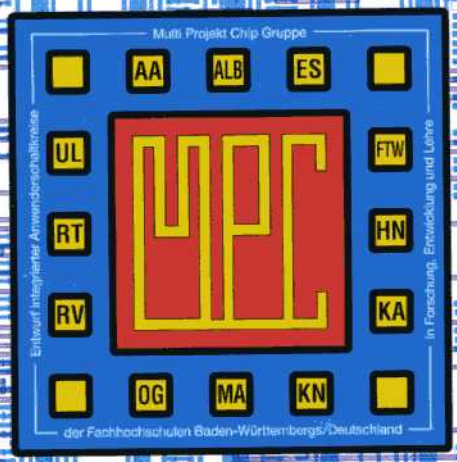
MULTIPROJEKT CHIP-GRUPPE

BADEN - WÜRTTEMBERG

Workshop Januar 1998

Albstadt-Sigmaringen



MULTIPROJEKT CHIP-GRUPPE

BADEN - WÜRTTEMBERG

Workshop Januar 1998

Albstadt-Sigmaringen

Herausgeber: Fachhochschule Ulm

© 1998 Fachhochschule Ulm

Das Werk und seine Teile sind urheberrechtlich geschützt. Jede Verwertung in anderen als den gesetzlich zugelassenen Fällen bedarf deshalb der vorherigen schriftlichen Einwilligung des Herausgebers Prof. A. Führer, Fachhochschule Ulm, Prittwitzstraße 10, 89075 Ulm.

Adressen der

MULTIPROJEKT-CHIP-GRUPPE (MPC-Gruppe)

BADEN - WÜRTTEMBERG

http://www.mpc.belwue.de

Fachhochschule Aalen
Prof. Dr. Kohlhammer, Postfach 1728, 73428 Aalen
Tel.: 07361/576-248, Fax: -324, Email: bernd.kohlhammer@fh-aalen.de

Fachhochschule Albstadt-Sigmaringen
Prof. Dr. Rieger, Johannesstr. 3, 72458 Albstadt-Ebingen
Tel.: 07431/579-124, Fax: -149, Email: rieger@fh-albsig.de

Fachhochschule Esslingen
Prof. Dr. Kampe, Flandernstr. 101, 73732 Esslingen
Tel.: 0711/397-4221, Fax: -4212, Email: kampe@ti.fht-esslingen.de

Fachhochschule Furtwangen
Prof. Dr. Rülling, Postfach 28, 78113 Furtwangen
Tel.: 07723/920-503, Fax: -610, Email: ruelling@fh-furtwangen.de

Fachhochschule Heilbronn
Prof. Dr. Clauss, Max-Planck-Str. 39, 74081 Heilbronn
Tel.: 07131/504-400, Fax: /252470, Email: clauss@fh-heilbronn.de

Fachhochschule Karlsruhe
Prof. Ritzert, Postfach 2440, 76012 Karlsruhe
Tel.: 0721/925-2238, Fax: -2259, Email: ritzert@fh-karlsruhe.de

Fachhochschule Konstanz
Prof. Dr. Voland, Postfach 100543, 78405 Konstanz
Tel.: 07531/983-631, Fax: -613, Email: voland@fh-konstanz.de

Fachhochschule Mannheim
Prof. Dr. Albert, Speyerer Str. 4, 68136 Mannheim
Tel.: 0621/2926-351, Fax: -454, Email: gerhard@eis.fht-mannheim.de

Fachhochschule Offenburg
Prof. Dr. Jansen, Badstr. 24, 77652 Offenburg
Tel.: 0781/205-267, Fax: -333, Email: d.jansen@fh-offenburg

Fachhochschule Ravensburg-Weingarten
Prof. Dr. Klotzbücher, Postfach 1261, 88241 Weingarten
Tel.: 0751/501-630, Fax: /49240, Email: klotzbuecher@fbe.fh-weingarten.de

Fachhochschule Reutlingen
Prof. Dr. Kreutzer, Federnseestr. 4, 72764 Reutlingen
Tel.: 07121/341-108, Fax: -100, Email: kreutzer@el.fh-reutlingen.de

Fachhochschule Ulm
Prof. Führer, Postfach 3860, 89028 Ulm
Tel.: 0731/502-8338, Fax: -8363, Email: fuehrer@fh-ulm.de

Inhaltsverzeichnis

Workshop-Vorträge

		Seite
1.	Ein Verfahren zur schnellen Berechnung der Sinus- und/oder Kosinusfunktion mit Hilfe eines Spezial-Chips H. Spiro, Böblingen, K. Najmann, Gärtringen	7
2.	Optimiertes Testen von analogen ICs mittels generischer ATE-Modelle und Testprogramm-Synthese J. Schuster, U. Jäger, FH Heilbronn	15
3.	Anwendung und Methoden der symbolischen Analyse beim Entwurf analoger Schaltungen V. Mutlu, G. Forster, FH Ulm	33
4.	Entwicklung einer Digitalschaltung unter VHDL zur Verarbeitung von MIDI-Signalen F. Messmer, H. Kreutzer, FH Reutlingen	47
5.	Entwicklung eines Einkanal-HDCL-Empfängers unter Verwendung der Hardwarebeschreibungssprache VHDL P. Kuppler, H. Kreutzer, FH Reutlingen	53
6.	Hochsprachen-Compiler für den FHOP O. Bischoff, D. Jansen, FH Offenburg	61
7.	Einfache Lösungen für Digital-Analog-Wandler und für Analog-Digital-Wandler M. Rieger, FH Albstadt Sigmaringen	69
8.	Selbsttest mit Hilfe der Signaturanalyse G. Stura, A. Führer, FH Ulm	75
9.	Entwicklung eines ASICs für Ethernet-Analysatoren J. Jesinger, FH Ravensburg-Weingarten	83

Firmenbesuche

10.	Teilnahme von MPC-Mitgliedern bei der CICC'97, Santa Clara, USA K. Schmidt, FH Furtwangen	93

Gefertigte Bausteine

	Seite
11. RDS-Empfänger 5. Semester Elektronik, F. Förster, W. Ludescher, FH Ravensburg-Weingarten	97
12. BiCMOS Allpass Delay-Line Continuous-Time Filter Testchip R. Koblitz, K. Schmidt, FH Furtwangen	98
13. tm_entprell A. Gauckler, W. Rülling, FH Furtwangen	99
14. Untersuchung und Weiterentwicklung von OP-Schaltungen für das Transistor-Array B500D J. Storch, H. Clauss, FH Heilbronn	100
15. Schwimmende Spannungsquelle G. Plappert, H. Clauss, FH Heilbronn	101
16. Empfänger-ASIC für ein Datenträgersystem K. Hartner, G. Forster, FH Ulm	102
17. Sender-ASIC eines Lichtszenenschalters A. Führer, FH Ulm	103
18. Thermologger V2 T. Klumpp, D. Jansen, FH Offenburg	104
19. ASIC-Solar A. Herb, B. Kohlhammer, FH Aalen	105
20. Steuerbaustein zur Diversityumschaltung in einem Funkmikrofonempfänger A. Lang, G. Busch, B. Kohlhammer, FH Aalen	106

Ein Verfahren zur schnellen Berechnung der Sinus- und/oder Kosinusfunktion mit Hilfe eines Spezial-Chips

Hans Spiro, Böblingen , **Knut Najmann**, Gärtringen

Durch einen speziellen Table Look-up, kombiniert mit stückweise linearer Approximation der Sinusfunktion (und/oder der Kosinusfunktion) können die mit hinreichender Genauigkeit vorzunehmenden Berechnungen des sin und/oder cos erheblich gegenüber denen mit herkömmlichen Verfahren beschleunigt werden. In diesem Aufsatz werden die Grundlagen des Verfahrens entwickelt und gezeigt, welche Komponenten das der schnellen sin- und/oder cos-Berechnung dienende Spezial-Chip mindestens enthalten muß.

1 Einige bekannte Verfahren und deren Nachteile

In Computern und hochfrequenztechnischen Geräten, in denen die Sinus- und/oder Kosinusfunktion benötigt wird, werden die Funktionen üblicherweise durch Reihenentwicklung berechnet. Beispielsweise kann man sich der "klassischen" Reihe

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots \qquad (1)$$

bedienen, die man bei einem i-ten Glied, für das

$$\left| \frac{x^{2i-1}}{(2i-1)!} \right| \leq \varepsilon \qquad (2)$$

gefunden wird, abbricht. Der Fehler gegenüber dem exakten Wert des Sinus ist dann kleiner oder höchstens gleich einer vorgegebenen Schwelle ε. Die Reihe (1) konvergiert sehr langsam, weshalb ***sehr viele*** rechenzeitaufwendige Divisionen (oder Multiplikationen mit $1/(2i-1)!$) und Additionen und/oder Subtraktionen benötigt werden.

Erheblich besser sind rascher konvergierende Reihen, wie z.B. das aus [1] entnommene und im *Bild 1* wiedergegebene Verfahren, das einen Fehler $\varepsilon \leq 2^{-28,1} \approx 3,5 \cdot 10^{-9}$ für die Sinus- und $\varepsilon \leq 2^{-24,57} \approx 4 \cdot 10^{-8}$ für die Kosinusfunktion garantiert. Da die Gleitkommazahlen bei der üblichen sogenannten "Single Precision" (= 32 bit) eine Mantissenlänge von 24 bit besitzen, ergibt sich für die Zahlendarstellung eine Genauigkeit von $\log_{10}(2^{24}) \approx 7,22$ Dezimalstellen. Folglich ist die Genauigkeit des im *Bild 1* dargestellten Verfahrens dem bestens angemessen. Jedoch benötigt auch dieses Verfahren (wie auch andere ähnlich aufgebaute) immer noch zu viele Multiplikationen, Additionen usw.

Abhilfe kann auf verschiedene Weise geschaffen werden:

- Man verwendet eine spezielle Arithmetikeinheit, die ohne zwischengeschaltete Hauptspeicherzugriffe sehr viele Additionen, Multiplikationen etc. in extrem kurzer Zeit hintereinander ausführen kann, weil Zwischenspeicherungen in Registern und/oder in sehr schnellen Caches vorgenommen werden. Dennoch bleibt die Zahl der arithmetischen Operationen viel zu hoch.

- Man speichert Tabellenwerte für die Sinusfunktion und ermittelt den aktuell gesuchten Funktionswert durch Table Look-up. Zwischenwerte werden durch lineare oder quadratische Interpolation zwischen den Stützpunkten ermittelt. Diese Methode ist aber in ihrer klassischen Form entweder recht ungenau oder erfordert sehr umfangreiche Tabellen, die entweder in einem Spezial-Chip abgespeichert sein müssen oder Hauptspeicherplatz beanspruchen. Ein Spezial-Chip für die Speicherung heranzuziehen, muß kein Nachteil sein; schließlich soll ja auch das in diesem Aufsatz erläuterte Verfahren zu einem Spezial-Chip führen. Jedoch wird zu viel Rechenzeit für den Table

1. Define $z = \frac{4}{\pi} \cdot |x|$ and separate z into its integer part (q) and its fraction part (r). Then $z = q + r$, and $|x| = \left(\frac{\pi}{4} \cdot q\right) + \left(\frac{\pi}{4} \cdot r\right)$.
2. If the cosine is desired, add 2 to q. If the sine is desired and if x is negative, add 4 to q. This adjustment of q reduces the general case to the computation of $\sin(x)$ for $x \geqq 0$ because
   $\cos(\pm x) = \sin\left(\frac{\pi}{2} + x\right)$, and
   $\sin(-x) = \sin(\pi + x)$.
3. Let $q_0 \equiv q \bmod 8$.
   Then, for $q_0 = 0, \sin(x) = \sin\left(\frac{\pi}{4} \cdot r\right)$,
   $q_0 = 1, \sin(x) = \cos\left(\frac{\pi}{4}(1 - r)\right)$,
   $q_0 = 2, \sin(x) = \cos\left(\frac{\pi}{4} \cdot r\right)$,
   $q_0 = 3, \sin(x) = \sin\left(\frac{\pi}{4}(1 - r)\right)$,
   $q_0 = 4, \sin(x) = -\sin\left(\frac{\pi}{4} \cdot r\right)$,
   $q_0 = 5, \sin(x) = -\cos\left(\frac{\pi}{4}(1 - r)\right)$,
   $q_0 = 6, \sin(x) = -\cos\left(\frac{\pi}{4} \cdot r\right)$,
   $q_0 = 7, \sin(x) = -\sin\left(\frac{\pi}{4}(1 - r)\right)$.
   These formulas reduce each case to the computation of either $\sin\left(\frac{\pi}{4} \cdot r_1\right)$ or $\cos\left(\frac{\pi}{4} \cdot r_1\right)$ where r_1 is either r or $(1 - r)$ and is within the range, $0 \leqq r_1 \leqq 1$.
4. If $\sin\left(\frac{\pi}{4} \cdot r_1\right)$ is needed, it is computed by a polynomial of the following form:
   $$\sin\left(\frac{\pi}{4} \cdot r_1\right) \cong r_1 (a_0 + a_1 r_1^2 + a_2 r_1^4 + a_3 r_1^6).$$
   The coefficients were obtained by interpolation at the roots of the Chebyshev polynomial of degree 4. The relative error is less than $2^{-28.1}$ for the range.
5. If $\cos\left(\frac{\pi}{4} \cdot r_1\right)$ is needed, it is computed by a polynomial of the following form:
   $$\cos\left(\frac{\pi}{4} \cdot r_1\right) \cong 1 + b_1 r_1^2 + b_2 r_1^4 + b_3 r_1^6.$$
   Coefficients were obtained by a variation of the minimax approximation which provides a partial rounding for the short precision computation. The absolute error of this approximation is less than $2^{-24.57}$.

Bild 1

Algorithmus zu Berechnung der Sinus- und der Kosinusfunktion laut [1]

Look-up und die lineare oder quadratische Interpolation zwischen zwei Stützpunkten (x_i, y_i) und (x_{i-1}, y_{i-1}) mit dem Abstand $\Delta x = x_i - x_{i-1}$, $i = 1, 2, \ldots n$ benötigt. Zusätzlich zu der für den Table Look-up verbrauchten Rechenzeit sind bei linearer Interpolation entsprechend

$$y = f(x, \Delta x, x_{i-1}, y_i, y_{i-1}) = y_{i-1} + \frac{x - x_{i-1}}{\Delta x} \cdot (y_i - y_{i-1}) \qquad (3)$$

immer noch 3 Additionen, eine Multiplikation und eine Division (bzw. Multiplikation mit $1/\Delta x$) durchzuführen. Wendet man quadratische Interpolation an, dann werden für die gleiche Genauigkeit zwar weniger Stützpunkte gebraucht, aber der Rechenaufwand ist gemäß

$$y = f(x, \Delta x, x_{i-1}, x_{i-2}, y_i, y_{i-1}, y_{i-2}) = \text{usw.} \qquad (4)$$

noch wesentlich größer, da (4) sogar mindestens 7 Additionen, 4 Multiplikationen und 2 Divi sionen erfordert.

- Man könnte ein Spezial-Chip bauen, das sowohl den nötigen Speicherplatz (z.B. in Form eines schnellen ROM) zur Speicherung der Tabellenwerte als auch eine schnelle Arithmetik für den Table Look-up und die Interpolationsrechnungen laut (3) oder (4) enthält. Dennoch, die Zahl der notwendigen Additionen und Multiplikationen wird dadurch nicht verringert. So besehen, stellt auch ein solches Chip noch nicht die optimale Lösung dar. Eine wesentliche Verbesserung meinen die Autoren erst durch das im folgenden Abschnitt 2 vorgestellte Verfahren zu erreichen.

2 Die Grundidee eines neuen Spezial-Chips

Die Idee besteht darin, den zeitaufwendigen Table Look-up zu vereinfachen und die Interpolation zwischen den Stützpunkten durch die einfache Berechnung der Ordinate y von Approximationsgeraden $y = a \cdot x + b$ zu ersetzen. Es kann gezeigt werden, daß lediglich eine einzige Addition und 2 Multiplikationen benötigt werden, um z.B. die gesamte Berechnung von $y = \sin x$ im Bereich $0 \le x \le \pi/2$ ***einschließlich*** dem "vereinfachten" Table Look-up mit hinreichender Genauigkeit (z.B. $\varepsilon \le 10^{-8}$) durchzuführen.

Dazu wird der Bereich $0 \le x \le \pi/2$ in n gleichbreite Segmente mit der Breite Δx unterteilt. In einem schnellen ROM werden aber ***nicht*** n Koordinaten x_i, y_i, sondern die $2n$ Koeffizienten a_i und b_i, $i = 1, 2, \ldots n$ der n Approximationsgeraden

$$\sin x \approx y = a_i \cdot x + b_i \qquad (5)$$

abgespeichert, so daß für (5) nur eine Addition und eine Multiplikation benötigt werden. Der "Quasi Table Look-up" beschränkt sich auf die Berechnung des Index i mit

$$i = 1 + \text{Integer}(q \cdot x) \qquad (6)$$

Wenn auch die Konstante

$$q = 1/\Delta x \qquad (7)$$

abgespeichert wird, dann benötigt man für (6) ebenfalls nur eine Addition mit 1 und eine Multiplikation. Schließlich kann man die Addition +1 sogar weglassen, so daß man mit

$$j = \text{Integer}(q \cdot x) \qquad (8)$$

die Größe j unmittelbar als Adresse benutzen kann, unter der die Koeffizienten a_i und b_i im ROM zu finden sind. Folglich sind, wie bereits oben erwähnt, für die ***gesamte*** $\sin x$ - Berechnung im Bereich $0 \le x \le \pi/2$ nur 1 Addition und 2 Multiplikationen erforderlich.

Die Autoren hatten diese Überlegungen bereits im Jahre 1986 gemacht, s. [2] u. [3]. Dabei wurde auch überlegt, ob es sinnvoll sei, statt mit Approximationsgeraden mit Approximationsparabeln

zu arbeiten, was für die gleiche Genauigkeit der sin-Berechnung die erforderliche ROM-Größe reduzieren würde. Jedoch wären dann je sin-Berechnung 2 Additionen und 3 Multiplikationen erforderlich, was die sin-Berechnung ca. 50% bis 60% verlangsamen würde.

Die Chip-Entwicklung hatte im Jahre 1986 noch nicht den heutigen Stand erreicht. Daher glaubte man damals, daß das erreichbare Nutzen/Kosten-Verhältnis die Entwicklung eines Spezial-Chips mit schnellem ROM und schneller Additions-Multiplikations-Arithmetik nicht rechtfertige (gleichgültig, ob man mit linearer oder quadratischer Approximation arbeiten würde). D.h. es bestand kein Interesse an einer Weiterführung dieser Arbeiten.

Heute hat sich die Situation geändert: Die Integrationsdichte auf den Chips ist stark angestiegen und der Speicherplatz wird immer billiger. Ein sehr umfangreiches schnelles ROM (groß genug, um mit der schnelleren linearen Approximation arbeiten zu können) ***und*** die erforderliche schnelle Arithmetik auf einem einzigen Chip zu realisieren, dürfte nur noch ein kleineres Problem darstellen. In [4] wurde gezeigt, daß man sich auch anderweitig um Geschwindigkeitssteigerungen mit Hilfe neuer Verfahren, die die Entwicklung neuer Chips erfordern, bemüht. Wegen des heutigen Standes der Chip-Entwicklung, und angeregt durch [4], haben die Autoren beschlossen, ihre Ideen von 1986 erneut aufzugreifen, in moderner Form vorzustellen und zur Entwicklung eines Spezial-Chips anzubieten.

3 Berechnung der Koeffizienten *a* und *b* für die stückweise lineare Approximation

Wegen der Periodizität des Sinus und wegen

$$\cos x = \sin (\pi/2 - x) \qquad (9)$$

braucht für alle weiteren Überlegungen nur die Sinusfunktion im Bereich $0 \le x \le \pi/2$ in Betracht gezogen werden. Hat man für diesen Bereich die Koeffizienten a und b der Approximationsgeraden ermittelt und abgespeichert, so lassen sich damit auch $\sin x$ für $-\infty < x < 0$ und für $\pi/2 < x < \infty$ sowie $\cos x$ für $-\infty < x < \infty$ mit nur äußerst geringem Mehraufwand berechnen.

Nachfolgend wird zunächst gezeigt, wie die zu speichernden Koeffizienten a und b ermittelt werden können:

Der Bereich von 0 bis $\pi/2$ wird in n gleichgroße Stücke (Segmente) unterteilt. Das nebenstehend wiedergegebene *Bild 2* zeigt die Approximationsgerade im 1. Segment. Für $x = 0$ ist auch der Fehler $\varepsilon = 0$. Da aber der Fehler der Approximation gegenüber der exakten Sinusfunktion im Bereich $0 \le x \le \pi/2$ absolut kleiner oder höchstens gleich ε_{max} sein muß, ist es durchaus zulässig, für den Endpunkt der Approximationsgeraden des 1. Segments laut *Bild 2*

$$y_1 = \varepsilon_{max} + \sin x_1 \qquad (10)$$

anzugeben. Durch Überschreiten von $\sin x_1$ um den maximal zulässigen Fehler ε_{max} kommt man mit insgesamt weniger Segmenten aus, als wenn man nur $y_1 = \sin x_1$ gewählt hätte. Mit (10) lassen sich

$$a_1 = y_1 / x_1 \text{ und } b_1 = 0 \qquad (11)$$

sofort angeben und abspeichern.

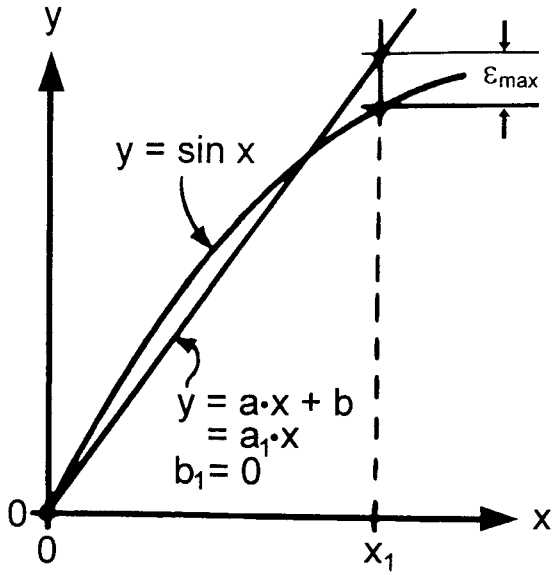


Bild 2 *Die 1. Approximationsgerade*

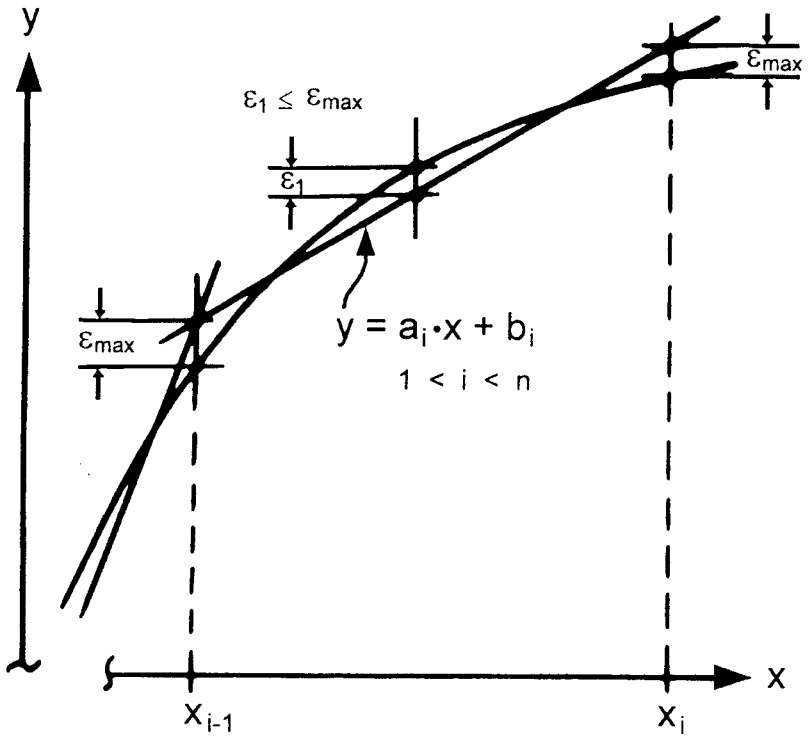


Bild 3

Die 2., 3., . . . (n-1)-te Approximationsgerade zusammen mit der exakten Sinusfunktion im 2., 3., . . . (n-1)-ten Segment

Die für die 2., 3., . . . (n-1)-te Approximationsgerade geltenden Verhältnisse sind im *Bild 3* dargestellt. Je am Anfang und am Ende des Segments überschreitet der y-Wert der Approximationsgeraden den exakten Wert des Sinus um ε_{max}. Damit wird bei den meisten Segmenten igendwo zwischen den Segment-Enden x_{i-1} und x_i ein Fehler $\varepsilon_1 < \varepsilon_{max}$ auftreten, sofern der x-Bereich in ***viele*** Segmente unterteilt wurde (n = groß). Jedoch wird bei mindestens einem Segment $\varepsilon_1 > \varepsilon_{max}$ auftreten, sofern der x-Bereich in nur ***wenige*** Segmente unterteilt wurde (n = klein). Unabhängig von der gewählten Segmentanzahl n ergibt sich laut *Bild 3* für das i-te Segment

$$a_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \quad , \qquad b_i = y_i - a_i \cdot x_i \tag{12}$$

was wiederum abgespeichert wird.

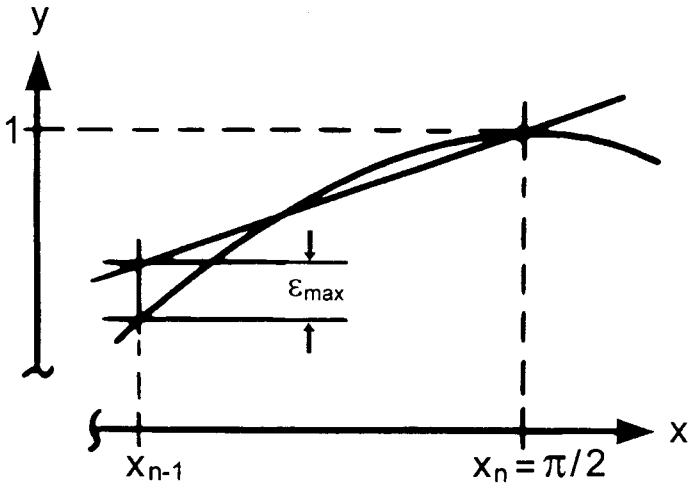


Bild 4

Die n-te (= letzte) Approximationsgerade

Die n-te Approximationsgerade wird laut *Bild 4* so gelegt, daß für $x = \pi/2$ der Fehler $\varepsilon = 0$ auftritt. Damit erhält man mit $i = n$, $x_i = x_n = \pi/2$ und mit $y_i = y_n = 1$ auch a_n und b_n entsprechend (12).

Im gesamten Bereich $0 \le x \le \pi/2$ darf der Fehler ε den zulässigen Maximalfehler ε_{max} niemals überschreiten. Jedoch wird man entsprechend *Bild 3* und obigen Ausführungen die geringstmögliche Anzahl der Segmente dann erhalten, wenn n so gewählt wird, daß man sich laut *Bild 3* dem Optimum $\varepsilon_1 = \varepsilon_{max}$ möglichst gut nähert. Die Abhängigkeit

$$n_{optimal} = f(\varepsilon_{max}) \qquad (13)$$

läßt sich jedoch nur auf iterativem Weg bestimmen: Beispielsweise haben die Autoren für ein gegebenes ε_{max} zunächst $n = 10$ gesetzt, damit die Koeffizienten a und b berechnet und dann dafür in 50000 Schritten den größten Fehler ε im gesamten Bereich $0 \le x \le \pi/2$ ermittelt. Danach wurde diese Prozedur mit $n = 10000$ wiederholt. Schließlich konnte, von diesen beiden Berechnungen ausgehend, eine "Binary Search"-Iteration aufgesetzt werden, bei der in nur ungefähr 16 bis 18 Iterationsschritten n verändert und so die optimale Segmentanzahl $n_{optimal}$ ermittelt wurde.

