Widerstände und Kondensatoren an genau definierten Plätzen auf dem Wafer vorhanden sind, die vom Entwickler nur noch mit einer letzten Aluminium-Leiterbahnebene verbunden werden müssen.
Als Nachteil muß hier angeführt werden, daß keinerlei Einflußmöglichlkeiten auf Bauteilewerte und Eigenschaften bestehen. Dem gegenüber steht jedoch die Möglichkeit, daß auch kleine Stückzahlen noch einigermaßen wirtschaftlich hergestellt werden können, dies ist besonders bei Prototypen ein nicht zu unterschätzender Vorteil. Außerdem lassen sich so Schaltungen entwerfen, deren Aufbau in diskreter Technik gar nicht möglich wäre, da kein einwandfreier Gleichlauf zwischen zwei Transistoren (Matching) gewährleistet werden kann.
Der verwendete Baustein B500 A besitzt auf einer Chipfläche von 12,9 mm^2 insgesamt 512 Komponenten. Diese Elemente sind in sogenannten Zellen zusammengefaßt, wobei der B500 aus 6 Standardzellen, einer Reglerzelle, einer Bandgabzelle und einer Stromquellenzelle besteht. Die Zellen haben ihren Namen nach einer möglichen Verwendung, sie können jedoch völlig frei und unabhängig benutzt werden. Das vorliegende Array ist vor allem für Anwendungen in der NF-Technik gedacht , die Herstellung erfolgte in einer bipolaren Standardtechnologie.

Ingesamt sind folgende Einzelkomponenten verfügbar:

NPN Transistoren	:	116
PNP Transistoren	:	73
Zenerdioden	:	6
Widerstände	:	305
Kondensatoren	:	12
Anschlußpads	:	42

Da dieses Skript nur als Überblick dienen soll, wird auf eine genaue Beschreibung des B500 an dieser Stelle verzichtet und auf AEG Unterlagen verwiesen.

3 Ausarbeitung eines Grundkonzeptes

3.1 Verschiedene Realisierungsmöglichkeiten

Nach der Erklärung der Aufgabenstellung und der Darstellung der zu Verfügung stehenden Bauelemente folgt jetzt die Umsetzung der Aufgabe in eine reale Schaltung.
Zuerst ist dabei die Frage zu klären, welche grundlegenden Möglichkeiten zu einer erfolgreichen Problemlösung bestehen, daran anschließend ist diejenige auszuwählen, die mit den vorhandenen Mitteln die beste Lösung bietet. Man kann davon ausgehen, daß es zu einer Realisierung sicherlich mehrere Möglichkeiten gibt, die sich hinsichtlich Aufwand und Eigenschaften sehr voneinander unterscheiden.

a) Verwendung eines Mikrocontrollers :

Eine sehr elegante Variante bietet die Benutzung eines Mikrocontrollers, bei dem man die lineare Temperaturabhängigkeit der Differenz zweier Basis-Emitterstrecken als Eingangsgröße nutzt (die Basis-Emitterspannung sinkt um ca. 2 mV pro °C ==> Verwendung der beiden Transistoren als Temperaturfühler, die Differenzbildung eliminiert dabei die Streuung der Durchlaßspannung und des Temperaturkoeffizienten), und sie über einen AD-Wandler direkt dem Prozessor als digitalen Datenfluß zuführt.
Durch entsprechende Programmierung , die jederzeit veränderbar ist, könnte man eine Temperaturkennlinie mit beliebiger Form erzeugen und über einen ausgangsseitigen DA-Wandler als analogen Spannungswert zur Verfügung stellen.
Dieses Verfahren ist sicherlich sehr genau, man hat jedoch einen relativ großen Aufwand, da ein Mikrocontroller und einiges mehr an Zusatzbeschaltung notwendig ist. Außerdem muß der Prozessor noch programmiert werden.
Da diese Möglichkeit aber natürlich auf keinen Fall mit dem B500 zu lösen ist, scheidet sie sofort aus, weil sie an dem Rahmen der Aufgabenstellung vorbeizielt.

b) Verwendung eines Funktionsnetzwerkes

Eine weitere Variante bietet sich an, wenn man sich die Kennlinie im Kap 1.1 nocheinmal genau ansieht. Sie gleicht im Prinzip einer Sinusschwingung und wäre somit vielleicht durch ein Funktionsnetzwerk zu approximieren, wie es zum Beispiel in einfachen Funktionsgeneratoren zu Erzeugung einer Sinusform verwendet wird.

Dieses Funktionsnetzwerk funktioniert nach dem Prinzip der stückweisen Approximation unter Verwendung von Dioden. Mit diesem Verfahren lassen sich recht gute Annährungen an die Sinusform erreichen, wenn man genug Dioden benutzt. Ein Nachteil besteht darin, daß die einzelnen Knickpunkte der Approximationskurve zuerst berechnet werden müssen und daraus die Bauteilewerte ermittelt werden. Ein weiterer Nachteil ist die geringe Flexibilität der Schaltung wenn es darum geht, die Kurvenform zu verändern.
Eine ausgiebige Erklärung dieser Funktionsnetzwerke ist in [2] zu finden, hier soll nicht weiter darauf eingegangen werden. Eine prinzipielle Realisierung wäre auf dem B500 zwar möglich, aber die Eigenschaften reichen für die gestellte Aufgabe nicht aus, da die Kennlinie in ihrer Form weitläufig verstellbar sein soll.

c) Verwendung von Differenzverstärkern

Als dritte Möglichkeit bietet sich die Verwendung von Differenzverstärkern an. Wegen seiner Eigenschaften ist der Differenzverstärker für eine Realisierung in diesem Fall besonders geeignet und er wurde aus diesem Grund in der entwickelten Schaltung auch wirklich benutzt.

Dafür sprechen im wesentlichen fünf Gesichtspunkte:

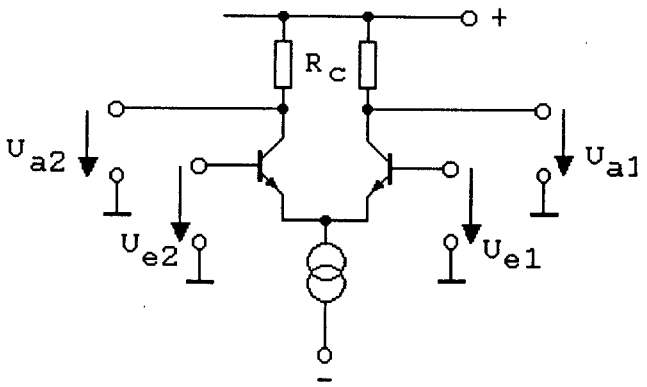
- Ein idealer Differenzverstärker ist von der Theorie völlig unabhängig gegenüber Temperatureinflüssen, die auf ihn einwirken.
- Die Großsignalübertragungskennlinie hat für den vorgesehenen Zweck einen idealen Verlauf.
- Ein Differenzverstärker ist mit den vorhandenen Transistoren in integrierter Technik sehr gut zu realisieren.
- Die Großsignalkennlinie ist mit zusätzlichen Emitterwiderständen in ihrer Steigung in weiten Grenzen einstellbar.
- Mittels Differenzverstärker ist eine sehr gute Gleichspannungkopplung von verschiedenen Stufen möglich.

Eine Schaltung mit Differenzverstärkern und dazugehörigen Baugruppen wie Stromquellen ist auf dem B500 auf jeden Fall zu realisieren.
Ausgehend von diesen Punkten, soll in den folgenden Kapiteln der Weg von der Grundidee bis zu einem vollständigen Blockschaltbild erläutert werden.

3.2 Grundlagen des Differenzverstärkers

In diesem Kapitel soll nur auf den grundsätzlichen Aufbau und die Eigenschaften des Differenzverstärkers eingegangen werden, dies erleichtert das Verstehen der folgenden Zusammenhänge. Eine ausführliche Berechnung wichtiger Merkmale wie Steilheit, Eingangwiderstand, Ausgangswiderstand oder Differenzverstärkung erfolgt zu einem späteren Zeitpunkt im Kapitel 6.

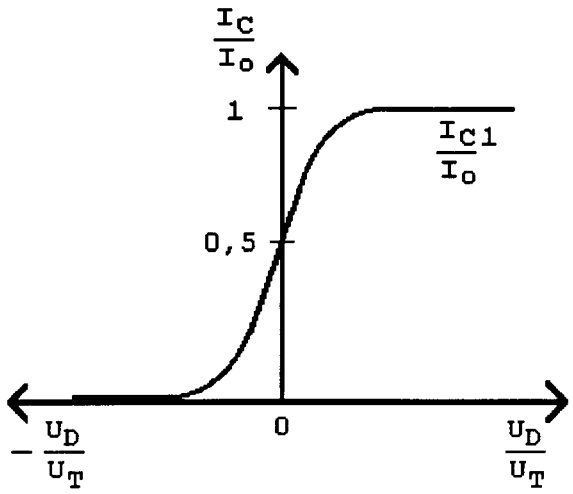
Die Grundschaltung eines Differenzverstärkers ist in der folgenden Abbildung dargestellt.



Ua1 : Ausgangsspannung 1
Ua2 : Ausgangsspannung 2
Ue1 : Eingangsspannung 1
Ue2 : Eingangsspannung 2

Abb. 3.1 einfacher Differenzverstärker

Es handelt sich um einen symmetrischen Gleichspannungsverstärker mit zwei Eingängen und zwei Ausgängen. Wichtige Merkmale sind der gemeinsame Emitteranschluß und die weiterführende Stromquelle. Bei einer Differenzeingangsspannung von 0V teilt sich dieser Strom genau zur Hälfte auf die beiden Transistoren auf, er verursacht so an den Kollektorwiderständen einen Spannungsabfall, d.h. es entsteht ein Kollektorruhepotential. Die gezeigte Schaltung besitzt keinerlei Stromgegenkopplung, da keine zusätzlichen Emitterwiderstände vorhanden sind.
Ein Differenzverstärker besitzt eine ganz charakteristische Großsignalübertragungskennlinie die auf der folgenden Seite gezeigt wird.



I_C = Kollektorstrom
U_D = Differenzspannung
U_T = Temperaturspannung
I_o = Strom der Stromquelle

Abb. 3.2 Großsignalübertragungskennlinie

Ein Differenzverstärker mit Stromgegenkopplung und eine besondere Schaltungsvariante sind in Abb. 3.3 gezeigt. Die paarweise vorhandenen Emitterwiderstände können zu einem gemeinsamen zusammengefaßt werden, der dann den doppelten Wert hat, dabei ist es zusätzlich notwendig die ursprünglich eine Stromquelle in zwei aufzuteilen, welche jetzt nur noch den halben Strom liefern müssen. Diese Schaltungsart hat den Vorteil, daß mit Verändern des Emitterwiderstands die Verstärkung geändert werden kann, ohne die Kollektorruhepotentiale zu beeinflussen.

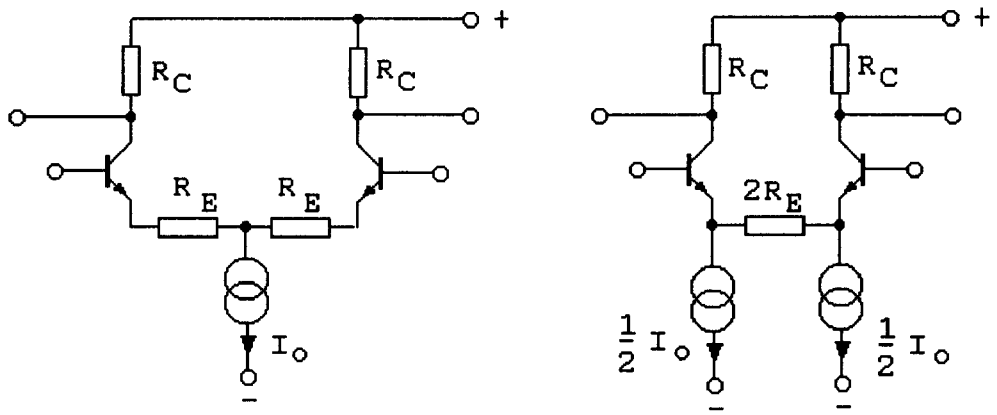


Abb 3.3 Differenzverstärker mit Stromgegenkopplung (links) und Schaltungsvariante (rechts)

3.3 Prinzipielle Erzeugung der geforderten Kennlinie

Um die Kennlinie zu erzeugen, werden mehrere gleichartige Differenzverstärker verwendet. Wenn ihre Großsignalkennlinien entsprechend benutzt und die Verstärker richtig zusammengeschaltet werden, dann ist es möglich eine gute Annährung an die Aufgabenstellung zu erreichen. In Abb. 3.4 ist dieser wichtige Zusammenhang zu erkennen.

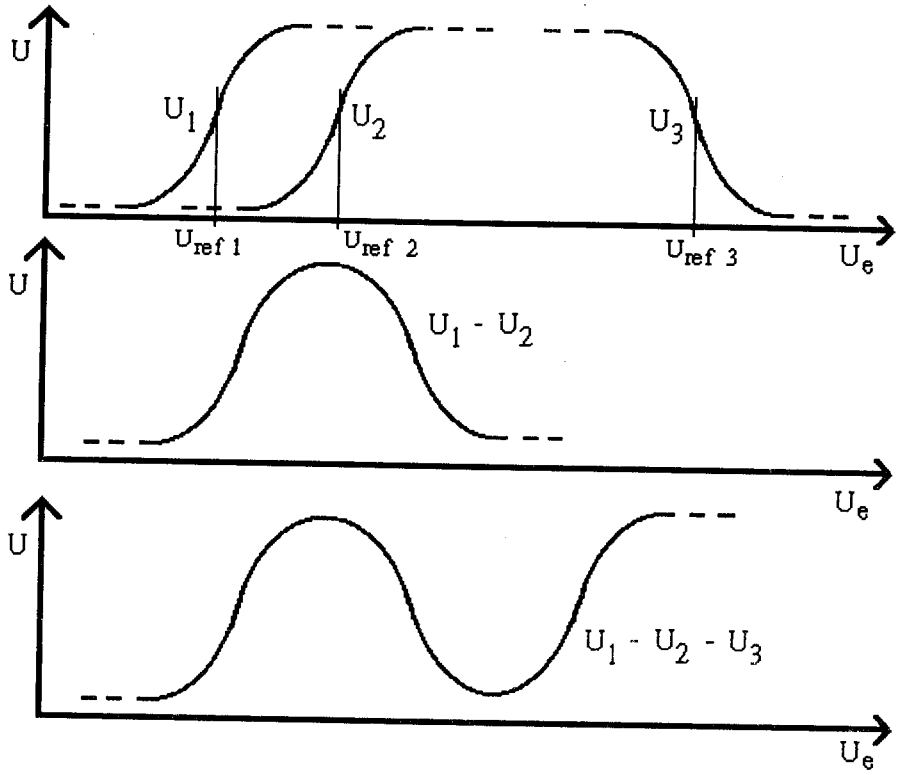


Abb 3.4 Zusammenwirken der einzelnen Differenzverstärkerkennlinien

Im oberen Koordinatensystem sind drei Kennlinien eingetragen. Man erkennt, daß die Lage der jeweiligen Kennlinie durch die Referenzspannung auf der Abszisse bestimmt wird. Diese Referenzspannung läßt sich beliebig nach links oder rechts verschieben, dies bedeutet, daß sich die Kennlinie in die gleiche Richtung bewegt. Die Amplitude ist zunächst nebensächlich, da es hier nur um das Prinzip geht (wird später ausführlich erläutert).
Wenn man nun drei Differenzverstärker hat und die drei Referenzspannungen entsprechend weit auseinander wählt, dann erhält man ein Bild wie es im oberen Koordinatensystem gezeigt ist. Es ist wichtig zu beachten, daß die drei Kurven untereinander völlig unabhängig sind d.h. keine wird von der Einstellung einer anderen beeinflußt.

5.12

Wird nun ein vierter Differenzverstärker verwendet, mit dem die Spannung U_2 von der Spannung U_1 subtrahiert wird, so entsteht U_1 - U_2 und somit die Kurve auf dem mittleren Koordinatensystem. Wird davon mit Hilfe eines fünften Differenzverstärkers die Spannung U_3 subtrahiert, so entsteht U_1 - U_2 - U_3 und somit die Ausgangsspannung auf dem unteren Koordinatensystem. Diese Spannung hat schon eine große Ähnlichkeit mit der Aufgabenstellung, am oberen und unteren Rand wird sie natürlich durch die Betriebsspannungsbedingungen begrenzt.
Diese relativ einfache Verschaltung von fünf Differenzverstärkern stellt das eigentliche Herz der Schaltung dar. Zum Betrieb sind neben der Versorgungsspannung nur eine linear steigende Eingangsspannung Ue (abhängig von der Temperatur), sowie 3 einstellbare Referenzspannungen notwendig.
Für eine voll funktionsfähige Schaltung sind natürlich noch einige andere Schaltungsteile von Bedeutung, deren Funktion im nächsten Kapitel erläutert wird.

3.4 Erstellung eines Blockschaltbildes

Um einen Überblick über die Gesamtschaltung zu erhalten, ist zweckmäßig zunächst ein Blockschaltbild zu entwerfen. Deshalb ist es notwendig alle benötigten Baugruppen am Anfang zu definieren.

Bestandteile:

- Hauptschaltung (bestehend aus fünf Differenzverstärkern)
- Temperatur-Spannungsumsetzer mit linearer Kennlinie
- Schaltung zur Einstellung der Referenzspannungen
- Erzeugung der Betriebsspannung mit zwei Spannungsreglern
- Referenzspannungsquelle
- eventuell eine Ausgangsstufe

Die Tatsache das zwei Spannungsregler gebraucht werden, sei hier einmal vorweggenommen, der Grund wird später ersichtlich sein.

Aus den oben genannten Bestandteilen läßt sich ein Blockschaltbild erarbeiten, daß auf der folgenden Seite als Abb. 3.5 gezeigt wird.

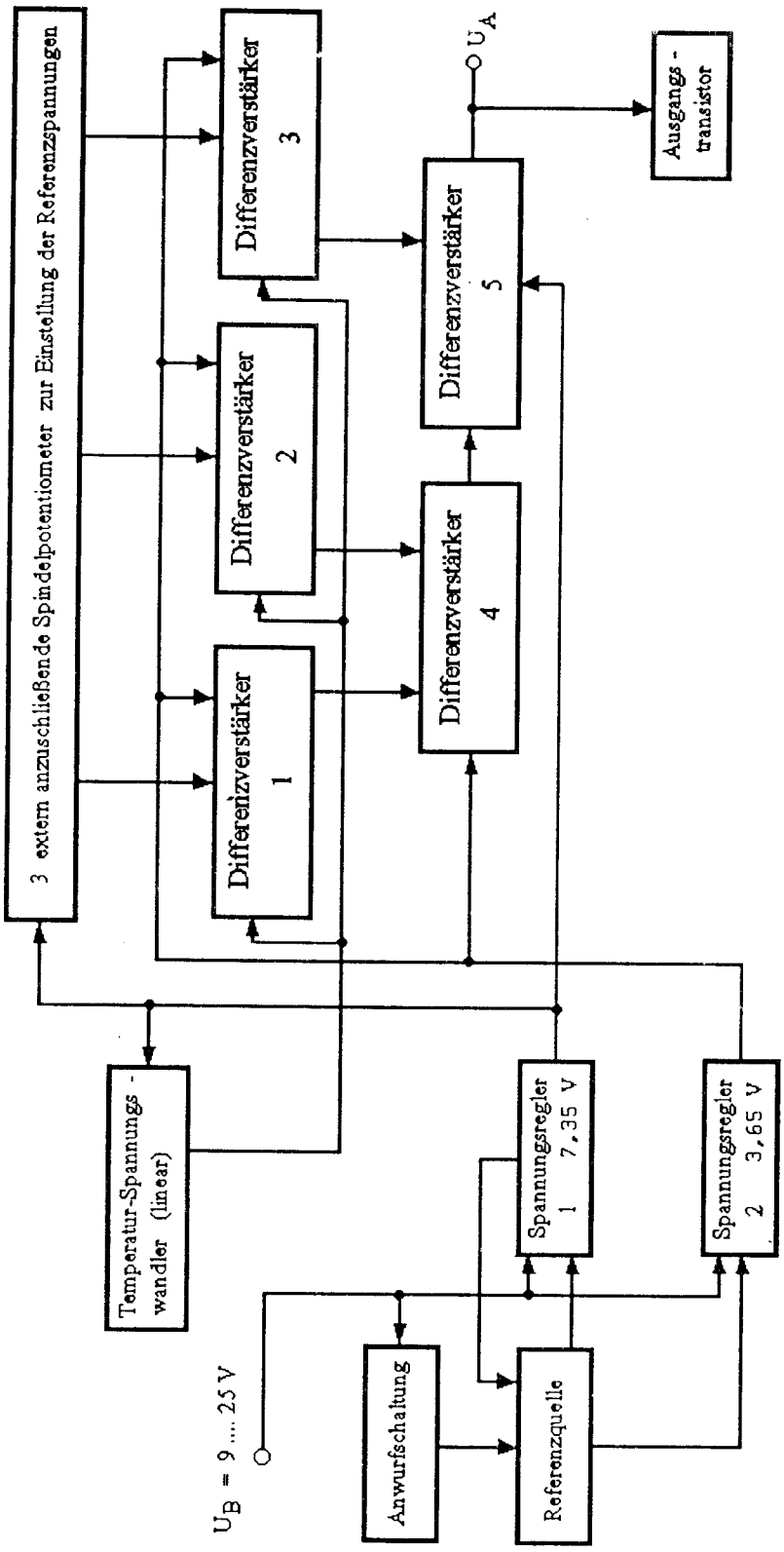


Abb. 3.5 Blockschaltbild des Temperatur-Spannungs-Umsetzers

5.14

4 Die Bestandteile des Blockschaltbildes

4.1 Der Spannungsregler

Zum Betrieb der Hauptschaltung ist natürlich eine stabile Gleichspannung nötig. Da es unsinnig ist eine integrierte Schaltung zu entwerfen, die dann nur mit einer einzigen exakten Spannung zu betreiben ist, muß eine interne Spannungsregelung vorgesehen werden, um die Schaltung aus einer ungeregelten Betriebsspannung zu versorgen.
Als Versorgungsspannung sind ca. 8,5 V bis zu 25 V möglich, es ist jedoch im Hinblick auf eine kleine Verlustleistung günstiger, sich an der unteren Grenze zu bewegen. Die hier vorliegende Schaltung braucht zwei Betriebsspannungen, nämlich einmal ca. 7,4 V und dann ca. 3,6 V (Eine Erklärung dafür folgt später). Diese Werte werden mit internen Spannungsreglern erzeugt, deren Funktion nachstehend beschrieben wird.