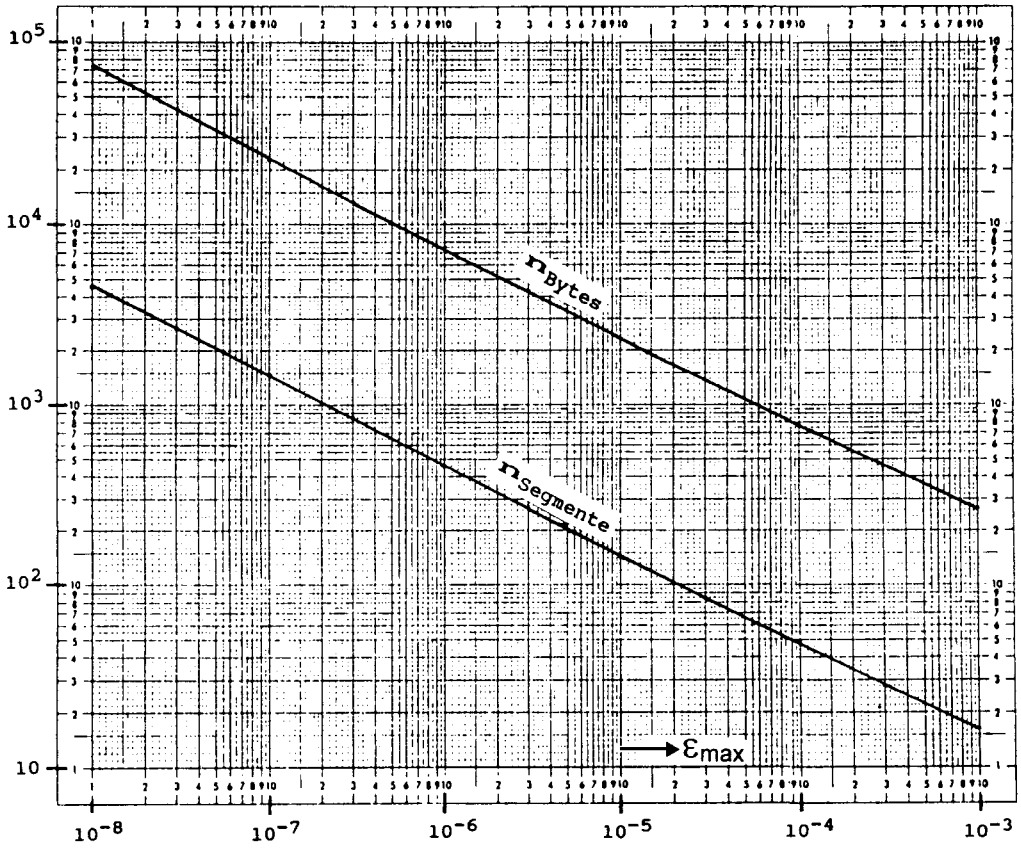
ε_{max}	Segmentanzahl genau	Segmentanzahl approx.	Speicher Bytes
10^{-8}	4584	4560	73352
$3 \cdot 10^{-8}$	2658	2647	42536
10^{-7}	1455	1459	23288
$3 \cdot 10^{-7}$	842	847	13480
10^{-6}	462	467	7400
$3 \cdot 10^{-6}$	267	271	4280
10^{-5}	146	149	2344
$3 \cdot 10^{-5}$	85	87	1368
10^{-4}	48	48	776
$3 \cdot 10^{-4}$	28	28	456
10^{-3}	16	15	264

Bild 5

Tabelle der optimalen Segmentanzahl und des Speicherplatzbedarfs als Funktion des maximal zulässigen Fehlers

Bild 6

Graphische Darstellung der Werte der Tabelle Bild 5 ↓



Die in der oben geschilderten Weise ermittelte Abhängigkeit (13) ist in den ersten beiden Spalten der Tabelle *Bild 5* bzw. in der unteren Kurve im *Bild 6* dargestellt. Jedoch läßt sich (13) auch durch

$$k = -0{,}495 \cdot \log_{10} \varepsilon_{max} \quad , \quad n_{optimal} = \text{Integer}\left(0{,}5 + 0{,}5 \cdot 10^{k}\right) \qquad (14)$$

zum Zwecke der schnellen Abschätzung der Segmentanzahl approximieren. Die Näherung (14) ist in der 3. Spalte der Tabelle *Bild 5* wiedergegeben.

Um eine der üblichen "Single Precision" (= 32 bit = 4 Bytes) entsprechende Genauigkeit der Sinus-Berechnung zu erreichen (z.B. mit $\varepsilon_{max} = 3 \cdot 10^{-8}$) müssen die Koeffizienten a und b mit weit größerer Genauigkeit berechnet und abgespeichert werden. Man wird daher zweckmäßigerweise die a- und b-Berechnungen und -Speicherungen mit der sogenannten "Double Precision" (= 64 bit = 8 Bytes pro Gleitkommazahl) vornehmen. Da außer den a- und b-Werten noch einmalig mindestens die Konstante q laut (7) abzuspeichern ist, muß das Spezial-Chip ein ROM mit mindestens

$$n_{Bytes} = 8 + 16 \cdot n_{optimal} \qquad (15)$$

Bytes enthalten, wie in der 4. Spalte der Tabelle *Bild 5* bzw. in der oberen Kurve im *Bild 6* angegeben. Bedenkt man jedoch, welche Speichergrößen auf heutigen Chips realisierbar sind, dann dürfte ein schnelles ROM mit z.B. > 40 KB und einer Organisation, die pro angelegter Adresse auf 16 Bytes zugreift (a und b zusammen), keine besondere Schwierigkeit darstellen.

In der Tabelle *Bild 7* sind a- und b-Werte in Gleitkomma-Darstellung mit 16 Dezimalstellen gezeigt. (8 Bytes entsprechen gemäß der dabei üblichen Mantissenlänge von 56 bit einer Genauigkeit von $\log_{10}(2^{56}) \approx 16,8$ Dezimalstellen.) Da *Bild 7* nur ein ***Beispiel*** ist, wurde die recht kurze Tabelle mit nur 28 Segmenten gewählt, die laut *Bild 5* die Sinus-Berechnung nur mit $\varepsilon_{max} = 3 \cdot 10^{-4}$

Bild 7

Tabelle der a- und b-Werte bei einer Einteilung des Bereichs $0 \le x \le \pi/2$ *in lediglich 28 Segmente*

Segment i	a_i	b_i
1	1.004817808469106	0.0000000000000000E+00
2	0.9963308301367069	0.4761183710757511E-03
3	0.9900512848875922	0.1180681700450922E-02
4	0.9806566720125237	0.2761791350002901E-02
5	0.9681765504392792	0.5562324082175413E-02
6	0.9526501872475270	0.9917458773262777E-02
7	0.9341264341200705	0.1615253949563411E-01
8	0.9126635736373783	0.2458098509781570E-01
9	0.8883291358990193	0.3550225521611161E-01
10	0.8611996860489982	0.4919988241437734E-01
11	0.8313605833734664	0.6593957987065735E-01
12	0.7989057127288510	0.8596743371161641E-01
13	0.7639371891453457	0.1095081887395083
14	0.7265650365352488	0.1367636359029621
15	0.6869068415170387	0.1679111094339242
16	0.6450873834443593	0.2031021011102396
17	0.6012382418040210	0.2424609986083865
18	0.5554973822182392	0.2860839543859029
19	0.5080087223537614	0.3340378909798921
20	0.4589216791036339	0.3863596480291784
21	0.4083906984663787	0.4430552757251425
22	0.3565747696017403	0.5040994787726760
23	0.3036369245919769	0.5694352143003596
24	0.2497437254826045	0.6389734465005514
25	0.1950647402165828	0.7125930601080468
26	0.1397720091108377	0.7901409341431966
27	0.8403950355376694E-01	0.8714321766544588
28	0.2270032014509487E-01	0.9643424204990145

zuläßt. Entsprechend (10) bis (12) und den *Bildern 2* bis *4* können die abzuspeichernden Tabellenwerte jedoch leicht für jede gewünschte Genauigkeit bzw. Segmentanzahl berechnet werden.

4 Schlußbemerkungen

Die Autoren glauben, daß es durchaus lohnend wäre, auf der Basis des hier vorgestellten Verfahrens ein Spezial-Chip zu entwickeln (und zu fertigen), das die schnelle sin- und cos-Berechnung im Bereich $-16^{63} < x < 16^{63}$ mit wenigstens $\varepsilon_{max} \leq 4 \cdot 10^{-8}$ ermöglicht. Dabei könnte möglicherweise auch das in [4] vorgestellte Verfahren zur schnellen Multiplikation mit eingesetzt werden, um die für die schnelle sin-cos-Berechnung laut (5) bis (8) immer noch nötigen wenigen Multiplikationen weiter zu beschleunigen.

Die Entwicklung könnte z.B. als Diplomarbeit(en) durchgeführt werden, eventuell im Rahmen der Zusammenarbeit der Mitglieder der *Multiprojekt-Chip-Gruppe Baden-Württemberg* (MPC-Gruppe).

Referenzen

[1] VS FORTRAN Application Programming: Library References, IBM Manual SC26-3989-2, Third Edition, 1983

[2] *Najmann, K.*: Unveröffentlichte Notizen und Protokolle zur Entwicklung eines Spezial-Chips, 1986

[3] *Spiro, H.*: Überlegungen zur schnellen Berechnung der Sinusfunktion durch lineare oder quadratische Approximation. Unveröffentlichter Aufsatz, 1986

[4] *Rülling, W.*: Schnelle Multiplizierer. Workshop Esslingen Juli 1997 der Multiprojekt-Chip-Gruppe Baden-Württemberg, S. 41 - 52, FH Ulm, 1997

Optimiertes Testen von analogen ICs mittels generischer ATE-Modelle und Testprogramm-Synthese

Dipl.-Ing. (FH) Jens Schuster

Prof. Dr. rer. nat. Uwe Jäger

Fachbereich Elektronik

Fachhochschule Heilbronn, Max-Planck-Straße 39, 74081 Heilbronn

Tel.: 07131-504419

Email: uwe.jaeger@fh-heilbronn.de

Bei dem hier vorgestellten Projekt handelt es sich um ein vom MWF gefördertes Innovatives Projekt, das an der Fachhochschule Heilbronn im Fachbereich Elektronik durchgeführt wird. Dieses Projekt ist in eine Kooperation mit der Firma TEMIC Semiconductors GmbH in Heilbronn und der University of Northumbria at Newcastle in England eingebunden.

Nach einer kurzen Einleitung wird der momentan aktuelle Stand der Technik im Bereich der Testprogrammgenerierung zusammengefaßt. Die anschließende Beschreibung des optimierten Testsystems für analoge und mixed-signal ICs wird durch die zugrunde liegenden theoretischen Grundlagen vervollständigt. Eine kurze Präsentation der bereits realisierten Vorarbeiten mündet schließlich in einem Ausblick auf weiterführende Schritte.

1 Einleitung

Wie bereits erwähnt handelt es sich bei dem beschriebenen Projekt um ein sogenanntes Innovatives Projekt, das vom MWF Baden-Württemberg über einen Zeitraum von 1.5 Jahren gefördert wird. Durch die Kooperation mit dem Halbleiterhersteller TEMIC Semiconductors GmbH wird in Form praktischer Anwendungen die Realitätsnähe gewährleistet, wohingegen die Zusammenarbeit mit der University of Northumbria at Newcastle, unter anderem durch Recherchemöglichkeiten, den nötigen akademischen Hintergrund bietet.

Das hier präsentierte Testsystem, das im folgenden noch detaillierter erläutert werden wird, wurde folgendermaßen konzipiert:

Eine sogenannte virtuelle Test-Bench soll mit Hilfe eines Pakets von Softwaretools ermöglicht werden, das dazu dient durch innovatives Vorgehen die Kosteneffizienz der mixed-mode IC Design- und Testaktivitäten deutlich zu verbessern. Die zu entwickelnden Tools stellen dem Designer umfassende Modelle von Testhardware für den Produktionstest und automatisierte Testprogramm-Generierungsmöglichkeiten zur Verfügung. Weitere Tools basieren auf dem Einsatz einer sogenannten Simulation Test-Bench, die in einer hochentwickelten Hardware Beschreibungssprache formuliert wird und die, im Gegensatz zu einem realen Hardware-Testsystem, als virtuelles Testsystem ausgeführt sein wird.

Die primäre Aufgabenstellung für dieses System läßt sich dann folgendermaßen formulieren: Das Forschungsvorhaben soll grundsätzlich das Testen von analogen und mixed-signal ICs (integrierten Schaltkreisen) beschleunigen, verbessern und flexibler gestalten, da dieser Bereich im Fertigungszyklus noch immer den gewichtigsten Kostenfaktor darstellt und zudem die Methoden und Verfahren für digitale ICs, die schon sehr umfangreich bearbeitet wurden, praktisch nicht adaptiert werden können.

Um dieses Ziel zu erreichen, soll wie folgt vorgegangen werden:

Nach diversen theoretischen Grundlagenarbeiten (Können analoge bzw. mixed-signal ICs mit einer AHDL / HDL beschrieben werden? Kann eine Test-Bench zur Designverifikation in dieser HDL realisiert werden? Hat die HDL die benötigten Features, um komplexe ATE-Systeme (Automatic Test Equipment) ausreichend beschreiben zu können?), die zum Teil bereits durchgeführt wurden, folgt die eigentliche Implementierung des generischen ATE-Systems als HDL-Modell. Im Anschluß daran findet die Implementierung der Software zur automatischen Testprogramm-Synthese mit einer anschließenden Verifikation des kompletten Simulationsmodells statt.

Den Abschluß bildet dann ein Vergleich der Ergebnisse DUT ⇔ Test-Bench-Simulation und DUT ⇔ ATE-Modell.

2 Stand der Technik

Ist hier die Rede vom Stand der Technik, so bezieht sich dies ausschließlich auf die Generierung eines Testprogrammes, wie es zur Prüfung eines integrierten Schaltkreises notwendig ist.

In der Vergangenheit wurde dir Testprogrammgenerierung vollständig manuell durchgeführt: Am Anfang stand normalerweise eine allgemeine Prüfvorschrift, die Angaben zur Abdeckung verschiedener Rahmenbedingungen enthielt. Diese Rahmenbedingungen setzten sich in der Regel aus applikationsspezifischen Kriterien, Umweltbedingungen, Simulationsergebnissen und anderen Anforderungen zusammen. Zweck dieser Prüfvorschrift war es, die Funktionalität des zugrunde liegenden ICs, innerhalb gewisser Toleranzen, zu überprüfen.

Diese allgemeine Prüfvorschrift für einen IC mußte daher, über eine zwischengeschaltete Übersetzung durch den Testingenieur, in ein Testprogramm für einen bestimmten ATE-Typ konvertiert werden. Dieses Verfahren war, einschließlich des anschließenden Debuggings, Grund für die enormen Kosten bei der Testprogrammerstellung.

Ein neueres Verfahren ist die sogenannte Fehlersimulation, wobei man hier Simulation After Test und Simulation Before Test unterscheiden muß. Beide Verfahren dienen zur Erkennung von katastrophalen, weichen und multiplen Fehlern eines Bauteils, unterliegen aber auch bedeutenden Einschränkungen. So kann das Verfahren Simulation Before Test zum Beispiel nur bei katastrophalen Fehlern angewandt werden.

Ein weiteres Verfahren der jüngeren Zeit sind die sogenannten Design For Testability Techniken, wie Boundary Scan, Built In Self Test, usw. Ihr Beitrag zur eigentlichen Testprogrammgenerierung ist allerdings als eher gering einzustufen.

Der bei weitem aussichtsreichste Ansatz der vergangenen Jahre ist der des Virtuellen Testers, wie er insbesondere von Teradyne, einem ATE-Hersteller, in Zusammenarbeit mit Analogy und TI praktiziert wird. Nachdem es sich in den vergangenen Jahren immer mehr herausgestellt hat, daß die Time-to-Market eines ICs vor allem durch das, teilweise sehr aufwendige Debugging des Testprogrammes geprägt wird, wurde die Forderung nach einem Off-Line Verfahren laut, welches es ermöglicht den eigentlichen Tester weiterhin im Produktionsprozeß zu belassen und dennoch ein Testprogramm entwickeln zu können - möglichst noch vor Fertigstellung des ersten Wafers.

Diese Forderung resultierte in der Entwicklung des sogenannten Virtuellen Testers, der als Simulationsmodell eines spezifischen ATE-Systems ausgeführt ist. Dieses Simulationsmodell hat die Fähigkeit ein Testprogramm abzuarbeiten, während es mit einem Verhaltensmodell des DUT verbunden ist. Dabei wird die Integrität des Testprogrammes verifiziert, ohne die eigentliche Hardware einsetzen zu müssen. Die momentan erhältlichen Tools neigen allerdings dazu, eng an einen großen ATE-Hersteller gebunden zu sein und sind daher in der Regel inkompatibel. Die Festlegung auf einen bestimmten ATE-Hersteller stellt normalerweise eine beträchtliche Investition dar, die manches kleine bis mittelgroße Unternehmen davon abhalten wird oben genannte Technik einzusetzen und zu erkunden. Die Art und Weise der Erfassung der Testkonfiguration mittels momentan erhältlicher Tools, basiert auf der Schaffung von einzigartigen Test-Blockschaltbildern, die sich auf einzelne Tests, oder kurze Testsequenzen, beziehen. Hieraus können sehr große Datenmengen für einen bestimmten Typ resultieren.

Den letzten Punkt in den neueren Testprogrammtechniken stellt dann das sogenannte Meßwert abhängige Messen dar, das auch unter Namen, wie Just Enough Test, bekannt ist. Hierbei werden Meßerkenntnisse eines bestimmten IC-Typs ausgewertet, um Meßzeit zu sparen.

Natürlich kann diese Aufzählung keinesfalls vollständig sein, aber die genannten Punkte stellen dennoch im wesentlichen den aktuellen Stand der Technik, und außerdem die zukunftsträchtigsten Ansätze in diesem Sektor dar.

3 Optimiertes Testsystem für analoge & mixed-signal ICs

3.1 Prinzip

Grundlage des gesamten Testsystems für analoge und mixed-signal ICs bildet ein virtuelles, generisches und rekonfigurierbares ATE-Modell.

In vielen Fällen ist heutzutage allerdings die Verwendung eines spezifischen ATE-Modells nicht unbedingt angebracht, da es dem Designer unnötige Beschränkungen auferlegt. Die hier vorgestellte Lösung ist ein sogenanntes generisches ATE-Modell. Dieses Modell simuliert weitgehend das allgemeine Verhalten von realem Testequipment, ohne jedoch die exakten Charakteristika eines speziellen ATE-Typs aufzuweisen. Die Verfügbarkeit eines generischen ATE-Modells ermöglicht es dem Designer und / oder Testingenieur die Realisierbarkeit eines

ATE Testprogramms zu erkunden, während er optional die Einflüsse alternativer und spezifischer ATE Parameter untersuchen kann, bevor eine Entscheidung über die exakte Natur der vorgesehenen Testplattform gefällt wird.

Nach Abschluß dieser Untersuchungen kann das generische ATE-Modell dann so konfiguriert werden, damit es genauso wie das gewählte herstellerspezifische, virtuelle ATE mit realem Verhalten agiert. Durch diesen Einsatz eines Modells der tatsächlichen Tester-Hardware, die anschließend im Produktionstest zum Einsatz kommen soll, wird eine Designverifikation ermöglicht.

Das generische ATE-Modell muß also als virtuelles ATE-Modell mit realem Verhalten rekonfiguriert werden können, d. h. es sind virtuelle Modelle von verschiedenen Testern nötig, die reales Verhalten simulieren.

Das Endprodukt ist dann ein resultierendes, reales Testprogramm, das direkt in der Produktion eingesetzt werden kann.

Die Vorteile dieses Prinzips sind, daß ein IC getestet werden kann, ohne eine beträchtliche Investition in eine spezifische Tester-Hardware tätigen zu müssen und somit die Entscheidung über den realen Testertyp auf einen späteren Zeitpunkt im Designprozeß verschoben werden kann. Außerdem kann die Verfügbarkeit eines rekonfigurierbaren ATE Simulationsmodelles, im Falle von getrennten IC-Design- und IC-Test-Häusern (als Sub-Contractor), deutlich die Entscheidungsfindung für ein bestimmtes, passendes Testhaus erleichtern.

3.2 Hintergrund

Der große Vorteil des vorgestellten Konzepts, ist die Tatsache, daß alle Komponenten, wie Test-Bench, ATE, Simulation IC, usw. in einer HDL- / AHDL-Kombination modelliert werden, so daß, ohne besondere Detailkenntnisse zu erfordern, eine gemeinsame Basis für Entwickler und Testingenieur geschaffen wird.

Außerdem soll durch dieses System eine Testprogramm-Synthese, vergleichbar der IC-Synthese, ermöglicht werden.

Im Gegensatz zu anderen Prinzipien, wird die Portierbarkeit der erzeugten Testprogramme durch den Einsatz einer High-Level Beschreibungsmethode sichergestellt, und nicht durch eine allgemeine Programmiersprache (entsprechend einer Low-Level Testsprache), mit der das Testprogramm für die diversen A [illegible]-Typen adaptiert wird. Um die Vorteile eines

generischen ATE-Modells voll ausschöpfen zu können, zielt dieses Vorhaben darauf ab, ein automatisches Software-Tool zur Testprogramm-Synthese für ATE zu entwickeln. Das prozedurale Low-Level Testprogramm wird dabei von einer HDL-basierten Test-Bench generiert, die zuvor vom IC Designer definiert wurde. Hierdurch wird der Gebrauch von zahlreichen „Test-Blockschaltbildern" für die Tests vermieden. Die Methode einen Test als Test-Blockschaltbild zu beschreiben ist auf einfache Sequenzen oder einzelne Tests beschränkt, wohingegen der Einsatz einer HDL-Test-Bench, in einer dem Designer vertrauten hardwareunabhängigen Art und Weise, auf Verhaltensebene eine komplette Beschreibung der Testsequenz ermöglicht, die nötig ist, um das DUT (Device Under Test) umfassend zu testen.

4 Theoretische Grundlagen

Die theoretischen Grundlagen, auf denen das optimierte Testsystem für analoge und mixed-signal ICs beruht, umfassen im wesentlichen folgende Bereiche: die Hardware Beschreibungssprachen, die Modellierung von automatischem Testequipment, die mixed-signal Test-Bench und schließlich die Prinzipien der Testprogrammgenerierung.

Bei den Hardware Beschreibungssprachen stehen verschiedene Gesichtspunkte im Vordergrund, bzw. müssen zur Auswahl der Projekt-HDL untersucht werden. Primär ist zu klären, welche Anforderungen die entsprechende HDL zu erfüllen hat. Damit einher geht die Frage nach der Verfügbarkeit dieser HDL, d. h. ob eine größere Investition getätigt werden muß. Weitere wichtige Aspekte sind sowohl die Kompatibilität der potentiellen HDL zu anderen Sprachen und bereits existierenden Simulatoren, als auch die Präsenz von vordefinierten Libraries für die verschiedenen Modellkomponenten. Neben anderen Dingen muß auch geklärt werden, ob es sinnvoll ist eine standardisierte (IEEE) AHDL / HDL zu verwenden, oder eine kommerzielle HDL-Kombination.

Im Bereich der Modellierung von automatischem Testequipment muß untersucht werden, worauf die Modellierung basieren kann. Dies bedeutet, daß ein gemeinsamer Faktor herausgefunden werden muß, der dann als Grundlage für die ATE-Modellierung dienen kann. Zusätzlich muß festgelegt werden, in welchem Maße die Modelle generische Eigenschaften aufzuweisen haben, bzw. wie diese generischen Eigenschaften an sich auszusehen haben.

Den letzten Gesichtspunkt der theoretischen Grundlagen bilden Überlegungen zu der Beschaffenheit der endgültigen mixed-signal Test-Bench und der eigentlichen Generierung

des Testprogramms. Von besonderem Interesse ist hierbei die Frage, was aus dem digitalen IC-Testsektor übernommen werden kann.

5 Vorarbeiten

5.1 Softwaresystem

An dieser Stelle soll kurz erläutert werden, wie die Oberfläche, bzw. die eigentliche Software des ATP-Systems (Automatic Test Programming) aufgebaut ist.

Im Prinzip kann man das ATP-Softwaresystem in die eigentlich Oberfläche, die zahlreiche Standard-Softwaretools integriert, die Windows-Conversion (s. 5.2) und den sogenannten Graphic Editor (s. 5.2) mit integrierten Symbol-Tools untergliedern.

Die Software ist als Windows-Software (Windows 3.x und Windows 95) ausgeführt, mit all den hieraus resultierenden Vorteilen, wie z. B. DDE, OLE, usw.

5.2 Wichtige Tools

Wie bereits erwähnt, umfassen die Entwicklungen der ATP-Software ein ganzes Paket von Tools, die zur Vorbereitung einer automatischen Testprogrammgenerierung geschaffen wurden.

Die funktional bedeutsamen Tools bestehen zur Zeit aus:

- einem Werkzeug, das die Erstellung einer Prüfvorschrift für ICs automatisiert,
- dem sogenannten Windows-Conversion Tool, das eine bereits erstellte Prüfvorschrift semi-automatisch in ein Testprogramm für bestimmte Testautomaten umwandelt,
- dem sogenannten Graphic Editor, der die Erstellung einer graphischen Prüfvorschrift auf Test-Blockschaltbild-Prinzip (d. h. das hardwaremäßige Setup eines jeden Tests kann als graphisches Blockschaltbild der einzelnen Instrumente dargestellt werden) ermöglicht,
- dem ManSym-Tool, das die manuelle Generierung von Symbolen für den oben erwähnten Graphic Editor gestattet,
- und schließlich der LibConvert Option, die sowohl die Verwendung von PSpice und OrCAD Symbol-Libraries, als auch die Integration von ganzen Schematics für den Graphic Editor ermöglicht.

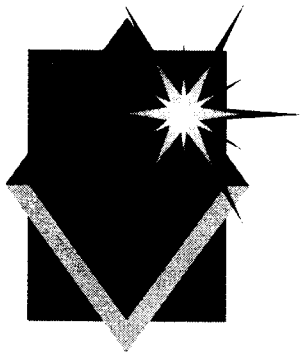
Diese Softwarebausteine dienen als Grundlage für alle weiteren Entwicklungen im Rahmen dieses Vorhabens.

6 Ausblick

Die zugrunde liegenden theoretische Untersuchungen implizieren die Reihenfolge der nächsten Schritte.

Nach der endgültigen Festlegung der Projekt-HDL, die momentan noch durch die Standardisierungsaktivitäten diverser HDLs verzögert wird, folgt zunächst eine Modellierung sämtlicher relevanten Komponenten. Diese Modellierung umfaßt in einem ersten Schritt nur die beteiligten Testhardware-Komponenten.

Danach wird dann die Generierung der mixed-signal Test-Bench realisiert, die abschließend in der Testprogrammerstellung mit diversen Features münden wird.



Optimiertes Testen von analogen ICs mittels generischer ATE-Modelle und Testprogramm-Synthese

Fachhochschule
Heilbronn
Fachbereich Elektronik

Dipl.-Ing. (FH) Jens Schuster 23. 01. 1998

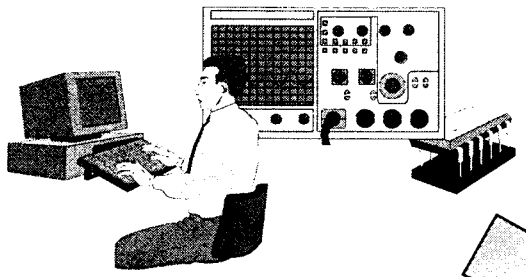
Optimiertes Testen von analogen ICs mittels generischer ATE-Modelle und Testprogramm-Synthese

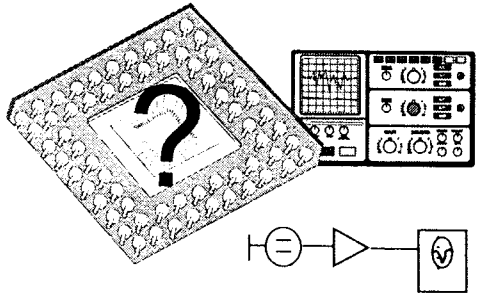
- Einleitung
- Stand der Technik
- Optimiertes Testsystem für analoge & mixed-signal ICs
- Theoretische Grundlagen
- Vorarbeiten
- Ausblick

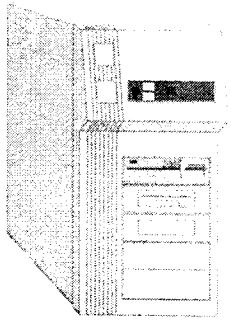
Einleitung

- Organisatorischer Hintergrund
- Kooperationen
- Kurzbeschreibung Testsystem
- Aufgabenstellung
- Vorgehensweise

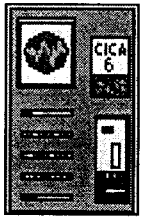
Stand der Technik





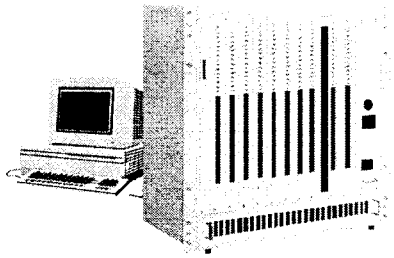


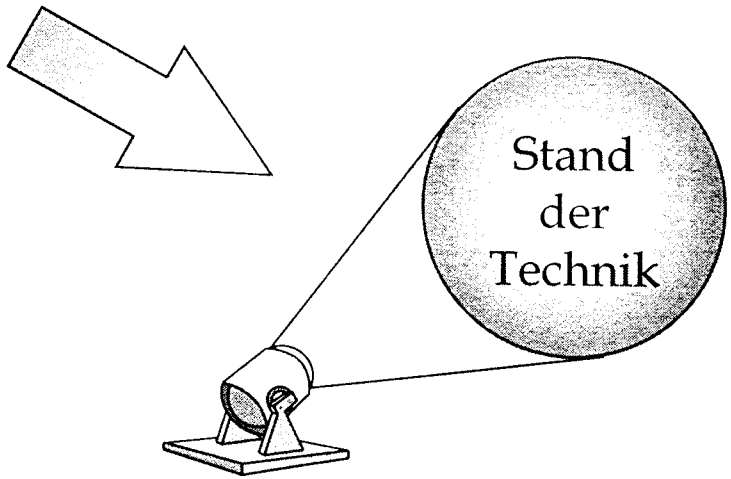
DFT



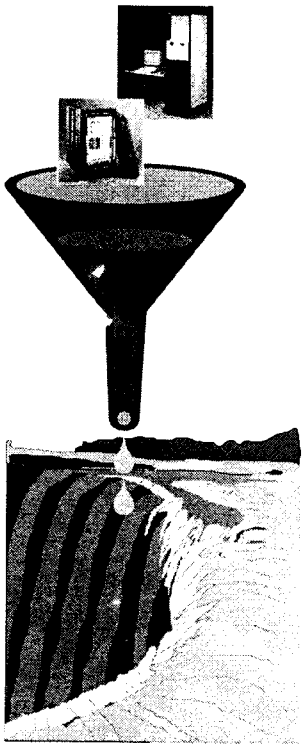
Teradyne

Analogy

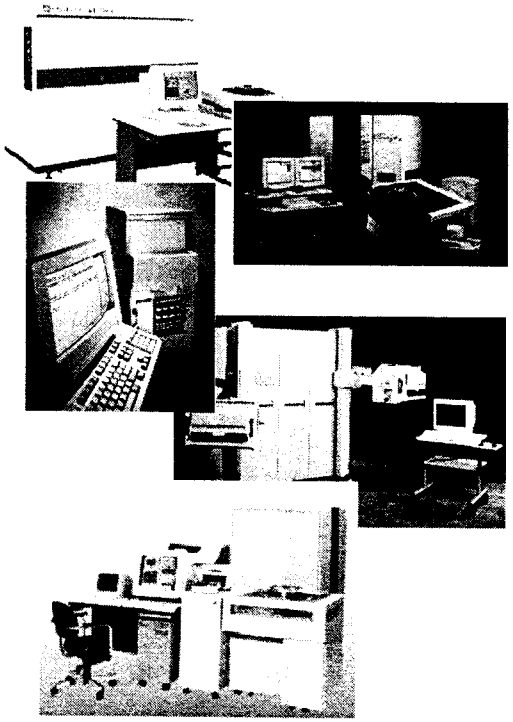




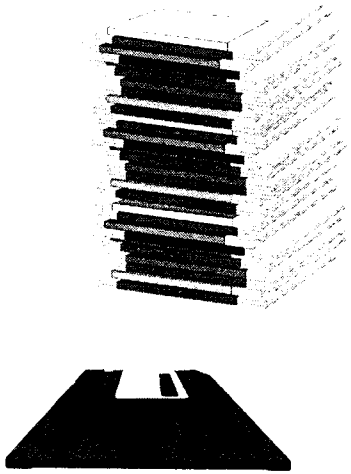
Optimiertes Testsystem für analoge & mixed-signal ICs 1



Generisches ATE-Modell

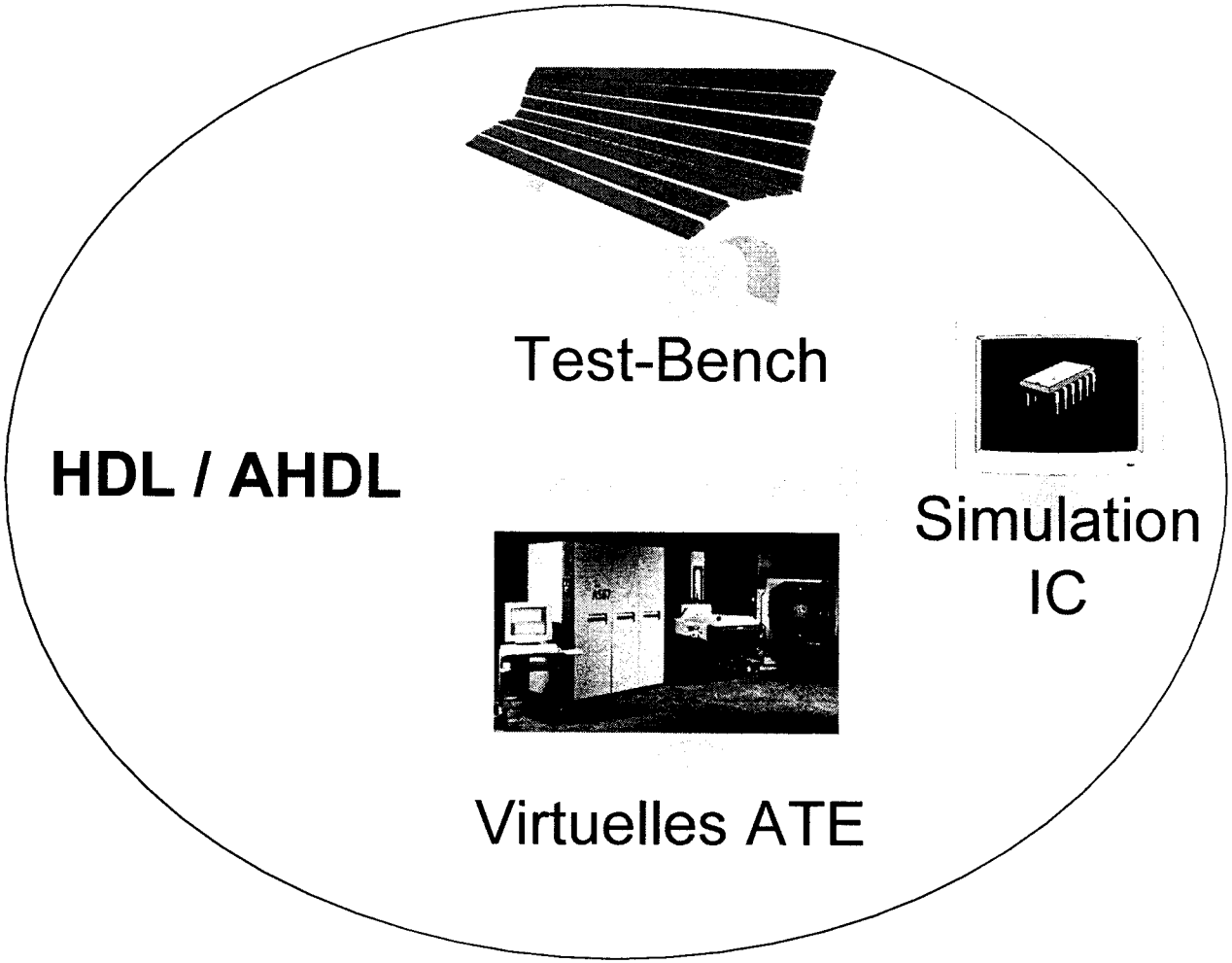


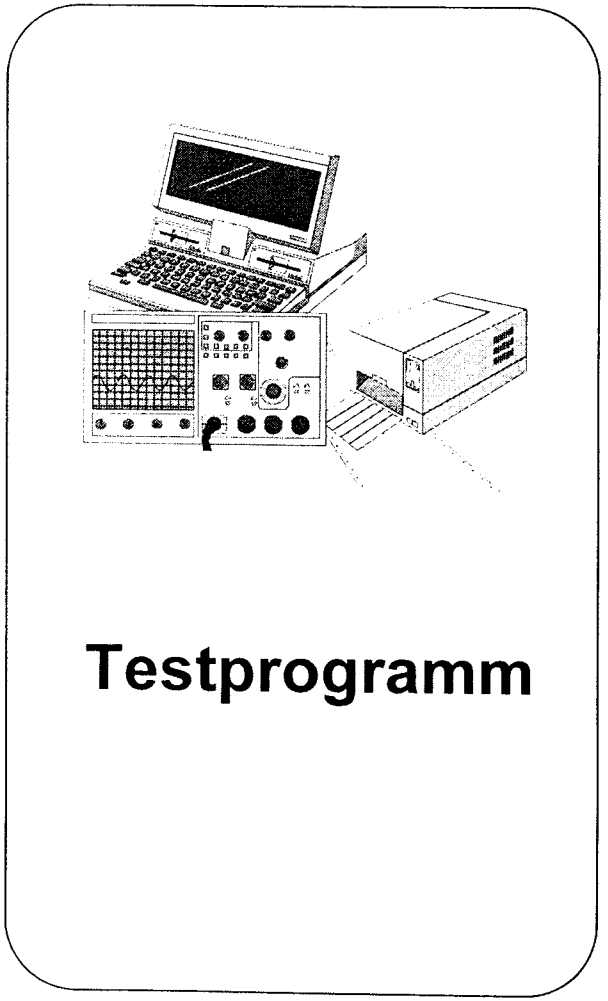
Virtuelle ATE-Modelle mit realem Verhalten



Reales Testprogramm

Optimiertes Testsystem für analoge & mixed-signal ICs 2

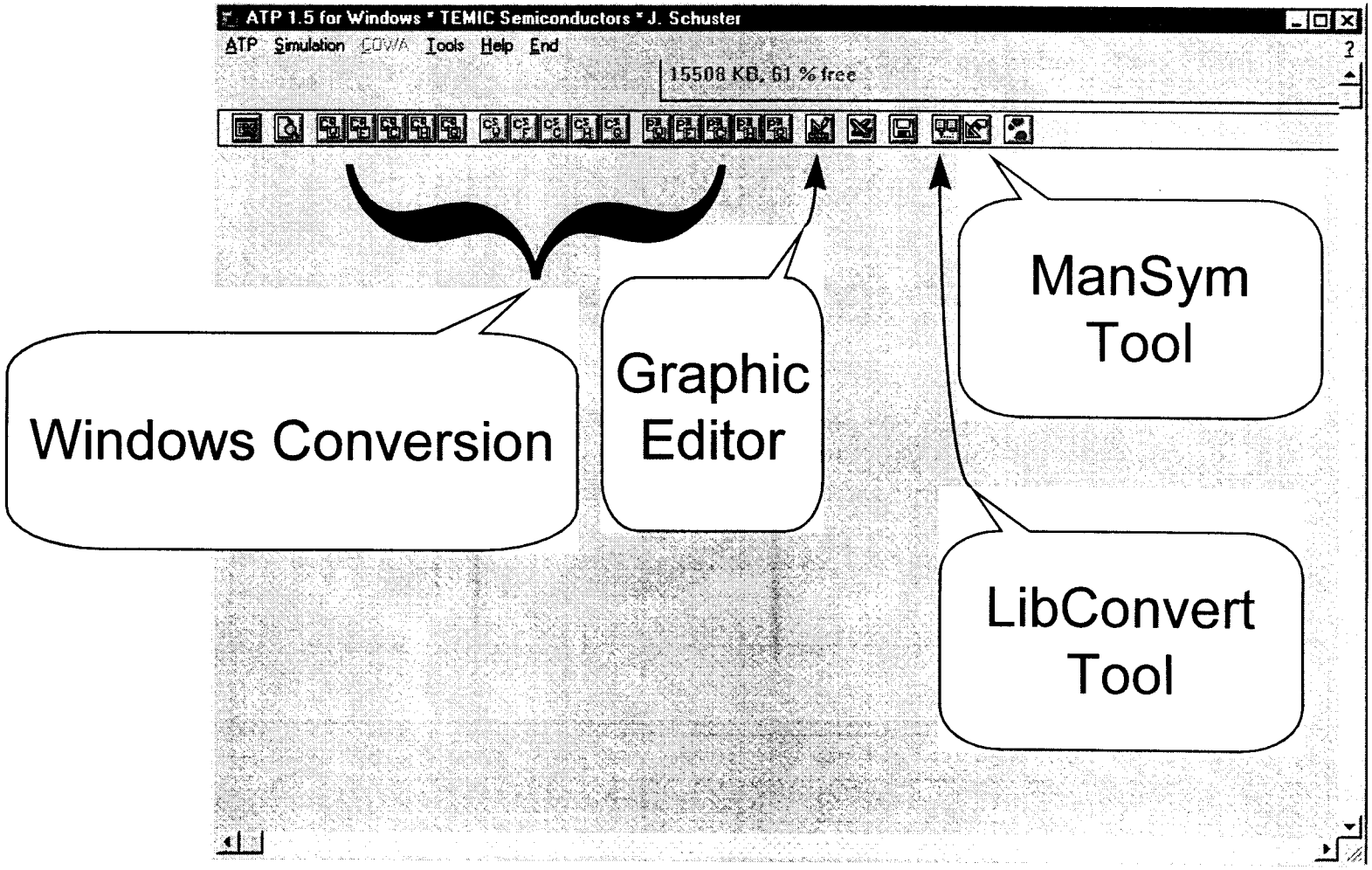




Theoretische Grundlagen

- Hardware-Beschreibungssprachen (Auswahl, Anforderungen,...)
- Modellierung automatisches Testequipment (gemeinsamer Faktor, generisches Modell,...)
- Mixed-signal Test-Bench
- Generierung Testprogramm

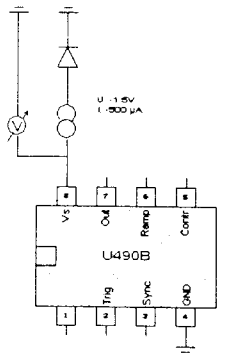
Vorarbeiten 1



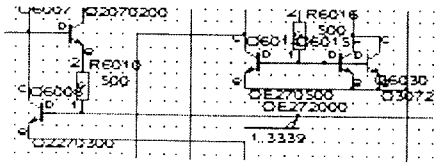
Vorarbeiten 2

- Umsetzung tabellarische Prüfvorschrift
- Graphische Prüfvorschrift
- Bereitstellung Symbole für Graphic Editor

U490B	Test no. 8			September 95	
Measurement contact test pin 8					
Parameter	Pin	Final test		Quality		Unit
		min	max	min	max	
OAS08	8	-0,9	-0,4	-1	-0,3	V



OrCAD®



Ausblick

- ⇨ Modellierung diverser Komponenten
- ⇨ Generierung mixed-signal Test-Bench
- ⇨ Automatisierung Testprogramm-erstellung (off-line, mehrfach bi-direktionaler Datenverkehr)

Anwendung von Methoden der symbolischen Analyse beim Entwurf analoger Schaltungen

Varol Mutlu, Gerhard Forster
Fachhochschule Ulm, Prittwitzstraße 10, 89075 Ulm

1 Motivation

Ziel der Dimensionierung einer Schaltung ist es eine bestimmte Spezifikation zu erfüllen. Die Spezifikation kann dabei Größen wie z.B. die Verstärkung, Bandbreite oder Verlustleistung enthalten (Abbildung 1.1).

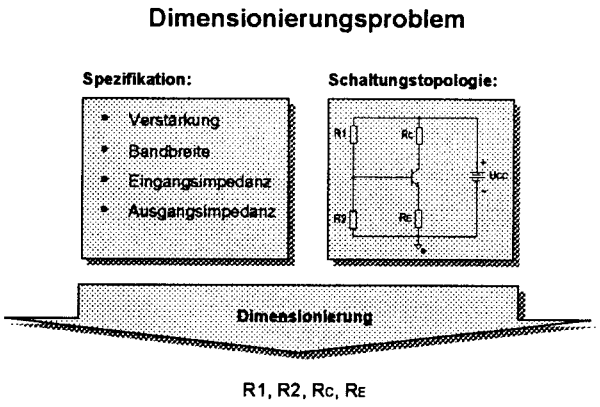


Abbildung 1.1: Das Dimensionierungsproblem

Bei der Art der Dimensionierung gibt es zwei verschiedene Methoden, die Simulationsmethode und die analytische Methode. Bei der Simulationsmethode werden für die Schaltungselemente ungefähre Zahlenwerte eingesetzt, die man entweder aus ähnlichen Schaltungen bereits kennt oder durch einfache Handanalysen berechnet. Mit einem Simulator (z.B. PSPICE) wird dann verifiziert, ob mit diesen Bauteilwerten die geforderte Spezifikation eingehalten werden kann. Falls nicht, werden die Bauelementewerte so lange optimiert, bis die gewünschte Spezifikation erreicht ist. Der Nachteil dieser Vorgehensweise ist, daß geeignete Startwerte für die Schaltungselemente gefunden werden müssen und viele Simulationsläufe benötigt werden.

Bei der analytischen Methode versucht man mathematische Beziehungen zwischen Schaltungselementen und der Spezifikation in Form von Gleichungen herzuleiten. Mit diesen Gleichungen werden dann die Werte für die Schaltungselemente bestimmt. Leider ist die Herleitung der Schaltungsgleichungen nicht immer einfach. Schon bei kleineren Schaltungen können komplizierte mathematische Formulierungen entstehen, die nicht mehr zu überblicken sind. Hier könnten Computer-Algebra-Systeme (CAS) eine wirksame Unterstützung leisten. Mit ihnen lassen sich symbolische Formeln mit einem Rechner bearbeiten.

2 Werkzeuge

2.1 CAS

Computer-Algebra-Systeme (CAS) sind Softwareprogramme, mit deren Hilfe unter anderem symbolische Mathematik betrieben werden kann. Symbolische Mathematik bedeutet formelmäßiges Rechnen. Die ersten CAS, die symbolisch rechnen konnten, entstanden vor etwa 20 Jahren. Sie wurden damals zur Bearbeitung von umfangreichen Formeln entwickelt und angewendet [1], vor allem in der Physik. Mit den heutigen CAS-Versionen lassen sich seitenlange Gleichungen handhaben, Gleichungssysteme lösen, Iterationen vornehmen usw. Ein weiterer Vorteil der CAS ist die Möglichkeit der grafischen Darstellung von Funktionen.

Zur Zeit gibt es mehrere CAS-Produkte von unterschiedlichen Herstellern auf dem Softwaremarkt. In der Tabelle 2.1 sind die bekanntesten CAS aufgelistet.

MAPLE
MATHEMATICA
MATHCAD
MACSYMA

Tabelle 2.1: Bekannte Computer-Algebra-Systeme

Unter der **Internet-Adresse [http://www.uni-frankfurt.de/~stst/ncrunch.html]** sind Testergebnisse aufgeführt, wobei mehrere Computer-Algebra-Systeme auf Funktionalität, Bedienung, Geschwindigkeit, Grafikfähigkeit und Kompatibilität mit anderen Software-Systemen geprüft und bewertet wurden.

2.2 MATHEMATICA

Das Computer-Algebra-System MATHEMATICA wurde von Wolfram Research Inc., USA, entwikkelt. Für die vorliegende Arbeit stand die Version MATHEMATICA 3.0.1 zur Verfügung. MATHEMATICA umfaßt folgende Funktionen:

- Numerisches Rechnen
- Symbolisches Rechnen
- Algebraische Manipulationen
- Lösen von Gleichungssystemen
- Mathematische Funktionen
- Grafikfunktionen
- Programmierfähigkeit
- Datei-Manipulationen

2.3 Analog Insydes (AI)

Analog Insydes ist ein Softwareprogramm, welches die Funktionalität von Computer-Algebra-Systemen (z.B. MATHEMATICA) erweitert, damit diese zur Entwicklung analoger elektronischer Schaltungen eingesetzt werden können. Solche Hilfen sind zum Beispiel die Aufstellung und Lösung von symbolischen oder gemischt symbolisch/numerischen Netzwerkgleichungen nicht nur nach Strömen und Spannungen, sondern auch nach gesuchten Elementewerten. AI bietet weiterhin die Möglichkeit der Darstellung verschiedener Grafiken wie z.B. den Bodeplot, Ortskurven oder den Pol/Null-Stellenplan.

Insydes ist hierbei eine Abkürzung für Intelligent Symbolic Design System und erinnert an den Begriff insights (Einsichten), denn symbolische Analyseverfahren vermitteln Einsichten in das Verhalten analoger elektronischer Schaltungen und die Funktionsweise ihrer Elemente. Analog Insydes wird an der Universität Kaiserslautern entwickelt. Für die vorliegende Diplomarbeit stand die Version 1.0 zur Verfügung.

Aufbau von AI
Das Softwareprogramm AI besteht aus Funktionen und Datentypdefinitionen. Innerhalb der Funktionen sind z.B. Algorithmen implementiert, die für die Aufstellung der Netzwerkgleichungen sorgen oder für das Lösen eines Gleichungsystems verantwortlich sind. Unterschiedliche Netzwerkelemente wie z.B. Widerstand, Kapazität, Induktivität, unabhängige Strom- und Spannungsquellen oder alle Arten von gesteuerten Quellen (VCVS, CCVS, VCCS, CCCS) und unabhängige Quellen sind als Datentypen verwendbar.

Die AI-Software ist für mehrere Computer-Algebra-Systeme (MATHEMATICA, MAPLE, MACSYMA) in der jeweiligen Syntax des CAS entwickelt worden.

2.4 Einführung in AI am Beispiel des unbelasteten Spannungsteilers

Um den Einstieg in das Arbeiten mit AI zu erleichtern, soll in dem folgenden Beispiel eine einfache Schaltung, nämlich ein unbelasteter Spannungsteiler (Abbildung 2.1) mit Hilfe von AI analysiert werden.

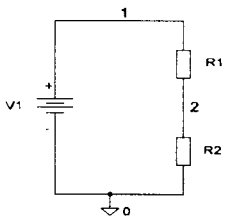


Abbildung 2.1: Spannungsteiler.

Die Spannung am Knoten 2 läßt sich wie bekannt mit der Formel

$$U_2 = \frac{R_2 \cdot V_1}{R_1 + R_2}$$

berechnen. Diese Formel soll nun mit AI hergeleitet werden.

Das Beispiel ist direkt aus dem MATHEMATICA-Notebook heraus kopiert. Dabei sind die hellgrauen Zonen (siehe unten) jeweils die Eingabe, die dunkelgrauen Zonen die dazugehörende Ausgabe in MATHEMATICA. Der Ablauf der Schaltungsanalyse ist in vier Schritte gegliedert, die im folgenden erklärt werden sollen.

1. Schritt:
Damit die Befehlsfunktionen von AI in MATHEMATICA verwendet werden können, muß als erstes das Master-Package von AI geladen werden.

2. Schritt:
Im zweiten Schritt kann nun die zu analysierende Schaltung in Form einer Netzliste eingegeben werden. Falls hier die Eingabe keine Syntax-Fehler enthält, erfolgt die Ausgabe *CheckedNetlist*, wobei die Netzliste ausgegeben wird.

3. Schritt
In dieser Phase wird die Netzwerkanalyse betrieben. Falls keine explizite Angabe für die Analysemethode vorgenommen wurde, wird standardmäßig die Modifizierte Knotenanalyse (MNA) angewendet. Die Ausgabe ist ein Gleichungssystem der Form **A*x = b**. Hierbei ist **A** die Koeffizientenmatrix, **b** der Vektor mit den unabhängigen Größen und **x** der Vektor mit den gesuchten Größen (Knotenspannungen, Zweigströme).

4. Schritt
In diesem letzten Schritt wird das Gleichungssystem **A*x = b** nach den Elementen von **x** aufgelöst. Die einzelnen Lösungen werden dabei in einer Liste ausgegeben.

1. Schritt: Laden der benötigten Packages

```
<< AnalogInsydes`Master`
```

2. Schritt: Eingabe der Netzliste

```
Spannungsteiler = Netlist[
  {V0, {1, 0}, V1},
  {R1, {1, 2}, R1},
   R2,  2, 0 , R2
```

```
CheckedNetlist  V0,  1, 0 , V1 ,  R1,  1, 2 , R1 ,  R2,  2, 0 , R2
```

3. Schritt: Aufstellen des Netzwerkgleichungssytems

```
Gleichungssystem1 =
 CircuitEquations Spannungsteiler, MatrixEquation -> False
```

$$I\$V0 + \frac{V\$1 - V\$2}{R1} == 0,\ \frac{V\$2}{R2} + \frac{-V\$1 + V\$2}{R1} == 0,\ V\$1 == V1\ ,\ V\$1, V\$2, I\$V0$$

4. Schritt: Lösen des Gleichungssystems

```
Lösungen1 = SolveCircuitEquations Gleichungssystem1
```

$$I\$V0 \to -\frac{V1}{R1 + R2},\ V\$2 \to \frac{R2\,V1}{R1 + R2},\ V\$1 \to V1$$

3 Dimensionierung eines Verstärkers mit AI/CAS

Im ersten Anwendungsbeispiel soll mit Hilfe von Analog Insydes ein einfacher Bipolar-NF-Verstärker in Emitterschaltung analysiert und dimensioniert werden. Die Schaltung ist in Abbildung 3.1 dargestellt.

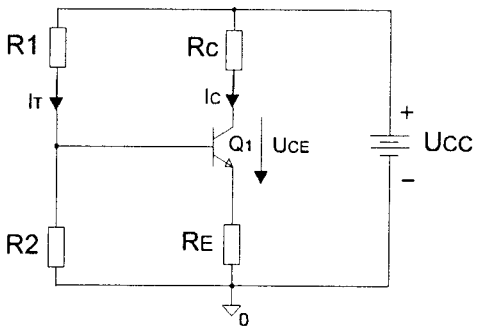


Abbildung 3.1: Einstufiger-Verstärker in Emitterschaltung

3.1 Die Emitterschaltung

Bei dieser Verstärkerschaltung dienen die Widerstände *R1* und *R2* zur Arbeitspunkteinstellung des NPN-Transistors Q1. Am Widerstand R_C fällt die Ausgangsspannung ab, der Widerstand R_E dient zur Stabilisierung des Arbeitspunktes und der Verstärkung. Die zu bestimmenden Größen sind alle Knotenspannungen, der Gesamtstrom der Schaltung und die Widerstandswerte. Insgesamt sind es 9 unbekannte Größen. Um diese 9 Größen ermitteln zu können, sind 9 Gleichungen notwendig. Zusammen ergibt dies ein lineares Gleichungssystem, welches sich dann zum Beispiel mit dem Gauß-Algorithmus [3] lösen läßt.

3.2 Die Spezifikation

Vorgegeben seien außer dem Schaltbild noch die Arbeitspunktkenndaten und eine zusätzliche Anforderung, die im folgenden aufgelistet werden:

Arbeitspunktdaten:
- Kollektorstrom I_C = 300 µA
- Teilerstrom I_T = 50 µA
- Betriebsspannung Ucc = 5 V
- Kollektor-Emitter Spannung U_{CE} = 2.5 V

Zusätzliche Anforderung:
- R_C/R_E = 10

3.3 Der Fixator

Um den Arbeitspunkt für den Transistor Q1 in Abbildung 3.1 angeben zu können wird dieser durch zwei Fixatoren ersetzt (Abbildung 3.2). Ein Fixator ist ein sogenanntes Ausgeartetes Element. Dieses Element ist eine Kombination aus Spannungsquelle und Stromquelle. Das Symbolzeichen ist in Abbildung 3.3 dargestellt. Da ein Fixator gleichzeitig durch zwei Elementebeziehungen ($U=U_{FIX}$ und $I=I_{FIX}$) beschrieben wird, bilden sich singuläre Analysegleichungssysteme, die in Bezug auf Ströme und Spannungen mehr Gleichungen als Variablen entstehen lassen (Überbestimmtheit). Da sich ein Fixator aus diesem Grund nicht physikalisch realisieren läßt, hat er nur in der Netzwerkanalyse seine Bedeutung.

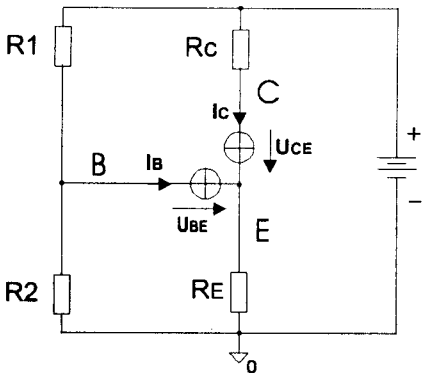


Abbildung 3.2: Transistor Q1 wird durch Fixatoren ersetzt

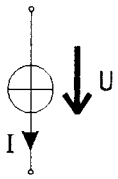


Abbildung 3.3: Symbolzeichen des Fixators

3.4 Dimensionierungsschritte

Die praktische Durchführung der symbolischen Schaltungsanlayse mit AI läßt sich in mehrere Schritte gliedern:

1. Schritt: Eingabe der Schaltung
2. Schritt: Herleitung der Netzwerkgleichungen
3. Schritt: Einsetzen zusätzlicher Gleichungen
4. Schritt: Lösen der Gleichungen
5. Schritt: Berechnen der gesuchten Größen

Zu jedem Schritt ist eine kurze Beschreibung der verwendeten Funktionen angegeben und darunter die Eingabe wie sie in MATHEMATICA erfolgen muß. Damit Ein- und Ausgabe unterschieden werden können, ist die Eingabe im hellgrauen Feld, die Ausgabe im dunkelgrauen Feld dargestellt.