Zu Anfang wird die grundsätzliche Wirkungsweise beschrieben:

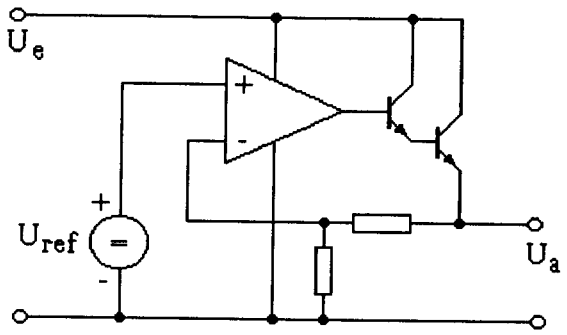


Abb. 4.1 Prinzipschaltung eines Spannungsreglers

Die Funktionsweise eines Spannungsreglers läßt sich aus dem Prinzipschaltbild sehr gut erkennen. Ein Operationsverstärker vergleicht an seinen Eingängen eine Referenzspannung mit einer heruntergeteilten Ausgangsspannung. Anhand des Ergebnisses dieses Vergleichs wird ein Längstransistor geregelt, über dessen Kollektor - Emitterstrecke ein Spannungsabfall entsteht. Durch diesen Regelvorgang wird die Ausgangsspannung auf einem konstanten Wert gehalten. Auf die Erzeugung der wichtigen Referenzspannung wird im Kapitel 4.2 eingegangen.
Der im oberen Bild gezeigte Operationsverstärker wird auf die einfachste denkbare Form reduziert, dies ist ein Differenzverstärker mit nachfolgendem Ausgangstransistor.

Der einfache Differenzverstärker wird mit Kollektorwiderständen betrieben, welche im wesentlichen die Differenzverstärkung bestimmen. Diese ist jedoch nicht groß genug um einen Operationsverstärker zu bauen, als Abhilfe werden darum die Kollektorwiderstände durch einen PNP Stromspiegel ersetzt, der als aktive Last wirkt. Der Strom durch die beiden Eingangstransistoren wird von einer NPN Stromquelle geliefert.

Damit kommt man zu folgendem Schaltbild :

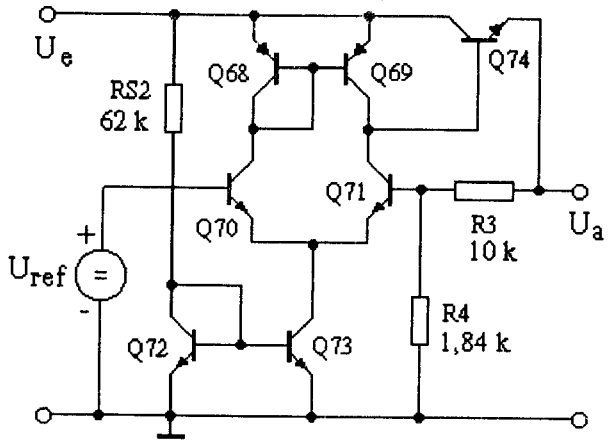


Abb. 4.2 Schaltbild des verwendeten Spannungsreglers

Für die Verstärkung ergeben sich zwei verschiedene Werte, einer für die Benutzung von Kollektorwiderständen, der andere bei Verwendung eines Stromspiegels.

Voraussetzungen : - Der Kollektorwiderstand betrage 10 kOhm
- Der mittlere Strom aus der Stromquelle Q_{72} und Q_{73} betrage

$$I = \frac{U_B}{R_{S2}} \qquad I = \frac{12V}{62000\Omega} \qquad I = 195\,\mu A \tag{4.1}$$

wegen der symmetrischen Stromaufteilung bleiben für jeden Transistor I = 97 μA, daraus ergibt sich die Verstärkung V zu

$$V = \frac{1}{2}\frac{I_C}{U_T} R_C \qquad V = 0,5\,\frac{97\mu A}{0,026mV}\,10k\Omega \qquad V = 18,65 \tag{4.2}$$

I_C = Kollektorstrom
R_C = Kollektorwiderstand
U_T = Temperaturspannung

Eine solch kleine Verstärkung reicht auf keinen Fall aus um die Schaltung als Operationsverstärker zu betreiben, wird jedoch ein Stromspiegels als aktive Last verwendet, so steigt die Verstärkung wegen des größeren Kollektorwiderstandes beträchtlich.

Der Stromspiegel wird hier ohne zusätzliche Emitterwiderstände betrieben, deshalb tritt als Ausgangswiderstand nur r_{CE} zu Tage:

$$r_{CE} = \frac{U_Y}{I_C} \qquad r_{CE} = \frac{137\,V}{97\,\mu A} \qquad r_{CE} = 1{,}412\,M\Omega \qquad (4.3)$$

U_Y = Earlyspannung

Dieser wirksame Kollektorwiderstand ist ungefähr um den Faktor 140 höher als beim vorangegangenen Fall.
Mit Gleichung 4.2 ergibt sich nun eine Verstärkung von 2634, dies entspricht 68,4 dB.

Mit einer Verstärkung in dieser Größenordnung ist ein Operationsverstärker für die Zwecke als Spannungsregler gut geeignet. Am Ausgang des Differenzverstärkers wird normalerweise ein Darlingtontransistor angeschlossen um die Verstärkung noch weiter zu erhöhen und um den Laststrom sicher verkraften zu können. Diese Maßnahme kann hier wegfallen, da nur sehr geringe Lastströme auftreten, man kommt also mit einem normalen Transistor aus. Auf den zweiten Eingang wird über einen Spannungsteiler ein Teil der Ausgangsspannung zurückgekoppelt, der mit der Referenzspannung verglichen wird.

Die Ausgangsspannung ergibt sich so zu:

$$U_A = \left(1 + \frac{R_3}{R_4}\right) U_{ref} \qquad (4.4)$$

U_{ref} = Referenzspannung
R3 , R4 = Spannungsteiler

Der erste Spannungsregler wurde mit seinem Widerstandsteilerverhältnis so eingestellt, daß seine Ausgangsspannung 7,36 V beträgt. Der zweite Regler ist absolut identisch mit dem ersten aufgebaut, lediglich sein Teilerverhältnis hat einen anderen Wert. Daraus ergibt sich für die zweite Ausgangsspannung ein Wert von 3,65 V.

4.2 Die Referenzspannungsquelle

Um einen Spannungsregler aufbauen zu können, braucht man eine Referenzspannungsquelle, die über einen möglichst weiten Temperaturbereich keine Spannungsänderung aufweist. Dies geht natürlich nur mit einer idealen Quelle, aber man kann diesem Ergebnis doch sehr nahe kommen. Das eigentliche Problem liegt in der Basis - Emitterspannung eines Transistors, die bekanntlich mit ca. 2 mV /°C driftet. Wenn man dazu eine Spannung in Reihe schaltet, die mit 2 mV /°C in die andere Richtung driftet, so wird der Temperaturgang kompensiert und eine stabile Spannung wird erzeugt.
Alle Schaltungen die nach diesem Prinzip funktionieren nennt man Bandgab-Referenzen, da sie die Bandgabspannung von Silizium erzeugen. Auch die folgende Schaltung gehört zu dieser Gruppe.

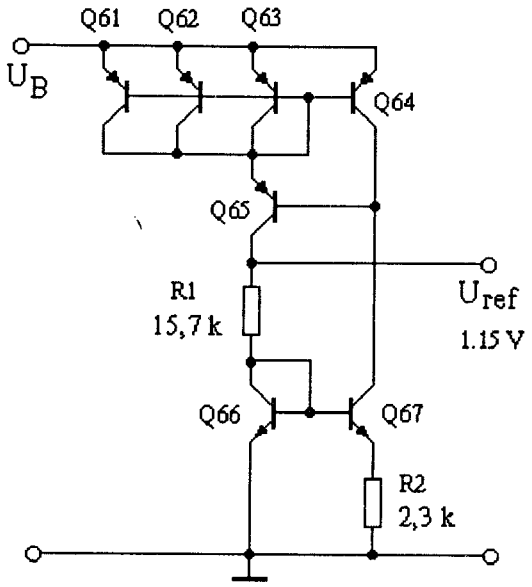


Abb. 4.3 Bandgabreferenz

Da mit Q_{61} , Q_{62} , Q_{63} drei Transistoren parallel geschaltet sind (n = Emitterverhältnis), fließt im linken Zweig der dreifache Strom wie im rechten. Die Spannung U (T) = U_{Ref} läßt sich wie folgt berechnen.

$$\text{Mit} \qquad I_1 = n\,I_2 \qquad I_1 = I_S \exp\left(\frac{U_{BE66}}{U_T}\right) \qquad I_2 = I_S \exp\left(\frac{U_{BE67}}{U_T}\right) \qquad (4.5)$$

linker Zweig rechter Zweig

folgt unmittelbar

$$\frac{I_1}{I_2} = \exp\left(\frac{U_{BE66} - U_{BE67}}{U_t}\right) \quad \text{und somit} \quad \Delta U_{BE} = U_{BE66} - U_{BE67} = U_T \ln n \qquad (4.6)$$

I_1 , I_2 : Kollektorströme
U_{BE} : Basis - Emitterspannung
I_S : Sperrsättigungsstrom

Mit Gleichung 4.6 (sie stellt den Spannungsabfall über R_2 dar) läßt sich der Strom durch diesen Widerstand berechnen. Da im linken Zweig der Strom dreimal so hoch ist wie im rechten, kann man durch eine Multiplikation mit n und R_1 die Spannung über R_1 herleiten.

Als Ausgangsspannung erhält man : $$U(T) = U_{BE66} + \frac{nR_1}{R_2} U_T \ln n \qquad (4.7)$$

Die Basis-Emitterspannung läßt sich ohne weiters nicht genau angeben, da sie ebenfalls temperaturabhängig ist, sie läßt sich aber über eine etwas längere Herleitung relativ genau berechnen, wenn einige Faktoren bekannt sind.

Als Ausgangspunkt dienen die beiden folgenden Beziehungen

$$1.) \quad I_C(T) = I_S(T) \exp\left(\frac{e\,U_{BE}}{kT} - 1\right) \qquad 2.) \quad I_S(T) = C\,T^m \exp\left(\frac{-e\,U_{Go}}{kT}\right) \qquad (4.8)$$

Eine Division von I_C (T) durch I_C (To) (wobei To ist eine frei wählbare Temperatur ist) führt zu folgender Gleichung

$$\frac{I_C\,(T)}{I_C\,(To)} = \frac{T^m \exp\left(\frac{-e\,U_{Go}}{kT}\right) \exp\left(\left(\frac{e\,U_{BE}}{kT}\right) - 1\right)}{To^m \exp\left(\frac{-e\,U_{Go}}{kTo}\right) \exp\left(\left(\frac{e\,U_{BE}}{kTo}\right) - 1\right)} \qquad (4.9)$$

Eine Vernachlässigung der 1 in Zähler und Nenner sowie einige weitere Umformungen führen schließlich zur entgültigen Gleichung für die temperaturabhängige Spannung U_{BE} .

$$U_{BE}(T) = U_{Go}\left(1 - \frac{T}{To}\right) + U_{BEO}\frac{T}{To} + m\frac{kT}{e}\ln\frac{To}{T} + \frac{kT}{e}\ln\frac{I_C}{I_{Co}} \qquad (4.10)$$

Es bedeuten :

U_{GO}	Bandgabspannung für Silizium	ca. 1,2 V
e	Elektronenladung	$1{,}602 \cdot 10^{-19}$ As
m	herstellungsbedingte Konstante	1...2
k	Boltzmannkonstante	$1{,}38 \cdot 10^{-23}$ J/K
T	absolute Temperatur in Kelvin	
U_{BEO}	Basis-Emitterspannung bei einer Bezugstemperatur To	
I_S	Sperrsättigungsstrom	
I_C	Kollektorstrom bei Temperatur T	
I_{CO}	Kollektorstrom bei Temperatur To	
C	Konstante	

Zu einer wirklich genauen Berechnung von U_{BE} muß jedoch diese Spannung bei einer bestimmten Bezugstemperatur bekannt sein. Dies ist ein etwas heikeler Punkt, da diese Information nur aus der Spicesimulation erhältlich ist und man sich darauf verlassen muß. Es wäre schön, wenn dafür ein genaues Meßergebnis vorläge. Ein weitere Ungenauigkeit liegt in der Konstanten m. Sie ist herstellungsabhängig und hat in der Regel einen Wert zwischen 1 und 2 ==> es wird dafür 1,4 angenommen
Diese Gleichung wird zu einem späteren Zeitpunkt noch einmal benötigt (im Kapitel 4.4 , wo die Erzeugung einer linearen Eingangsspannung beschrieben wird).

Zusammenfassendes Ergebnis :

In Gleichung (4.7) spiegelt sich das Ergebnis wieder. Die Ausgangsspannung setzt sich aus zwei Anteilen zusammen. Der Term U_{BE} liefert eine linear mit der Temperatur fallende Spannung, aber der zweite Term steigt dafür mit der Temperatur ==> die beiden Teile addieren sich und heben sich in Ihrer Wirkung gegenseitig auf, es entsteht eine Gerade mit der Steigung 0, die für diesen Anwendungsfall sehr wichtig ist.
Die Schaltung liefert eine Ausgangsspannung von 1,15 V und hat eine maximale Abweichung von 1 mV d.h. sie hat über den gesamten Temperaturbereich von -50°C bis +100°C eine Genauigkeit von < 0,1% bzw. 0,00066 % /°C. Dies stellt einen sehr guten Wert dar.

5.20

Wenn man eine ideale Schaltung voraussetzt, dann wäre diese zusätzlich völlig unabhängig von der Betriebsspannung. Dies ist in der Realität nicht der Fall, da die PNP-Transistoren eine Earlyspannung aufweisen, die keinesweg unendlich groß ist (horizontaler Verlauf der Kurven im Ausgangskennlinienfeld des Transistors). Um diesen ungünstigen Einfluß weitgehend zu verhindern, ist der Transistor Q_{65} eingebaut, damit ist U_{CE61} ungefähr gleich U_{BE} und der Early-Effekt wird weitgehend vermieden.

Da insgesamt zwei Spannungsregler benutzt werden, wird die Referenzspannung gleichzeitig beiden Reglern auf den jeweiligen Eingang des Differenzverstärkers gegeben. Durch die relativ hochohmigen Eingänge, stellen sie keine große Belastung dar und beeinflussen die Referenzspannung nicht.

4.3 Die Anwurfschaltung

Um die Einflüsse von Betriebsspannungsschwankungen möglichst weit zu unterdrücken, wird die Referenzspannungsquelle aus der geregelten Eingangsspannung versorgt. Dazu muß aber die Ausgangsspannung des Reglers einen einigermaßen stabilen Wert erreicht haben und genau das ist in der Einschaltphase nicht gewährleistet. Es wird deshalb eine besondere Schaltung angewandt um diesen Betriebszustand zu umgehen.

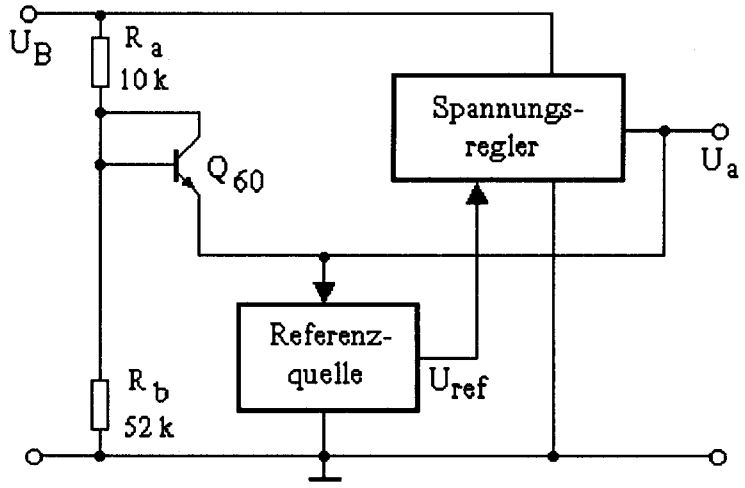


Abb 4.4 Anwurfschaltung für die Spannungsreferenz

Über den Spannungsteiler Ra , Rb und den als Diode geschalteten Transistor Q_{60} wird die Referenzquelle nach dem Einschalten mit Spannung versorgt. Da der Ausgang des Reglers voll auf den Emitter dieses Transistors zurückgekoppelt ist, beginnt Q_{60} zu sperren sobald die Ausgangsspannung die Größe des Basispotentials erreicht hat und trennt die Anwurfschaltung ab. Nach diesem Zeitpunkt erfolgt die Versorgung aus dem Regelausgang. Da eine vollständige Rückkopplung vorliegt und die Transistorkapazitäten voll zum Tragen kommen, besteht grundsätzlich eine Schwingneigung, dies muß bei der Simulation unbedingt überprüft werden, eine Abhilfe wäre das zusätzliche Anbringen eines kleinen Kompensationskondensators.

4.4 Eigentlicher Temperatur - Spannungsumsetzer

4.4.1 Allgemeines

Die Aufgabe dieser Schaltung ist es, aus dem Temperaturbereich von -50°C bis +100°C eine linear steigende Ausgangsspannung zu erzeugen. Diese muß aber zu der nachfolgenden Hauptschaltung "passen" d.h. sie muß einen richtigen Startwert und einen ausreichenden Spannungshub besitzen. Die Vorstellungen lagen bei etwa 1V Startwert und bei einem Spannungshub von ca. 2V.
Für diesen Zweck könnte man eigentlich die Kennlinie eines PNP - Transistors, der als Diode geschaltet ist, benutzen. Es müßte nur ein ausreichend großer Widerstand in Reihe geschaltet werden, um ihren Verlauf zu linearisieren. Hier treten jedoch einige Probleme auf, da man aus Erfahrungswerten und mit Auswertung der Gleichung 4.10 weiß, in welchen Bereichen sich diese Spannung bewegt. Man erkennt, daß selbst bei einem Temperaturunterschied von 150 °C die maximale Differenz der Basis-Emitterspannung höchstens ca. 300 mV beträgt.
Der Spannungshub ist also viel zu klein, es muß nach einer Schaltung gesucht werden, die diesen Hub vergrößert und gleichzeitig einen richtigen Anfangswert in der Gegend von 1V bereitstellt.

4.4.2 Erste Schaltungsvariante

Als erste Variante wurde eine Schaltung simuliert, die nach dem oben erwähnten Prinzip arbeitet. Dazu sind im wesentlichen nur Transistoren notwendig, deren Temperaturabhängigkeit voll ausgenutzt wird.