1. Schritt:
Die Schaltung in Abbildung 3.2 wird in Form einer Netzliste eingegeben. In diesem Beispiel erhält die Netzliste die Bezeichnung EAmp1. EAmp1 ist die Abkürzung für den Schaltungsnamen (Emitteramplifier) und kann vom Anwender willkürlich gewählt werden. Die Funktion *Circuit* enthält als Parameter die Funktion *Netlist*. Als Parameter für die Funktion *Netlist* werden die Schaltungselemente mit ihren Knotenbezeichnungen und zusätzlichen optionalen Angaben angegeben. Für genauere Informationen über die einzelnen Funktionen (Befehle) von AI sei an dieser Stelle auf die Online-Hilfe von AI verwiesen.

```
EAmp1 = Circuit[
 Netlist[
        {Vcc, {1, 0}, Type -> VoltageSource, Value -> Vcc},
        {Rc, {1, C}, Type -> Resistor, Value -> Rc},
        {Re, {E, 0}, Type -> Resistor, Value -> Re},
        {R1, {1, B}, Type -> Resistor, Value -> R1},
        {R2, {B, 0}, Type -> Resistor, Value -> R2},
        {FIX1, {B, E}, {IB0, VBE0}},
        {FIX2, {C, E}, {IC0, VCE0}}

       ];
```

2. Schritt:
Mit der Funktion *CircuitEquations* werden die Netzwerkgleichungen automatisch von AI hergeleitet. Als Parameter muß die Schaltung (EAmp1) angegeben werden. Bei der Ausgabe der Netzwerkgleichungen kann zwischen der Matrixdarstellung oder der Darstellung einzelner Gleichungen gewählt werden. Als Default ist die Ausgabe in Matrixdarstellung eingestellt. Falls einzelne Gleichungen gewünscht sind, muß die Option *MatrixEquation-> False* als Parameterwert stehen. Der Rückgabewert der Funktion *CircuitEquations* enthält außer den Netzwerkgleichungen noch die Bezeichnungen der Knotenspannungen und Ströme. Diese werden zusammen in einer doppelten Liste an die Variable mit der Bezeichnung GS1 ausgegeben. Zur besseren Übersicht kann die Ausgabe mit der Funktion *TableForm* spaltenweise dargestellt werden.

```
GS1 = CircuitEquations FlatEAmp1, MatrixEquation -> False ;
```

```
I$Vcc + (V$1-V$B)/R1 + (V$1-V$C)/Rc == 0
(V$B)/R2 + (-V$1+V$B)/R1 == -IB0
(-V$1+V$C)/Rc == -IC0                          V$1
{(V$E)/Re == IB0 + IC0                         V$B , V$C }
V$1 == Vcc                                     V$E
V$B - V$E == VBE0                              I$Vcc
V$C - V$E == VCE0
```

3. Schritt

Um an das Ziel, ein lineares Gleichungssystem mit allen gesuchten Größen, zu gelangen, wird das Gleichungssystem GS1 um die vier gesuchten Widerstände *R1*, *R2*, R_C, R_E und eine zusätzliche Gleichung (R_C/R_E = 10) erweitert. Nach diesen Erweiterungen entsteht schließlich ein Gleichungssystem (GS3) mit 9 Gleichungen und 9 Unbekannten.

```
GS2 = {Insert[
 Insert[
  GS1[[1]],
  Rc  Re == 10, 8 ,
   IT + IC0 + I$Vcc == 0, 9 , GS1  2   ;
```

```
I$Vcc + (V$1-V$B)/R1 + (V$1-V$C)/Rc == 0
(V$B)/R2 + (-V$1+V$B)/R1 == -IB0
(-V$1+V$C)/Rc == -IC0                          V$1
{(V$E)/Re == IB0 + IC0                         V$B , V$C }
V$1 == Vcc                                     V$E
V$B - V$E == VBE0                              I$Vcc
V$C - V$E == VCE0
Rc/Re == 10
IC0 + IT + I$Vcc == 0
```

```
GS3 = {GS2[[1]],
  Insert[
   Insert[
    Insert[
     Insert[
      GS2[[2]],
      Rc, 6],
      Re, 7],
      R1, 8 ,
      R2, 9  ;
```

```
I$Vcc + (V$1-V$B)/R1 + (V$1-V$C)/Rc == 0
(V$B)/R2 + (-V$1+V$B)/R1 == -IB0               V$1
(-V$1+V$C)/Rc == -IC0                          V$B
                                               V$C
{(V$E)/Re == IB0 + IC0                         V$E , I$Vcc}
V$1 == Vcc                                     Rc
V$B - V$E == VBE0                              Re
V$C - V$E == VCE0                              R1
Rc/Re == 10                                    R2
IC0 + IT + I$Vcc == 0
```

4. Schritt:

Mit der Funktion *SolveCircuitEquations* wird das Gleichungssystem GS3 nach den unbekannten Größen aufgelöst. Die Variable mit der Bezeichnung L1 enthält alle Lösungen als symbolische Formeln.

```
L1 = SolveCircuitEquations GS3 ;
V$E → (IB0+IC0) (Vcc-VCE0) / (IB0+11 IC0)
V$C → (IB0 Vcc+IC0 (Vcc+10 VCE0)) / (IB0+11 IC0)
I$Vcc → -IC0 - IT
Re → (Vcc-VCE0) / (IB0+11 IC0)
V$B → (IB0 (VBE0+Vcc-VCE0) +IC0 (11 VBE0+Vcc-VCE0)) / (IB0+11 IC0)
V$1 → Vcc
R2 → (-IB0 (VBE0+Vcc-VCE0) +IC0 (-11 VBE0-Vcc+VCE0)) / ((IB0+11 IC0) (IB0-IT))
Rc → (10 (Vcc-VCE0)) / (IB0+11 IC0)
R1 → (IB0 -VBE0+VCE0 +IC0 -11 VBE0+10 Vcc+VCE0) / (IB0+11 IC0 IT)
```

5. Schritt:

Mit der Funktion *ReplaceAll* werden die Zahlenwerte der bekannten Größen in die zuvor hergeleiteten Formeln eingesetzt und dadurch die gesuchten Spannungs-, Strom- und Widerstandswerte berechnet. Die Zahlenwerte befinden sich in der Liste mit der Variablenbezeichnung dp.

```
TableForm ReplaceAll L1, dp
V$E → 0.22933696639418703`
V$C → 2.72933696639418643`
I$Vcc → -7/20000
Re → 756.8876778685082 12`
V$B → 0.973336963941864 75`
V$1 → 5
R2 → 20709.297157323121`
Rc → 7568.87677868508212`
R1 → 80533.2606721162846`
```

3.5 Ergebnisse

Zum Test der oben ermittelten Widerstandswerte wurde eine numerische Simulation mit PSPICE durchgeführt. Dabei wurde die in Abbildung 3.1 dargestellte Verstärkerschaltung mit den berechneten Widerstandswerten für R1, R2, Rc und R_E eingegeben. In der Tabelle 3.1 sind die Simulationsergebnisse aufgeführt.

Spezifikation	Ergebnisse mit AI/ Mathematica		Verifikation mit SPICE
	Dimensionierung	Erwartungswerte	
I_c = 300µA	R_1 = 80.5 kOhm	I_c = 300µA	I_c = 301 µA
I_T = 50µA	R_2 = 20.7 kOhm	I_T = 50µA	I_T = 50 µA
U_{CE} = 2.5 V	R_c = 7.57 kOhm	U_{CE} = 2.5 V	U_{CE} = 2.49 V
U_{CC} = 5V	R_e = 757 Ohm	I_B = 2.3 µA	I_B = 2.66 µA
R_c/R_e = 10		U_{BE} = 0.744 V	U_{BE} = 0.749 V

Tabelle 3.1: Ergebnisse nach Dimensionierung und Simulation

4 Dimensionierung einer Differenzverstärker-Stufe mit AI/MATHEMATICA

Im zweiten Anwendungsbeisiel wird eine Differenzverstärkerstufe betrachtet (siehe Abbildung 4.1). Auch für diese Schaltung ist eine Spezifikation vorgegeben. Ziel ist es wieder, mathematische Gleichungen herzuleiten, die den Zusammenhang zwischen Schaltungselementen und der Spezifikation wiedergeben.

Anwendungsbeispiel 2
Schaltungsdimensionierung ohne Arbeitspunktvorgabe

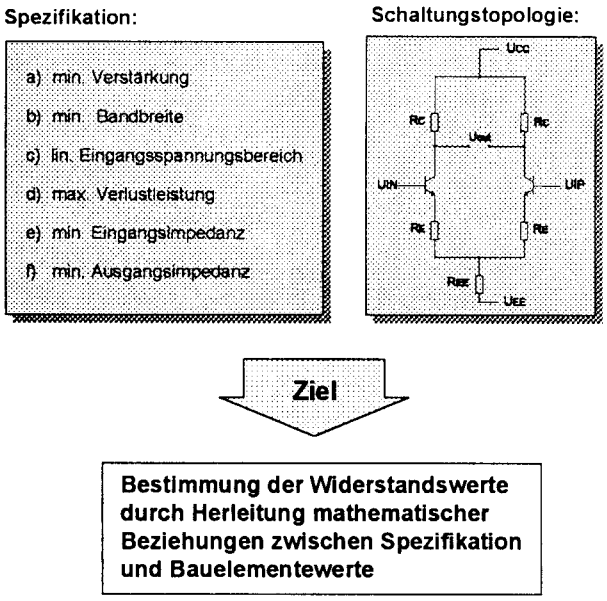


Abbildung 4.1 Schaltungstopologie und Spezifikation

In Abbildung 4.2 wird der Zusammenhang zwischen der Spezifikationsgröße „Verstärkung" und den Schaltungselementen R_C, R_E und dem Kollektorstrom I_C angedeutet. Die Verstärkung ist eine von den Schaltungselementen abhängige Größe. Die Widerstände R_C, R_E und der Kollektorstrom I_C sind unabhängige Größen. Sind z.B. die Werte der Widerstände R_C, R_E und der Kollektorstrom I_C bekannt, so kann die Verstärkung der Schaltung berechnet werden. Umgekehrt kann man aber auch auf einen Widerstand (z.B. R_C) rückschließen, falls die Größen R_E, I_C und die Verstärkung gegeben sind.

Beispiel:

Verstärkung = f (R_C, R_E, I_C)

Umkehrung

R_C = f (Verstärkung, R_E, I_C)

Abbildung 4.2: Abhängigkeiten zwischen Spezifikationsgröße und Schaltungselementen.

Gelingt es nun, durch zusätzliche Gleichungen die Anzahl der unabhängigen Größen auf eine unabhängige zu reduzieren, so kann der Wert dieser Größe anhand der Spezifikationsdaten ermittelt werden. Dieser Wert kann dann für die Bestimmung der anderen unabhängigen Größen eingesetzt werden.

Abbildung 4.3 zeigt eine mögliche Darstellung der oben genannten Situation. Als einzige unabhängige Größe ist der Arbeitsstrom I_C gewählt (dicke Linie unten). Für I_C kann je nach Spezifikation eine Lösungsmenge entstehen (Bereich zwischen unterer und oberer Grenze).

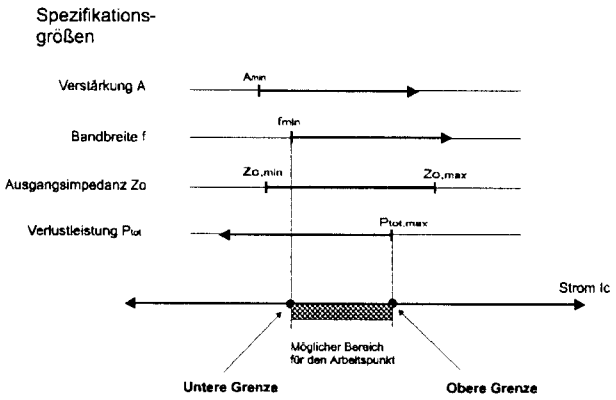


Abbildung 4.3: Abbilden der Spezifikationsgrößen auf die unabhängige Stromgröße I_C

Im folgenden wird nun die Dimensionierung der DV-Stufe von Abbildung 4.1 mit Hilfe der Werkzeuge AI/MATHEMATICA durchgeführt. Der Dimensionierungsablauf ist in 4 Schritte aufgeteilt. Abbildung 4.4 zeigt diesen Ablauf.

Ablauf der Dimensionierung

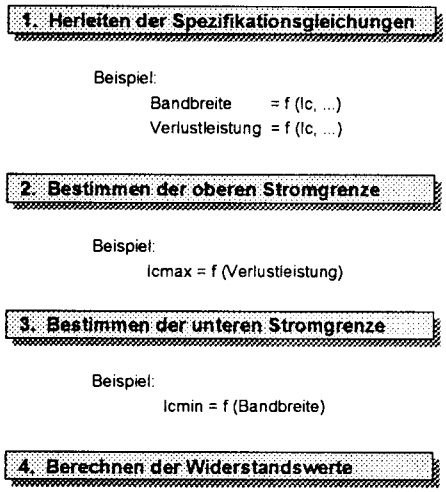
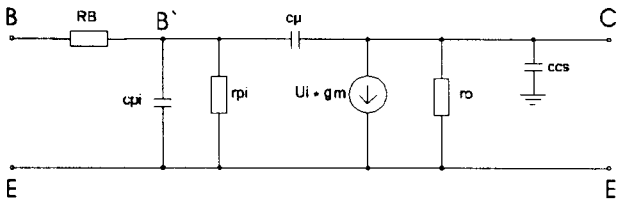


Abbildung 4.4 Dimensionierungsablauf

4.1 Herleitung der Spezifikationsgleichungen

In diesem Abschnitt werden für die Spezifikationsgrößen analytische Formeln hergeleitet. Bei den Spezifikationsgrößen handelt es sich um die Verstärkung A, die Bandbreite f_{-3dB}, den linearen Eingangsspannungsbereich U_{idcp}, die Verlustleistung und die Ein- und Ausgangsimpedanzen Z_{in} und Z_{out}. Soweit es möglich ist, werden hierfür die Gleichungen durch Handanalyse bestimmt. Um die Schaltungsgleichungen herleiten zu können, werden die Transistoren werden durch Ersatzschaltbilder ersetzt. Zur Berechnung der Übertragungsfunktion wird das vollständige Transistor-Kleinsignalmodell verwendet. Lediglich die Bahnwiderstände r_C und r_E sind vernachlässigt.



Der Handanalyse liegen allerdings vereinfachte Modelle zugrunde. Nachfolgend sind die Dimensionierungsgleichungen zusammengestellt, wobei versucht wird, abhängige Variablen durch die unabhängige Variable I_C auszudrücken.

Verlustleistung

Für die Verlustleistung P_{tot} der DV-Stufe gilt

$$P_{tot} = I_{EE}\left(U_{CC} - U_{EE}\right) \qquad (4.1)$$

Linearer Eingangsspannungsbereich

Der lineare Eingangsspannungsbereich U_{idcp} gibt den oberen Grenzwert der Eingangsspannung an, welcher noch unverzerrt (linear) verstärkt wird. Die Formel hierfür lautet:

$$U_{idcp} = 2 \cdot U_T + 2 \cdot U_{RE} \qquad (4.2)$$

Ausgangsimpedanz

Es gilt:

$$Z_{out} = 2\left(\frac{R_C \cdot R_o}{R_C + R_o}\right) \approx 2\left(\frac{R_C \cdot r_o}{R_C + r_o}\right) \qquad (4.3)$$

R_o ist der Ausgangswiderstand der Transistoren unter Berücksichtigung der Emittergegenkopplung, r_o der Ausgangswiderstand des Transistors. Da die Differenzverstärkerstufe aus zwei symmetrischen Emitterschaltungen aufgebaut ist, kommt der Faktor 2 zustande. Für den differentiellen Ausgangswiderstand r_o gilt:

$$r_o = \frac{U_A}{I_c} \qquad (4.4)$$

Wird die Gleichung (4.4) in die Gleichung (4.3) eingesetzt, so entsteht die Formel (4.5) für die Ausgangsimpedanz der Schaltung. U_A ist die Early-Spannung, I_C der Kollektorstrom.

$$Z_{out} = 2\left(\frac{R_C \cdot U_A}{I_C \cdot R_C + U_A}\right) \qquad (4.5)$$

Eingangsimpedanz

Der Eingangswiderstand Z_{in} der Schaltung wird durch den differentiellen Widerstand r_{pi} der beiden Transistoren Q1 und Q2 und die Emittergegenkopplung R_E bestimmt.

$$Z_{in} = 2 \cdot r_{pi}\left(1 + g_m \cdot R_E\right) \qquad (4.6)$$

Nun gilt

$$r_{pi} = \frac{\beta}{g_m} \qquad (4.7)$$

und

$$g_m = \frac{I_C}{U_T} \qquad (4.8)$$

Durch Einsetzen der Gleichungen (4.7) und (4.8) in (4.6) erhält man schließlich

$$\boxed{Z_{in} = 2 \cdot BF\left(\frac{U_T}{I_C} + R_E\right)} \qquad (4.9)$$

Übertragungsfunktion

Die Gleichungen für die Verstärkung und für die Bandbreite werden aus der Übertragungsfunktion der Schaltung gewonnen. Hierfür gilt

$$\ddot{U}F = \frac{U_{out}}{U_{in}} = \frac{U_{out}}{UIP - UIN} \qquad (4.10)$$

Sie wird mit Hilfe von Analog Insydes und MATHEMATICA hergeleitet. Zu diesem Zweck wird die Schaltung in Abbildung 4.1 in Form einer Netzliste unter MATHEMATICA eingegeben. Durch nachfolgendes Kürzen bekommt man die Übertragungsfunktion ÜF:

```
ÜF = -(Rc (Re - gm ro rpi + cu Re ro s + cu Re rpi s +
      cpi Re rpi s + cu ro rpi s + cu gm Re ro rpi s + cu cpi Re ro rpi s^2)) /
  (RB Rc + RB Re + Rc Re + RB ro + Re ro + Rc rpi + Re rpi + ro rpi + gm Re ro rpi +
   cjs RB Rc Re s + cjs RB Rc ro s + cu RB Rc ro s + cu RB Re ro s + cjs Rc Re ro s +
   cu Rc Re ro s + cu RB Rc rpi s + cpi RB Rc rpi s + cu RB Re rpi s + cpi RB Re rpi s +
   cjs Rc Re rpi s + cu Rc Re rpi s + cpi Rc Re rpi s + cu RB ro rpi s + cpi RB ro rpi s +
   cjs Rc ro rpi s + cu Rc ro rpi s + cu gm RB Rc ro rpi s + cpi Re ro rpi s +
   cu gm RB Re ro rpi s + cjs gm Rc Re ro rpi s + cu gm Rc Re ro rpi s +
   cjs cu RB Rc Re ro s^2 + cjs cu RB Rc Re rpi s^2 + cjs cpi RB Rc Re rpi s^2 +
   cjs cu RB Rc ro rpi s^2 + cjs cpi RB Rc ro rpi s^2 + cu cpi RB Rc ro rpi s^2 +
   cu cpi RB Re ro rpi s^2 + cjs cpi Rc Re ro rpi s^2 + cu cpi Rc Re ro rpi s^2 +
   cjs cu gm RB Rc Re ro rpi s^2 + cjs cu cpi RB Rc Re ro rpi s^3 ;
```

An der gebrochenrationalen Darstellung der Übertragungsfunktion ÜF sieht man, daß die komplexe Frequenzvariable s enthalten ist. Die Ausgangsspannung U_{out} ist also frequenzabhängig. Weiter ist zu erkennen, daß die Übertragungsfunktion 3 Polstellen (Nennerterm) und 2 Nullstellen (Zählerterm) besitzt. Die Polstellen sind für die Bandbreite der Schaltung verantwortlich, auf die aber etwas später eingegangen werden soll. Zunächst soll die Verstärkungsformel für tiefe Frequenzen (f -> 0) bestimmt werden.

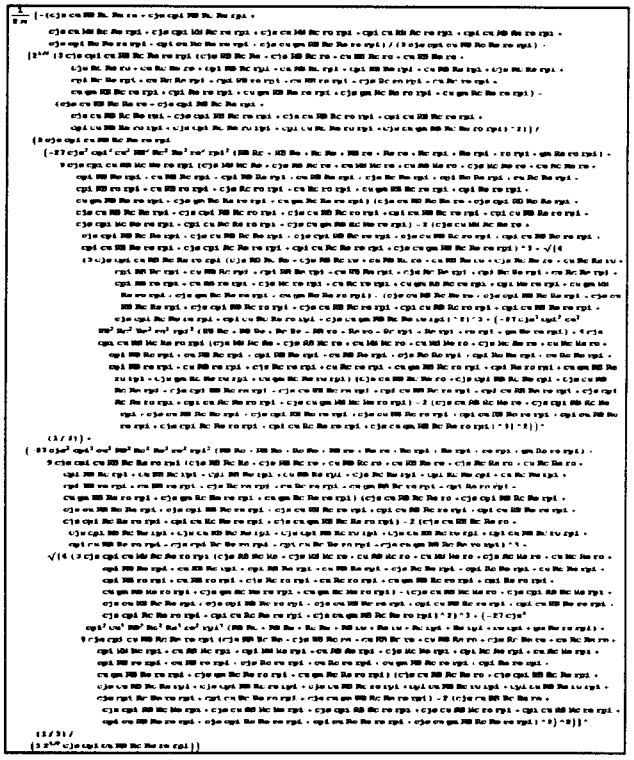
Verstärkung

Die DC-Kleinsignalverstärkung wird dadurch bestimmt, daß die Frequenzvariable s der ÜF gleich Null gesetzt wird. Man erhält den folgenden Ausdruck, womit die niederfrequente Verstärkung der Schaltung berechnet werden kann:

$$\boxed{A = -\frac{Rc\ Re - gm\ ro\ rpi}{RB\ Rc + RB\ Re + Rc\ Re + RB\ ro + Re\ ro + Rc\ rpi + Re\ rpi + ro\ rpi + gm\ Re\ ro\ rpi}} \qquad (4.11)$$

Bandbreite

Aus dem Nennerpolynom der Übertragungsfunktion läßt sich die Bandbreite-Formel herleiten. Wird nämlich das Nennerpolynom der Übertragungsfunktion gleich Null gesetzt und nach der Frequenzvariablen s aufgelöst, so entstehen 3 Pol-Gleichungen (höchster Grad = 3) für s. Da bis hierhin nur mit Symbolzeichen gerechnet wurde, sind dementsprechend diese Ergebnisse ziemlich unübersichtlich. Wie lang diese Ausdrücke werden können, soll am Beispiel der ersten Polgleichung nachfolgend gezeigt werden.



Im Augenblick soll die kurze Darstellung der Pol-Gleichungen in (4.12), (4.13) und (4.14) für die weiteren Maßnahmen genügen.

Pol_1 = f (Schaltungselemente) (4.12)

Pol_2 = f (Schaltungselemente) (4.13)

Pol_3 = f (Schaltungselemente) (4.14)

4.2 Bestimmung der oberen Stromgrenze

Aus bestimmten Anforderungen, die an eine Schaltung gestellt werden, lassen sich Informationen gewinnen, mit denen sich die obere Grenze des Arbeitspunktstroms festlegen läßt. Als Beispiel kann die Vorgabe der maximalen Verlustleistung $P_{tot,max}$ als eine Begrenzung des Arbeitspunktes (Arbeitsstromes) nach oben hin angesehen werden.

Die Verlustleistung wird in Bezug auf unseren Differenzverstärker mit der Gleichung (4.1) berechnet. Durch einfaches Umstellen kann der Arbeitsstrom I_{EE} ermittelt werden, wenn Verlustleistung und die Betriebsspannungen vorgegeben sind. Da für die Verlustleistung einer Schaltung normalerweise ein Höchstwert angegeben wird und die Betriebsspannungen konstant sind, müßte bei der Umstellung der Gleichung (4.1) nach I_{EE} eine Ungleichung entstehen. In unserem Fall interessiert jedoch gerade der Wert des Stromes, mit dem die maximale Verlustleistung erreicht wird. Somit kann die Gleichung (4.15) geschrieben werden.

$$I_{EE,\max} = \frac{P_{tot,\max}}{U_{CC} - U_{EE}} \qquad (4.15)$$

In der Praxis wird der Arbeitspunktstrom I_{EE} weit unterhalb der maximal vorgegebenen Verlustleistung gelegt, um z.B. eine längere Lebensdauer der Bauteile und somit auch der Schaltung zu erreichen.

Weitere Angaben, wie die minimale Eingangs- bzw. Ausgangsimpedanz der Schaltung, begrenzen ebenfalls den Arbeitsstrom nach oben hin. Aus der Gleichung (4.9) für die Eingangsimpedanz entsteht die Gleichung (4.16). Dabei ist $Z_{in,min}$ die geforderte Mindest-Eingangsimpedanz der Schaltung, *BF* das ideale Gleichstromverhältnis und U_T die Temperaturspannung bei 300 Kelvin.

$$I_{C,\max 1} = \frac{2 \cdot BF \cdot U_T}{Z_{in,\min} - 2 \cdot BF \cdot R_E} \qquad (4.16)$$

Für die Ausgangsimpedanz-Gleichung (4.5) gilt nach der Umstellung die Gleichung

$$I_{C,\max 2} = \frac{U_A\left(2 \cdot R_C - Z_{out,\min}\right)}{R_C \cdot Z_{out,\min}} \qquad (4.17)$$

Hier ist $Z_{out,min}$ die geforderte Mindest Ausgangsimpedanz der Schaltung, U_A die Early-Spannung und R_C der Kollektorwiderstand.

Zwischen dem Strom I_{EE} aus (4.15) und den Strom I_C aus (4.16) oder (4.17) besteht die Beziehung (4.18). Hierbei wird der Basisstrom wegen $I_B << I_C$ vernachlässigt.

$$I_C = \frac{I_{EE}}{2} \qquad (4.18)$$

Setzt man schließlich (4.18) in (4.15) ein, so erhält man

$$\boxed{I_{C,\max 3} = \frac{P_{tot,\max}}{2\left(U_{CC} - U_{EE}\right)}} \qquad (4.19)$$

In den Gleichungen (4.16) und (4.17) sind die noch zu bestimmenden unabhängigen Größen R_C und R_E enthalten. Um bei diesen Gleichungen den jeweiligen Strom I_C berechnen zu können, sind zwei weitere unabhängige Gleichungen mit den Variablen R_C und R_E nötig. Diese zusätzlich notwendigen Gleichungen werden mit den Gleichungen (4.2) und (4.11) auf folgende Art hergeleitet:

Erste notwendige Gleichung:
Die Spannung U_{RE} in der Gleichung (4.2) ist gerade der Spannungsabfall am Widerstand R_E. Für ihn kann auch geschrieben werden:

$$U_{RE} = I_C \cdot R_E \qquad (4.20)$$

Setzt man (4.20) in (4.2) ein und formt nach R_E um, so erhält man

$$\boxed{R_E = \frac{U_{idcp} - 2 \cdot U_T}{2 \cdot I_C}} \qquad (4.21)$$

Zweite notwendige Gleichung:
An der Verstärkungsformel aus (4.11) werden ebenfalls Substitutionen vorgenommen. Wie an ihr zu erkennen ist, ist die Verstärkung von den Größen g_m, R_B, R_C, R_E, r_o und r_{pi} abhängig. Für die Transistorparameter können die entsprechenden Gleichungen (4.4), (4.7) und (4.8) eingesetzt werden. R_B ist der Basis-Bahnwiderstand des Transistors und wird aus der SPICE-Transistorparameterliste entnommen (R_B = 250 Ohm). Für R_E wird die Gleichung (4.21) eingesetzt. Nach der gesamten Substitution ist die Verstärkung nur noch abhängig vom Strom I_C und von dem Widerstand R_C. Dies soll die Funktion (4.22) darstellen.

$$\textit{Verstärkung } A = \mathrm{f}\left(I_C, R_C\right) \qquad (4.22)$$

Um die zweite Gleichung zu bekommen, wird für die Verstärkung A aus der Gleichung (4.22) der geforderte Verstärkungswert aus der Spezifikation eingesetzt und nach R_C aufgelöst.

$$\boxed{R_C = f(I_C)} \qquad (4.23)$$

Nachdem also die Gleichungen (4.23), (4.22), (4.19), (4.17) und (4.16) ermittelt wurden, können die Ströme ($I_{C,max1}$, $I_{C,max2}$, $I_{C,max3}$) bestimmt werden. Für die obere Stromgrenze I_{Cmax} wird der kleinste Wert der Lösungen verwendet (siehe 4.24). Dieser darf nicht überschritten werden, da sonst mindestens einer der geforderten Spezifikationen der Schaltung nicht eingehalten wird.

$$I_{Cmax} = \mathrm{Min}\left[I_{C,max1},\ I_{C,max2},\ I_{C,max3},\ \ldots\right] \qquad (4.24)$$

4.3 Bestimmung der unteren Stromgrenze

Damit eine vorgegebene Spezifikation eingehalten werden kann, muß ein Mindeststrom fließen. Umgekehrt hängt dieser Strom von der Spezifikation ab. Bei unserem Schaltungsbeispiel sind der lineare Eingangsspannungsbereich, die Mindestverstärkung und die Mindestbandbreite für die untere Stromgrenze verantwortlich. Für diese Eigenschaften wurden unter 4.1 bereits Formeln hergeleitet. Im folgenden wird nun gezeigt, wie aus den Pol-Gleichungen die Bandbreite-Formel bestimmt wird. Danach wird diese Bandbreite-Formel für die Berechnung des Mindeststroms (untere Grenze des Arbeitspunktes) verwendet.

Die Bestimmung der Bandbreite-Formel

Bei der Herleitung der Spezifikationsgleichungen in Abschnitt 4.1 wurden für die Bandbreite-Formel gleich drei mögliche Lösungen (4.12), (4.13) und (4.14) berechnet. Mit diesen drei Gleichungen werden weiter unten, durch Einsetzen von Zahlenwerten die Polstellen der Übertragungsfunktion ermittelt. Für die Bandbreite der Schaltung ist dabei nur ein Pol verantwortlich, der dominante Pol. Was unter diesem Begriff zu verstehen ist, wird im folgenden erläutert.

Die Übertragungsfunktion ist eine gebrochen rationale Funktion in der komplexen Frequenzvariablen $s = \sigma + j\,\omega$. Sie läßt sich als Quotient eines Zählerpolynoms und eines Nennerpolynoms darstellen. Die Nullstellen des Nennerpolynoms stellen die Pole der Übertragungsfunktion dar. Ursache für diese Pole sind die im Signalweg der Schaltung liegenden Tiefpäße, verursacht durch transistorinterne Widerstände und Kapazitäten. Liegen nun alle Pole auf der negativen reellen Achse der komplexen Frequenzebene und gibt es einen dominanten Pol P1, so bestimmt dieser die Bandbreite

der Schaltung. Es gilt dann ω_{-3dB} = |P1|. Der dominante Pol liegt am dichtesten beim Ursprung mit einem Abstand, der höchstens der Hälfte des nächsten Pols entspricht.

Die Übertragungsfunktion unserer Bespielschaltung weist 3 Pole auf, denn bei den Ersatzschaltungen der Transistoren wurden drei parasitäre Kapazitäten (C_{pi}, C_μ und C_{cs}) berücksichtigt [5]. Sie lassen sich durch folgende Beziehungen (4.25), (4.26), (4.27) berechenen:

$$C_{pi} = TF \cdot g_m \qquad (4.25)$$

$$C_\mu = \frac{CJC}{\left(1+\frac{UCB}{VJC}\right)^{MJC}} \qquad (4.26)$$

$$C_{cs} = \frac{CJS}{\left(1+\frac{UCS}{VJS}\right)^{MJS}} \qquad (4.27)$$

Die Spannung *UCB* in der Gleichung (4.26) gibt die Kollektor-Basis-Spannung an. Für diese Schaltung kann *UCB* durch die Gleichung (4.28) ersetzt werden. Für die Kollektor-Substrat-Spannung *UCS* aus (4.27) gilt dasselbe. Sie wird durch (4.29) ersetzt. Bei der Gleichung (4.25) für die eingangsseitige Kapazität C_{pi} kann die Steilheit g_m durch die bekannte Formel (4.8) ersetzt werden.

$$UCB = U_{CC} - I_C \cdot R_C \qquad (4.28)$$

$$UCS = U_{CC} - I_C \cdot R_C \qquad (4.29)$$

Nach dem Einsetzen entstehen die neuen Gleichungen (4.30), (4.31) und (4.32). Wie zu erkennen ist, sind die parasitären Kapazitäten C_{pi}, C_μ und C_{cs} vom Kollektorstrom I_C abhängig.

$$C_{pi} = \frac{TF \cdot I_C}{U_T} \qquad (4.30)$$

$$C_\mu = \frac{CJC}{\left(1+\frac{U_{CC} - I_C \cdot R_C}{VJC}\right)^{MJC}} \qquad (4.31)$$

$$C_{cs} = \frac{CJS}{\left(1+\frac{U_{CC} - I_C \cdot R_C}{VJS}\right)^{MJS}} \qquad (4.32)$$

Die drei Pol-Gleichungen (4.12), (4.13) und (4.14) zeigen ebenfalls eine Abhängigkeit von den Schaltungselementen. Diese Abhängigkeit ist in folgender Funktionsgleichung dargestellt:

$$(Pol\_1, Pol\_2, Pol\_3) = f(C_{pi}, C_{cs}, C_\mu, R_B, R_C, R_E, r_{pi}, r_o, g_m) \qquad (4.33)$$

Werden bei (4.33) die Schaltungselemente durch die Beziehungen (4.4), (4.7), (4.8), (4.21), (4.23), (4.30), (4.31), (4.32) ersetzt, so entsteht eine neue Funktion, die die Polstellen nur noch in eine Abhängigkeit vom Strom I_C stellt.

$$(Pol\_1, Pol\_2, Pol\_3) = f(I_C) \qquad (4.34)$$

Für den Strom I_C in der Funktion (4.34) wird zunächst die in Abschnitt 4.2 ermittelte obere Stromgrenze I_{Cmax} eingesetzt:

$$(Pol\_1, Pol\_2, Pol\_3) = f(Ic = Ic_{max}) \qquad (4.35)$$

Aus den resultierenden Ergebnissen von 4.35 bestimmt der kleinste Wert den dominanten Pol:

$$\text{dominanter Pol P1} = \text{Min}[Pol\_1, Pol\_2, Pol\_3] \qquad (4.36)$$

Damit ist bekannt, welche Polgleichung für die Berechnung der Bandbreite zur Anwendung kommt.

Für die Bandbreite erhält man dann

$$\omega_{-3dB} = -P1 \qquad (4.37)$$

Berechnung des Mindeststroms

Für die Berechnung des Mindeststromes I_{Cmin} aus der Bandbreite-Formel (4.37) wird das Newton-Verfahren angewendet, da im vorliegenden Fall MATHEMATICA keine symbolische Lösung für I_C finden konnte

Das Verfahren ist bereits als fertige Funktion in MATHEMATICA implementiert. Als Startwert setzt man die in Kapitel 4.2 ermittelte obere Stromgrenze I_{Cmax} ein. Das Ergebnis wird nach zahlreichen Iterationsschritten, die übrigens verschieden eingestellt werden können [10], ausgegeben. Falls gewünscht, kann dieser Wert für eine weitere Näherung wieder als Startwert verwendet werden, um eine genauere Lösung zu bekommen.

4.4 Ermittlung der Widerstandswerte

In den Abschnitten 4.2 und 4.3 wurden die obere und untere Grenze des Arbeitsstromes festgelegt. Zwischen diesen Grenzen, die durch die Werte I_{Cmin} und I_{Cmax} gegeben sind, kann ein Wert als Arbeitspunkt I_{AP} für die Schaltung in Abbildung 4.1 verwendet werden.

$$I_{AP} = [I_{Cmin} \ldots\ldots I_{Cmax}] \qquad (4.38)$$

Widerstand R_{EE}
Bei unserer Beispielschaltung wird der Strom I_{AP} mit dem Widerstand R_{EE} erzeugt. Dieser Widerstand ist durch die Gleichung (4.39) bestimmt.

$$\boxed{R_{EE} = \frac{U_{REE}}{I_{EE}}} \qquad (4.39)$$

Vernachlässigt man die Widerstände R_E, so kann der Spannungsabfall am Widerstand R_{EE} wie folgt berechnet werden:

$$U_{REE} = U_{EE} - U_{BE} \qquad (4.40)$$

Die Basis-Emitterspannung U_{BE} berechnet sich mit der Formel (4.41). Für I_C wird dabei der Strom I_{CAP} eingesetzt.

$$U_{BE} = U_T \cdot \ln\left(\frac{I_{CAP}}{I_S}\right) \qquad (4.41)$$

Durch Einsetzen von (4.41), (4.40) und (4.18) in die Widerstandsgleichung (4.39) erhält man R_{EE} .

$$\boxed{R_{EE} = \frac{U_{EE} - U_T \cdot \ln\left(\frac{I_{CAP}}{I_S}\right)}{2 \cdot I_{CAP}}} \qquad (4.42)$$

Widerstände R_C und R_E
Die Kollektorwiderstände R_C und die Emitterwiderstände R_E der Schaltung lassen sich mit den Formeln (4.21) und (4.23) durch Einsetzten des Arbeitspunktstromes I_{AP} berechnen.

4.5 Ergebnisse

Zum Test der unter Abschnitt 4 beschriebenen Methode zur Dimensionierung der Differenzverstärkerstufe (Abbildung 4.1) wurden verschiedene Zahlenbespiele untersucht. In der Tabelle 4.1 sind die Spezifikationsgrößen aufgelistet. In der mittleren Spalte der Tabelle ist ein Satz von Werten eingetragen, die mit der Schaltung erreicht werden sollen. Nach der Dimensionierung der DV-Stufe mit der oben dargestellten Methode wurde die Schaltung mit den berechneten Widerstandswerten in einen Simulator eingegeben und getestet. Die Ergebnisse des Simulators sind in der dritten Spalte der Tabelle aufgeführt.

Spezifikations-größen	Angestrebte-Ergebnisse	Simulations-ergebnisse
Verstärkung	> 15	14.7
Bandbreite	> 25 MHz	25.4 MHz
Linearer Eingangsspannungsbereich	> 4 U_T	4 U_T
Verlustleistung	< 2 mW	0.8 mW
Eingangsimpedanz	> 20 kOhm	76.2 kOhm
Ausgangsimpedanz	> 10 kOhm	19.2 kOhm

Tabelle 4.1: Dimensionierungsergebnis

Wie zu bemerken ist, werden außer der Verstärkung sonst alle anderen Vorgaben eingehalten. Die geforderte Verstärkung konnte deshalb nicht ganz erreicht werden, da in den Gleichungen keine Hochstromeffekte bei den Transistoren berücksichtigt wurden [11].

5 Zusammenfassung

Zum Aufgabengebiet des Schaltungsentwicklers gehört die Dimensionierung analoger und digitaler Schaltungen. Dabei stehen dem Ingenieur zahlreiche Hilfsmittel zur Verfügung. Es existieren verschiedene Simulationswerkzeuge, mit denen sich analoge, digitale oder gemischt analog/digitale Schaltungen verifizieren lassen. Diese Simulationswerkzeuge arbeiten nach numerischen Verfahren, bei denen für jedes Schaltungselement ein Zahlenwert vor der Simulation eingesetzt werden muß.

In letzter Zeit werden aber auch Versuche gemacht, Werkzeuge zu entwickeln, die eine symbolische Analyse elektronischer Schaltungen ermöglichen. Der Vorteil dieser analytischen Methode ist, daß exakte Werte für die zu dimensionierenden Größen berechnet werden können. Unter Abschnitt 2 wurde ein solches Werkzeug mit dem Namen Analog Insydes (AI) vorgestellt. In Kombination mit einem Computer-Algebra-System (CAS) kann es für die Schaltungsberechnung eingesetzt werden. In den Abschnitten 3 und 4 sind zwei Schaltungsbeispiele dargestellt, die mit Hilfe von AI/CAS dimensioniert wurden. Bei der ersten Schaltung wurden die Widerstandswerte (R_C, R_E, R_1, R_2) der Emitterschaltung bestimmt. Hierzu war eine Spezifikation vorgegeben, die die Arbeitspunktdaten und eine zusätzliche Forderung enthielt. Aus diesen Vorgaben wurden mit AI/CAS Netzwerkgleichungen hergeleitet, mit denen sich schließlich die Widerstandswerte berechnen ließen. Im nächsten Beispiel (Abschnitt 4) wurde eine Differenzverstärkerstufe mit AI/CAS dimensioniert. Für eine vorgegebene Spezifikation waren die Widerstandswerte für R_C, R_E und R_{EE} zu bestimmen. Da in diesem Fall keine Angaben für den Arbeitspunkt gegeben waren (in der Praxis der häufigste Fall), mußte dieser zuerst bestimmt werden. Hierzu wurden Spezifikationsgleichungen hergeleitet, mit denen eine obere und eine untere Stromgrenze (Grenzen für den Arbeitspunktstrom) definiert werden konnte. Innerhalb dieser Grenzen konnte dann für die Bestimmung der Widerstandswerte ein Arbeitsstrom angenommen werden.

In beiden Beispielen (Kapitel 3 und 4) ist es gelungen, eine Dimensionierungsroutine zu entwickeln, die jederzeit mit neuen Zahlenwerten der Spezifikationsgrößen und Transistorparameter „gefüttert" werden kann. Dadurch sind sehr kurze Entwicklungszeiten beim Redesign möglich. Die Programmierung der Routinen erfolgt in der jeweiligen CAS-Sprache. Bei MATHEMATICA ist es auch möglich, C-Code zu generieren, um höhere Verarbeitungsgeschwindigkeiten zu erreichen. Leider enthielt die erste AI-Version zur Entstehungszeit dieser Arbeit keine grafische Schaltungseingabe. Bei größeren Schaltungen ist das Eingeben der Netzliste dadurch recht mühsam. Es muß immer darauf geachtet werden, daß die Schaltungselemente an den richtigen Knotenpunkten hängen. Nachteilig wirkt sich auch die Abarbeitungszeit eines Befehls innerhalb des CAS aus, wenn nämlich die zu analysierende Schaltung komplizierter wird. Der Grund liegt darin, daß bei der Herleitung der Netzwerkgleichungen die Anzahl der Abhängigkeiten zwischen den Schaltungsgrößen steigt und dadurch auch die Gleichungen wachsen. Bezogen auf den rechnerinternen Ablauf bedeutet das höhere Datenmengen, die im Speicher gespeichert und verwaltet werden müssen.

Auf der anderen Seite fördert die mathematische Analyse mit AI/CAS das Verständnis für Analogschaltungen, da Gleichungen hergeleitet werden können, die das Schaltungsverhalten in Abhängigkeit der Schaltungselemente darstellen. Zum Beispiel sieht man an der Verstärkungsformel der DV-Stufe, welche Abhängigkeiten zwischen der Verstärkung und den Widerständen R_C und R_E und der Steilheit g_m des Transistors bestehen. Der interaktive Dimensionierungsprozeß mit AI/CAS erlaubt dem Entwickler auch eigene Gleichungen in die Berechnung einer Schaltung einzugeben, Vereinfachungen durchzuführen und Zwischenergebnisse grafisch darzustellen. Mit der analytischen Methode bietet sich auch die Möglichkeit, die gesamte Schaltungsdimensionierung völlig zu automatisieren. Für verschiedene Schaltungstopologien könnten hierzu Dimensionierungsroutinen in einer Art Datenbank zusammengefaßt werden. Durch die Vorgabe der Spezifikation kann schließlich eine geeignete Routine ausgewählt werden, mit der dann eine fertige Schaltungslösung berechnet und ausgegeben wird.

Literaturverzeichnis

[1] Physik mit Maple V
Von Newton zu Feyman
Michael Komma
1996

[2] Analyse elektronischer Schaltkreise
Grundlagen, Berechnungsverfahren,
Anwendungen
Band 1: Stationäres Verhalten
Dieter Mildenberger, 1975

[3] Mathematik für Ingenieure und
Naturwissenschaftler
Band 2
Lothar Papula

[4] Einführung in die Elektronik 1
Bauelemente der Elektronik und
ihre Grundschaltungen
H. Böger / F. Kähler / G. Weigt 1986

[5] Analysis and Design of
Analog Integrated Circuits
Third Edition
Paul R. Gray / Robert G. Meyer

[6] Symbolische Analyse analoger
BiCMOS-Schaltungen
Fortschrittberichte VDI, Reihe 5: Elektronik,
Nr. 213
Gerhard Nebel

[7] Analyse elektronischer Schaltkreise
Grundlagen, Berechnungsverfahren,
Anwendungen
Band 2: Quasistationäres Verhalten
Dieter Mildenberger, 1976

[8] Grundgebiete der Elektrotechnik
Band 1: Stationäre Vorgänge
A. Führer / K. Heidemann / W. Nerreter
1990

[9] Praxis nichtlinearer Gleichungen
W. Heitzinger / I. Troch / G. Valentin
1985

[10] THE MATHEMATICA BOOK
Third Edition
Stephen Wolfram, 1996

[11] Integrierte Bipolarschaltungen
Halbleiter-Elektronik
H. M. Rein / R. Ranfft
1980

[12] Anwendung von Methoden
der symbolischen Analyse
beim Entwurf analoger
Schaltungen (Diplomarbeit)
Varol Mutlu, Prof. G. Forster
Fachhochschule Ulm
Wintersemester 97/98

Entwicklung einer Digitalschaltung unter VHDL zur Verarbeitung von MIDI-Signalen

Dipl.-Ing.(FH) Frank Messmer,
Prof. Dr.-Ing. Hans Kreutzer,
Fachhochschule Reutlingen, Alteburgstr. 150, 72762 Reutlingen
Tel. 07121/341-108, Fax 07121/341-100
E-Mail: hans.kreutzer@fh-reutlingen.de

Im Rahmen der Diplomarbeit am Fachbereich Elektronik wurde eine Schaltung für ein MIDI-Mischpult entworfen. Dazu wurde zunächst die Spezifikation für das Mischpult erstellt. Aufbauend auf dieser Spezifikation konnte eine Zerlegung der Gesamtaufgabe in Teilaufgaben durchgeführt werden, die sich in Teilkomponenten in der Schaltung wiederfinden. Die Funktionen der Teilkomponenten wurden mit Hilfe von Zustandsübergangsdiagrammen mit dem Werkzeug Renoir von Mentor Graphics beschrieben. Aus diesen Beschreibungen wurde VHDL-Code erstellt, der dann synthetisiert und optimiert wurde. Eine in VHDL erstellte Testumgebung gestattete die funktionale Überprüfung des VHDL-Codes, sowie der synthetisierten und optimierten Schaltung.

1. Einführung

Im Bereich der Musikelektronik werden unterschiedlichste Geräte, beispielsweise Klangerzeuger (Synthesizer, Sampler), Rhythmusmaschinen (Sequenzer), Synchronisationsgeräte (Synchronizer), Computer mit Steuerungs-, Bearbeitungs- und Speicherungssoftware, Klangmanipulatoren (Effektgeräte), Aufzeichnungs- und Abspielgeräte (Bandmaschinen, Harddiskrecording-Systeme) eingesetzt. Damit die Kommunikation dieser Geräte untereinander möglich ist, existiert seit 1983 eine von der IMA (International MIDI Associaton) genormte Schnittstelle: die MIDI-Schnittstelle (Musical Instrument Digital Interface).

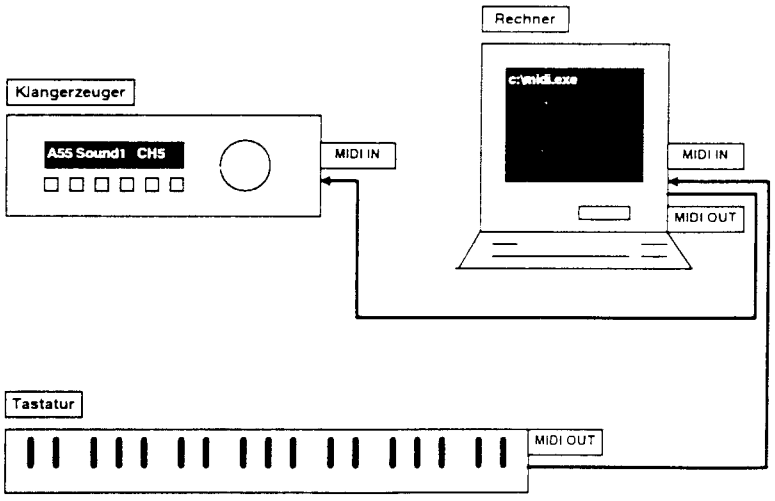


Bild 1 Beispiel zur Verbindung von Musikinstumenten

MIDI ist eine asynchrone serielle Schnittstelle mit einer Baudrate von 31,25 kBaud. Je Byte werden innerhalb von 320 µs ein Startbit, ein Stoppbit und acht Datenbits übertragen.
Das MIDI Protokoll legt Befehle, Befehlsfolgen und deren Bedeutung fest. Ein Befehl setzt sich aus einer Folge von Bytes zusammen. Befehle bestehen aus ein, zwei, drei oder beliebig vielen Bytes, wobei das erste Byte ein Statusbyte sein muß.
In der Betriebsart running mode wird bei langen Befehlsfolgen mit gleichem Status lediglich zu Beginn das Statusbyte übertragen, um den Empfänger in den richtigen Status zu versetzen. Daraufhin werden nur noch die dazugehörigen Datenbyteblöcke übertragen, bis ein Statuswechsel erfolgt.

In Bild 1 wird eine häufig vorkommende Anwendung der Verbindung von Musikgeräten dargestellt. Der im Bild dargestellte Aufbau stellt einen Minimalaufbau dar. Beim Spiel auf der Klaviatur werden MIDI-Daten (beispielsweise Tonhöhen-, Tondauer-, Anschlagstärke- und Kanalinformation) zum Rechner übertragen. Im Rechner läuft eine Anwendung zur Aufnahme und Wiedergabe dieser Daten in Echtzeit. Gleichzeitig wird der Datenstrom evt. in veränderter Form an die angeschlossenen Klangerzeuger weitergeleitet, die das auf der Klaviatur Gespielte zum klingen bringen.

Die musikalischen Informationen lassen sich auf dem Rechner beeinflussen. Dazu wird in einer graphischen Oberfläche ein Mischpult mit Drehreglern und Fadern (Schieberegler) nachgebildet.

Neben einigen Vorteilen des am Rechner dargestellten und zu bedienenden Mischpultes liegt das Hauptproblem dieses Mischpultes in der Bedienbarkeit. Die Bedienelemente sind entsprechend nachgebildete Drehregler bzw. Fader oder Felder zur Direkteingabe von Zahlenwerten - wie üblich steuerbar über Maus und Tastatur. Diese Art der Bedienung ermöglicht weder schnelles noch intuitives arbeiten, wie man es von einem Hardwaremischpult gewohnt ist.

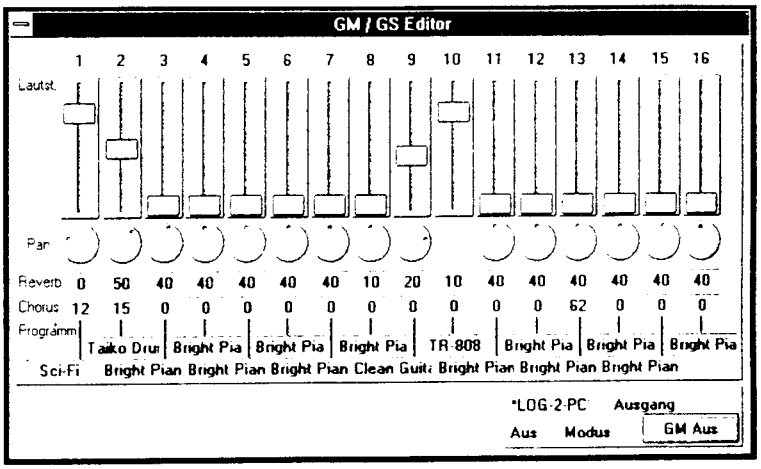


Bild 2 Softwaremischpult

2. Aufgabenbeschreibung

Die oben geschilderte Problematik hat zu dem Wunsch geführt, anstatt oder zusätzlich, ein Hardwaremischpult zu realisieren, das das Softwaremischpult ersetzt oder ergänzt.
Im folgenden soll eine Teilmenge der Anforderungen an das Gerät aufgelistet werden:

- Es soll ein 16-kanaliger MIDI-Mixer sein.
- Im Maximalausbau soll das Gerät je Kanalzug vier frei definierbare Controllerarten aufweisen. Im Maximalausbau befinden sich also 64 Regler auf der Frontplatte des Gerätes
- Ankommende MIDI-Daten müssen das Gerät unbeeinflußt passieren können.
- Im Gerät müssen ankommende MIDI-Daten mit den selbst erzeugten MIDI-Controller-Daten richtig gemischt werden (Merging)
- Die ankommenden MIDI_Daten werden eingelesen, erkannt und evt. neu generiert ausgesandt.

3. Realisierungskonzept

Unter Berücksichtigung der Datenrate und der Aufgabe wäre zur Realisierung ein Prozessorkonzept sehr geeignet. Da der bearbeitende Student aber die in der Vorlesung erworbenen Kenntnisse bezüglich FPGA-Entwurf mit Hilfe von VHDL vertiefen wollte, wurde eine reine Hardwarelösung gewählt. Alle Komponenten sollen in einem FPGA integriert werden, lediglich die AD-Wandler zur Umwandlung der Reglerwerte in digitale Werte werden durch externe Bausteine realisiert. Das in Bild 3 dargestellte Blockschaltbild zeigt die Schaltungsteile, die in dem FPGA untergebracht sind.

Der **SP-Wandler** wandelt die seriellen Daten des eingehenden Datenstromes in parallele Daten um.

Der **Filterblock** läßt nur die Kanäle passieren, die freigegeben sind.

Das **Fifo1** gewährleistet, daß die Daten aus dem eingehenden seriellen Datenstrom zwischengehalten werden können, solange Daten aus dem Fifo2 in den ausgehenden Datenstrom eingeschleust werden.

Ungekehrt werden in dem **Fifo2** Daten zwischengehalten solange eingehende Daten direkt an den Ausgang durchgeschleust werden müssen.

Der **Merger** mischt die vom Fifo1 und vom Fifo2 kommenden Datenfolgen. Dabei haben die Daten des Fifo1 Vorrang und es müssen die verschiedenen Betriebsmodi unterschieden werden.

Fifo3 dient sicherheitshalber zur Pufferung des Ausgangsdatenstromes, da die Verarbeitung im FPGA wesentlich schneller abläuft, als die Daten auf der Ausgangsseite abgegeben werden können.

Der **PS_Wandler** erstellt aus den Rohdaten wieder ein serielles Datenformat mit Start- und Stoppbits.

Die Komponente **AD_Steuerung** beinhaltet die Ansteuerung des AD-Wandlers und die Zuordnung der von dem AD-Wandler eingelesenen Daten zur Kanalnummer und Reglergruppe. Dabei ist zu berücksichtigen, daß nur ein AD-Wandler gemultiplext für die 16 Regler verwendet wird.

Die Aufgabe der Komponente **Generator** ist die Erzeugung und Ausgabe der Control-Change-Befehle bei Werteänderung eines Reglers.

Alle Komponenten kommunizieren untereinander im Handshakeverfahren.

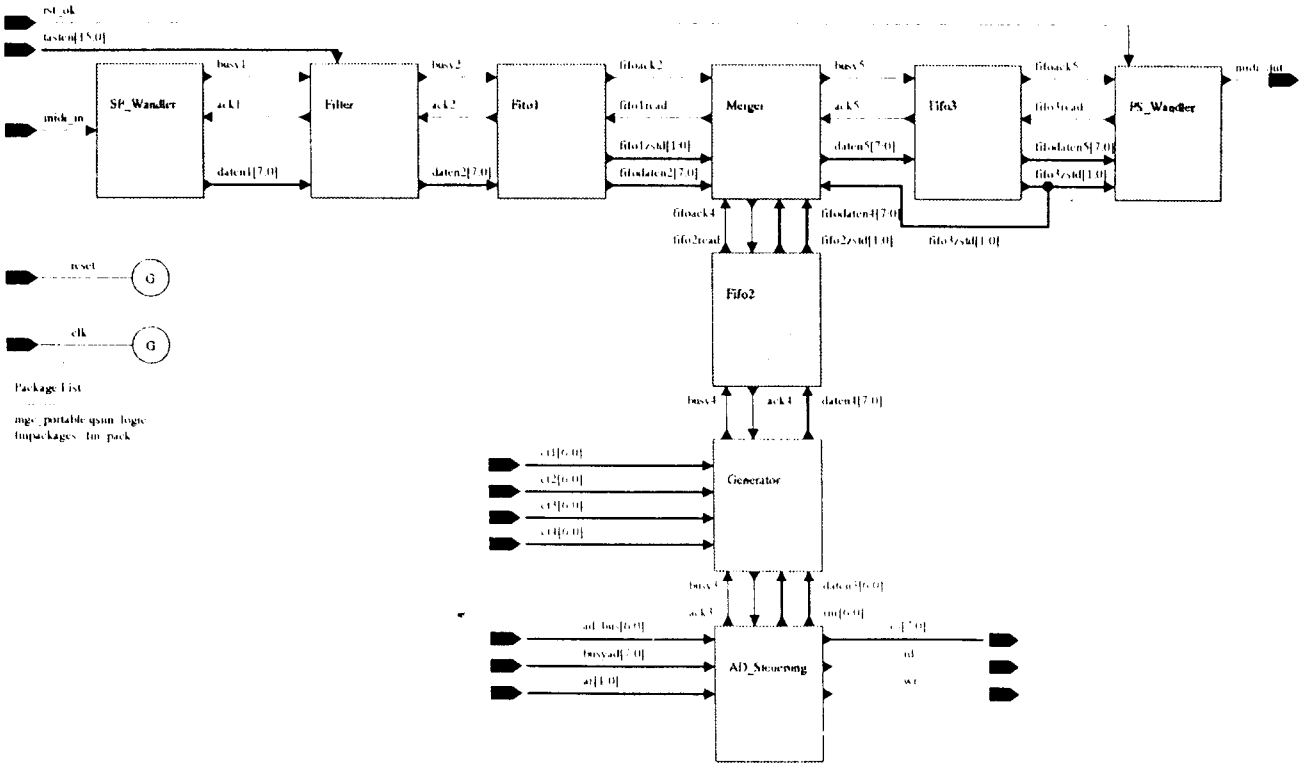


Bild 3 Blockschaltbild der Gesamtschaltung

4. Realisierung

Die somit definierten Komponenten der Gesamtschaltung wurden mit Hilfe von Zustandsübergangsdiagrammen funktional beschrieben. Diese Beschreibung ist an dem Beispiel des SP_Wandlers in Bild 4 veranschaulicht.

Das dargestellte Blockdiagramm in Bild 3 und das dargestellte Zustandsübergangsdiagramm in Bild 4 wurden mit dem Programm Renoir von Mentor Graphics erstellt. Dieses Programm gestattet die Generierung von VHDL-Code aus den jeweiligen Diagrammen.

Der compilierte VHDL-Code wurde dann mit dem Programm Quicksimll simuliert, wobei zur Generierung und Überprüfung der Testdaten eigene Komponenten (eine Testbench) in VHDL erstellt wurden. So ist in Bild 5 zu erkennen, daß ein Sender und Empfänger sowie ein AD-Wandler als zusätzliche Komponenten erstellt wurden.