5.22

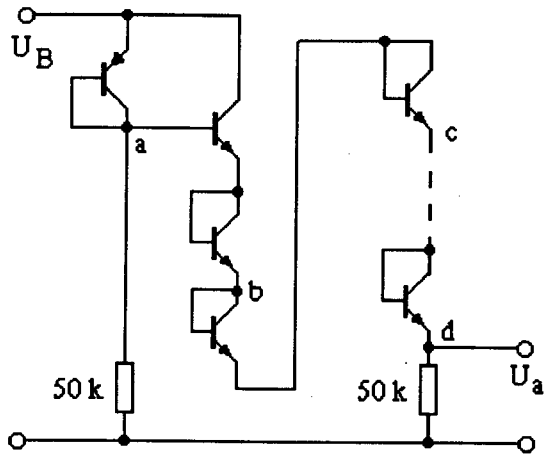


Abb. 4.5 Schaltbild der ersten Schaltungsvariante

Über dem linken Widerstand ist eine Spannung meßbar, die mit der Temperatur steigt, sie liegt jedoch im Mittel nur ca. 700 mV unter dem Betriebsspannungspotential. Das hat zur Folge das die Spannung auf den richtigen Wert umgesetzt werden muß. Dies geschieht mit dem Emitterfolger und den dazu in Reihe geschalteten "Dioden". Der Emitterfolger und jede Diode bewirken eine Spannungsverschiebung von, im Mittel, 0,7 V. Da die Basis-Emitterspannung bei -50°C größer ist als bei +100°C (ergibt sich aus Gleichung 4.10 und aus der Erfahrung), wird mit jeder Stufe, die Steigung des Spannungverlaufs sowie der absolute Hub erhöht, wie auf der folgenden Abbildung verdeutlicht werden soll.

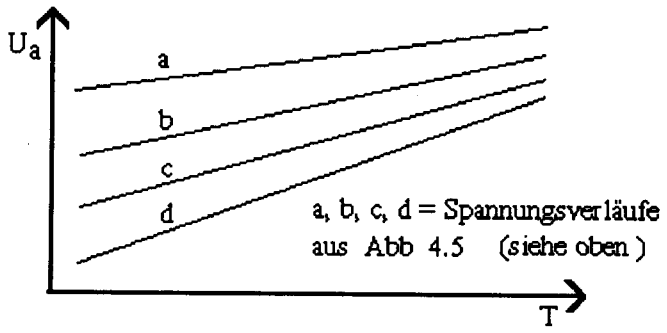


Abb. 4.6 Erzeugung einer steigenden Ausgangskennlinie (Prinzip)

Mit dieser Methode ist das gewünschte Ergebnis zu erzielen, man muß nur die Zahl der Dioden den Anforderungen anpassen. Das Verfahren hat nur einen wichtigen Nachteil, es funktioniert mit einer Spannungsverschiebung. Diese sollte man in einer integrierten Schaltung aber möglichst vermeiden, da eine absolute Toleranz von U_{BE} eingeht. Mit Pspice läßt sich zwar diese Spannung in Abhängigkeit von der Temperatur berechnen, aber die Simulation kann keine fertigungsbestimmten Toleranzen von U_{BE} einrechnen. Diese liegen im Höchstfall bei ca. 5-10% der Basis-Emitterspannung und werden unter anderem vom Faktor m mitbestimmt (vgl. Gleichung 4.10).
Wenn nun wie im oberen Fall quasi bis zu 8 oder 10 Dioden in Reihe geschaltet sind, kann sich die Ausgangsspannung relativ weit von den Simulationsergebnissen wegbewegen, ohne das man dieses vorhersehen könnte. (Wie man später sieht, tritt diese Problematik bei den anderen Schaltungsteilen nicht auf).
Da diese Schaltung mit einigen Schönheitsfehlern behaftet war, mußte nach einer anderen Lösung gesucht werden.

4.4.3 Verwendeter Umsetzer

Als Alternative bietet sich eine Schaltungsart an, die schon einmal benutzt wurde. Es handelt sich um die Referenzspannungsquelle aus Kapitel 4.2. Für den vorliegenden Fall wird diese Schaltung jedoch zweckentfremdet.
Aus Gleichung 4.7 ist die Ausgangsspannung bekannt, man erkennt , daß sie sich aus zwei Anteilen zusammensetzt. Wenn der zweite Term, der eine positive Steigung aufweist, sehr stark überwiegt, so liefert die Schaltung die gewünschte Abhängigkeit von der Temperatur. Dazu reicht eine starke Vergrößerung des Widerstandes R_1 völlig aus, da dieser im Zähler des Bruches in Gleichung 4.7 steht. An R_1 ergibt sich infolgedessen ein Spannungsabfall, der wesentlich größer ist als U_{BE}.
Die entstehende Ausgangsspannung hat damit schon einen Hub, der relativ groß ist, sie muß nur noch mit einem Emitterfolger und zwei "Dioden" in die richtige Lage gebracht werden. Natürlich tritt hier genau das gleiche Problem auf wie in Kapitel 4.4.1, zu dessen Lösung wird ein Stromspiegel verwendet, der aus den Transistoren Q_{99} und Q_{100} besteht. Er bildet zusammen mit dem Emitterfolger und den beiden Dioden einen Spannungsteiler, so daß eine etwaige Änderung von U_{BE} sich nicht auswirkt, da sie gleichmäßig an allen Transistoren auftritt und so nach außen keine Wirkung zeigt.

Man gelangt somit zu der Schaltung auf der nächsten Seite.

5,24

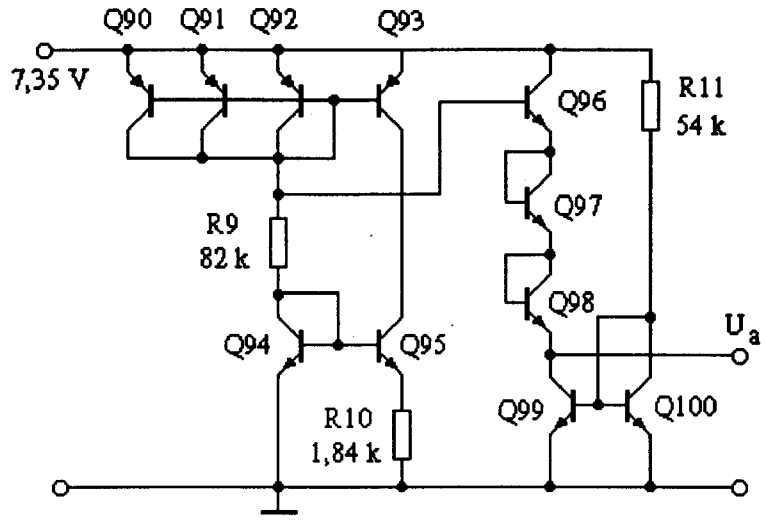


Abb. 4.7 Schaltbild des Temperatur-Spannungsumsetzers

Die Schaltung erzeugt eine Spannung die sich im Bereich von 0,91 V bis 3,1 V bewegt und dabei absolut linear verläuft. Eine Darstellung des Ergebnisses und einer Berechnung der Spannung erfolgt in einem späteren Kapitel. Die Ausgangsspannung kann nach der Funktion $Ue = 0{,}0146 \cdot T + 1{,}65\ V$ berechnet werden (Geradengleichung).

4.5 Erzeugung der drei einstellbaren Referenzspannungen

Bei diesem Schaltungsteil handelt es sich nur um drei Spindelpotentiometer (50 kOhm), die am Ausgang des ersten Spannungsreglers angeschlossen werden. Die Abgriffe sind direkt mit den Eingängen der drei folgenden Differenzverstärker verbunden. Trotz eines höheren Preises sind Spindeltrimmer normalen Potentiometern vorzuziehen, da sie viel genauer eingestellt werden können. In einer weiteren Entwicklungsphase, wäre es eine Überlegung wert, ob die Potis nicht durch eine eigene Schaltung zu ersetzen sind, bei der nur noch ein einzelner Widerstand zum Einsatz kommt. Allerdings muß die leichte, universelle Einstellbarkeit der Referenzspannungen dabei unbedingt erhalten bleiben, sodaß dieser Widerstand nicht mitintegriert werden darf, da sonst nur noch eine einzige festgelegte Kennlinie entstehen würde. Somit wäre die ganze Schaltung nur noch für genau einen einzigen Anwendungsfall verwendbar. Aus diesem Grund ist es nicht so einfach die Potentiometer zu ersetzen, an irgendeiner Stelle muß immer "gedreht" werden um die Einstellung vorzunehmen, sodaß die Potentiometer vielleicht doch die beste und zugleich einfachste Lösung darstellen.

4.6 Die Schaltung zur Kurvenerzeugung

Wie schon erläutert wurde, dient dieser wichtigste Teil der gesamten Schaltung dazu, die einstellbare und nichtlineare Kennlinie zu bilden. Zum genaueren Verständnis hilft das Schaltbild, daß aus Platzgründen auf der nächsten Seite zu finden ist.
Man erkennt sofort die Bestandteile, welche schon im Blockschaltbild (Abb. 3.5) zu finden waren, es sind die fünf Differenzverstärker sowie die 3 Spindelpotentiometer aus Kap. 4.5 und ein einzelner Ausgangstransistor. Die Differenzverstärker sind genau so aufgebaut wie in Abb. 3.3, dies hat mehrere Gründe die nocheinmal kurz zusammengefaßt werden sollen.

- Einsparung eines Emitterwiderstandes pro Differenzverstärker
- Einstellung der Verstärkung mit dem einzelnen Emitterwiderstand, ohne dabei die Kollektorruhepotentiale zu verändern.
- Mit dem Emitterwiderstand wird die Steigung der Großsignalkennlinie beeinflußt, deshalb muß dieser wegen der Einstellbarkeit an externe Anschlußpins geschaltet werden, ==> es ergibt sich eine Verringerung der benötigten Pins von 3 auf 2 für die ersten drei Differenzverstärker.
- Noch weniger Einfluß bezüglich der Temperatur wegen der Stromgegenkopplung

Als Nachteil erkauft man sich dafür die doppelte Anzahl von Stromquellen, dies ist jedoch nicht weiter schlimm, da alle fünf Differenzverstärker durch eine gemeinsame Strombank gespeist werden. Diese besteht aus den zwölf Transistoren $Q_{12,12.1}$ $Q_{10,11}$ $Q_{20,21}$ $Q_{30,31}$ $Q_{40,41}$ und $Q_{50,51}$. Die Einstellung der Ströme geschieht durch den Widerstand R_{S1} in Verbindung mit Q_{12}. Der entstehende Referenzstrom wird wegen des Spiegelverhältnisses von 1 (alle Emitterflächen sind gleich) an alle Stromquellentransistoren übertragen. Der Transistor $Q_{12.1}$ liefert dabei die notwendigen Basisströme, ohne den Transistor dürfen sie bei der Berechnung nicht mehr einfach vernachlässigt werden.

Transistor Q_{12} kann eigentlich als Diode betrachtet werden, somit ergibt sich der Referenzstrom aus Betriebsspannung und Vorwiderstand zu :

$$I_{ref} = \frac{U_{B1}}{R_{S1}} \qquad I_{ref} = \frac{3,65\ V}{20000\ \Omega} \qquad I_{ref} = 185\ \mu A \qquad (4.11)$$

Von diesem Wert gehen noch ca. 5 μA als Basisstrom für $Q_{12.1}$ ab , sodaß ca. 180 μA als Referenzstrom verbleiben. Dieser Strom fließt theoretisch auch durch die 10 Stromquellentransistoren. Es zeigte sich jedoch bei den Simulationsergebnissen, daß dort im Mittel nur ca. 140 μA fließen, der Unterschied liegt wahrscheinlich an den nichtidealen Transistoren.

5.26

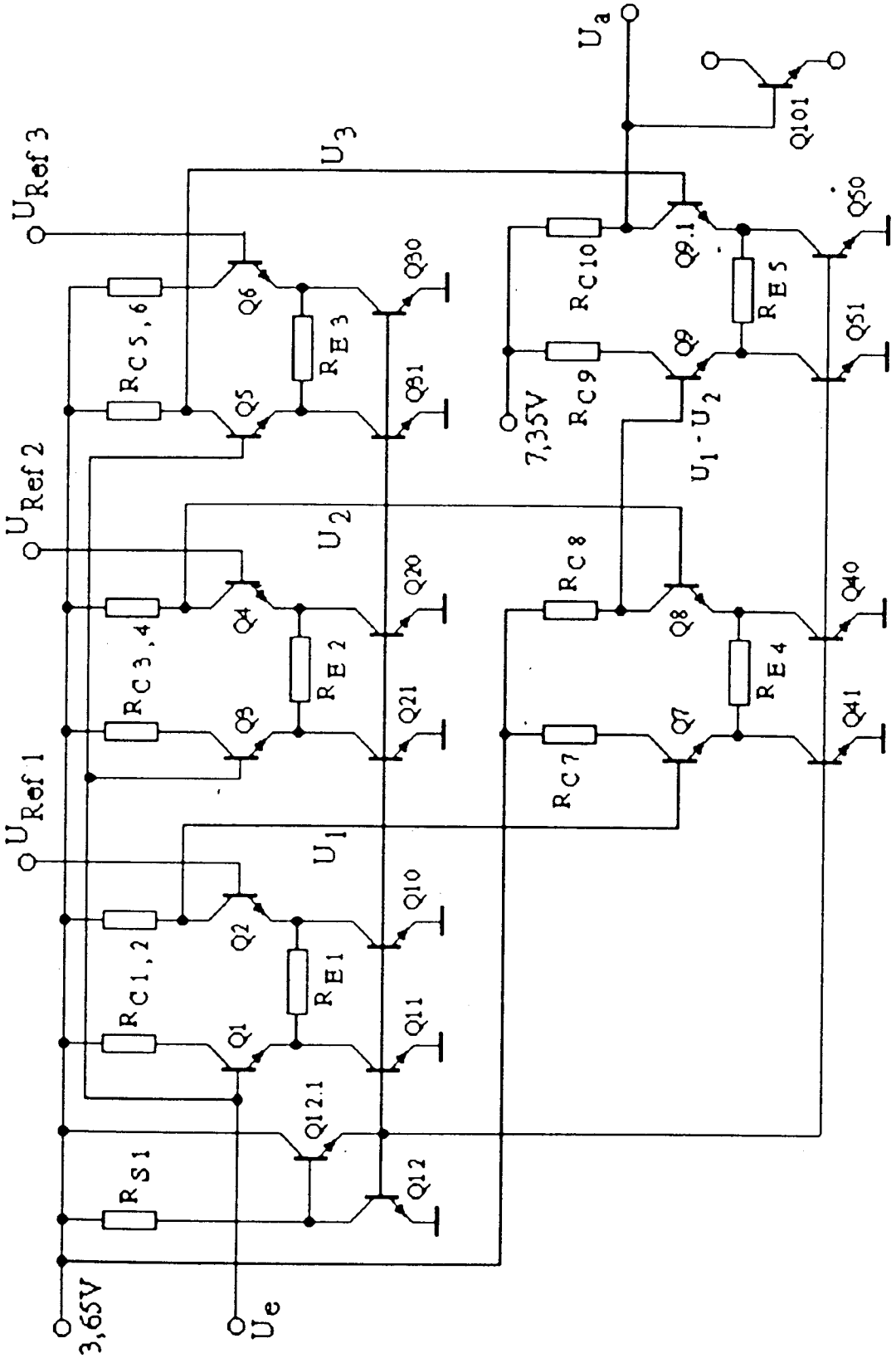


Abb. 4.8 Schaltbild zur Erzeugung der nichtlinearen Kennlinie

Hier tritt noch ein weiteres Problem zu Tage: Der Referenzstrom hängt direkt von R_{S1} ab, der jedoch Herstellungstoleranzen und eine Temperaturabhängigkeit aufweist, und so nicht konstant ist. Auf die Lösung dieses und anderer damit verbundener Probleme wird im Kapitel 8 noch etwas näher eingegangen.

Die Differenzverstärker sind genau wie im Blockschaltbild miteinander verschaltet. Die Transistoren Q_1 , Q_3 , Q_5 bilden die jeweils ersten Eingänge der drei kennlinienbestimmenden Differenzverstärker, sie sind direkt mit der linearen Eingangsspannung verbunden. Die Transistoren Q_6 , Q_{16} , Q_{26} bilden die jeweils zweiten Eingänge, sie sind mit den Potentiometern der Referenzspannungseinstellung verbunden. Ein Durchfahren der Eingangsspannung läßt eine Kennlinie wie in Abb. 3.2 entstehen.
Die ersten beiden Ausgänge (Kollektor von Q_6 und Q_{16}) gehen direkt auf die Eingänge der nachfolgenden 4. Stufe (Q_7 und Q_8). Die Ausgänge von Stufe 3 (Kollektor Q_5) und Stufe 4 (Kollektor Q_8) gehen auf die Eingänge der 5. Schaltungsstufe (Q_9 und $Q_{9.1}$). Den Ausgang der Hauptschaltung bildet der Kollektoranschluß von Transistor $Q_{9.1}$
Wie weiter oben schon erwähnt wurde, müssen die Emitterwiderstände R_{E1} , R_{E2} und R_{E3} wegen der Einstellmöglichkeiten an externen Anschlußpins angeschlossen sein, sie werden deshalb nicht wie alle anderen Widerstände integriert.
Die Ausgangsspannung kann also direkt abgegriffen werden, eine zweite Möglichkeit besteht darin, den eingebauten Ausgangstransistor zu benutzen. Sein Emitter und Kollektor sind nach außen geführt und bieten so dem Anwender volle Freiheit d.h. man kann ihn zum Beispiel als Emitterfolger oder als weitere Verstärkerstufe benutzen.

Die ersten vier Stufen werden mit einer Spannung von 3,65 V betrieben, die fünfte Stufe mit 7,35 V. Die Ausgangspotentiale der dritten und vierten Stufe liegen im Bereich von 3,3 V bis 3,65 V d.h. knapp unter der Betriebsspannung. Um mit der letzten Stufe eine ausreichende Verstärkung und einen Spannungshub von ca. 2,5 V zu erreichen ist eine höhere Betriebsspannung notwendig, da das Kollektorpotential von Q9.1 nicht unter das Basispotential fallen kann.
Die ausführliche Berechnung der Differenzverstärker wird im Kapitel 6.2 behandelt.

Zum Abschluß dieses Kapitels, in dem die Funktion der Einzelschaltungen beschrieben wurde, soll auf der folgenden Seite noch einmal das komplette Schaltbild in zusammenhängender Form gezeigt werden. Da es schon relativ umfangreich ist, soll auf die Bauteilebezeichnungen und die Widerstandswerte verzichtet werden, diese wurden schon in den Einzelschaltungen angegeben. Allerdings sind die Knotennummern des Spice-Listings eingetragen.

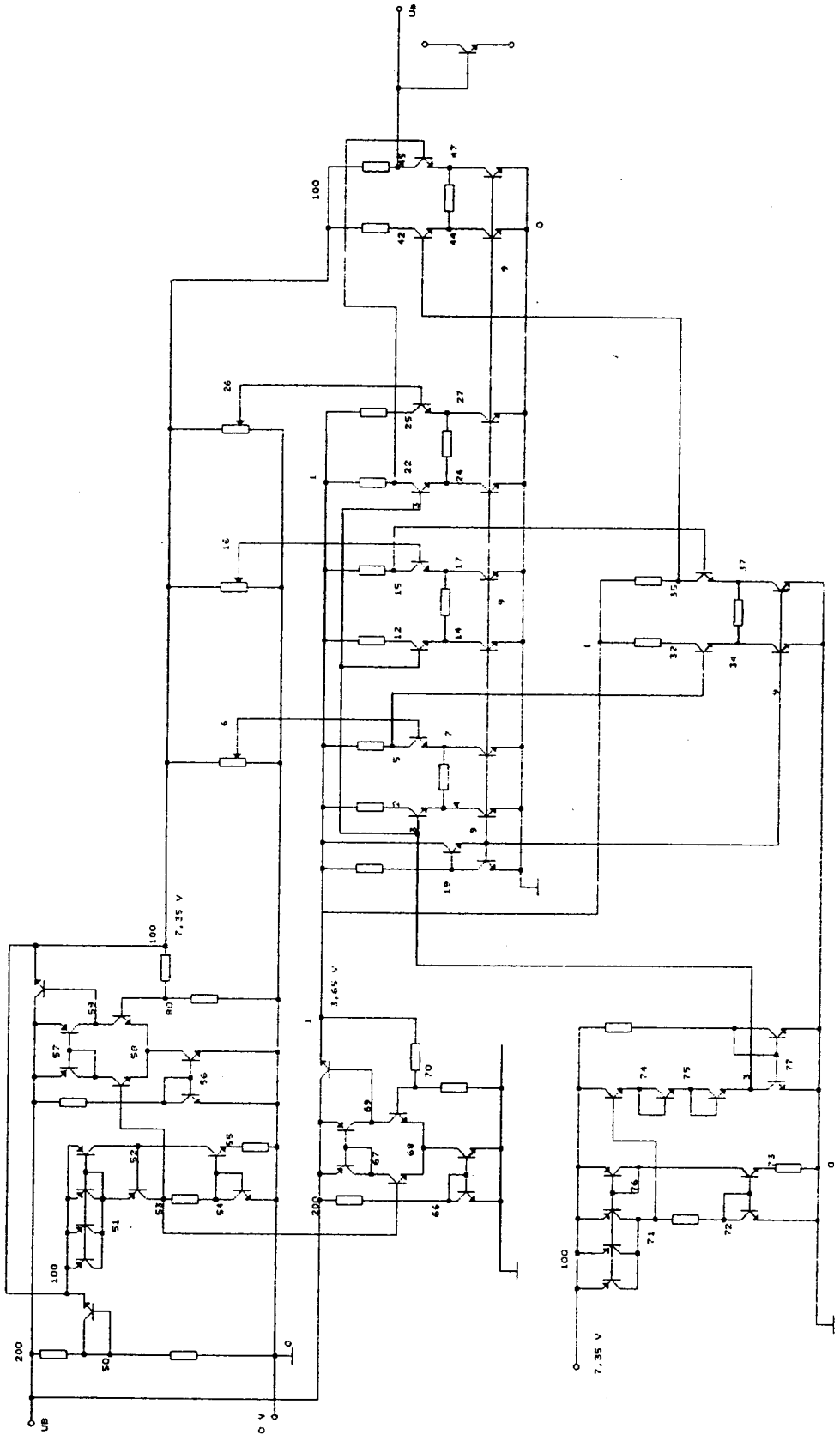


Abb. 4.9 Gesamtschaltung des Temperaturspannungs-Umsetzers

5 Funktionserklärung an einem konkreten Beispiel

5.1 Allgemeines

In diesem Kapitel soll, an einem in allen Einzelheiten durchsimulierten Beispiel, die komplette Wirkungsweise der ganzen Schaltung erläutert werden. Dazu werden als Hilfsmittel die Ausdrucke benutzt, welche mit "Pspice" unter Zuhilfenahme des Softwareoszilloskops "Probe" erstellt wurden. Bei der Simulation sind die Temperaturabhängigkeit und die Herstellungstoleranzen der Bauelemente voll berücksichtigt worden.
Die maximale Ausdehnung der Kurven U_1 U_2 und U_3 auf der Spannungsachse wird durch die jeweiligen Kollektorwiderstände bestimmt. Wenn sie alle gleich groß wären, dann hätten die drei Kurven alle die gleichen Y-Ausdehnungen. Die Kollektorwiderstände eines jeden Differenzverstärkers sind aber unterschiedlich. Dies führt dazu, daß Kurve 2 die kleinste (mit 135 mV), Kurve 1 die mittlere (mit 185 mV) und Kurve 3 die größte vertikale Ausdehnung mit ca. 405 mV besitzt. Ein solches Vorgehen war erforderlich um die absoluten Extremwerte der Kurve (bei -50°C und bei +100°C) unterhalb bzw. oberhalb des positiven bzw. negativen Umkehrpunktes anzusiedeln, dadurch wird dem Kurvenverlauf ein besseres Aussehen geboten.