Mit Autologic wurde aus dem VHDL-Code die Gesamtschaltung synthetisiert und optimiert und als Schaltplan und Netzliste dargestellt. Aufgrund des Umfanges der Schaltungen soll hier auf die Darstellung der Schaltungen verzichtet werden. Es ist zwar prinzipiell möglich, aber nicht ratsam, die synthetisierten Schaltungen zu betrachten und zu analysieren. Stattdessen wurde die synthetisierte und optimierte Gesamtschaltung mit der gleichen Testumgebung (Testbench) wie der VHDL-Code untersucht.

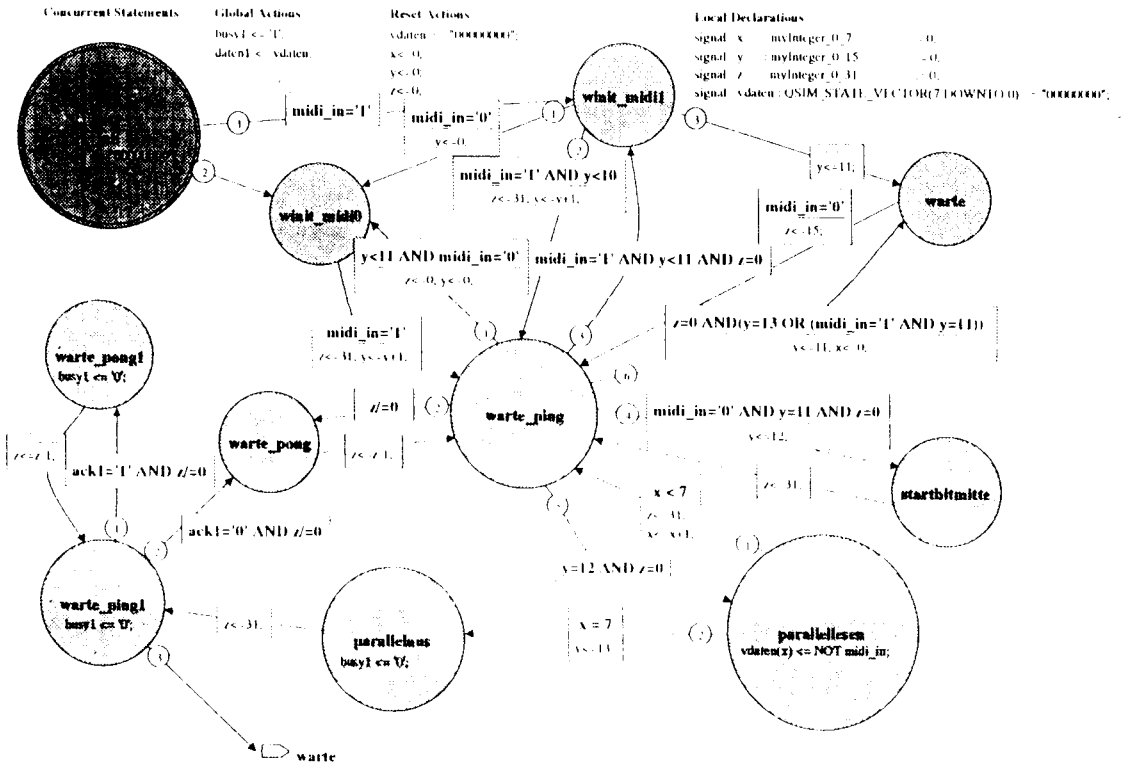


Bild 4 Zustandsübergangsdiagramm SP-Wandler

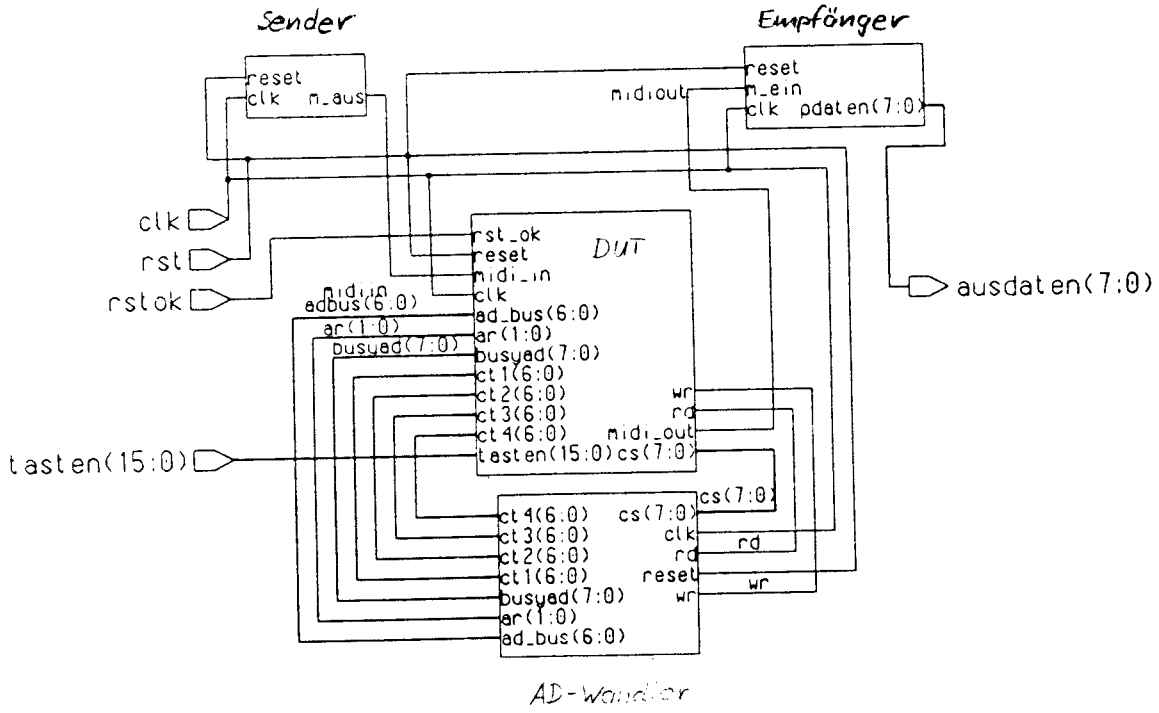


Bild 5 Testumgebung des DUT (Device under test)

5. Ergebnis und Ausblick:

Der Entwurf der MIDI-Mischpultschaltung wurde auf die Zieltechnologie Actel 1200 FPGA angewendet. Die Simulation des VHDL-Codes, sowie der synthetisierten und optimierten Schaltung ergab das gewünschte Verhalten der Schaltung. Zum Zeitpunkt der Diplomarbeitsfertigstellung konnte, aufgrund der Tatsache, daß die zu diesem Zeitpunkt verwendeten Mentor Graphics Werkzeuge (AutologicII) keine FPGA-Synthese zuliessen, kein FPGA programmiert werden.
Diese Aufgabe und den Aufbau eines fertigen Gerätes soll im Rahmen einer Studienarbeit weitergeführt werden.

Fachhochschule Reutlingen
Hochschule für
Technik u. Wirtschaft

Entwicklung eines Einkanal-HDLC-Empfängers unter Verwendung der Hardwarebeschreibungssprache VHDL

Dipl.-Ing.(FH) Peter Kuppler,
Prof. Dr.-Ing. Hans Kreutzer,
Fachhochschule Reutlingen, Alteburgstr. 150, 72762 Reutlingen
Tel. 07121/341-108, Fax 07121/341-100
E-Mail: hans.kreutzer@fh-reutlingen.de

Im Rahmen einer Diplomarbeit wurde an der Fachhochschule Reutlingen ein Einkanal - HDLC - Empfänger entworfen, der es ermöglicht HDLC - Blöcke, die in Kanälen des PCM - Datenstromes gemultiplext vorkommen, zu erkennen und zu analysieren. Die HDLC - Blöcke können dabei Datenraten von 8 bis 64 kBit/s besitzen. Die HDLC - Blöcke werden aufgrund ihrer Flags erkannt und neben einigen Eigenschaftsüberprüfungen, wie zum Beispiel die Länge der Blöcke,auch deren CRC überprüft. Der Entwurf wurde so gestaltet, daß eine Erweiterung auf 16 zu untersuchende Kanäle möglich ist. Sämtliche Komponenten sind in der Hardwarebescheibungssprache VHDL entworfen worden. Der Einkanal - HDLC - Empfänger wurde in ein FPGA EPF10K30 der Firma Altera implementiert und ist erfolgreich getestet.

1. Einführung

Die immer komplexer werdende Telekommunikationstechnik erfordert entsprechende Meß- bzw. Prüfeinrichtungen, um deren Fehlerfreiheit zu gewährleisten. Für den Bereich GSM (Mobilfunk) wird dafür von der Firma Wandel & Goltermann der Mobilfunkanalysator MA-10 angeboten. Dabei handelt es sich um eine PC-Einsteckkarte deren Funktionalität durch entsprechende Software für unterschiedliche Meßaufgaben programmiert werden kann. So stehen Softwarepakete für Analyseaufgaben der GSM-Daten, Bitfehlermessung, sowie Simulationsanwendungen zur Verfügung.
Beim MA-10 wird für die Dekodierung der HDLC (High-level-data-link-control) formatierten Eingangsdaten, wie sie im Bereich GSM zum Einsatz kommen, bisher ein käuflicher Kommunikations-Controller der Fa. Siemens verwendet.
Da in einem neuen Meßgerät der Fa. Wandel & Goltermann die Funktionalität erweitert werden soll, kann dieser Baustein nicht mehr verwendet werden. Aus diesem Grund sollte im Rahmen dieser Diplomarbeit ein Einkanal-HDLC-Empfänger als Grundbaustein entworfen werden. Der Entwurf sollte berücksichtigen, daß dieser HDLC-Kern letztendlich zu einem 16-Kanal-Empfänger erweitert werden soll. Die HDLC-Rahmen sollen von dem Empfänger dann an einen Prozessor weitergegeben werden. Dazu sind Schaltungsteile zu entwerfen, die mit dem Prozessor kommunizieren. Diese Schaltungsteile waren allerdings nicht mehr Aufgabe dieser Arbeit.

2. Der HDLC- und PCM-Rahmenaufbau

Beim HDLC-Protokoll handelt es sich um ein Sicherungs-Protokoll, das im OSI-Referenzmodell in die Schicht 2 einzuordnen ist [1].

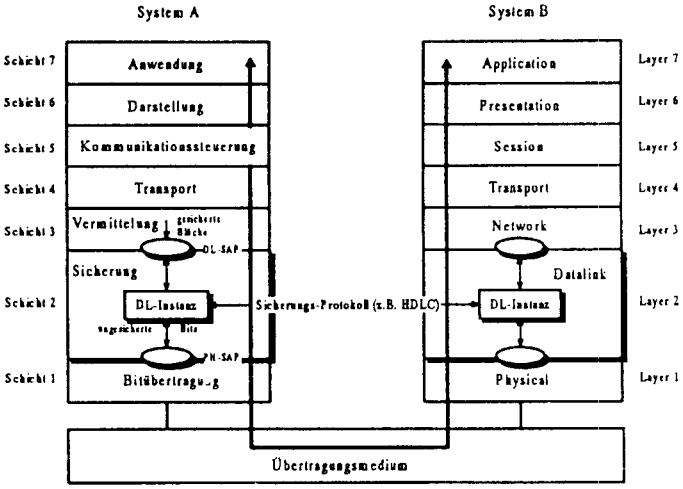


Bild 1 Einordnung des HDLC-Protokolls im OSI-Referenzmodell

2.1 Rahmenaufbau des HDLC-Rahmens

Das international standardisierte HDLC-Protokoll verwendet einen bitorientierten Rahmenaufbau. Der Grundrahmen besteht dabei aus folgenden Feldern:

- einer Rahmenbegrenzung durch die Bitkombination 01111110, die als Flag bezeichnet wird. Sie dient

dem Empfänger als Rahmensynchronisation,
- einem Header, bestehend aus 8 Bit Adressfeld und 8 Bit Steuerfeld,
- einem Informationsfeld beliebiger Länge,
- einem FCS-Feld (Frame-Check-Sequence), das ein Blocksicherungsfeld mit der Länge von 16 bzw. 32 Bit darstellt.

Bild 2 zeigt den Rahmenaufbau in graphischer Darstellung.

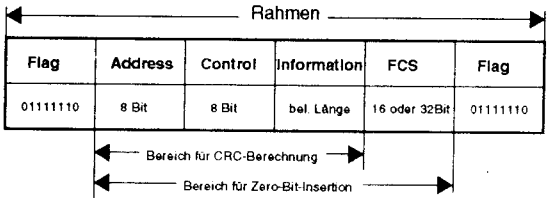


Bild 2 Aufbau eines HDLC-Rahmens

2.2 Die Felder eines Rahmens und ihre Eigenschaften

2.2.1 Rahmenbegrenzung (Flag)

Jeder Rahmen beginnt und endet mit einem Flag (Bitkombination 01111110). Dabei kann ein einzelnes Flag sowohl das Ende eines Rahmens als auch den Beginn eines neuen Rahmens darstellen (shared flag).
Zusätzlich wird als Sonderfall die Bitkombination 011111101111110 als zwei Flags gewertet (shared zero), d.h. das letzte Null-Bit eines Ende-Flags und das erste Null-Bit eines Start-Flags, werden zu einer Null zusammengefaßt.
Bild 3 zeigt anhand von Beispielen die verschiedenen Möglichkeiten der Rahmenbegrenzung.

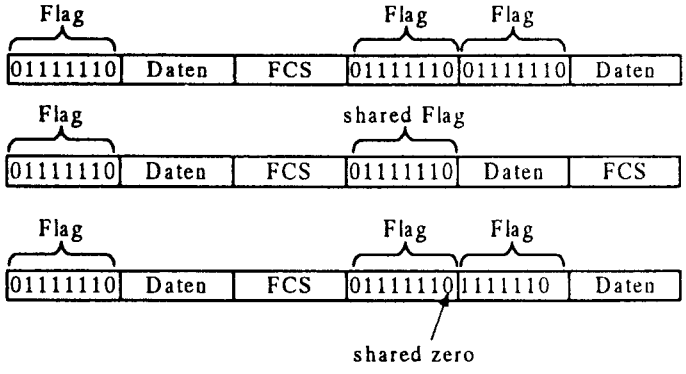


Bild 3 Arten der Rahmenbegrenzung

2.2.2 Adress- u. Steuerfeld

Das Adress- und das Steuerfeld sollen in der geplanten Anwendung lediglich angezeigt werden, aus diesem Grund wird auf diese beiden Felder nicht näher eingegangen.

2.2.3 Informationsfeld

Das Informationsfeld besteht aus einer beliebigen Bitfolge sowie einer beliebigen Anzahl von Bits. Meistens setzt sich das Informationsfeld jedoch aus einer Folge von Oktets (8-Bit lange Datenblöcke) zusammen. Um Codetransparenz zu erreichen, müssen an geeigneter Stelle Null-Bits eingefügt werden.

2.2.4 Blocksicherungsfeld

Das Blocksicherungsfeld enthält eine Bitkombination zur Fehlersicherung, die aus den Bits des Adress-, Kontroll- und des Informationsfeldes berechnet wird (nach dem Entfernen der eingefügten Null-Bits für Codetransparenz). Die Länge der Blockprüfzeichenfolge (FCS) beträgt 16.
Die 16-Bit FCS ist das Einerkomplement des Restes, der sich aus der Division von
$x^{16}*G(x)+x^{k}*(x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^{9}+x^{8}+x^{7}+x^{6}+x^{5}+x^{4}+x^{3}+x^{2}+x+1)$
durch das Generatorpolynom $x^{16}+x^{12}+x^{5}+1$ ergibt. Die Addition mit $x^{k}*(x^{15}+ \ldots +x+1)$ entspricht dabei der Initialisierung des Restes mit dem Wert 1111 1111 1111 1111. Bei fehlerfreier Übertragung des Rahmens beträgt der Wert des Restes 0001 1101 0000 1111 für (x15 ... x0).

2.2.5 Codetransparenz

Die Codetransparenz (Bitfolgeunabhängigkeit) der Übertragung erfordert, daß die Bitkombination eines Flags zwischen den Rahmenbegrenzungen nicht wieder vorkommt. Dazu fügt der Sender überall dort, wo fünf aufeinanderfolgende Einsen erkannt werden, eine zusätzliche Null ein (zero bit insertion). Der Empfänger entfernt dann jede Null, wenn sie unmittelbar nach fünf aufeinanderfolgenden Einsen auftaucht.
Beispiel:

Es soll das Datenwort 0111111011111111 übertragen werden.
Nach Einfügen der Null-Bits wird die Bitfolge
01111101011111011 im Imformationsfeld übetragen.

Eingefügte Null-Bits

2.2.6 Zeitüberbrückung zwischen Rahmen (inter-frame time fill)

Eine Zeitüberbrückung zwischen Rahmen soll durch das Senden von Dauer-Flags oder 7 bis 14 Einsen erfolgen. Diese beiden Methoden können auch kombiniert werden.

2.2.7 Ungültige Rahmen

Folgende Rahmen sind ungültig:
- Ein Rahmen, der nicht durch zwei Flags begrenzt

ist.
- Ein Rahmen, der ohne Flags kürzer als 32 Bit, bei Verwendung der 16-Bit Prüfzeichenfolge, ist.
- Ein Rahmen mit einer falschen Blockprüfzeichenfolge (CRC-Fehler).

2.3 Der Rahmenaufbau von PCM-Rahmen

Der HDLC Datenblock ist bei GSM in einen PCM-Rahmen gemultiplext. Die Daten sind in den einzelnen Zeitkanälen eines 2Mbit/s PCM-Datenstromes nach G.704 [2] enthalten. Der Rahmenaufbau eines PCM-30-Systems ist im Bild 4 dargestellt.

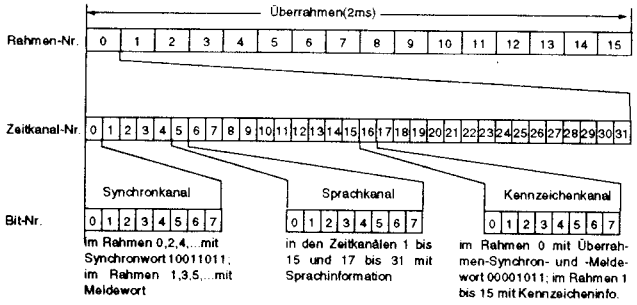


Bild 4 Rahmenaufbau des Systems PCM-30

Die Besonderheit bei dem System besteht nun darin, daß Datenraten mit 8, 16, 24, 32, 40, 48, 56, 64 kBit/s realisiert werden. Bei 64 kBit/s werden alle 8 Bit eines PCM-Kanals für die Datenübertragung verwendet. Bei 8 KBit/s wird nur ein Bit des PCM-Kanales für einen HDLC-Rahmen verwendet. Die anderen Bits in dem PCM-Kanal können für weitere HDLC-Rahmen ausgenutzt werden. Bild 5 zeigt die Verteilung der Bits für einen HDLC-Kanal mit 8 kbit/s und 24 kBit/s, die in dem PCM-Kanal 5 untergebracht sind.

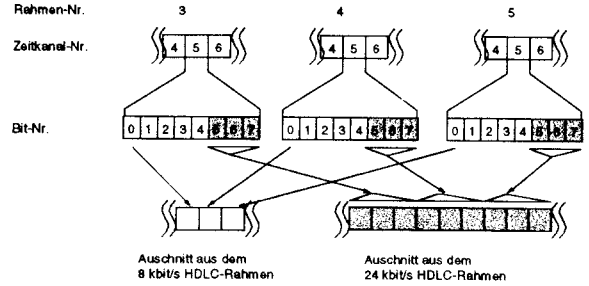


Bild 5 Ein 8 kBit/s und ein 24 kBit/s HDLC-Rahmen eingebettet in den PCM-Kanal 5.

3. Aufgabenbeschreibung

Im Rahmen dieser Arbeit sollte ein FPGA programmiert werden, das es gestattet einen HDLC-Rahmen aus den PCM-Rahmen zu demultiplexen, zu analysieren und abzuspeichern, damit der nachfolgende Rechner die Daten am Bildschirm darstellen kann. Daraus ergeben sich folgende Anforderungen an den HDLC-Empfänger:

- Demultiplexung der HDLC-Blöcke, die im PCM-Rahmen eingebettet sind.
- Subchanneling in jedem Zeitschlitz, d.h. die 8-Bit breiten Daten eines PCM-Zeitkanals werden als 8 bis 64 kBit/s Datenströme verwendet.
- Automatische Flag-Erkennung, auch shared flag und shared zero.
- Zero-Bit-Deletion
- CRC-Berechnung und Test derselben
- Einstellbare maximale Rahmenlänge (MFL)
- Erkennung von zu kurzen Rahmen (short frames)
- Erkennung des Bitmusters für einen Rahmenabbruch (Abort)
- Erweiterbarkeit auf 16 HDLC-Kanäle

4. Auswahl der Werkzeuge

Für die Erstellung des HDLC-Empfängers wurden folgende Entwicklungswerkzeuge verwendet.

Für die graphische Eingabe der Schaltung:

- Renoir V.2.053 von Mentor Graphics (MGC)

Erstellung des VHDL-Codes:

- Design Architect von MGC

Kompilieren und simulieren des VHDL-Codes:

- QuickVHDL-Compiler qvhcom
- QuickVHDL-Simulator qhsim

Synthese des VHDL-Codes:

- Vantage VHDL-Compiler
- Synopsys Design Compiler

Plazieren und Routen der synthetisierten Schaltung:

- Altera MAX+PLUS II

Für die Erstellung des Programms zur Generierung der Testdaten:

- Borland Delphi V.3.0
- GNU flex V. 2.5.3
- GNU bison V.1.4
- GNU C/C++-Compiler V.2.7.1

5. Realisierung des HDLC-Empfängers

Die Aufgabe wurde schrittweise in kleinere Einheiten zerlegt. Sämtliche Blockschaltbilder und Schaltungsteile wurden mit dem Eingabewerkzeug Renoir von Mentor Graphics in der Form von Zustandsdiagrammen bzw. Flußdiagrammen erstellt. Aus diesen Diagrammen wird von Renoir der VHDL-Code automatisch generiert, der anschließend mit einem VHDL-Compiler bearbeitet werden kann. An diese Kompilierung schließt sich die Simulation des VHDL-Codes an.

In Bild 6 ist die Gesamtschaltung des Einkanal-HDLC-Empfängers dargestellt. Sie setzt sich aus 3 Modulen zusammen,

- dem HDLC-Receiver-Core, der die Dekodierung der HDLC-Rahmen vornimmt,
- einem Demultiplexer, der die in einem 2 Mbit/s PCM-Datenstrom eingebetteten HDLC-Rahmen herausfiltert,
- sowie einem CPU-Interface, das die Kommunikation mit einem Mikroprozessor übernimmt.

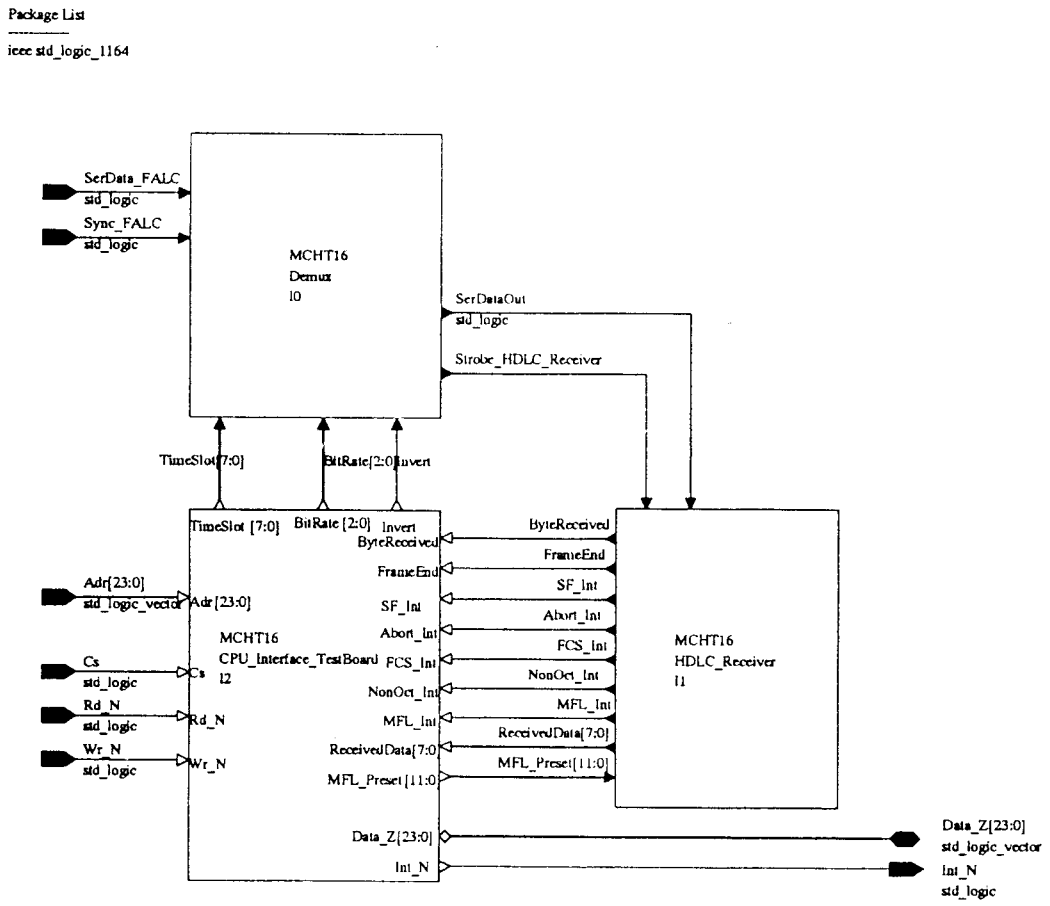


Bild 6 Gesamtschaltung des HDLC-Empfängers

Um die HDLC-Blöcke analysieren zu können, wurde der HDLC_Receiver-Block in 7 funktionale Module aufgeteilt. Im einzelnen sind dies:

- Flag_Detection (Flag-Erkennung)
- Bit_Counter (Zählt die empfangenen Bits)
- MFL_Counter (Zählt die empfangenen Bytes)
- FCS_Calculator (Berechnung und Überprüfung der Blockprüfzeichenfolge)
- SP_Converter (Serien-Parallel-Wandlung)
- Receive_Register (Ausgangsregister)
- Receive_Control (Ablaufsteuerung)

Im Bild 7 ist das Blockschaltbild des HDLC-Receivers dargestellt.

Die Funktion der Module wird nun mit Zustandsübergangsdiagrammen oder Flußdiagrammen beschrieben. In Bild 8 ist das Zustandsübergangsdiagramm für die Flagerkennung (Flag_Detection) dargestellt. In diesem Block Flagerkennung ist zusätzlich die Abort-Erkennung und die Zero-Bit-Deletion untergebracht

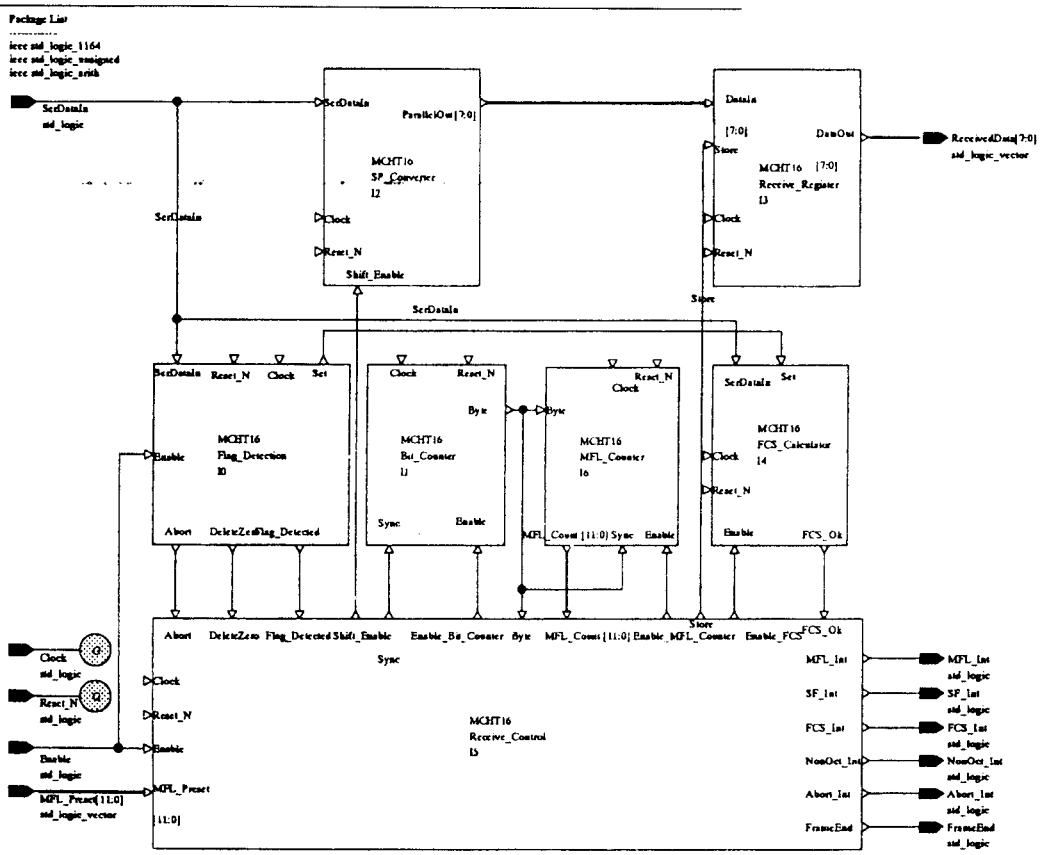


Bild 7 Blockschaltbild des HDLC-Empfänger-Cores.

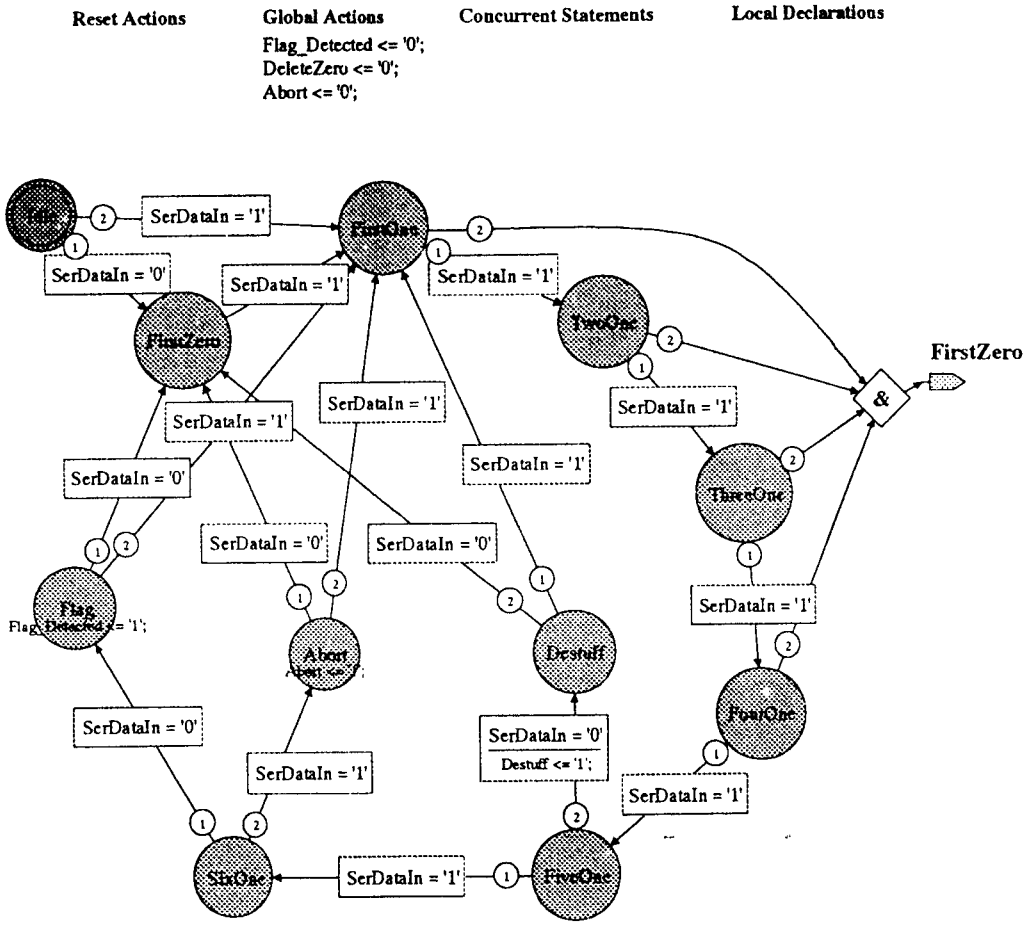


Bild 8 Zustandsautomat für den Block Flag_Detection

In Bild 9 wird das Flußdiagramm des FCS_Calculator dargestellt. Der FCS_Calculator berechnet die Blockprüfzeichenfolge und vergleicht diese mit dem Rest für eine fehlerfreie Übertragung. Zur Berechnung wird das Generatorpolynom 16. Grades $x^{16} + x^{12} + x^5 + 1$ nach ISO 3309 verwendet.
Der FCS_Calculator besteht im wesentlichen aus einem rückgekoppelten Schieberegister. Die Anzahl der Registerstufen des Schieberegisters ergibt sich aus der Länge der Blockprüfzeichenfolge. Eine Blockprüfzeichenfolge von 16 Bits erfordert daher ein 16-stufiges Schieberegister.

Zu Beginn der CRC-Berechnung wird das Schieberegister mit 16 Einsen initialisiert.
Der dargestellte FCS_Calculator läßt sich leicht auf andere Generatorpolynome anpassen. Bleibt der Grad des Generatorpolynoms gleich, so muß im VHDL-Code lediglich die Konstante CRCPOLY auf den entsprechenden Wert geändert werden. Ebenso kann der Wert für eine fehlerfreie Übertragung, sowie der Initialisierungswert des Schieberegisters mit den Konstanten REMAINDER bzw. INICRC, den Erfordernissen angepasst werden.

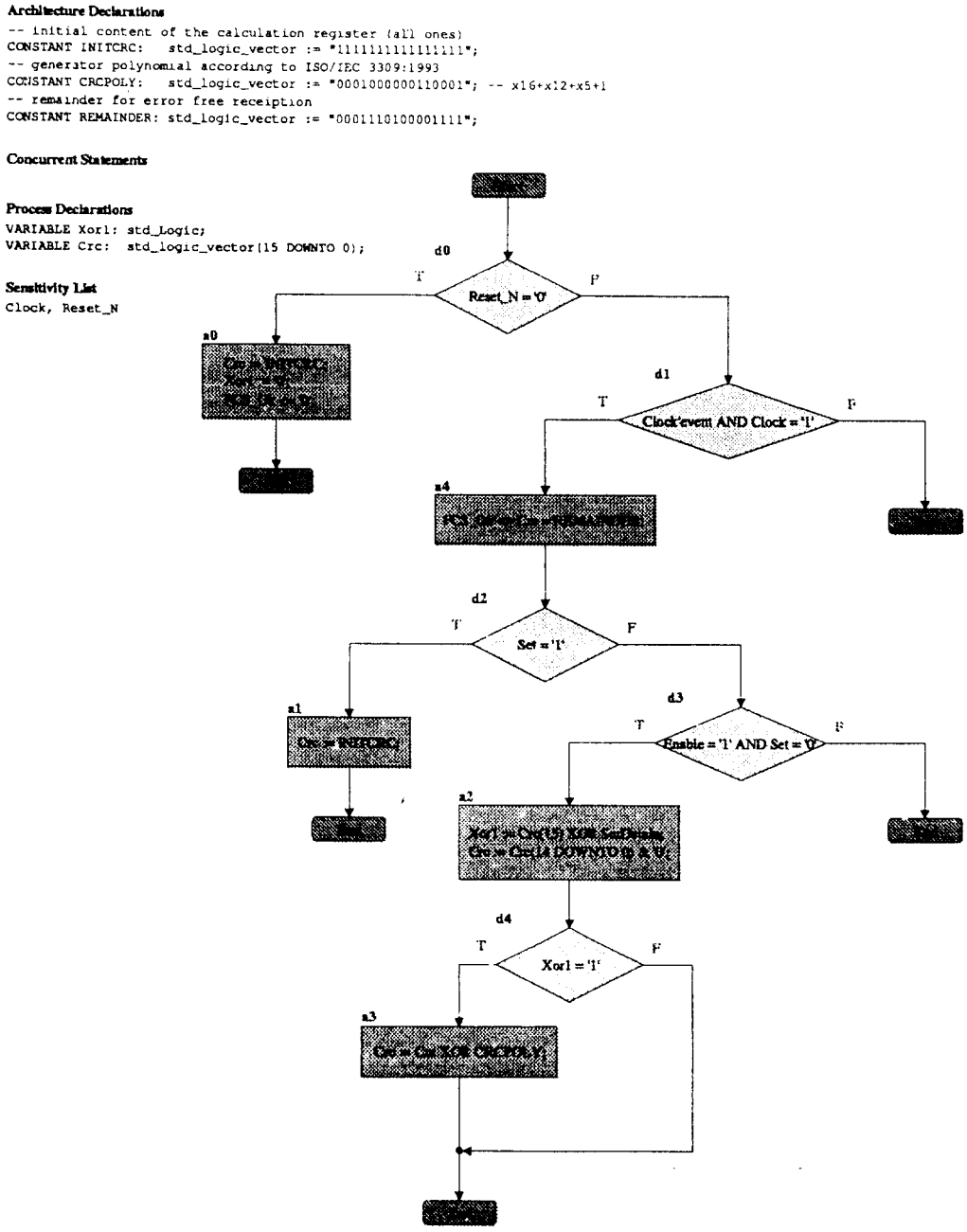


Bild 9 Flußdiagramm des FCS-Calculators

6. Simulation des VHDL-Codes

Für den beschriebenen HDLC-Empfänger wurde eine Testumgebung in VHDL erstellt, die für die Erzeugung der entsprechenden Eingangssignale und die Überprüfung der Ergebnisse sorgt. Da die Testumgebung nicht synthetisiert wird, kann hier der gesamte Sprachumfang von VHDL, auch sämtliche Schleifenarten, Dateioperationen, Funktionen, usw. verwendet werden.

Die Testumgebung muß im vorliegenden Fall folgende Aufgaben erfüllen:

- Generierung eines 2 Mbit/s PCM-Datenstromes mit Synchronisations- und Taktsignal.
- Generierung von HDLC-Rahmen, die in den einzelnen Zeitkanälen des 2 Mbit/s Datenstromes übertragen werden.
- Nachbildung der Schreib- und Lesezugriffe des Prozessors auf die internen Register des HDLC-Empfängers.
- Auswerten der vom HDLC-Empfänger gelieferten Ergebnisse.

Bild 10 zeigt das Blockschaltbild der dazu verwendeten Schaltung.

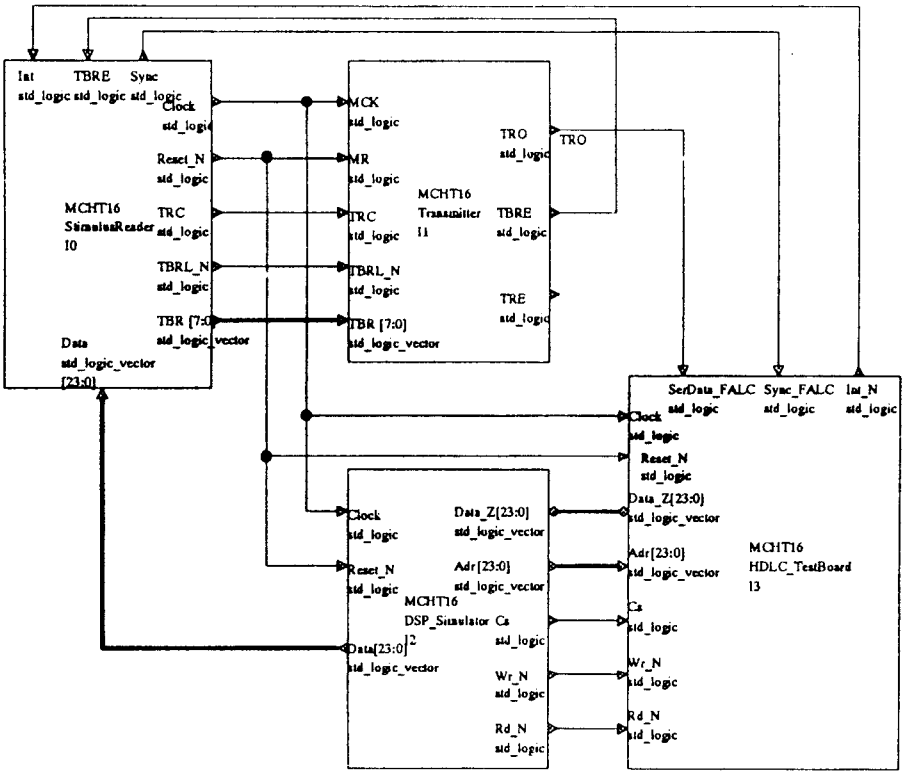


Bild 10 Blockschaltbild der Testumgebung

7. Ergebnis und Ausblick

Der oben skizzierte Einkanal-HDLC-Empfänger wurde synthetisiert und in einem FPGA vom Typ FLEX EPF10K30 implementiert. Funktionsprüfungen haben gezeigt, daß der Empfänger die an ihn gestellten Forderungen erfüllt. In einem weiteren Schritt soll der nun bestehende Einkanal-Empfänger auf 16 Eingangskanäle erweitert werden.

8. Literaturverzeichnis

[1] DIN ISO3309: High-level data link control (HDLC) procedures - Frame Structure. Dec. '93

[2] ITU Q.921: ISDN User-Network Interface - Data Link Layer Specification.

Hochsprachen-Compiler für den FHOP

Dipl.-Ing. (FH) Oliver Bischoff,
Prof. Dr.-Ing. Dirk Jansen, ASIC-Design-Center
Fachhochschule Offenburg, Badstr. 24, 77652 Offenburg
Tel. 0781/205-267, Fax 0781/205-242,
E-Mail: d.jansen@fh-offenburg.de

An der Fachhochschule Offenburg wurde ein C-Compiler entwickelt. Es handelt sich dabei um einen Hochsprachen-Compiler für den an der Fachhochschule entwickelten Prozessor FHOP (First Homemade Operational Prozessor), der ANSI-C-Code in Maschinencode umsetzt.
Um den Maschinencode zu erzeugen, wird die Hilfe des bereits existierenden Crash - Assemblers in Anspruch genommen. Der C-Compiler Assemblercode erzeugt, welcher dann durch den Crash-Assembler in Maschinencode umgesetzt wird.
Der Compiler soll als Tool bei der Entwicklung neuer Software für den FHOP dienen. Hierbei wird vor allem an die Entwicklungsgeschwindigkeit gedacht, die für C-Code wesentlich höher ist, als für Assemblercode, der die selbe Funktionalität besitzt.

1 Einleitung

Die Programmiersprache „C" erfreut sich großer Beliebtheit und weiter Verbreitung. Dies beruht auf der Tatsache, daß C eine sehr übersichtliche und einfach zu erlernende Programmiersprache ist. Aufgrund der Vielzahl an Prozessoren stehen auch eine Menge an Compiler zur Verfügung; ein Teil ist „Retargetable", d. h. es besteht die Möglichkeit, mit diesen Compilern Maschinencode für Prozessoren zu erzeugen, die nicht vom selben Prozessortyp sind, wie der, auf dem der Sourcecode compiliert wurde. Diese Compiler, z. B. GNU, erfordern eine 32-Bit-Architektur des Prozessors, so daß eine Anpassung dieser Compiler für den FHOP nicht möglich war. Deshalb entschloß ich mich, mit Hilfe von Compilerbau-Tools, einen Compiler selbst zu schreiben. Dieser Compiler sollt sich so nah wie möglich an dem ANSI-C-Standard halten, damit es später möglich ist, bestehende C-Bibliotheken einzubinden.

2 Hochsprache! - Warum?

Das Programmieren von Prozessoren ist eine angenehme Arbeit, sofern die Tools zum Programmieren des Prozessors vorhanden sind. Maschinencode direkt einzugeben, ist sicher keine sehr angenehme Arbeit, zumal die Verwaltung des Speichers und sämtlicher Adressen in den Händen des Programmierers liegt. Variablen existieren auf dieser Programmierebene nicht. Der Code entbehrt jeglicher Übersicht. Eine Erleichterung bietet der Assembler, der das Erzeugen von Maschinencode komfortabler gestaltet. Er übernimmt die Adressenverwaltung und erlaubt das Anlegen von Variablen. Jedoch bleiben Zuverlässigkeit und Übersichtlichkeit weiterhin ein Problem des Assemblerprogramms. Diese Nachteile weißt ein Hochsprachen-Compiler nicht mehr auf. Sowohl die Übersichtlichkeit, als auch die Zuverlässigkeit des Codes sind weitaus besser, als bei einem Assemblercode. Dabei besticht vor allem die Lesbarkeit. Durch Unterteilung in einzelne Funktionen, erhält man ein übersichtlichen und lesbaren Code. Die Zuverlässigkeit wird durch erweiterte Syntaxprüfungen erhöht. Bei einem Assemblerprogramm kann das Vergessen eines Push- oder Pop-Befehls den Prozessor zum Abstürzen veranlassen, während man in einer Hochsprache dazu schon mehr braucht.
Ein weiteres Argument, welches die Vorteile eines C - Compilers unterstreicht, ist die Entwicklungszeit. Ein Vergleich (Abbildung 1) von zwei Programmausschnitten wird dies gut darstellen.

C - Code

```
main() {
    initialize();
    for(i=0;i<=3;i++) {
      tast_in();
      if(taste == kode[i])
        state++;
      else
        state=0;        }
  if(state==4)
    Tuer_out();
  else {
    state=0;
  Fehler_out();         }
}
```

Assembler-Code

```
main: CAL initialize
      LDI i_global
      PSH A
      LDI #0000
      POP B
      STA M
_FOR_1:
      LDA i_global
      PSH A
      LDI #0003
      POP B
      CAL KLGL_INT
      CMI 0
```

Abbildung 1 Vergleich C - / Assembler - Code

Dabei hat der unterstrichene C-Code links genau die gleiche Funktionalität wie der Assemblercode rechts.

Man sieht schon hier einen deutlichen Unterschied im Umfang.
Außerdem läßt sich der linke Code in Abbildung 1 wesentlich besser lesen. Durch die Anlehnung des C-Befehlssatzes an das Englische Vokabular kann sehr schneller verstanden werden, was ein bestimmter Programmteil ausführt. (z. B. if-Anweisungen). So läßt sich auch nach größeren Zeiträumen schneller nachvollziehen, welche Funktionalität der C - Code erfüllen soll. Ein Nachteil der Hochsprache der Umfang des erzeugten Codes. Bei Assembler-programmierung läßt sich der Code geringer halten. Jedoch ist der Speicher des FHOP so groß ausgelegt, so daß speicherplatzkritische Anwendungen bis jetzt nicht aufgetreten sind. Deshalb wird dieser Nachteil gerne in Kauf genommen, da die oben an - gesprochenen Vorteile überwiegen.

3 Wie wird ein Compiler erstellt?

Um die Vorteile der verwendeten Tools zu erkennen, möchte ich hier kurz auf die Funktionsweise eines Compilers eingehen.
Zuerst wird der Source-Code einer Überprüfung unterzogen. Wird diese erfolgreich abgeschlossen, d.h. der Source-Code ist fehlerfrei, wird versucht, den Source-Code durch eine Baumstruktur darzustellen. Abbildung 2 zeigt eine solche Baumstruktur.

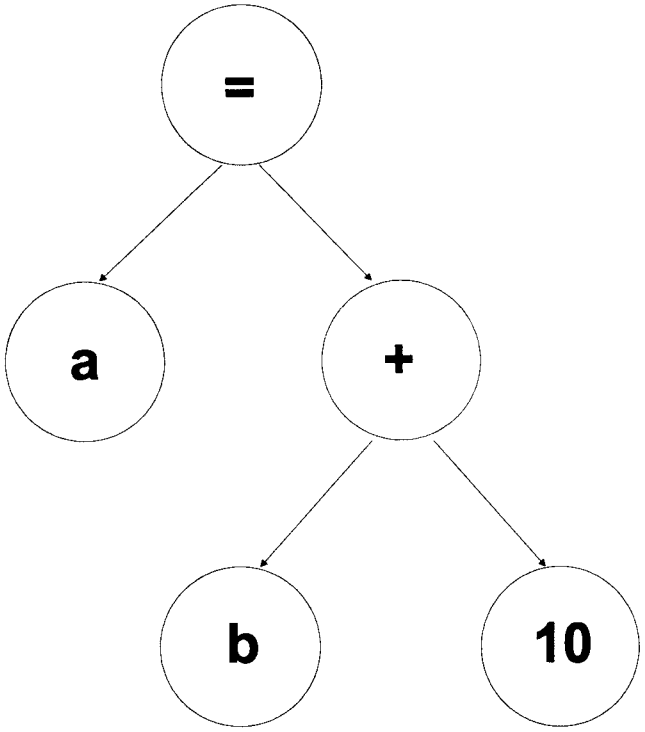


Abbildung 2 Baumstruktur

Die abgebildete Baumstruktur ergab die C-Anweisung „a = b + 10". Nachdem nun der gesamte Source-Code durch solche Baumstrukturen abgebildet wurde, versucht der Compiler diese Strukturen mit Assemblerbefehlen zu überdecken, d.h. in Assemblercode umzusetzen. Für die Baumerzeugung und die Überdeckung wurden nun Compilerbau-Tools verwendet. In Abbildung 3 wird der Ablauf „Source - Code → Maschinencode" schemenhaft dargestellt.

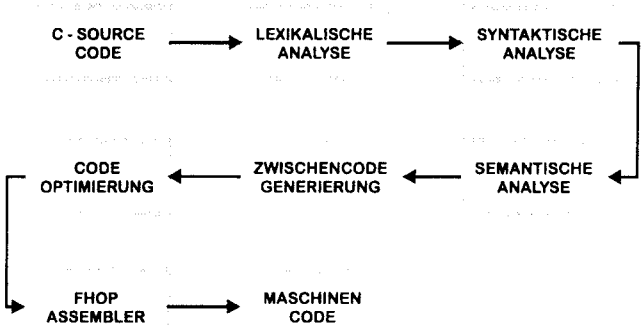


Abbildung 3 - Stufen der Codeerzeugung

Anhand dieses Schemas lassen sich die einzelnen Stufen der Umsetzung vom Source-Code in Assemblercode gut nachvollziehen. Das C-Source-File ist der Ausgangspunkt. Diese wird zuerst auf Richtigkeit überprüft und dann in 2 Schritten in Maschinencode umgesetzt. Schritt 1 ist die Erzeugung von Assemblercode. Dieser wird dann durch den Crash-Assembler in Maschinencode gewandelt. Der erste Schritt wird mit Hilfe von Compilerbau-Tools durchgeführt. Diese Tools helfen dem Anwender bei der Umsetzung der einzelnen Compilerfunktionen. Folgende Tools kamen dabei zum Einsatz:

3.1 flex: Scanner - Tool

Der Compiler erhält das vollständige File. Dieses muß zuerst auf Fehler durchsucht werden. Um diese Überprüfung durchführen zu können, muß dem Compiler bekannt sein, welche Ausdrücke zulässig sind und welche nicht .
Das Absuchen des C-Sourcefiles „von Hand" wäre sehr umständlich. Diese Arbeit übernimmt das Tool flex. flex ist die GNU-Variante des Standard-UNIX-Tools lex, welches bei allen UNIX-Betriebsystemen mit ausgeliefert wird. flex hat den Vorteil, daß es ein Freeware-Tool ist.
Diesem Tool werden nun die erlaubten Stringfolgen bzw. Befehle in einem Definitionsfile bekannt gemacht. Aus diesem File erzeugt flex dann ein C-File, welches dann compiliert die Funktion des Scanners übernimmt, der das gesamte File nach Fehlern in den Stringketten absucht. Diese Fehlersuche entspricht der lexikalischen Analyse, dargestellt als 2 Block in Abbildung 3.

3.2 bison: Parser - Tool

Die darauf folgende syntaktische Analyse erfolgt mit Hilfe des Tools bison. Bison ist ebenso wie flex eine GNU-Variante eines Standard-UNIX-Tools.
Wie auch bei lex, wird hier mit Hilfe eines Definitionsfiles dem Tool bekanntgemacht, welche Ausdrücke zulässig sind und welche nicht.
Bison wandelt das Definitionsfile in C-Code um, der, nachdem er compiliert wurde, die Funktion eines Parsers übernimmt.
Beim Parsen des Files wird nun nicht mehr auf die Richtigkeit einzelner Zeichenketten geachtet, sondern auf den sinnvollen Zusammenhang der Zeichenketten, die vom Scanner erkannt wurden.
So müssen bei bestimmten Operationen gewisse Bedingungen erfüllt sein. Bei einer mathematischen Operation, wie z.B. der Addition, müssen zwei Operanden zur Verfügung stehen. Dies heißt nun für den Parser, das beim Auffinden eines Additionszeichens der Scanner genau zwei Operanden liefern muß. Diese können direkt Zahlenwerte, Variablen oder auch beliebig komplexe Ausdrücke sein. Jedoch muß die Bedingung der zwei Operanden erfüllt sein. Ist dies nicht der Fall, so wird das Parsen mit einer Fehlermeldung abgebrochen.

3.3 Ox: Scanner & Parser - Tool

Ox ist ein Tool, welches die beiden vorherigen Tools, flex und bison unterstützt.
Beim Parsen und Scannen ist es notwendig, mehr Informationen als nur die vorgefundene Zeichenkette weiterzugeben. Dies läßt relativ gut an einem Beispiel zeigen.
Der Ausdruck „int a", definiert eine Variable „a" vom Typ „int". Trifft man nun im späteren Verlauf wieder auf diese Variable „a", z.B. bei „a = 0", so erkennt man nun zwar den Variablennamen „a", der Typ wird jedoch nicht mehr mit angegeben. Diese Information muß also irgendwo gespeichert werden. Nun wäre es möglich bei flex und bison eine Information auszutauschen. Jedoch ist es nicht möglich mehrere solcher Informationen untereinander auszutauschen. Und hier schaltet sich Ox ein. Ox ermöglich diesen Datenaustausch. Und dies auf eine sehr einfache und elegante Weise, so daß beliebig viele Informationen ausgetauscht werden können.

3.4 iburg: Code - Generating Tool

Nachdem nun mit Hilfe der ersten drei Tools der C-Code in eine Baumstruktur umgesetzt werden konnte, steht nun das Problem an, wie setzte ich diese Struktur wieder in ausführbaren Code um, im Falle des FHOP-C-Compilers in Crash-Assemblercode.
Zur Lösung dieses Problems gib es nun das Tool iburg. iburg hilft bei der Umsetzung der Baumstruktur in Assemblercode. Wie bei flex und bison wird diesem Tool durch ein Definitionsfile mitgeteilt, welche Assemblerbefehle zulässig sind. Aus diesem Definitionsfile erzeugt iburg ein C-File. Diesem File, nachdem es compiliert wurde, wird die erzeugte Baumstruktur zugeführt. Iburg erzeugt dann aus der zugeführten Baumstruktur und den Assemblerbefehlen ein lauffähiges Assemblerfile.

3.5 Alle Tools zusammen -> Compiler

Nimmt man nun alle C-Codefiles, die von den Compilerbau-Tools erzeugt wurden, und linkt diese zusammen, erhält man einen Compiler. Dieser Compiler setzt dann, die ihm zugeführte Hochsprache, im Falle des FHOP-C-Compilers die Sprache C, in Maschinencode um.
Diese Art des Compilerbaus ist sehr komfortabel, weil die standardisierten Abläufe bei der Compilierung eines Sourcecodes durch die Tools übernommen werden.
Mit diesen Tools lassen sich auch Compiler für andere Hochsprachen erzeugen, sofern man die Befehle und Grammatik für den Sprache besitzt.
Aber auch die Generierung von Assemblern mit diesen Tools ist möglich, da bei Assembler keine komplizierten Grammatiken vorliegen, so daß man mit diesen Tools sehr schnell zu einem Ergebnis gelangen kann.

4 Was kann der Compiler?

Der Compiler wurde als Kommandozeilen - Compiler konzipiert. Dabei soll in der Befehlszeile außer dem Aufruf verschiedene Dinge angegeben werden können. So z.B. den Beginn des Data-Segmentes, das Data-Limit, der Beginn des Stackbereichs, u. a.. Da zunächst die Tools zum Compilerbau nur auf UNIX zur Verfügung standen, wurde begonnen auf UNIX zu entwickeln. Nachdem jedoch die Tools erfolgreich auf DOS-Ebene portiert wurden, konnte der Compiler ebenfalls auf der DOS-Plattform generiert werden.
Auf der DOS-Ebene sind bis jetzt folgende Befehle implementiert:

- Variablen:
  - int - Variablendeklaration
  - const int - Variablendeklaration
  - Eindimensionale int - Arrays
- Mathematische Operationen:
  - +, -, *, /, Modulo-Fkt
  - Incrementieren & Decrementieren
  - Klammersetzung

- Bitoperationen:
  - AND
  - OR
  - XOR
- Logische Operationen
  - AND
  - OR
- Schiebeoperationen
  - >> (Schiebe nach rechts)
  - << (Schiebe nach links)
- Vergleichsoperationen
  - == (Gleich)
  - != (Ungleich)
  - < (Kleiner als)
  - > (Größer als)
  - >= (Größer gleich)
  - <= (Kleiner gleich)
- Kommentarzeilen
- Inline - Befehle
  - 1 zu 1 - Übernahme der angeführten Assemblerbefehle
- if - Abfrage
  - einfache if - Abfrage
  - if - else - Abfrage
  - if - else if - Abfrage
  - if - else if - else Abfrage
- while - Schleifen
- for - Schleifen

Dieser Befehlsumfang, so hat sich gezeigt, genügt in erster Linie den Ansprüchen der Problematik der Assemblerprogrammierung. Deshalb wird dieser Befehlsumfang zuerst in einigen Projekten tiefergehend getestet. Dabei soll die Tauglichkeit und Zuverlässigkeit des Compilers geprüft werden. Spätere Erweiterungen des Befehlsumfanges sind angedacht und jederzeit möglich, so daß einer Erweiterung des Befehlsumfanges nach erfolgreichen Tests nichts im Wege steht.