5.2 Einstellungen der verschiedenen Parameter

Die Form der Ausgangskennlinie hängt neben der Temperatur von insgesamt sechs Parametern ab. Dazu zählen die drei Emitterwiderstände R_{E1} R_{E2} R_{E3} sowie die drei Referenzspannungen U_{Ref1} U_{Ref2} U_{Ref3} . Alle diese Parameter lassen sich unabhängig voneinander einstellen. Mit den Emitterwiderständen läßt sich, wie schon gesagt, die Steigung der drei Kennlinienäste beeinflussen.

Sie besitzen hier folgende Werte :

R_{E1} = 1600 Ohm
R_{E2} = 1400 Ohm
R_{E3} = 1400 Ohm

Auf die Auswirkungen der Referenzspannungseinstellungen soll im Diagramm auf der nächsten Seite eingegangen werden.

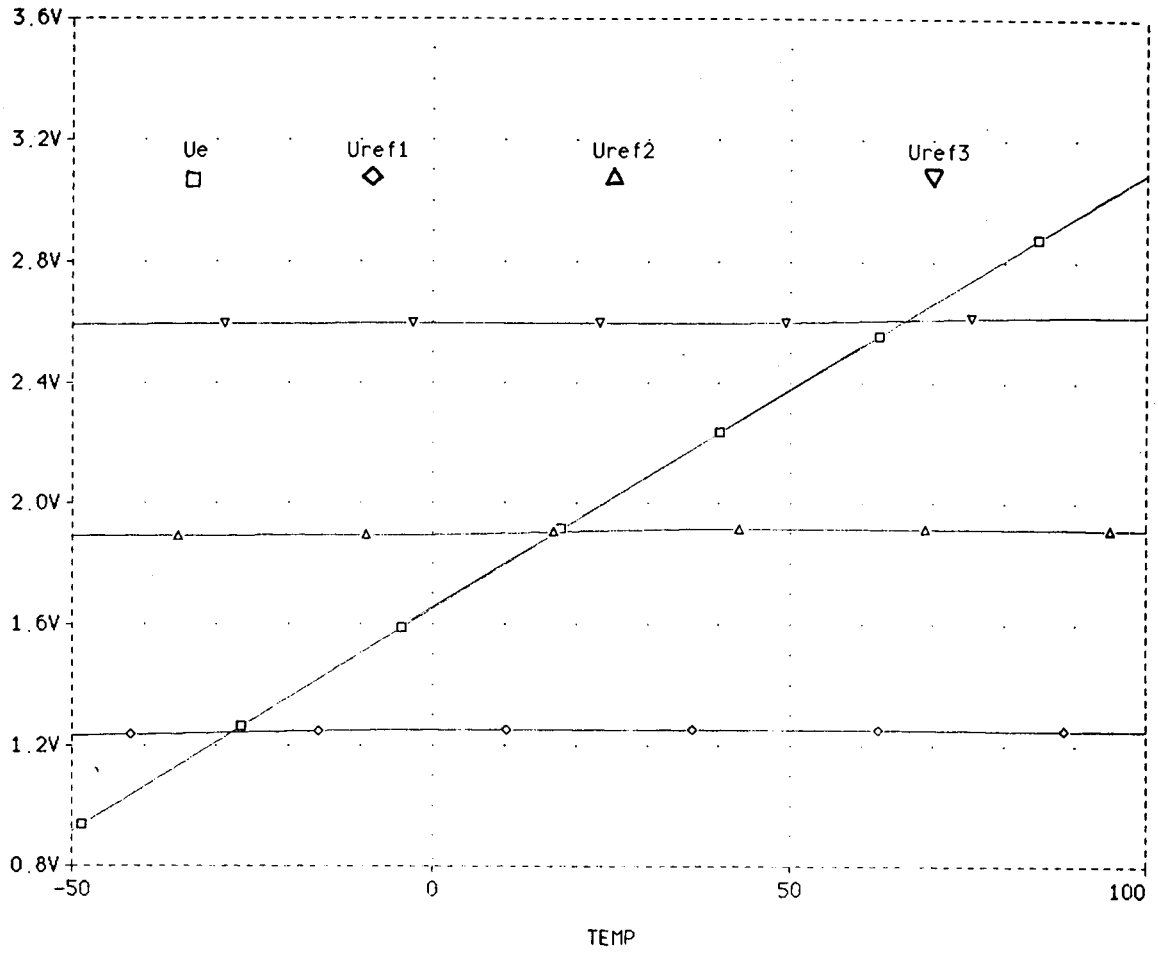


Abb. 5.1 Darstellung der Eingangs- und Referenzspannungen

Auf dem Ausdruck sind die Lage der Eingangsspannung und die der drei Referenzspannungen zueinander dargestellt. Die Eingangsspannung bewegt sich im Bereich von 0,9 V bis zu 3,1 V . Die Referenzspannungen bilden horizontale Geraden, welche die steigende Gerade an drei Stellen kreuzen. Diese drei Stellen sind die genauen Einstellwerte der Referenzspannungen und bilden die Punkte, an denen die Kennlinienäste ihre maximale Steigung erreichen (dort entstehen bei der Ausgangskennlinie Wendepunkte).
Aus dem Diagramm kann man zu jeder eingestellten Referenzspannung am Kreuzungspunkt die zugehörige Temperatur ablesen.

Hier sind eingestellt	1.Punkt :	1,25 V	- 28°C
	2.Punkt :	1,9 V	+17°C
	3.Punkt :	2.6 V	+75°C

5.3 Darstellung der Ergebnisse des Beispiels

Aus den eben festgelegten Werten ergeben sich im folgenden Diagramm die Kurven U_1 U_2 U_3 sowie U_1-U_2 genau so, wie sie im Prinzipbild in Abb. 3.4 dargestellt worden sind.

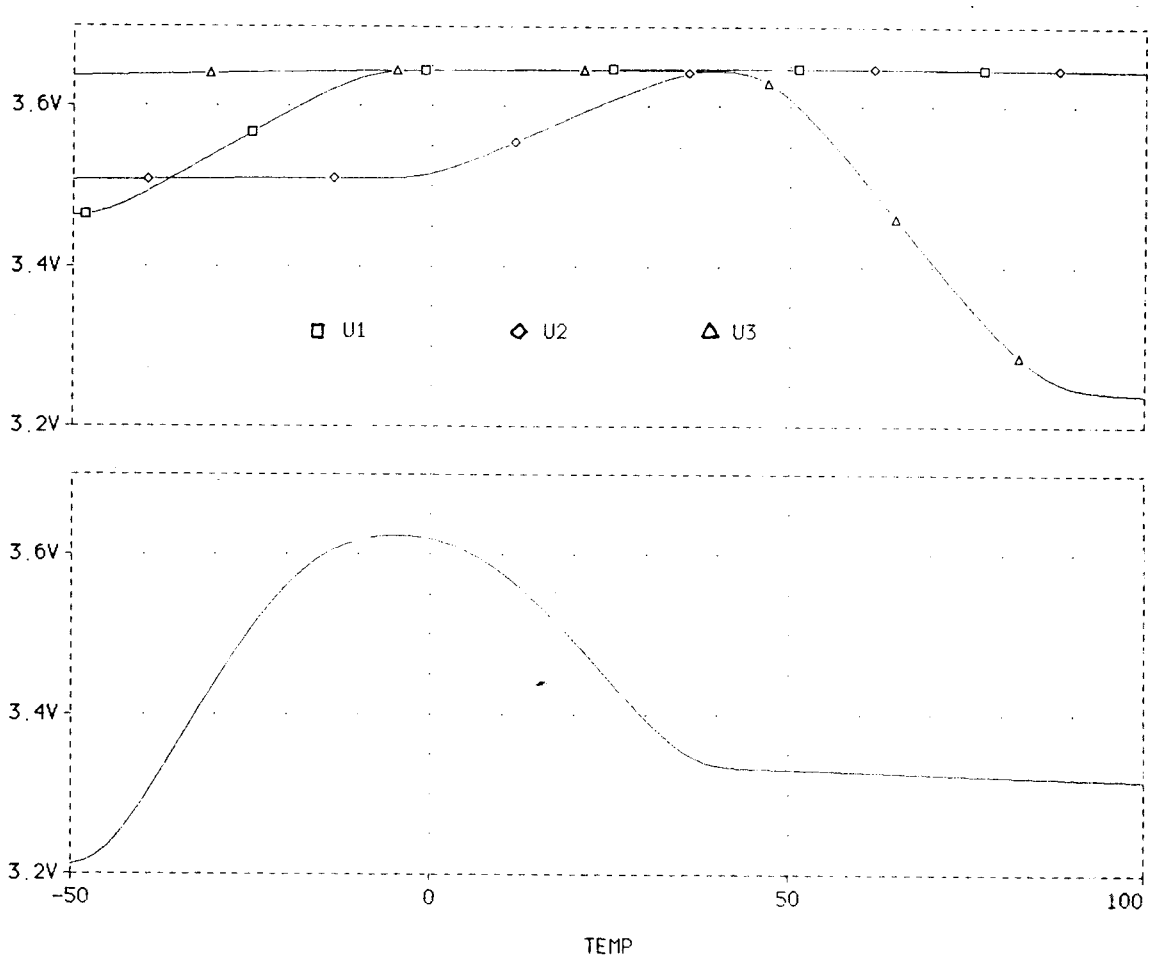


Abb. 5.2 Die Kurvenformen der Ausgangsspannungen von Differenzverstärker 1 - 4

Im ersten der beiden Diagramme sind die Spannungen U_1 U_2 und U_3 dargestellt, sie haben unterschiedliche Y-Ausdehnungen wie es vorher beschrieben wurde. Die wichtigen Spannungswerte lassen sich leicht berechnen. (Differenzeingangsspannung = 0V).

Kurve 1: Kollektorruhepotential (Wendepunkt) bei : U = 3,55 V

$U = U_{B1} - R_C \cdot I_O$

$=>$ 3,65V - 700 Ohm · 135 μA = 3,55V

maximaler Spannungshub : 2 · 700 Ohm · 135 μA

U = 189 mV

Kurve 2 : Kollektorruhepotential (Wendepunkt) bei : U = 3,58 V

=> 3,65V - 520 Ohm · 135 μA = 3,58V
maximaler Spannungshub : 2 · 520 Ohm · 135 μA
U = 140 mV

Kurve 3: Kollektorruhepotential (Wendepunkt) bei : U = 3,45 V

=> 3,65V - 1440 Ohm · 135 μA = 3,45V
maximaler Spannunghub : 2 · 1440 Ohm · 135 μA
U = 390 mV

Diese berechneten Spannungswerte lassen sich alle in dem Diagramm wiederfinden. Die Kurvenverläufe sehen im wesentlichen so aus, wie sie vorhergesagt wurden.
Im unteren Diagramm die Kurve, die durch die Subtraktion von U_1 und U_2 entsteht. Dabei kann man am fast horizontalen Verlauf am rechten Bildrand das Kollektorruhepotential von Differenzverstärker 4 ablesen, da dort dessen Eingangsspannungsdifferenz zu Null geworden ist.

Kurve 4 : Kollektorruhepotential bei U = 3,33 V
=> 3,65V - 2300 Ohm · 135 μA = 3,33V

Mit der Subtraktion der Spannung U_3 von der Kurve 4 kommt man zu der entgültigen Ausgangsspannung, die in Abb. 5.3 dargestellt ist.

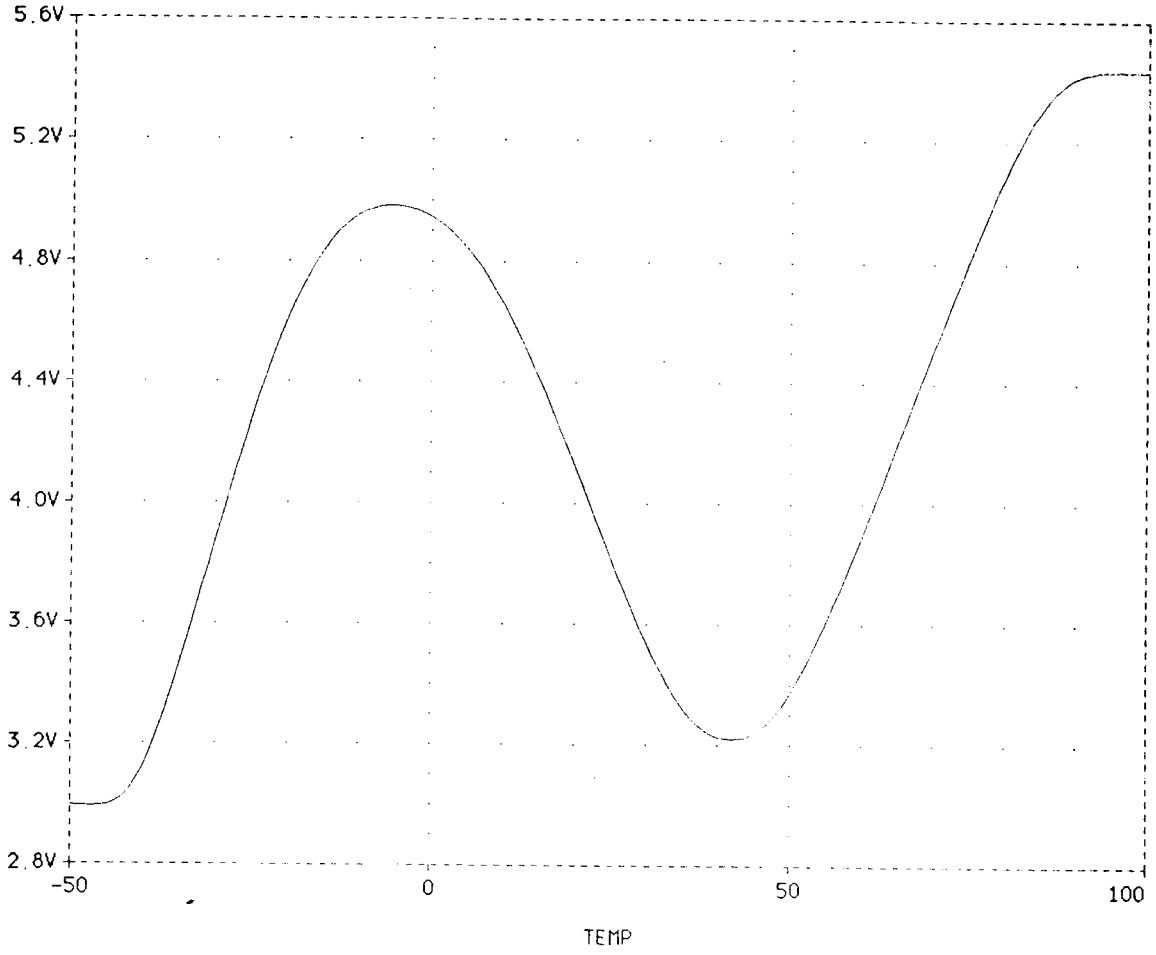


Abb. 5.3 Ausgangspannung des Temperatur-Spannungsumsetzers (Beispiel)

Die Kurve besitzt einen absoluten Spannungshub von ca. 2,5 V. In der Mitte der drei geraden Teilstücke liegen jeweils die drei Referenztemperaturen wie sie am Anfang eingestellt wurden. Der Verlauf entspricht somit der geforderten Vorgabe unter Abb. 1.1 in der Einleitung. ==> Ein Teilpunkt der Diplomarbeit wäre gelöst.

Mit entsprechenden Maßnahmen läßt sich die Grundform jetzt weitläufig verändern, diese Einstellmöglichkeiten werden im Kapitel 9 behandelt.
Eine genaue mathematische Beschreibung der Gesamtkennlinie wird in Kapitel 7 hergeleitet, so weit dies überhaupt möglich ist. Man kann an dieser Stelle jedoch schon andeuten, daß eine gute Nährungsformel hier die wahrscheinlich bessere und weit einfachere Lösung darstellt.

Eine Kurve, bei der die beiden äußeren Referenzspannungen etwas weiter in die Mitte rükken ist in Abb 5.4 dargestellt. Dadurch wird die Amplitude der beiden Umkehrpunkte kleiner, im Extremfall werden sie völlig verschwinden, an dieser Stelle ergibt sich ungefähr eine Gerade.

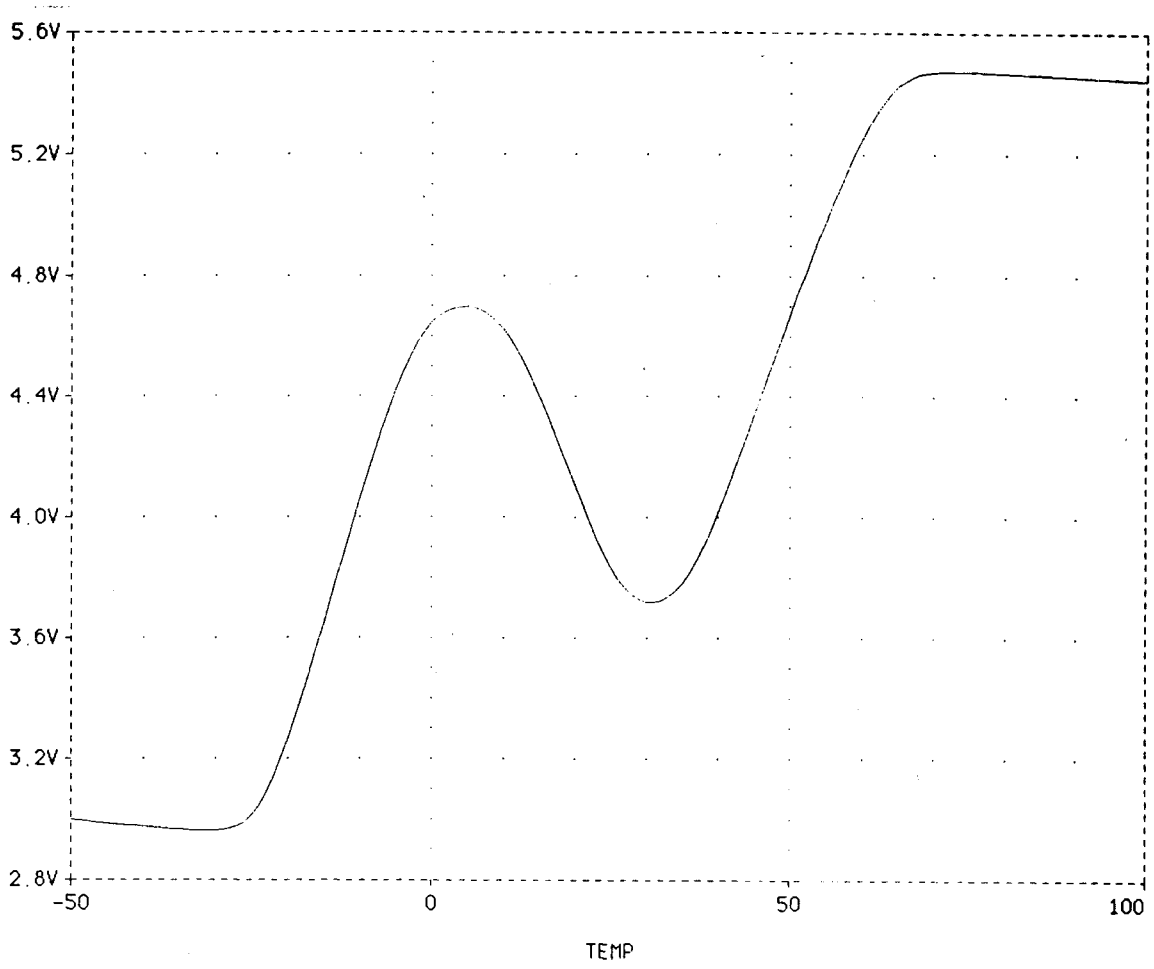


Abb. 5.4 Zweites Beispiel einer Ausgangsspannung

7 Entwicklung einer Näherungsgleichung für die Gesamtkennlinie

7.1 Einleitung

Die Gleichung für den Kollektorstrom eines Differenzverstärkers mit Stromgegenkopplung läßt sich nach einigen mathematischen Schwierigkeiten herleiten. Es stellt sich heraus, daß sie transzendent ist und eine genaue mathematische Lösung sehr schwer fällt.
Da es aber sehr sinnvoll ist die gesamte Kennlinie beschreiben zu können (Vorhersage des Kurvenverlaus bei beliebigen Einstellungen), muß nach einer Lösung gesucht werden, die wesentlich einfacher zu handhaben ist. Dabei liegt der Gedanke nah, eine Näherungslösung anzugeben, deren Genauigkeit im Bereich von $\pm 5\%$ liegt.
Die Näherung, die in diesem Kapitel ausgearbeitet wird, basiert auf dem Grundprinzip der Verschaltung der Differenzverstärker, wie sie im Blockschaltbild und bei Abb. 4.8 wiedergegeben ist.

7.2 Grundgedanke der Näherungsgleichung

Nach "Tietze/Schenk (Halbleiterschaltungstechnik)" läßt sich die Kennlinie eines einzigen Differenzverstärkers ohne Stromgegenkopplung eindeutig beschreiben.

$$U = U_B - \frac{R_C I_O}{2}\left(1 + \tanh\frac{U_D}{2\,U_T}\right) \qquad (7.1)$$

U	jeweilige Kollektorspannung
R_C	Kollektorwiderstand
U_B	Betriebsspannung
U_D	Differenzspannung
I_O	Strom der Stromquelle

Für eine genaue Anpassung auf den vorliegenden Fall werden noch zwei Veränderungen eingeführt :

- Das in der Gleichung stehende ½ I_O wird durch I_O ersetzt, weil bei der verwendeten Schaltung die eine Stromquelle in zwei, mit je dem halben Strom, aufgeteilt wurde.
- Die Differenzspannung U_D entsteht jetzt aus der Differenz der Eingangsspannung Ue (0,9 V bis 3,1 V) und der einstellbaren Referenzspannung U_{ref}.

Die Gleichung 7.1 stellt sich nun so dar

$$U = U_B - R_C I_O \left(1 + \tanh \frac{U_{ref} - U_e}{2 U_T}\right) \qquad (7.2)$$

Mit dieser endgültigen Form lassen sich die drei Ausgangsspannungen U_1 U_2 U_3 in Abhängigkeit von Kollektorwiderstand, Referenzspannung und Eingangsspannung beschreiben. Natürlich ist darin auch die Abhängigkeit von der Temperatur enthalten, die in U_T steckt.

$U_T = \frac{k T}{e}$ Wegen der Übersichtlichkeit wird weiterhin nur U_T benutzt.

Prinzipiell läßt sich die Kennlinie genau so beschreiben, wie sie mit der Schaltungstechnik erzeugt wird. Dazu sind fünf Schritte notwendig. (Siehe auch Abb. 3.4)

- Bildung von U_1 U_2 U_3
- Bildung von U_1 - U_2
- Verstärkung dieser Spannung und anheben auf das Kollektorruhepotential durch die vierte Stufe
- Bildung von U_1 - U_2 - U_3
- Verstärkung dieser Spannung und anheben auf das Kollektorruhepotential durch die fünfte Stufe

Mit diesen Vorarbeiten kann man die Grundform der verwendeten Näherungsgleichung aufstellen.

Die Ausgangsspannung ergibt sich, wenn die Spannung U_2 von der Spannung U_1 subtrahiert wird. Das Ergebnis wird mit der Verstärkung von Differenzverstärker 4 multipliziert und dessen Kollektorruhepotential wird addiert. Von diesem Zwischenergebnis wird die Spannung U_3 subtrahiert, daß ganze wird mit der Verstärkung von Differenzverstärker 5 multipliziert und dazu wird noch dessen Kollektorruhepotential addiert.

U_1 Ausgangsspannung des 1. Differenzverstärkers
U_2 Ausgangsspannung des 2. Differenzverstärkers
U_3 Ausgangsspannung des 3. Differenzverstärkers

Trotz allem sind immer noch einige Unbekannte vorhanden,dies sind die Verstärkungen, die Ruhepotentiale und der Einfluss der Emitterwiderstände

7.3 Ermittlung der Verstärkungen und der Ruhepotentiale

Hier sind nur die Werte der Stufen vier und fünf interessant, da nur sie in der Näherungsgleichung gebraucht werden. Die Spannungen U_1 U_2 U_3 können berechnet werden. Die Berechnung der Ruhepotentiale ist schon erläutert worden, der Vollständigkeit halber sind sie noch mal aufgeführt.

Stufe 4: $U_R = 3{,}65\ V - 2300\ Ohm \cdot 135\ \mu A = \underline{3{,}339\ V}$

Stufe 5: $U_R = 7{,}35\ V - 17000\ Ohm \cdot 135\ \mu A = \underline{5{,}055\ V}$

Diese Ergebnisse stimmen mit den Werten, welche man aus den Diagrammen ablesen kann, genau überein !

Verstärkungsfaktoren :

Die Verstärkungsfaktoren können nach den Gleichungen berechnet werden, die in Kapitel 6.3 angegeben sind. Wegen der Schwankungen des Kollektorstroms und der Temperaturspannung sind diese aber auch nicht exakt richtig. Deshalb wurden die realen Werte aus vergrößerten Diagrammen des Beispiels in Kapitel 5 gewonnen. Das Verfahren ist recht einfach:

Da das Ruhepotential der 4. Stufe bekannt ist, kann man an einer definierten Temperatur den Abstand bis zum entsprechenden Y-Wert der Kurve 4 genau ablesen ==> dies ergibt die verstärkte Ausgangsspannung. An der selben definierten Stelle wird der Abstand der Kurven 1 und 2 ermittelt ==> dies ergibt die Differenzeingangsspannung. Indem man den ersten Wert durch den zweiten dividiert ergibt sich ein Verstärkungsfaktor für die 4. Stufe. Um einen zuverlässigen Faktor zu erhalten führt man das Verfahren mehrere Male an verschiedenen Stellen durch und bildet einen Mittelwert. Für die vierte Stufe ergaben sich so folgende Werte :

V = 124 mV : 48 mV	V = 2,58
V = 283 mV : 129 mV	V = 2,19
V = 106 mV : 40 mV	V = 2,65
V = 171 mV : 65 mV	V = 2,63
V = 184 mV : 70 mV	V = 2,62

Als Mittelwert wurde eine Verstärkung von V = 2,59 gewählt.

Zum Vergleich ergibt eine Rechnung nach Kapitel 6.3 einen Wert von V = 2,73 bei einem mittleren Kollektorstrom von 135 μA. Die beiden Zahlen stimmen also recht gut überein.

Zur Ermittelung des Verstärkungsfaktors von Stufe 5 wird genauso vorgegangen. Das Ruhepotential ist bekannt, an definierten Stellen werden die Werte zwischen Ruhepotential und Kurve 5 einerseits sowie die Differenz zwischen Kurve 3 und 4 andererseits abgelesen. Daraus wird dann ebenfalls ein Verstärkungsfaktor errechnet

V = 485 mV : 75 mV	V = 6,45
V = 1605 mV : 267 mV	V = 6,01
V = 775 mV : 125 mV	V = 6,21
V = 1173 mV : 198 mV	V = 5,92
V = 176 mV : 30 mV	V = 5,86

Als Mittelwert wurde eine Verstärkung von V = 6,09 gewählt.

Eine Vergleichsrechnung ergibt hier einen Wert von V = 3,69. Die Diskrepanz kann damit erklärt werden, daß fast im ganzen Arbeitsbereich eine Differenzspannung anliegt, die negativ ist. Deshalb darf hier nicht mit einem mittleren Kollektorstrom von 135 μA gerechnet werden. Einen Einfluß haben sicherlich auch die Nichtlinearitäten der Transistoren.

7.4 Der Einfluß des Emitterwiderstandes

Die einzige noch verbleibende Unbekannte war der Einfluß des Emitterwiderstandes. Wie schon erwähnt, läßt sich dadurch der Kollektorstrom nicht berechnen. Der Emitterwiderstand hat eine direkte Wirkung auf die Form der Kennlinie. Je höher der Widerstand, desto flacher wird der Verlauf und desto größer wird der lineare Bereich der Kennlinie.

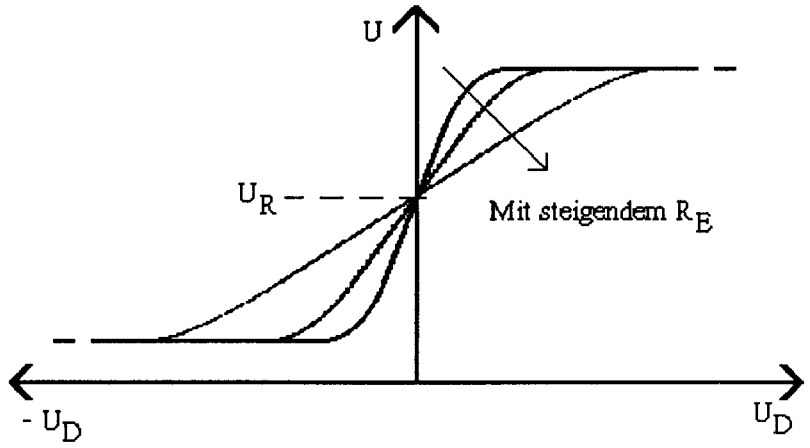


Abb. 7.1 Einwirkung des Emitterwiderstandes auf die Kennlinie

Wie kann dieser Einfluß in der Näherungsgleichung eingeführt werden? Die Gleichung, welche die Kollektorspannung beschreibt, müßte um einen geeigneten Korrekturterm erweitert werden, der die Verringerung der Steigung ausdrückt.

$$U = U_B - R_C\, I_O \left(1 + \tanh \frac{U_{ref} - U_e}{2\, U_T}\right) \qquad \text{Gleichung 7.2 von oben}$$

Um den Verlauf anzupassen, muß das Argument des tanh verändert werden. Da die Steigung mit wachsendem Emitterwiderstand immer kleiner wird, muß vom Argument etwas subtrahiert werden. Um sicherzustellen das durch den Korrekturterm kein Fehler entsteht, müssen einige Bedingungen eingehalten werden.

- Bei einer Differenzeingangsspannung von 0V darf sich das Ruhepotential nicht ändern (sonst würde keine Drehung um den Wendepunkt erfolgen)
- Wenn kein Emitterwiderstand vorhanden ist, muß der Korrekturterm verschwinden

5,40

- Der Emitterwiderstand selbst muß enthalten sein
- Die jeweilige Referenzspannung muß enthalten sein

Aus diesen Forderungen ergibt sich für den Korrekturterm folgendes Aussehen.

$$\frac{(U_{ref} - Ue)\,R_E}{F} \qquad \text{(allgemeine Form)}$$

Der Faktor F im Nenner muß noch bestimmt werden, durch das Vorhandensein von U_{ref} , Ue und R_E werden die oben aufgestellten Bedingungen eingehalten. Zusammen mit Gleichung 7.2 ergibt sich nun die nachstehende Form.

$$U = U_B - R_C I_O \left[1 + \tanh\left[\left(U_{ref} - Ue\right)\left(\frac{1}{2\,U_T} - \frac{R_E}{F}\right)\right]\right] \qquad (7.3)$$

Damit das Argument dimensionslos wird, muß der Fakor F eine eigene Einheit erhalten.

$$[F] = \frac{V^2}{A}$$

Wie läßt sich der wichtige Faktor F ermitteln ?

Man kann mit "Pspice" für jeden der ersten drei Differenzverstärker eine seperate Simulation durchführen. Dabei macht man den Emitterwiderstand variabel und läßt sich für 5 oder 6 Werte je einen Ausdruck anfertigen.
Danach gibt man die Gleichung 7.3 mit dem relevanten Emitterwiderstand und den anderen bekannten Variablen in ein Mathematikprogramm ein und setzt einen sinnvollen Wert z B. F = 20 willkürlich ein und läßt sich die Funktion zeichnen. Die entstandene Kurve wird mit dem Pspice-Ausdruck verglichen und danach der Faktor F verbessert. In der Regel gelangt man durch das Ausprobieren von 3 oder 4 Faktoren schon auf eine sehr gute Übereinstimmung. Dieser so gefundene passende Faktor F gehört nur zu dem einen Emitterwiderstand mit dem er ausprobiert wurde.
Nachdem das ganze Verfahren für 4 oder 5 Emitterwiderstände durchgeführt wurde, hat man genausoviele verschiedene Faktoren F und kann daraus einen Zusammenhang ableiten.

Obwohl das Verfahren auf systematischem Ausprobieren beruht, ist es doch sehr genau und liefert die gewünschten Ergebnisse in relativ kurzer Zeit.

Für Differenzverstärker 1 ergaben sich dabei folgende Werte:

R_E in Ohm	A
200	38
700	65
1200	91
1600	112
2000	133
2400	152

Wenn man diese Werte in ein Diagramm einträgt, dann ergibt sich ein sehr überraschendes Bild, es entsteht eine Gerade !!!
Da sich eine absolute Gerade ergibt, kann man mit einer Geradengleichung den Faktor F in Abhängigkeit vom Emitterwiderstand errechnen. Da der Faktor nur für den ersten Differenzverstärker gültig ist, wird er nun in A umbenannt (B für den 2. und C für den 3. Differenzverstärker)

Es gilt : $$A = 0{,}0522 \cdot R_E + 28{,}09$$

Was sofort auffällt, ist die Zahl 0,0522 . Sie erinnert stark an $2 \cdot U_T$. Dafür gibt es wahrscheinlich einen genauen mathematischen Zusammenhang, der bis jetzt noch nicht ergründet ist.

Für Differenzverstärker 2 lauteten die Werte :

R_E in Ohm	B
100	27
500	50
1100	83
1600	109
2000	129
2400	150

Auch hier läßt sich mit einer Geradengleichung eine Abhängigkeit angeben

Es gilt : $$B = 0{,}0529 \cdot R_E + 23.24$$

Es ist ebenfalls eine fast vollständige Übereinstimmung mit 2 U_T festzustellen.

Für den dritten Differenzverstärker ergibt sich :

R_E in Ohm	C
500	53
1200	89
2200	141
3000	181
4000	235
5000	287

Mit der Geradengleichung gilt : $C = 0{,}0517 \cdot R_E + 27{,}43$

Auch hier zeigt sich wieder diese auffällige Ähnlichkeit mit $2 \cdot U_T$

Nachdem es nun möglich ist die Korrekturfaktoren A , B und C zu berechnen, kann nun die vollständige Näherungsformel aufgestellt werden. Sie orientiert sich genau an dem Merksatz in Kapitel 7.2 .

Von der Theorie her, hängt der Korrekturfaktor nicht vom Kollektorwiderstand ab, da er nur im tanh steckt. Die drei Geradengleichungen zeigen aber eine kleine Abweichung. Deshalb könnte man aus ihnen noch eine Art von Mittelwert bilden um später nur einen Korrekturfaktor berechnen zu müssen. Bei Werten des Faktors bis fast 300 macht eine Abweichung von 1 oder 2 nichts aus.

$$F = 0{,}0522 \cdot R_E + 26{,}25$$

Da die Auswertung sowieso mit dem Rechner erfolgt, ist es gleichgültig ob nun Faktor F (wäre für alle drei gültig) oder die Faktoren A, B, C benutzt werden.

7.5 Endgültige Nährungsgleichung

Bei der folgenden Gleichung sind aus Gründen der Übersichtlichkeit die Variablen U_T und die Korrekturfaktoren A , B und C direkt eingesetzt. Die dafür notwendigen Berechnungsformeln, die bekannt sind, werden erst später direkt in einem Computerprogramm eingefügt, daß die Kurve ausplotten kann. (Ruhepotential und Verstärkung von Differenzverstärker vier und fünf sind schon eingesetzt).

Wenn man zuerst U_1 - U_2 bildet und mit der vierten Stufe verarbeitet ergibt sich :

$$Ua = 3,33V + 2,59\left[R_{C2}\cdot I_O\left[1+\tanh\left[\left(U_{ref2}-U_e\right)\cdot\left(\frac{1}{2U_T}-\frac{R_{E2}}{B}\right)\right]\right]\right.$$
$$\left.-R_{C1}\cdot I_O\left[1+\tanh\left[\left(U_{ref1}-Ue\right)\cdot\left(\frac{1}{2U_T}-\frac{R_{E1}}{A}\right)\right]\right]\right] \qquad (7.4)$$

Mit der Hinzunahme von U3 und der fünften Stufe ergibt sich das entgültige Aussehen.

$$Ua = \left[3,33V + 2,59\left[R_{C2}\cdot I_O\left[1+\tanh\left[\left(U_{ref2}-U_e\right)\cdot\left(\frac{1}{2U_T}-\frac{R_{E2}}{B}\right)\right]\right]\right.\right.$$
$$\left.-R_{C1}\cdot I_O\left[1+\tanh\left[\left(U_{ref1}-Ue\right)\cdot\left(\frac{1}{2U_T}-\frac{R_{E1}}{A}\right)\right]\right]\right]$$
$$\left.-3,65V + R_{C3}\cdot I_O\left[1+\tanh\left[\left(U_e-U_{ref3}\right)\cdot\left(\frac{1}{2U_T}-\frac{R_{E3}}{C}\right)\right]\right]\right]\cdot 6,06 + 5,05V$$
$$(7.5)$$

Mit dieser Gleichung läßt sich die gesamte Kennlinie näherungsweise beschreiben.In welchen Bereichen die Genauigkeit liegt, wird die Auswertung mit einem noch zu erstellenden Computerprogramm zeigen (wahrscheinlich $\pm 10\%$).

Literaturverzeichnis

1 M.Seifart Analoge Schaltungen
Hüthig Verlag , Heidelberg
3.Auflage 1990

2 U.Tietze , Ch. Schenk Halbleiterschaltungstechnik
Springer Verlag Berlin , Heidelberg
9.Auflage 1990

3 E.E.E. Hoefer , H.Nielinger Spice - Analyseprogramm
Springer Verlag Berlin , Heidelberg
1.Auflage 1985

4 Bosch Manuskript - Bandgabquellen als Spannungreferenzen in monolitisch integrierten Schaltungen (4.Dezember 1978)

5 AEG Manuskript - Transistor-Arrays für integrierte Analogschaltungen Serie A

6 Vorlesungsskript Entwurf integrierter Schaltungen
Prof. Dr. G. Albert , FHT Mannheim
Sommersemester 1992

7 W.Leupold Analysis für Ingenieure
Verlag Harri Deutsch Frankfurt/Main 1987

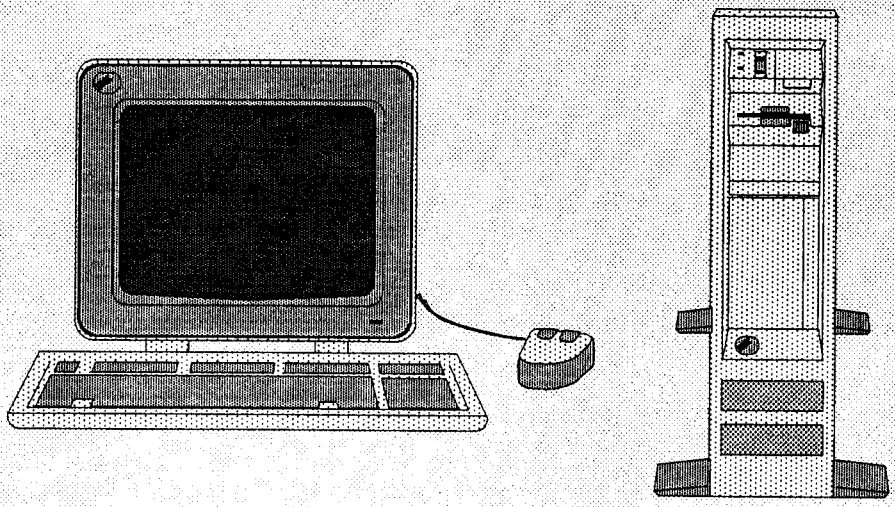
8 K.H. Rollke Das Turbo Pascal 6.0 Buch
Sybex Verlag Düsseldorf 1991

9 R.Paul Elektronische Halbleiterbauelemente
Teubner Studienskripten 1989

10 Pspice Arbeitsbuchbuch der Firma Hoschar Systemelektronik

11 U. Leingang Entwicklung eines Tinitus Maskers
Diplomarbeit FHT Mannheim 1990

ERFAHRUNGEN MIT DER BOARD-STATION VON MENTOR-GRAPHICS



Elke Mackensen

Fachhochschule Offenburg

Im Februar 1993

Erfahrungen mit der BOARD-Station von MENTOR-Graphics

Elke Mackensen

Fachhochschule Offenburg

Mit zunehmend komplexer werdenden Schaltungen wachsen auch die Anforderungen an die Entwicklung einer entsprechenden Leiterplatte. Mit der BOARD-Station von MENTOR-Graphics können professionelle Leiterplatten entwickelt werden.
Im Rahmen dreier Entwicklungsprojekte an der Fachhochschule Offenburg wurden mehrere aufwendige Layoutentwürfe mit der BOARD-Station in verschiedenen Diplomarbeiten durchgeführt.
Im Folgenden wird über die dabei gewonnenen Erfahrungen berichtet.

1. Vorstellung der Projekte

In den letzten drei Jahren wurde die BOARD-Station in mehreren Projekten erfolgreich eingesetzt.

Bei dem ersten Projekt handelt es sich um die Entwicklung eines GPS-Empfängers (GPS = Global Positioning System) für den Einsatz im KFZ-Bereich. Dieser GPS-Empfänger besteht aus einer Antenneneinheit, einem Tuner, einem Netzteil, einem Analogteil und einem Digitalteil. Bei dem schaltungstechnisch recht aufwendigen Analog- bzw. Digitalteil wurden die Platinen mit der BOARD-Station entwickelt (Abb.1-1 u. Abb.1.-2). Es wurden folgende Anforderungen an die Platinen gestellt:

Platzmässig ist man mit den Platinen auf die Größe eines Autoradios beschränkt, wobei man den Platzbedarf des Tuners und des Netzteiles abziehen muß. Wegen dieser Platzeinschränkung und des schaltungstechnisch großen Umfanges wurden

- die Platinen in SMD-Technik und als Multilayerplatinen ausgeführt.
- große Teile der Schaltung in LCA´s, GAL´s und in einen ASIC integriert, und diese bei der Platinenentwicklung miteinbezogen.

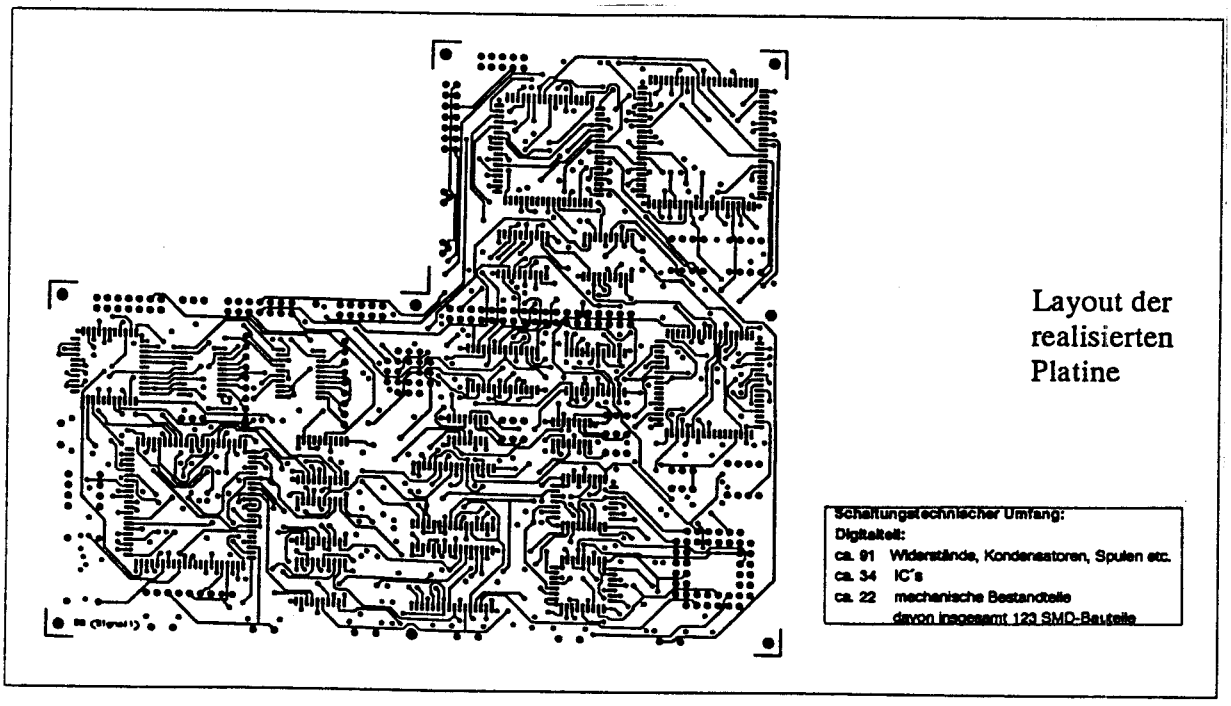


Abb.1-1: Das GPS-Projekt (Projekt 1): Digitalteil

6.2

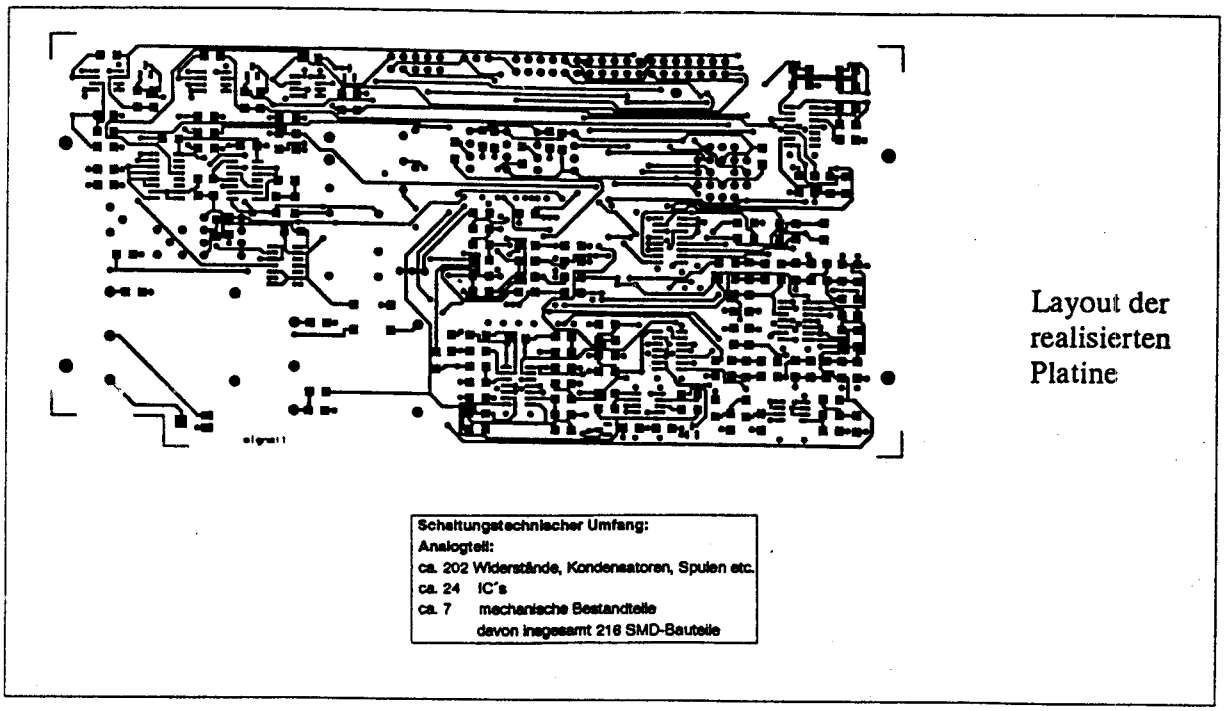


Abb.1-2: Das GPS-Projekt (Projekt 1): Analogteil

Bei dem zweiten Projekt handelt es sich um ein Industriegerät mit Display, Tastatur und IEC-Bus zur Auswertung der Daten eines optischen Wegsensors, der zur Messung von Abständen zu einem Objekt benutzt wird. (Abb.1-3). An die Platine wurden folgende Anforderungen gestellt:

- Wegen des schaltungstechnisch großen Umfangs (siehe Abb.1-3) und wegen hoher Taktfrequenzen Fertigung der Platine in SMD-Technik und als Multilayerplatine.
- Unterbringung der Schaltung auf einer Platine mit doppeltem Europakarten-Format, also nicht etwa Trennung von Analog- und Digitalteil.

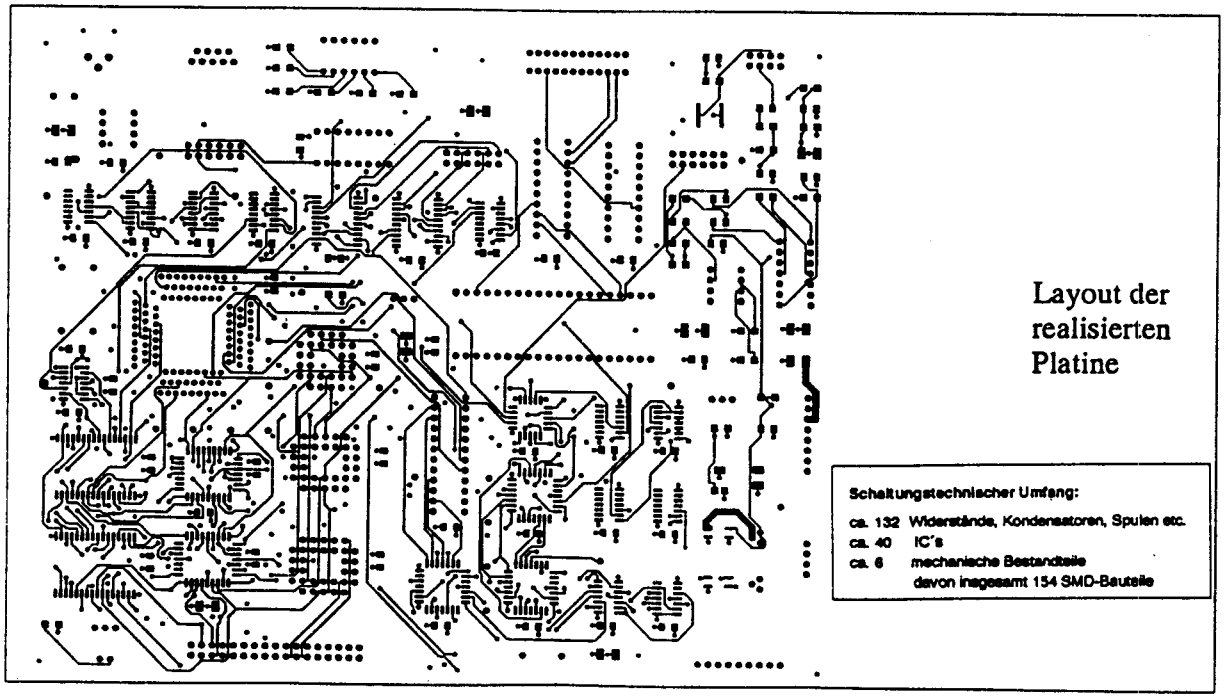


Abb.1-3: Industriegerät zur optischen Wegmessung (Projekt 2)

Bei dem dritten Projekt handelt es sich um die Entwicklung einer Elektronik für optische Datenübertragung in einem Drehbearbeitungszentrum zur Vermessung von Distanzen, Kanten und Oberflächen von Getriebewellen. Die Anforderungen an die Platine in diesem Falle waren:

- Erstellung einer runden Platine mit einem ϕ von 16 cm
- Erstellung einer zweilagigen Platine, um die Platine noch mit Mitteln der Fachhochschule fertigen zu können.
- Wegen des Schaltungstechnisch großen Umfanges auch weitestgehend Fertigung in SMD-Technik und Einsatz von Logic-Cell-Array´s (LCA´s).

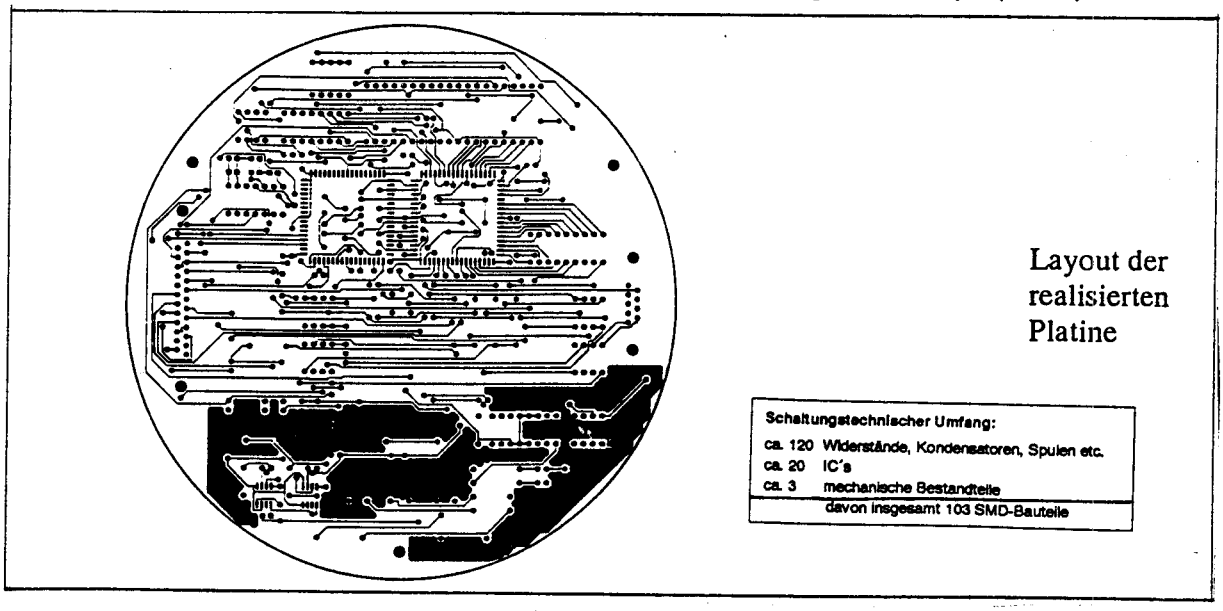


Abb.1-4: Messung von Distanzen bei einer Drehbearbeitung (Projekt 3)

2. Entwicklungsgang auf der BOARD-Station

Der prinzipielle Entwicklungsgang auf der BOARD-Station , um von dem bestehenden Schaltungsentwurf zur fertigen Platine zu kommen, ist in Abb.2-1 dargestellt. Es müssen **8 verschiedene Programmodule** durchlaufen werden:

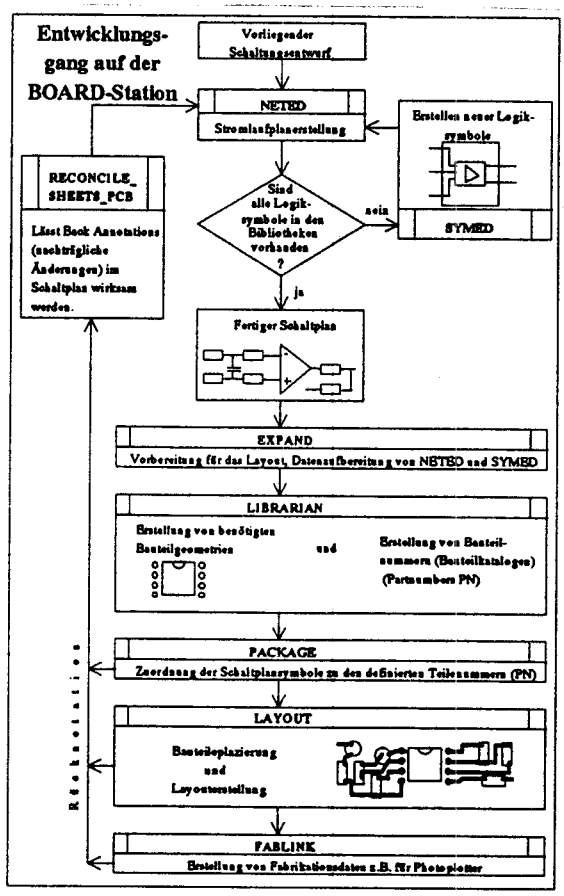


Abb.2-1: Entwicklungsgang auf der BOARD-Station

6.4

3. Erfahrung mit den einzelnen Programmodulen

3.1. NETED/SYMED

Der erste Schritt für den Layoutentwickler ist die Eingabe seines Stromlaufplanes in den Schaltplaneditor NETED. Zur Schaltplanerstellung stehen dem Entwickler im sogenannten STATIC-Menü in

55 Libraries ca. 1800 Symbole für Bauteile

zur Verfügung. Ist man sich jedoch der Vielzahl real existierender Bauteile bewußt, so kann man sich vorstellen, daß gerade bei größeren Schaltplaneingaben viele Symbole neu erstellt bzw. bestehende Symbole abgeändert werden müssen. Hierzu steht dem Anwender der Symboleditor SYMED zur Verfügung. Die neu erstellten Symbole speichert man in ein Unterverzeichnis z.B. *symbols_pcb* ab, aus dem sie jederzeit aufrufbar sind.
Für unserere Projekte mußten trotz der angebotenen Symbole eine recht hohe Anzahl eigener Symbole erzeugt werden:

Projekt 1:	Analogteil:	33 neue Symbole
	Digitalteil:	42 neue Symbole
Projekt 2:		40 neue Symbole
Projekt 3:		47 neue Symbole

Ein gravierendes Problem von NETED ist, daß die **Einbindung eigener Symbole schlecht unterstützt** wird. Unter dem Menü PARTS —> PARTS BY NAME im EDIT-Fenster kann das entsprechende selbsterstellte Symbol unter Angabe seines gesamten Pfades ausgewählt werden:

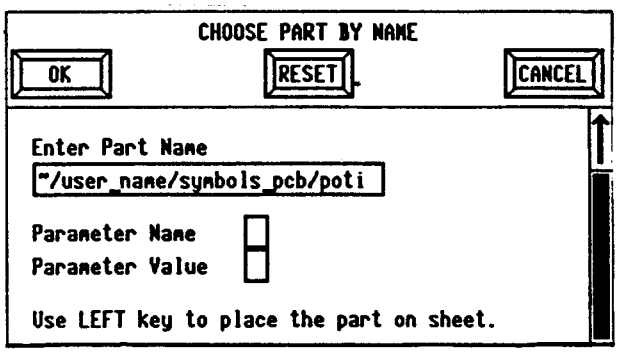


Abb.3-1: Aufruf eines eigenen Symbols unter NETED

Die Nachteile liegen auf der Hand:

- zeitaufwendige Arbeit,
- komplizierte Prozedur für viele selbsterstellte Symbole,
- man muß sich zu lange Pfadnamen merken.

Die Lösung dieses Problems haben wir durch die Erstellung einer eigenen Library gefunden, die nach Aufruf von NETED wie die anderen Standard-Libaries im STATIC-Menü erscheint. Als Beispiel ist hier die Erstellung der *fho_lib* dargestellt. Ein Symbol Name "Potentiometer" soll in der Library als "Poti" auswählbar sein. Die entsprechenden Befehle müssen in NETED in der Command-Line eingegeben werden:

1. Voraussetzung:

Das Symbol ist mit SYMED erstellt worden und liegt in einem Unterverzeichnis z.B. symbols_pcb.

2. Definition der betreffenden Library:

DEFine MEnu "fho_lib"

3. Definition des Symboles, das unter der Library stehen soll (Poti) und Angabe, wo das Symbol zu finden ist (symbols_pcb):

DEFine Item "fho_lib/poti" "transcript off; act comp ~/user_name/symbols_pcb/potentiometer -Hidden -NOUNdo

4. Schreiben der Library bzw. eines compilierten Binärfiles:

WRite MEnu fho_menu -replace

Um nun die erzeugte Library auch nach Aufruf von NETED im STATIC-Menü erscheinen zu lassen, muß nur noch die Startup-Datei für NETED (neted.startup3) abgeändert werden, wie dies in Abb.3-2 dargestellt ist. Dieses Startup-File muß sich unter dem Userspezifischen Verzeichnis *user_data* befinden und enthält schließlich die zusätzliche Befehlszeile:

READ MEnu fho_menu

Der Aufruf von NETED erfolgt schließlich mit:

NETED *filename* -st

In NETED müssen zudem weitere Informationen an Hand von sogenannten Properties für die spätere Bearbeitung der Schaltung eingebracht werden. Dies sind z.B. elektr.Daten, Pinbelegung, Symbolname usw.. Eine Vergabe bzw. Abänderung solcher Properties ist durch eine gute Menü-Führung in NETED leicht möglich. Eine Erläuterung, der für die Erstellung eines Layoutes benötigten Properties, ist in dem "IDEA Series Properties Reference Manual" übersichtlich dargestellt.

```
#-------------------------------------------------------------------------
#
# FILE:    neted.startup3
#
# SOURCE:  -/elke/user_data
#
# HISTORY: 17/10/90  FHO  Reorganized in preparation for v6.1
#
#-------------------------------------------------------------------------

IF ^($Easy_Interface = $TRUE) THEN
  EXecute /idea/sys/hi/startup/easy_neted.startup  ^$arg_1 ^$arg_2 ^$arg_3 ^$arg_4 ^$arg_5  ^$arg_6 ^$arg_7
^$arg_8 ^$arg_9   # compiled version
ELSE
  DO /idea/sys/hi/startup/original_neted.startup ^$arg_1 ^$arg_2 ^$arg_3 ^$arg_4 ^$arg_5 ^$arg_6 ^$arg_7
^$arg_8 ^$arg_9
END IF


# Eigenes STATIC-MENU einlesen und anzeigen
# ----------------------------------------
READ MEnu fho_menu
```

Abb.3-2: Neues Startup-File für NETED

3.2. EXPAND

Das Programm EXPAND, das die Datenstrukturen von NETED und SYMED für die weitere Bearbeitung aufbereitet, erzeugt nach Aufruf von EXPAND_PCB automatisch ein Design-File *pcb_design.erel*. Dieses Programm ist unproblematisch, setzt aber voraus, daß unter NETED keine Fehler gemacht wurden. Falls also ein EXPAND nicht richtig durchgeführt wird, so liegt die Ursache nur an Fehlern in NETED.

3.3. LIBRARIAN

Einer der recht zeitaufwendigsten und abstraktesten Programme ist LIBRARIAN.

3.3.1 Aufgaben von Librarian

a) Benötigte Bauteil-Geometrien aus vorhandenen Bibliotheken in eine eigene Bibliothek zu kopieren bzw. **abzuspeichern.** Hierzu stehen dem Layoutentwickler von MENTOR, wie auch schon in NETED, unter dem Menü LIBRARIES —> VIEW PARTS LIBRARIES —> MENTOR

7 Bibliotheken mit ca. 110 Bauteilgeometrien

zur Verfügung.

Der Anwender sollte sich jedoch nicht zu früh über diese recht große Anzahl von Bauteilgeometrien freuen. Wir mußten bei unseren Entwürfen immer wieder feststellen, daß man die vorhandenen Geometrien mit den Geometrien des realen Bausteines vergleichen sollte. Viele vorhandenen Geometrien stimmen z.B. im Rastermaß nicht mit denen des Datenblattes überein.

Beispiel: Die Geometrie *jedec_sq_84* (PLCC84-Gehäuse) liegt nicht im Zoll-Raster

Oder die Nummerierung der Pins der vorhandenen Geometrien stimmt nicht mit der Zählweise des Herstellers überein.

Beispiel: Die Geometrie *jedec_rec_32* weicht von der Nummerierung mit der Zählweise eines INTEL-Flash-Speicher ab.

Bei der Übernahme von vorhandenen Bauteilgeometrien sollte also immer ein Vergleich mit dem Datenblatt des betreffenden Bauteiles erfolgen !!!

b) Erstellung eigener Bauteilgeometrien, Pad´s und Via´s. Dies ist mit dem Menü LIBRARIES —> CREATE COMPONENT —> GENERIC COMPONENT möglich.

c) Erstellung eines Bauteilekataloges, d.h. man ordnet den Logiksymbolen aus dem Schaltplan ein oder mehrere Bauteilgeometrien zu. Dies läßt sich auch mit dem Menü LIBRARIES —> CREATE PART NUMBER relativ leicht realisieren, wie dies in Abb.3-3 gezeigt ist.

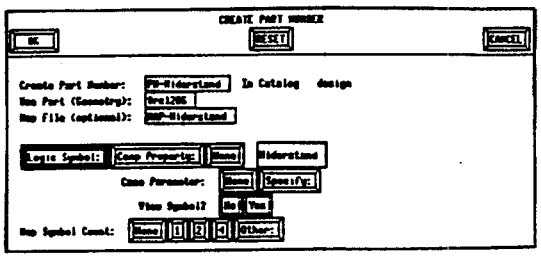


Abb.3-3: Erstellung eines Bauteilekataloges

6.6

Was hier als besonders Positiv auffiel, war die Möglichkeit, einem Gehäuse mehrere unterschiedliche Symbole aus NETED zuzuordnen. Dies bezeichnet man als sogenanntes "Inhomogenes Zusammenpacken" bzw. als "Non-homogeneous Packaging" (Abb.3-4). Diese Möglichkeit mußten wir bei dem Analogteil des Projektes 1 nutzen. Hier wurden für den verwendeten ASIC vier verschiedene Symbole erstellt, da auf dem ASIC ein ZF-Verstärker, drei verschiedene Multiplizierer und ein Addierer integriert waren.
An Hand dieser Option des inhomogenen Zusammenpackens kann man in NETED/SYMED mehrere Symbole für einen Baustein erstellen, um im Schaltplan die Übersicht zu wahren. Wenn man daran denkt, daß in zunehmendem Maße LCA's, ASIC's usw. in Schaltungen eingesetzt werden, erweist sich dies als vorteilhaft.

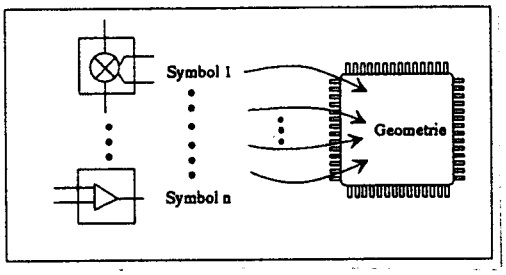


Abb.3-4: Inhomogenes Zusammenpacken.

d) Erstellung des Leiterplattenboardes.

3.3.2. Gründe für die Komplexität von LIBRARIAN

a) Aufwendige Struktur des Programmes durch viele verschiedene Menüs, die gerade für den Einsteiger zur Verwirrung führen.

b) Es müssen viele Informationen für die spätere Bearbeitung unter PACKAGE, LAYOUT und FABLINK eingebracht werden:

Diese Informationen vergibt man mit sogenannten Attributen, die entweder zwingend oder optional sind. Hier gibt es genau **74 verschiedene Attribute.** Auf die richtige Vergabe der Attribute muß geachtet werden, da einige in der späteren Bearbeitung des Designs nicht mehr abänderbar sind, andere wiederum in der späteren Bearbeitung wesentlich einfacher zu vergeben sind.

Mit einem Attribut kann man folgende Dinge festlegen:

- Plazierungs-Eigenschaften
- Routing-Eigenschaften
- Eigenschaften für eine thermische Analyse

Man unterscheidet die Attribute in

- BOARD-Attribute, die sich auf die Leiterplatte beziehen.
- Component-Attribute, die sich auf die Bauteilgeometrien beziehen.
- Pin-Padstack-Attribute, die sich auf alle Arten von Pad's beziehen.
- Via-Padstack-Attribute, die sich auf alle Arten von VIA's beziehen.

Diese Attribute unterteilt man wiederum in Graphical-Attributes und Non-Grapical-Atributes. Ein Graphical Attribute ist in der Zeichnung des Bauteiles oder des Boardes direkt sichtbar. Ein Non-Graphical-Attribute wird zusätzlich vergeben, ist jedoch nicht aus der Zeichnung ersichtlich.

In Abb.3-5 ist die Erstellung einer Spulen-Geometrie und ihrer Attribute dargestellt. Bei der Vergabe von Component-Attributes sollte man nun folgendes beachten:
Eine Bauteilgeometrie besitzt 2 zwingende Attribute und je nach Anwendung 16 optionale Attribute. Die unbedingt erforderlichen Attribute sind Graphical-Attributes und werden beim Zeichnen der Geometrie angelegt:

- Pin-Definition
  (Component_Pin_Definition: Attr.3+4)
- Bauteile-Umrandung
  (Component_Placement_Definition: Attr.5)

Hierbei tritt folgendes Problem auf:

Optionale Attribute von Components lassen sich in der späteren Bearbeitung in LAYOUT nicht mehr abändern, obwohl die meisten von ihnen dort erst wirksam werden.
Folge: Vergessene Attribute in LIBRARIAN bedeuten einen großen Zeitaufwand für den Layoutentwickler, da er wieder in LIBRARIAN zurück und die Bauteile erneut aufrufen und bearbeiten muß. Besonders nachteilig ist dies natürlich bei vielen selbsterzeugten Geometrien.

Beispiel:
Bei den Layoutentwürfen der Fachhochschule entschloss man sich bei den meisten Platinen eine beidseitige Plazierung der Bauteile durchzuführen. In LIBRARIAN läßt sich nun dafür das optionale Attribut COMPONENT_LAYOUT_SURFACE 'BOTH' vergeben (Attr.9 in Abb.3-5), was dem Bauteil die Berechtigung gibt, auf beiden Seiten einer Platine plaziert zu werden. Vergißt man dieses Attribut, so lassen sich die Bauteile nur auf einer Seite der Platine plazieren.

Deshalb kann hier die Empfehlung gegeben werden, sich alle optionalen Attribute der Bauteilgeometrien genau zu betrachten, um auftretende Probleme in LAYOUT zu vermeiden.

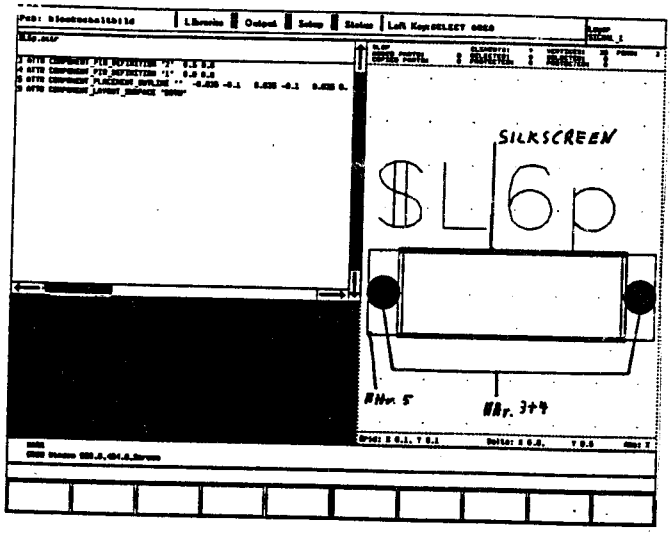


Abb.3-5: Erstellung einer Spulen-Geometrie

Bei der Vergabe von Board-Attributen kann man 12 zwingende und 24 optionale Attribute vergeben. Im Gegensatz zu den Component-Attributes sind die meißten optionalen Attribute eines Boardes in LAYOUT abänderbar, da diese im Wesentlichen auf den Routing-Vorgang bzw. die Design-Rules für den Routing-Prozeß bezogen sind. So sollte man auf die Vergabe von optionalen Attributen bei einem Board verzichten, da diese in LAYOUT wesentlich einfacher zu vergeben sind.

c) Die Fertigungsmöglichkeiten des Platinenherstellers müssen miteinbezogen werden:

Während der Layout-Entwicklung unserer Projekte sind drei gravierende Dinge aufgefallen, die sich erst nach der Fertigung bemerkbar machten:

- Man hat die Möglichkeit, in LIBRARIAN eine Siebdruckvorlage (SILKSCREEN) um die Bauteile zu zeichen (siehe Abb.3-5). Verwendet man auf der Platine hauptsächlich SMD-Bauteile, so muß man hier auf Folgendes achten: Überdeckt der Positionsdruck SMD-Pads, so lassen sich die SMD-Bauteile nur noch von Hand anlöten, ein Reflowlöten der Bauteile ist nicht mehr möglich, da der Siebdrucklack während des Lötvorganges nicht verdrängt wird und somit eine gute Lötung der SMD-Bauteile nicht mehr gewährleistet ist (siehe Abb.3-6). Diese Erfahrung mußten wir bei 2 Platinen machen. Bei der Digitalplatine von Projekt 1 mußte während der Fertigungszeit nochmals eine völlig neue Siebdruckvorlage erstellt werden.

Bei der Analogplatine von Projekt 1 wurde die Siebdruckvorlage etwas zu dick erstellt und wurde der Siebdruck während der Fertigung leicht versetzt, so daß zahlreiche SMD-Pads von dem Siebdruck überdeckt waren. Diesen Siebdruck konnte man nur noch durch vorsichtiges Abkratzen entfernen.

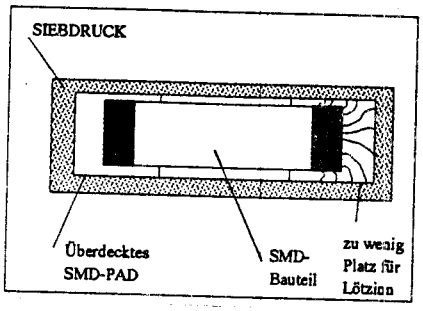


Abb.3-6: Schlechter Siebdruck bei SMD-Pad's

- Bei der Erstellung der Lötstopmaske (SOLDER_MASK) von Pad's und Via's sollte man auf folgendes achten:

Die Lötstopmaske wird wie in Abb.3-7.a dargestellt etwas größer als das eigentliche Pad bzw. Via gewählt, da man davon ausgeht, daß die Positionierung der Lötstopmaske nicht 100% genau ist. Hier muß jedoch darauf geachtet werden, daß die Lötstopmaske nicht zu groß gewählt wird, wie das z.B. bei unserem dritten Projekt passiert ist. Dadurch wurden an die Pad's grenzende Leiterbahnen freigelegt (Abb.3-7.b), was schließlich beim Löten zu möglichen Kurzschlüssen führen kann, gerade wenn minimale Abstände von Leiterbahn zu Pad gewählt wurden.

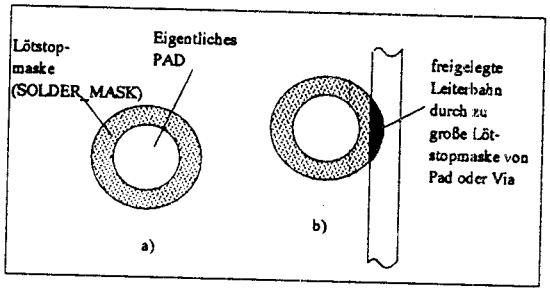


Abb.3-7: Lötstopmaske bei Pad's und Via's

- Bevor man an die Definition der Leiterplatte geht sollte man wissen, wieviele Lagen die Platine haben soll.

Mit der BOARD-Station ist eine Definition beliebiger Anzahl von Lagen möglich. Es gibt jedoch viele Firmen, die aus fertigungstechnischen Gründen nur geradzahlige Multilayerplatinen fertigen können. Ein Beispiel hierfür ist die Erstellung der Analogplatine aus Projekt 1. Diese war als 5-lagige Platine konzipiert worden. Gefertigt werden konnten jedoch entweder nur 4 oder 6 Lagen. Deshalb mußte bei dieser Platine eine Lage verdoppelt werden.

Daraus ergaben sich folgende Nachteile:

- Größerer Kostenaufwand bei mehr Lagen.
- Wäre man gleich von 6 Lagen ausgegangen, so hätte man eine bessere Aufteilung der Lagen vornehmen können.

d) Es muß bei der Erstellung von Geometrien und Board's die richtige Einstellung des Grids gewählt werden:

Die Einstellung des Grids hat später Auswirkung bei der Plazierung der Bauteile und beim Routen im Programm LAYOUT.
Bei der Einstellung des richtigen Grids sollte man sich an der Praxis orientieren. Die meißten Bauteile sind in INCH-Maßen gefertigt. Das Standard Zoll-Raster beträgt 0,1" = 2,54 mm (z.B. Abstände bei den Pins an einem IC).
Grundsätzlich kann man sagen, daß die Einstellung des Grids zur Erstellung der Bauteile ein Vielfaches des Standard-Zoll-Rasters betragen sollte. Ein Beispiel für eine falsche Grideinstellung ist in Abb.3-8 dargestellt. Pin 1 liegt im Raster des Routing-Grids, Pin 2 liegt nicht mehr im Routing-Grid. Hier ist die richtige Anbindung des Pins beim Routing-Vorgang nicht mehr garantiert.

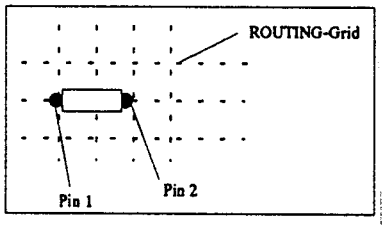


Abb.3-8: Falsche Grideinstellung

5. Die Erstellung von Leiterplatten, die eine abweichenden Form von der eines Rechteckes habe, wird von LIBRARIAN schlecht unterstüzt:

Die Definition eines Boardes ist recht einfach, wenn die Platine eine rechteckige Form hat, denn dann geschieht die Erstellung des Boardes mit dem Menü LIBRARIES —> CREATE NEW PART —> CREATE BOARD. Hier werden die für das BOARD zwingenden und die optionalen Attribute durch systematische Abfrage automatisch vergeben. Ebenso geschieht die Definition der Physikalischen Lagen automatisch. Sobald man jedoch ein nicht rechteckiges BOARD erzeugen möchte, muß man alles von Hand einstellen trotz der großen Anzahl vorhandener Menüs.

Zunächt sollte man beachten, daß ein Board kein Bauteil ist und deswegen während des Designs nur einmal vorkommen kann. Um überhaupt ein BOARD erstellen zu können muß man in die Command-Line den Befehl

Create BOARD

eingeben.

Ein Punkt aus dem man schließen kann,daß die BOARD-Station nicht dazu geschaffen ist, jedes Platinenformat zu definieren, war die Tatsache, daß die graphischen Attribute BOARD_PLACEMENT_OUTLINE (Gebiet, in dem Komponenten plaziert werden dürfen) und BOARD_ROUTING_OUTLINE (Gebiet in dem geroutet werden darf) nur in Form von Polygonen mit 90°- oder 45°-Linien vergeben werden können. Dies machte vor allen Dingen bei der Erstellung der runden Platine aus Projekt 3 Schwierigkeiten (Abb.3-9). Hier mußte ein sehr feines Grid gewählt werden, um eine annähernd runde Placement_Outline bzw. Routing_Outline zu erstellen.

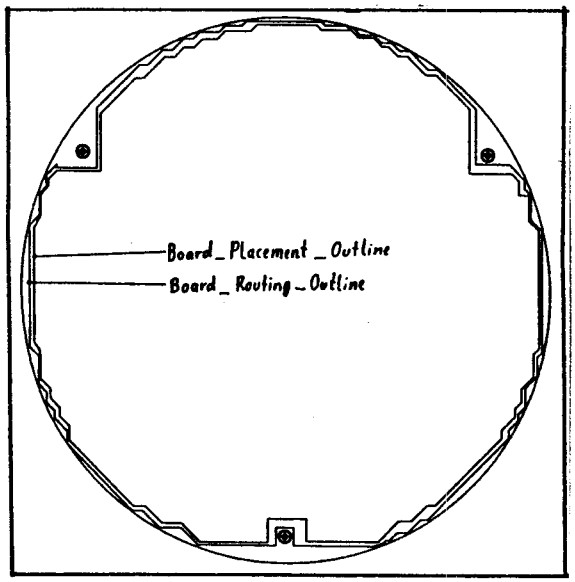


Abb.3-9: Erstellung einer runden Platine

Ein wichtiger Punkt ist die richtige Definition der einzelnen Lagen bei einer Platine, falls man das BOARD manuell erstellt. Dies geschieht mit dem Menü SETUP —> PHYSICAL LAYERS.
Man ordnet hier den sogenannten Physical-Layers (Physical_1 ... Physical_n) bestimmte Logical-Layers zu. Logical Layers sind die Lagen, die man sich auf dem Bildschirm anzeigen lassen kann, z.B. Signallage 1..n, Padlage 1 und 2 , Power-Lagen 1..n, Board_Outline usw..Insgesamt kann der Anwender hier **99 verschiedene Lagen** definieren.
Ein Problem, auf das wir während des Entwurfes der Leiterplatte bei Projekt 2 gestoßen sind, ist, daß die Definition der Physikalischen Lagen nicht den realen physikalischen Lagen entspricht. In Abb.3-10 ist die Definition der Physical-Layers am Beispiel der Platine

aus Projekt 2 erklärt. Erstellt werden sollte eine 6-lagige Multilayerplatine, wobei die Anordnung der Lagen entsprechend der Abb.3-10 vorgenommen werden sollte. Bei den mittleren Lagen wollte man eine sogenannte SPLITTING POWER PLANE durchführen, daß heißt die Power-Lagen nicht nur mit einerSpannung zu belegen, sondern aus zwei Spannungen zu kombinieren, in diesem Beispiel pos12V und VCC bzw. GND und GNDa.

Bei der Definition der Physical-Layers kann man nun beiden äußeren Lagen mehr als ein Logical-Layer zuordnen. Bei der Definition der inneren Lagen darf man nur einen Logical-Layer vergeben. D.h. man kann also nicht folgende Definition durchführen:

Physical_3 = "Power_1" "Power_2"

Deshalb muß man hier mehr physikalische Lagen definieren, als in Wirklichkeit vorhanden sind. Darauf sollte man bei der Definition der Physical Layers unbedingt achten.

3.4. PACKAGE

PACKAGE ist wiederum eines der Programmodule, das ohne größere Probleme zu überwinden ist. Hier wird allen vorhandenen Logiksymbolen mit der Funktion BUILD eine Geometrie aus dem Bauteilekatalog zugewiesen.
Aber auch hier kann gerade der Anfänger Fehler machen. Wie man in Abb.3-11 sieht, kann man in Package nicht nur Geometrien zuordnen, sondern auch die Plazierung (Spalte LOCATION) auf dem BOARD abändern.
Das Problem ist, daß in LAYOUT alle Bauteile plaziert werden, ein BACK-ANNOTATION gemacht wird und man feststellt, daß dem einen oder anderen Bauteil nochmals eine andere Geometrie gegeben werden muß.

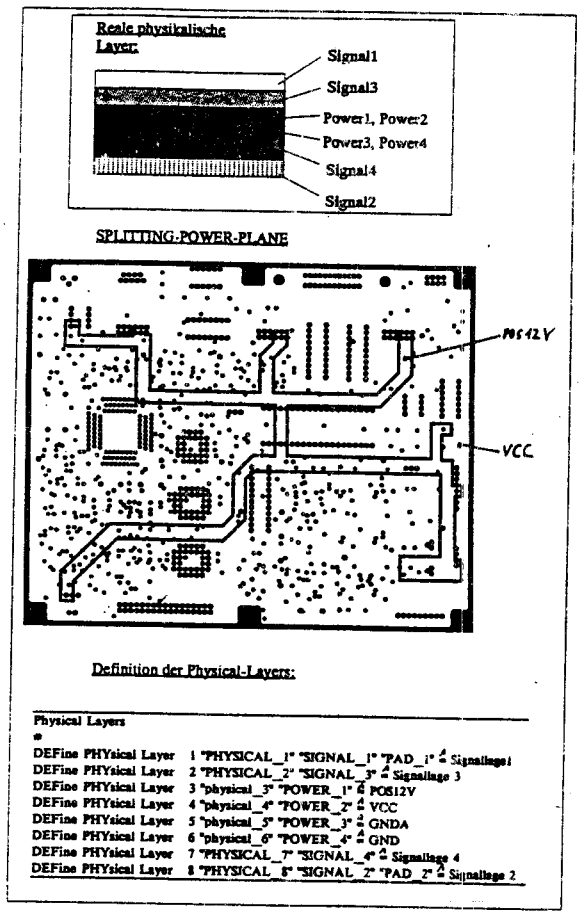


Abb3-10: Definition der Physical-Layer

Wenn der Anwender bei diesem Schritt nicht zuerst in NETED geht und ein anschließendes EXPAND durchführt ist trotz Back-Annotation nach einem erneuten Build in PACKAGE die ganze Plazierung verloren, denn eine **Back-Annotation wird nur durch Aufruf von NETED** wirksam. D.h. bei aufwendigen Layoutentwürfen sind möglicherweise mehrere Tage Arbeit zunichte geworden.

PACKAGE EDIT SYMBOL

LINE	FLG	SCHEMATIC SYMBOL	GEOMETRY	MAPPING	REFERENCE	INSTANCE	PART NUMBER	LOCATION	SYMBOL PROPERTY VALUES
1	A	CAPACITOR	$rc0805	MAP-C0805	C1	/I$121/C1	PN-C0805	64.77 66.04 2 180 MM 6477000 6604000	(
2	A	CAPACITOR	$rc0805	MAP-C0805	C2	/I$121/C2	PN-C0805	64.77 62.865 2 180 MM 6477000 6286500	
3	A	CAPACITOR	$rc1206	MAP-C1206	C3	/I$121/C3	PN-C1206	64.77 59.69 2 180 MM 6477000 5969000	(
4	A	CAPACITOR	$rc1206	MAP-C1206	C4	/I$121/C4	PN-C1206	49.53 52.705 1 270 MM 4953000 5270500	
5	A	CAPACITOR	$rc1206	MAP-C1206	C5	/I$121/C5	PN-C1206	74.93 52.705 2 90 MM 7493000 5270500	
6	A	CAPACITOR	$rc1206	MAP-C1206	C6	/I$121/C6	PN-C1206	86.36 58.165 1 0 MM 8636000 5816500	
7	A	CAPACITOR	$rc1206	MAP-C1206	C7	/I$121/C7	PN-C1206	86.36 43.815 1 0 MM 8636000 4381500	
8	A	CAPACITOR	$rc1206	MAP-C1206	C8	/I$121/C8	PN-C1206	86.36 40.64 1 0 MM 8636000 4064000	(
9	A	CAPACITOR	$rc1206	MAP-C1206	C9	/I$22/C9	PN-C1206	95.885 43.815 1 0 MM 9588500 4381500	

Abb.3-11: PACKAGE-Möglichkeiten

3.5. LAYOUT

Im Programm LAYOUT wird schließlich das eigentliche LAYOUT der Leiterplatte erstellt. Der Entwickler kann sein Layout interaktiv oder per AUTO-Router entwerfen. Bei größeren Entwürfen sollte man nur mit dem Autorouter arbeiten. Ein interaktives Routing ist zu aufwendig.

Die Probleme in Zusammenhang mit dem Programm LAYOUT liegen vor allen Dingen in einer sinnvollen Plazierung der Bauteile , und der Wahl der Voreinstellungen, damit ein reibungsloser Routing-Prozeß abläuft.

a) Plazieren der Bauteile:

Hier gibt es zwei Möglichkeiten:

- Automatisches Plazieren der Bauteile:

Bei allen drei Projekten erwies sich dies jedoch nicht als sinnvoll, da z.B.
- alle Widerstände, IC´s usw. Gruppenweise plaziert wurden.
- eine viel zu enge Plazierung der Bauteile vorgenommen wurde.
- eine funktional schlechte Plazierung durchgeführt wurde.

- Interaktives Plazieren der Bauteile:

Hierbei sollte man vor allen Dingen auf die Plazierungsdichte der Bauteile achten. Ab einer Plazierungsdichte von ca. 60% kann der Router keine Lösungen mehr finden. Die Plazierungsdichte kann man sich mit einem Histogramm anzeigen lassen. Abb.3-12 zeigt als Beispiel die Digital-Platine von Projekt 1. Die Plazierung nimmt einen recht großen Zeitraum in Anspruch.

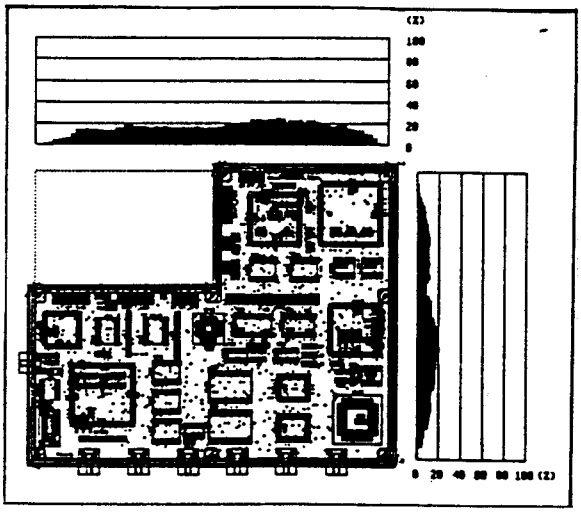


Abb.3-12: Histogramm für Plazierungsdichte

b) Grideinstellung:

Es gibt drei verschiedene Gridarten in LAYOUT einzustellen:

a) DISPLAY-Grid:

> Grideinteilung auf dem Bildschirm; kann beliebig gewählt werden.

b) PLACEMENT-Grid:

> Grid auf dem die Bauteile beim Plazieren einrasten. Sollte genauso groß wie das ROUTING-Grid eingestellt werden, damit gewährleistet ist, daß die Pins auf dem Routing Grid liegen und somit ein exaktes Routen stattfindet.

c) ROUTING-Grid:

> Aufgeteilt in Wire-Grid, Via-Grid und Pad-Grid. Grid auf dem geroutet wird.

Aus Erfahrungswerten während der Entwürfe hat sich folgende Routinggrideinstellung als günstig erwiesen:

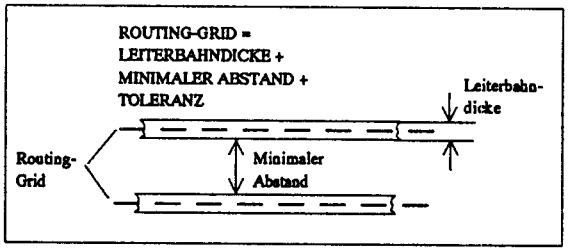


Abb.3-13: ROUTING-Grid-Einstellung

Im LAYOUT-User-Manual ist eine Formel angegeben, die zudem Pad- und Viagröße zur Berechnung des Routing-Grids miteinbezieht. Dies erwies sich jedoch als nicht so günstig.

Ist das Board in Librarian im mm-Maßstab gefertigt worden, so werden hier alle spezifischen Werte für die Grideinstellung auch in mm eingegeben. Aber auch hier gilt wiederum, nur Werte als Vielfaches von 0,1" bzw. 2,54 mm anzugeben.

6.11

c) Einstellung günstiger Design-Rules

Die Design-Rules für das Routen müssen mit dem Menü AUTO —> SETUP ROUTER eingegeben werden. Hier werden folgende Dinge festgelegt:

- Leitungsdicke (Wire-Width)
- Min. Abstände zwischen Leitungen, Leitungen und Vias (Track-Clearance)
- Min. Anstände zw. Pads, Pads und Vias, und Pads und Leitungen (Pad-Clearance)
- Benutzung von VIA´s
- Benutzung von T-Junctions beim Routen
- Diagonales Routen
- Bevorzugte Routing-Richtung auf den Signallagen (horizontal oder waagerecht)
- Nach welchen Gesichtspunkten geroutet werden soll, z.B. nach dem Gesichtspunkt erst der "kürzesten Verbindungen" zu routen.

Bevor man die Design-Rules einstellt sollte man sich über die Herstellmöglichkeiten der Firma klar sein, d.h. wie dünn dürfen Leiterbahnen sein, wie groß müssen minimale Abstände sein usw..
Z.B. wurden bei der Digitalplatine von Projekt 1 Leitungsdicken von 0,254mm eingestellt. Bei der Analogplatine von Projekt 1 wurden minimale Abstände von 0,212 mm gewählt. Hiermit war man schon an der Grenze der Fertigungsmöglichkeit einer Firma.

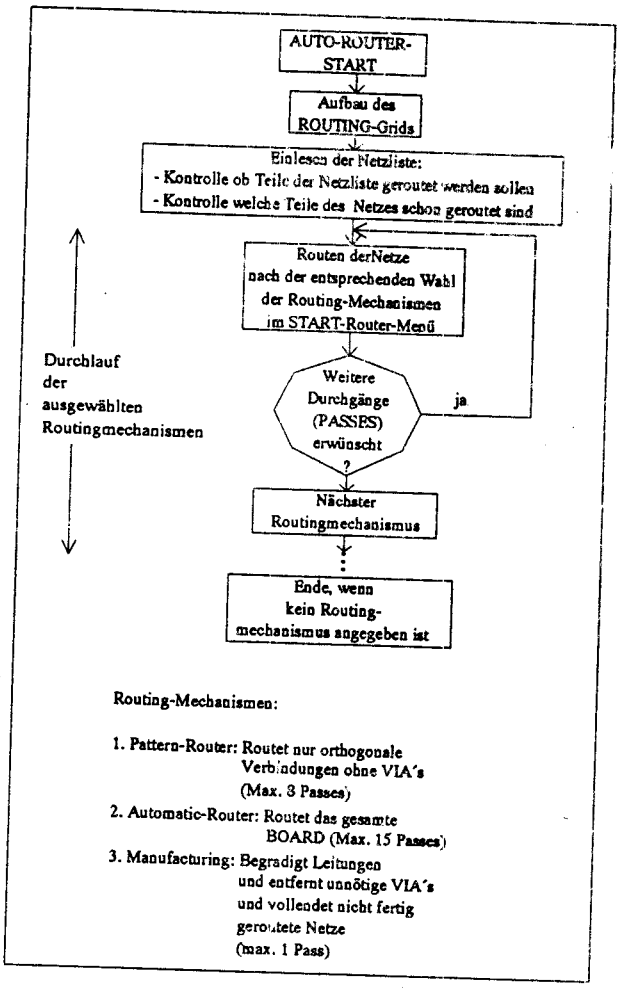
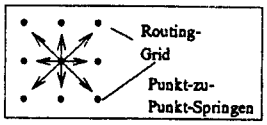


Abb.3-14: Ablauf des Autoroutings

d) Richtige Einstellung des Routers

Der hier verwendete Router ist ein MAZE-Runner, der von Grid-Punkt zu Grid-Punkt springt und nach der kürzesten Verbindung zwischen Anfangs- und Endpunkt eines Netzes sucht.



Der Ablauf des AUTO-Routings ist in Abb.3-14 dargestellt. Dem Layoutentwickler wird hier die sinnvolle Auswahl der 3 verschiedenen Routing-Mechanismen überlassen. Wichtig ist auch die Auswahl der Anzahl der Durchgänge (PASSES) der Routing-Mechanismen. Hier sollte man die max. Anzahl der Passes gerade bei komplexen Layoutentwürfen nutzen. Die verschiedenen Routing-Mechanismen kann man übrigends auch mehr als einmal aufrufen.

e) Vorrouten bestimmter Netze

Bestimmte Netze sollte man sich vorrouten lassen, so z.B. die Versorgungslagen der Platinen GND, VCC usw., da der Router dann direkt von einem Pin an dem betreffenden Netz mit einem VIA nach unten auf die Versorgungslage geht. Lässt man alle Netze aufeinmal routen, so verbaut sich der Router selbst Wege durch zu lange Leitungsführung bei den Netzen, die mit einem VIA direkt an eine Lage angebunden werden könnten.

Bei der richtigen Einstellung der Gridarten, der Design-Rules und des Routers hat man im Routing-Durchlauf sehr gravierende Unterschiede bemerkt:
Im Allgemeinen benötigt der Auto-Router für solch aufwendige Layoutentwürfe wie unsere **ca.2-4 Stunden Zeit zum Routen.**
Bei falschen Voreinstellungen benötigt der Router einen **Zeitraum von ca. 5 Stunden** und weist danach immer noch **offene Netze** auf.
Nachträglich mußte man bei den Layoutentwürfen der Fachhochschule keine manuelle Bearbeitung durchgeführt werden; das Routing-Ergebnis war sehr gut.

3.5. FABLINK

In Fablink werden die Fabrikationsdaten für die Fertigung der Platine erstellt:

1. Die Konfiguration zur Erzeugung der Photoplotdaten im Gerber-Format und die Erstellung der Blendetabellen (Aperture table) für den Blendenteller des Photoplotters.
2. Die Konfiguration für Fräsdaten im Excellon- bzw. Sieb & Meyer-Format
3. Die Konfiguration für Bohrdaten im Excellon- bzw. Sieb & Meyer-Format

Dies geschieht in FABLINK weitestgehend automatisch.

Bevor man die Vorlagen für die Photoplots erstellt (Artworks), muß man diese definieren. Wie auch schon bei den Physical-Layers weist man den Artworks verschiedene Logical-Layers zu, die sich auf den Photoplots befinden sollen.
Ein Beispiel für die Erstellung von Photoplot-Definitionen ist in Abb.3-15 dargestellt.

ARTWORK LAYER	ARTWORK FILE	LOGICAL LAYERS	
1	artwork_1	signal_1,pad_1,placement_keepout	} Signallagen
2	artwork_2	signal_2,pad_2,placement_keepout	
3	artwork_3	silkscreen_1,board_outline	} SIEBDRUCK
4	artwork_4	silkscreen_2,board_outline	
5	artwork_5	power_1,board_outline	} Versorgungslagen
6	artwork_6	power_2,board_outline	
7	artwork_7	power_3,board_outline	
8	artwork_8	solder_mask_1,board_outline	} Lötstopmasken
9	artwork_9	solder_mask_2,board_outline	
10	artwork_10	paste_mask_1,placement_keepout	} Klebemasken für SMD-Bauteile
11	artwork_11	paste_mask_2,placement_keepout	

Abb.3-15: Definition der Photoplotvorlagen

Bevor man nun die Photoplotdaten erstellt, sollte man sich die Definition der Blendetabellen für den Photoplotter betrachten. Abb.3-16 stellt den Auszug aus der Definition einer Blendetabelle dar.

Es gibt kreisförmige (Circel) oder rechteckförmige (Rectangular) Blenden. Zum Einen gibt es Blenden, die blitzartig auf- und zugemacht werden (Flash) und somit dafür sorgen, daß es unterschiedliche Pad´s oder Via´s gibt. Zum Anderen gibt es Blenden, die über den Film gezogen werden (Trace) und somit Linien, Text oder Leiterbahnen erzeugen. Verfügt eine Firma über eine Laserplotter, so ist man in der Zusammenstellung der Blenden vollkommen uneingeschränkt, d.h. man kann jede Art von Blende an jeder Blendentellerposition erzeugen. Wird kein Laserplotter verwendet, so ist man auf die Firmenüblichen vordefinierten Blendenteller angewiesen, und muß sich mit seinem Layout daran orientieren.

In Bezug auf die spätere Fertigung sollte man sich die Einstellung der Blenden etwas genauer betrachten. Benutzt man z.B. für den Siebdruck eine zu große Blende, so entsteht ein unschöner, breiter Druckstrich mit Gefahr des Verlaufens.

# DEFine	APerture	Position	Shape	Type	Height(Y)/Diameter	Width(X)	Kommentar: zugehöriges Symbol
DEFine	APerture	1	Circle	Trace	0.254000	0.000000	Leiterbahnen,Silkscreen(Umrisse)
DEFine	APerture	2	Circle	Flash	1.000000	0.000000	Via1(Signallage)
DEFine	APerture	3	Circle	Flash	1.500000	0.000000	Via1(Power-Antip.,Lötst.),Pad1(Sign.)
DEFine	APerture	4	Circle	Flash	1.100000	0.000000	Pad_taste (Signal)
DEFine	APerture	5	Circle	Flash	2.600000	0.000000	Befestigungslöcher für Platine
DEFine	APerture	6	Circle	Trace	0.127000	0.000000	Board_Outline,Silkscreen(Text)
DEFine	APerture	7	Rectangular	Flash	2.032000	0.635000	SMD-Pad 80x25(Signal,Paste-Mask)

Abb.3-16: Auszug einer Blendentellerdefinition für den Photoplotter

6.73

4. Entwicklungszeiten

In Tabelle 4.1 sind die Entwicklungszeiten der einzelnen Projekte dargestellt.

Tabelle 4.1: Entwicklungszeiten

	Reine Entwicklungszeit
Projekt 1	
Digitalteil	ca. 8 Wochen
Analogteil	ca. 6-8 Wochen
Projekt 2	ca. 6 Wochen
Projekt 3	ca. 4-6 Wochen

Hierbei muß man erwähnen, daß die Projekte jeweils im Rahmen einer Diplomarbeit durchgeführt wurden, und die Diplomanden vorher noch nicht mit dem System und der BOARD-Station vertraut waren.

Die meiste Zeit bei der Layoutentwicklung mit der BOARD-Station mußte also in die Einarbeitung der verschiedenen Programme investiert werden.

Den Zeitraum, den die Arbeit an den einzelnen Programme in Anspruch nimmt, kann man nicht genau angeben, da dieser immer anwenderspezifisch ist. Denn ja nach Projekt müssen einmal mehr Geometrien erstellt oder ein anderes Mal ein komplexers Layout geroutet werden usw..

Sicherlich kann man jedoch sagen, daß man durch Berücksichtigung der gewonnenen Erfahrungen, den gesamten Layoutentwicklungsprozeß an der BOARD-Station verkürzen kann.