Zum Schluß möchte ich noch kurz zwei Screen-Shots zeigen: die Help-Seite und eine erfolgreiche Compilierung:

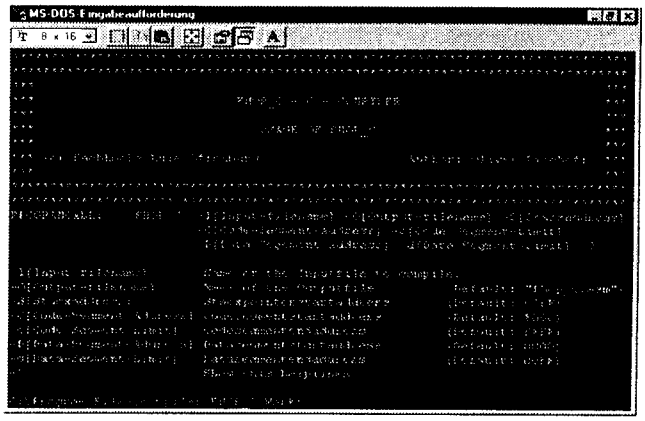


Abbildung 4 Die Help-Seite

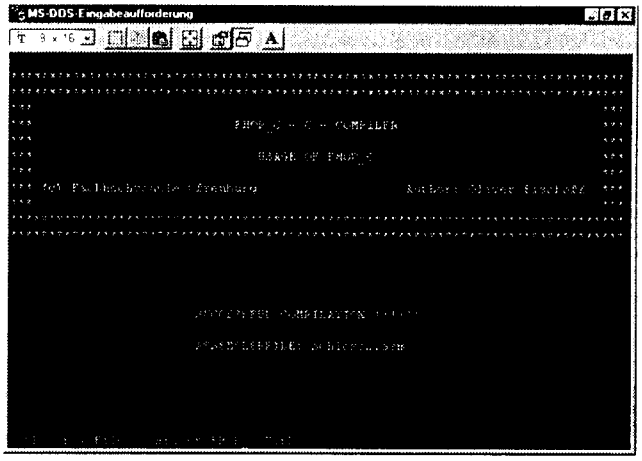


Abbildung 5 Eine erfolgreich Compilierung

5 Literatur

[1] HP-UX Reference Volume 1
SECTION 1: User Commands
Release 8.05

[2] Kurt M. Bischoff
Ox, Tutorial Introduction
November 1993

[3] Christopher W. Fraser
Engineering a Simple, Efficient Code Generator Generator

[4] Brian W. Kernighan
Dennis M. Richtie
The C - Programming Language 2nd Ed. Prentice Hall, 1988

Hochsprachen-Compiler für den FHOP

Dipl.-Ing. (FH) Oliver Bischoff,
Prof. Dr.-Ing. Dirk Jansen, ASIC-Design-Center
Fachhochschule Offenburg, Badstr. 24, 77652 Offenburg
Tel. 0781/205-267, Fax 0781/205-242,
E-Mail: d.jansen@fh-offenburg.de

An der Fachhochschule Offenburg wurde ein C-Compiler entwickelt. Es handelt sich dabei um einen Hochsprachen-Compiler für den an der Fachhochschule entwickelten Prozessor FHOP (First Homemade Operational Prozessor), der ANSI-C-Code in Maschinencode umsetzt.
Um den Maschinencode zu erzeugen, wird die Hilfe des bereits existierenden Crash - Assemblers in Anspruch genommen. Der C-Compiler Assemblercode erzeugt, welcher dann durch den Crash-Assembler in Maschinencode umgesetzt wird.
Der Compiler soll als Tool bei der Entwicklung neuer Software für den FHOP dienen. Hierbei wird vor allem an die Entwicklungsgeschwindigkeit gedacht, die für C-Code wesentlich höher ist, als für Assemblercode, der die selbe Funktionalität besitzt.

1 Einleitung

Die Programmiersprache „C" erfreut sich großer Beliebtheit und weiter Verbreitung. Dies beruht auf der Tatsache, daß C eine sehr übersichtliche und einfach zu erlernende Programmiersprache ist. Aufgrund der Vielzahl an Prozessoren stehen auch eine Menge an Compiler zur Verfügung; ein Teil ist „Retargetable", d. h. es besteht die Möglichkeit, mit diesen Compilern Maschinencode für Prozessoren zu erzeugen, die nicht vom selben Prozessortyp sind, wie der, auf dem der Sourcecode compiliert wurde. Diese Compiler, z. B. GNU, erfordern eine 32-Bit-Architektur des Prozessors, so daß eine Anpassung dieser Compiler für den FHOP nicht möglich war. Deshalb entschloß ich mich, mit Hilfe von Compilerbau-Tools, einen Compiler selbst zu schreiben. Dieser Compiler sollt sich so nah wie möglich an dem ANSI-C-Standard halten, damit es später möglich ist, bestehende C-Bibliotheken einzubinden.

2 Hochsprache! - Warum?

Das Programmieren von Prozessoren ist eine angenehme Arbeit, sofern die Tools zum Programmieren des Prozessors vorhanden sind. Maschinencode direkt einzugeben, ist sicher keine sehr angenehme Arbeit, zumal die Verwaltung des Speichers und sämtlicher Adressen in den Händen des Programmierers liegt. Variablen existieren auf dieser Programmierebene nicht. Der Code entbehrt jeglicher Übersicht. Eine Erleichterung bietet der Assembler, der das Erzeugen von Maschinencode komfortabler gestaltet. Er übernimmt die Adressenverwaltung und erlaubt das Anlegen von Variablen. Jedoch bleiben Zuverlässigkeit und Übersichtlichkeit weiterhin ein Problem des Assemblerprogramms. Diese Nachteile weißt ein Hochsprachen-Compiler nicht mehr auf. Sowohl die Übersichtlichkeit, als auch die Zuverlässigkeit des Codes sind weitaus besser, als bei einem Assemblercode. Dabei besticht vor allem die Lesbarkeit. Durch Unterteilung in einzelne Funktionen, erhält man ein übersichtlichen und lesbaren Code. Die Zuverlässigkeit wird durch erweiterte Syntaxprüfungen erhöht. Bei einem Assemblerprogramm kann das Vergessen eines Push- oder Pop-Befehls den Prozessor zum Abstürzen veranlassen, während man in einer Hochsprache dazu schon mehr braucht.
Ein weiteres Argument, welches die Vorteile eines C - Compilers unterstreicht, ist die Entwicklungszeit. Ein Vergleich (Abbildung 1) von zwei Programmausschnitten wird dies gut darstellen.

C - Code

```
main() {
    initialize();
    for(i=0;i<=3;i++) {
      tast_in();
      if(taste == kode[i])
        state++;
      else
        state=0;        }
  if(state==4)
    Tuer_out();
  else {
    state=0;
  Fehler_out();         }
}
```

Assembler-Code

```
main: CAL initialize
      LDI i_global
      PSH A
      LDI #0000
      POP B
      STA M
_FOR_1:
      LDA i_global
      PSH A
      LDI #0003
      POP B
      CAL KLGL_INT
      CMI 0
```

Abbildung 1 Vergleich C - / Assembler - Code

Dabei hat der unterstrichene C-Code links genau die gleiche Funktionalität wie der Assemblercode rechts.

Man sieht schon hier einen deutlichen Unterschied im Umfang.
Außerdem läßt sich der linke Code in Abbildung 1 wesentlich besser lesen. Durch die Anlehnung des C-Befehlssatzes an das Englische Vokabular kann sehr schneller verstanden werden, was ein bestimmter Programmteil ausführt. (z. B. if-Anweisungen). So läßt sich auch nach größeren Zeiträumen schneller nachvollziehen, welche Funktionalität der C - Code erfüllen soll. Ein Nachteil der Hochsprache der Umfang des erzeugten Codes. Bei Assembler-programmierung läßt sich der Code geringer halten. Jedoch ist der Speicher des FHOP so groß ausgelegt, so daß speicherplatzkritische Anwendungen bis jetzt nicht aufgetreten sind. Deshalb wird dieser Nachteil gerne in Kauf genommen, da die oben an - gesprochenen Vorteile überwiegen.

3 Wie wird ein Compiler erstellt?

Um die Vorteile der verwendeten Tools zu erkennen, möchte ich hier kurz auf die Funktionsweise eines Compilers eingehen.
Zuerst wird der Source-Code einer Überprüfung unterzogen. Wird diese erfolgreich abgeschlossen, d.h. der Source-Code ist fehlerfrei, wird versucht, den Source-Code durch eine Baumstruktur darzustellen. Abbildung 2 zeigt eine solche Baumstruktur.

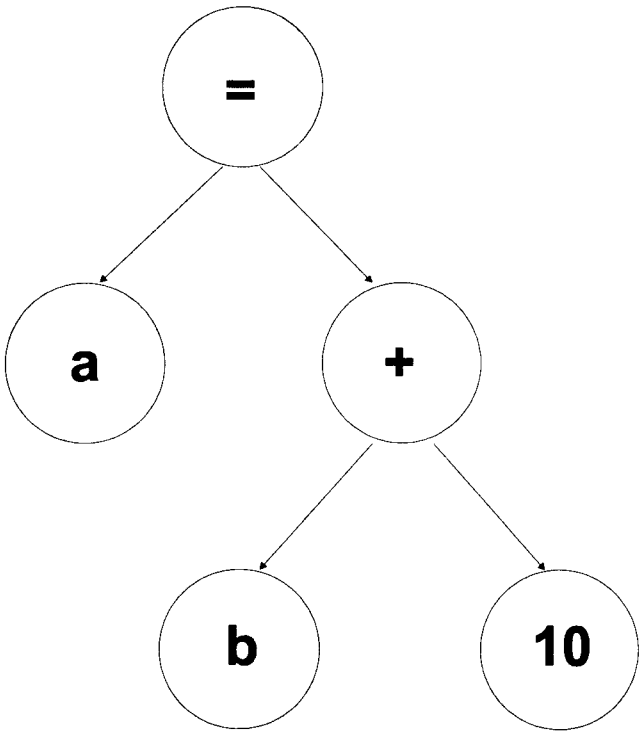


Abbildung 2 Baumstruktur

Die abgebildete Baumstruktur ergab die C-Anweisung „a = b + 10". Nachdem nun der gesamte Source-Code durch solche Baumstrukturen abgebildet wurde, versucht der Compiler diese Strukturen mit Assemblerbefehlen zu überdecken, d.h. in Assemblercode umzusetzen. Für die Baumerzeugung und die Überdeckung wurden nun Compilerbau-Tools verwendet. In Abbildung 3 wird der Ablauf „Source - Code → Maschinencode" schemenhaft dargestellt.

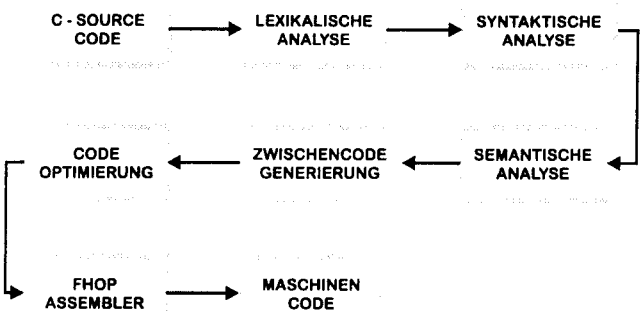


Abbildung 3 - Stufen der Codeerzeugung

Anhand dieses Schemas lassen sich die einzelnen Stufen der Umsetzung vom Source-Code in Assemblercode gut nachvollziehen. Das C-Source-File ist der Ausgangspunkt. Diese wird zuerst auf Richtigkeit überprüft und dann in 2 Schritten in Maschinencode umgesetzt. Schritt 1 ist die Erzeugung von Assemblercode. Dieser wird dann durch den Crash-Assembler in Maschinencode gewandelt. Der erste Schritt wird mit Hilfe von Compilerbau-Tools durchgeführt. Diese Tools helfen dem Anwender bei der Umsetzung der einzelnen Compilerfunktionen. Folgende Tools kamen dabei zum Einsatz:

3.1 flex: Scanner - Tool

Der Compiler erhält das vollständige File. Dieses muß zuerst auf Fehler durchsucht werden. Um diese Überprüfung durchführen zu können, muß dem Compiler bekannt sein, welche Ausdrücke zulässig sind und welche nicht .
Das Absuchen des C-Sourcefiles „von Hand" wäre sehr umständlich. Diese Arbeit übernimmt das Tool flex. flex ist die GNU-Variante des Standard-UNIX-Tools lex, welches bei allen UNIX-Betriebsystemen mit ausgeliefert wird. flex hat den Vorteil, daß es ein Freeware-Tool ist.
Diesem Tool werden nun die erlaubten Stringfolgen bzw. Befehle in einem Definitionsfile bekannt gemacht. Aus diesem File erzeugt flex dann ein C-File, welches dann compiliert die Funktion des Scanners übernimmt, der das gesamte File nach Fehlern in den Stringketten absucht. Diese Fehlersuche entspricht der lexikalischen Analyse, dargestellt als 2 Block in Abbildung 3.

3.2 bison: Parser - Tool

Die darauf folgende syntaktische Analyse erfolgt mit Hilfe des Tools bison. Bison ist ebenso wie flex eine GNU-Variante eines Standard-UNIX-Tools.
Wie auch bei lex, wird hier mit Hilfe eines Definitionsfiles dem Tool bekanntgemacht, welche Ausdrücke zulässig sind und welche nicht.
Bison wandelt das Definitionsfile in C-Code um, der, nachdem er compiliert wurde, die Funktion eines Parsers übernimmt.
Beim Parsen des Files wird nun nicht mehr auf die Richtigkeit einzelner Zeichenketten geachtet, sondern auf den sinnvollen Zusammenhang der Zeichenketten, die vom Scanner erkannt wurden.
So müssen bei bestimmten Operationen gewisse Bedingungen erfüllt sein. Bei einer mathematischen Operation, wie z.B. der Addition, müssen zwei Operanden zur Verfügung stehen. Dies heißt nun für den Parser, das beim Auffinden eines Additionszeichens der Scanner genau zwei Operanden liefern muß. Diese können direkt Zahlenwerte, Variablen oder auch beliebig komplexe Ausdrücke sein. Jedoch muß die Bedingung der zwei Operanden erfüllt sein. Ist dies nicht der Fall, so wird das Parsen mit einer Fehlermeldung abgebrochen.

3.3 Ox: Scanner & Parser - Tool

Ox ist ein Tool, welches die beiden vorherigen Tools, flex und bison unterstützt.
Beim Parsen und Scannen ist es notwendig, mehr Informationen als nur die vorgefundene Zeichenkette weiterzugeben. Dies läßt relativ gut an einem Beispiel zeigen.
Der Ausdruck „int a", definiert eine Variable „a" vom Typ „int". Trifft man nun im späteren Verlauf wieder auf diese Variable „a", z.B. bei „a = 0", so erkennt man nun zwar den Variablennamen „a", der Typ wird jedoch nicht mehr mit angegeben. Diese Information muß also irgendwo gespeichert werden. Nun wäre es möglich bei flex und bison eine Information auszutauschen. Jedoch ist es nicht möglich mehrere solcher Informationen untereinander auszutauschen. Und hier schaltet sich Ox ein. Ox ermöglich diesen Datenaustausch. Und dies auf eine sehr einfache und elegante Weise, so daß beliebig viele Informationen ausgetauscht werden können.

3.4 iburg: Code - Generating Tool

Nachdem nun mit Hilfe der ersten drei Tools der C-Code in eine Baumstruktur umgesetzt werden konnte, steht nun das Problem an, wie setzte ich diese Struktur wieder in ausführbaren Code um, im Falle des FHOP-C-Compilers in Crash-Assemblercode.
Zur Lösung dieses Problems gib es nun das Tool iburg. iburg hilft bei der Umsetzung der Baumstruktur in Assemblercode. Wie bei flex und bison wird diesem Tool durch ein Definitionsfile mitgeteilt, welche Assemblerbefehle zulässig sind. Aus diesem Definitionsfile erzeugt iburg ein C-File. Diesem File, nachdem es compiliert wurde, wird die erzeugte Baumstruktur zugeführt. Iburg erzeugt dann aus der zugeführten Baumstruktur und den Assemblerbefehlen ein lauffähiges Assemblerfile.

3.5 Alle Tools zusammen -> Compiler

Nimmt man nun alle C-Codefiles, die von den Compilerbau-Tools erzeugt wurden, und linkt diese zusammen, erhält man einen Compiler. Dieser Compiler setzt dann, die ihm zugeführte Hochsprache, im Falle des FHOP-C-Compilers die Sprache C, in Maschinencode um.
Diese Art des Compilerbaus ist sehr komfortabel, weil die standardisierten Abläufe bei der Compilierung eines Sourcecodes durch die Tools übernommen werden.
Mit diesen Tools lassen sich auch Compiler für andere Hochsprachen erzeugen, sofern man die Befehle und Grammatik für den Sprache besitzt.
Aber auch die Generierung von Assemblern mit diesen Tools ist möglich, da bei Assembler keine komplizierten Grammatiken vorliegen, so daß man mit diesen Tools sehr schnell zu einem Ergebnis gelangen kann.

4 Was kann der Compiler?

Der Compiler wurde als Kommandozeilen - Compiler konzipiert. Dabei soll in der Befehlszeile außer dem Aufruf verschiedene Dinge angegeben werden können. So z.B. den Beginn des Data-Segmentes, das Data-Limit, der Beginn des Stackbereichs, u. a.. Da zunächst die Tools zum Compilerbau nur auf UNIX zur Verfügung standen, wurde begonnen auf UNIX zu entwickeln. Nachdem jedoch die Tools erfolgreich auf DOS-Ebene portiert wurden, konnte der Compiler ebenfalls auf der DOS-Plattform generiert werden.
Auf der DOS-Ebene sind bis jetzt folgende Befehle implementiert:

- Variablen:
  - int - Variablendeklaration
  - const int - Variablendeklaration
  - Eindimensionale int - Arrays
- Mathematische Operationen:
  - +, -, *, /, Modulo-Fkt
  - Incrementieren & Decrementieren
  - Klammersetzung

- Bitoperationen:
  - AND
  - OR
  - XOR
- Logische Operationen
  - AND
  - OR
- Schiebeoperationen
  - >> (Schiebe nach rechts)
  - << (Schiebe nach links)
- Vergleichsoperationen
  - == (Gleich)
  - != (Ungleich)
  - < (Kleiner als)
  - > (Größer als)
  - >= (Größer gleich)
  - <= (Kleiner gleich)
- Kommentarzeilen
- Inline - Befehle
  - 1 zu 1 - Übernahme der angeführten Assemblerbefehle
- if - Abfrage
  - einfache if - Abfrage
  - if - else - Abfrage
  - if - else if - Abfrage
  - if - else if - else Abfrage
- while - Schleifen
- for - Schleifen

Dieser Befehlsumfang, so hat sich gezeigt, genügt in erster Linie den Ansprüchen der Problematik der Assemblerprogrammierung. Deshalb wird dieser Befehlsumfang zuerst in einigen Projekten tiefergehend getestet. Dabei soll die Tauglichkeit und Zuverlässigkeit des Compilers geprüft werden. Spätere Erweiterungen des Befehlsumfanges sind angedacht und jederzeit möglich, so daß einer Erweiterung des Befehlsumfanges nach erfolgreichen Tests nichts im Wege steht.

Zum Schluß möchte ich noch kurz zwei Screen-Shots zeigen: die Help-Seite und eine erfolgreiche Compilierung:

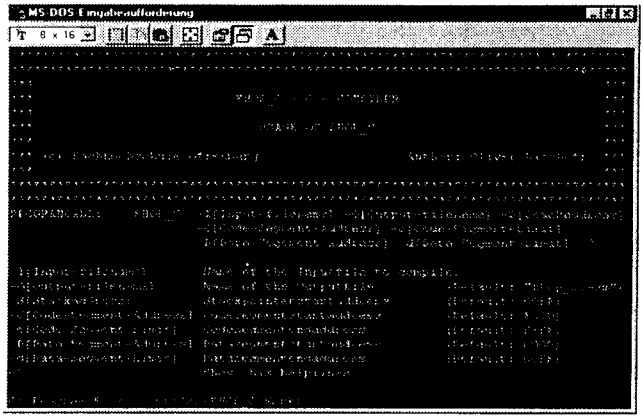


Abbildung 4 Die Help-Seite

Abbildung 5 Eine erfolgreich Compilierung

5 Literatur

[1] HP-UX Reference Volume 1
SECTION 1: User Commands
Release 8.05

[2] Kurt M. Bischoff
Ox, Tutorial Introduction
November 1993

[3] Christopher W. Fraser
Engineering a Simple, Efficient Code Generator Generator

[4] Brian W. Kernighan
Dennis M. Richtie
The C - Programming Language 2nd Ed. Prentice Hall, 1988

Einfache Lösungen für Digital-Analog-Wandler und für Analog-Digital-Wandler

Martin Rieger
Fachhochschule Albstadt-Sigmaringen
Johannesstraße 3, 72458 Albstadt,
Tel. 07431/579-124, email: rieger@fh-albsig.de

Kurzfassung

Es werden Wege aufgezeigt, wie in rein auf Digitalschaltungen ausgerichteten Technologien, Digital-Analog-Wandler und Analog-Digital-Wandler implementiert werden können. In unserem Labor wurden verschiedene Entwürfe zu diesem Themenbereich in PLDs realisiert. Die Tests zeigen, daß die Entwürfe funktionsfähig sind. Sigma-Delta-Wandler für Videosignale, sind wesentlich kompakter als vergleichbare Parallelwandler. Ein experimenteller Sigma-Delta-Analog-Digital-Wandler, der in einer Silizium-Technologie mit f_T = 20 GHz realisiert wurde, zeigt die Leistungsfähigkeit dieses Konzepts.

1. Einleitung

Seit einigen Jahren ist in der Halbleitertechnik der Trend hin zur Digitaltechnik und zu CMOS-Technologien klar erkennbar [1].
Insgesamt gesehen verlieren analoge Komponenten gegenüber digitalen Komponenten an Bedeutung. Daher wird bei der Entwicklung neuer Halbleitertechnologien mehr Gewicht auf die Optimierung digitaler Komponenten gelegt, die Entwicklungsgeschwindigkeit analoger Komponenten ist deutlich geringer.
Aus dieser Beobachtung heraus entsteht der Wunsch, Schaltungen, die bisher aus analogen Komponenten aufgebaut waren, soweit wie möglich aus digitalen Komponenten aufzubauen. Im Rahmen dieser Arbeit werden mehrere Ansätze für Digital-Analog-Wandler (DAC) und Analog-Digital-Wandler (ADC) gezeigt, bei denen die Anforderungen hinsichtlich der analogen Komponenten stark reduziert sind.
Ein weiteres Ziel der Arbeiten mit DACs und ADCs ist es, für einschlägige Vorlesungen und Praktika interessante und anschauliche Beispiele zu schaffen.

2. Digital-Analog-Wandler

2.1. Digital-Analog-Wandler mit Widerstandsnetzwerk

Grundlegende Ansätze für CMOS-DACs sind seit langem bekannt [2].
Eine einfache Lösung ist in Bild 1 dargestellt.

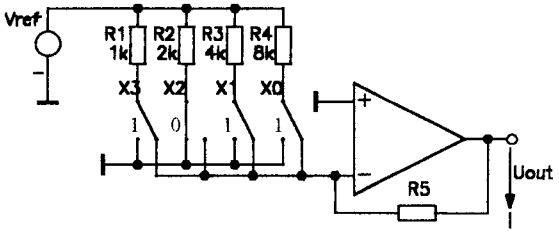


Bild 1: DAC mit binär gewichteten Widerständen und Operationsverstärker

Die binär gewichteten Widerstände verursachen entsprechend binär gewichtete Strömen, die je nach Schalterstellung X_i zur Masse oder zum Summationspunkt am Operationsverstärker fließen. Die binären Werte bestimmten die Schalterstellungen X_i, wobei die höchstwertige Stelle X_3 den höchsten Strom schaltet. Damit erhält man für V_{out} in Bild 1

$$V_{out} = -V_{ref} \cdot \left(\frac{1}{2} \cdot X_3 + \frac{1}{4} \cdot X_2 + \frac{1}{8} \cdot X_1 + \frac{1}{16} \cdot X_0 \right)$$

$$= -V_{ref} \cdot \frac{X}{X_{max} + 1},$$

wobei X der zu wandelnde binäre Wert und X_{max} der größte binäre Wert ist.

Für den in Bild 1 dargestellten DAC werden die Komponenten

Binär gestuftes Widerstandsnetzwerk,
Um- oder Einschalter und
Verstärker

benötigt. In rein digital ausgerichteten Technologien stehen diese Komponenten so nicht zur Verfügung und es erhebt sich die Frage, wie Ersatz beschafft werden kann.

Für die Widerstände kann in einfacher Weise kein direkter Ersatz gefunden werde, so daß diese Komponente außerhalb des ICs als externe Komponente verwendet werden müssen.

Anstelle der Schalter können Transfergates eingesetzt werden.

Der Verstärker kann durch einen Inverter ersetzt werden, wie dies in Bild 2 dargestellt ist. Allerdings sind Einschränkungen gegenüber

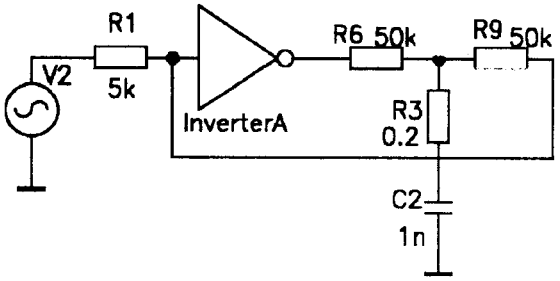


Bild 2: CMOS-Inverter als Operationsverstärker

einem üblichen Operationsverstärker zu beachten. Der Inverter erfordert aus Stabilitätsgründen eine Kompensation. Zudem kann am Eingang des Inverters das Bezugspotential nicht frei gewählt werden, vielmehr liegt das Bezugspotential fest bei der Entscheiderschwelle des Inverters.

Bild 3 zeigt eine Realisierung, wie sie in unserem Labor erprobt wurde. Die Spannungsquellen V_1 bis V_4 repräsentieren den zu wandelnden binären Wert. Die gestrichelt gezeichneten Einheiten sind in einem PLD integriert.

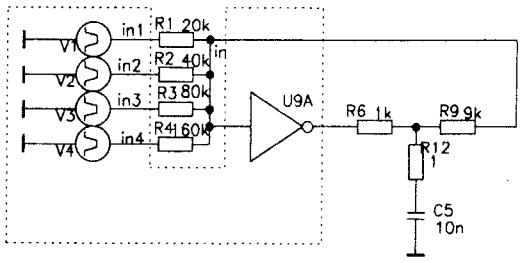


Bild 3:DAC mit CMOS-Inverter und binär gestuftem Widerstandsnetzwerk

Es wurden noch weitere DACs mit Widerstandsnetzwerk untersucht wobei R-2R-Netzwerke Verwendung fanden. Diese Netzwerke haben gegenüber binär gestuften Widerstandsnetzwerken den Vorteil von gleichartigen und somit besser matchenden Widerständen.

2.2. Digital-Analog-Wandler mit Pulsweitenmodulator

In rein digital ausgerichteten Technologien können Widerstandsnetzwerke nicht integriert werden, so daß sie als externe Komponenten hinzugefügt werden müssen. Eine rein digitale Lösung ohne externe Komponenten wäre hier vorzuziehen. Einen Ansatz hierzu bildet der Pulsweitenmodulator (Bild 4a).

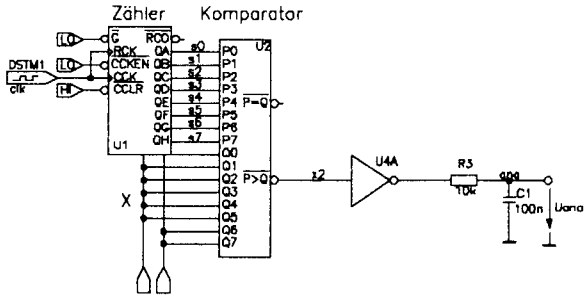


Bild 4a: Pulsweitenmodulator als DAC

Ein digitaler Komparator vergleicht den zu wandelnden binären Wert X mit dem Zählerstand s. Der Zähler inkrementiert seinen Stand mit jedem Clockimpuls, bis der maximale Zählerstand erreicht ist, dann springt der Zählerstand auf 0 und das Inkrementieren wird

fortgesetzt. Der Komparator-Ausgang x_2 ist high, solange s > X ist. Sobald s < X wird springt der Pegel bei x_2 auf low (Bild 4b).

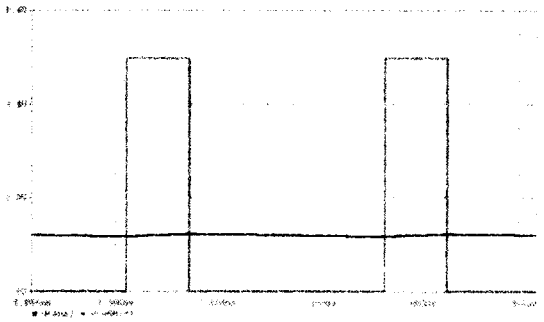


Bild 4b: Simulationsergebnis zu Bild 4a: V(U4) (Pulse), V(ana) (Gleichwert)

Natürlich ist diese Kurvenform an x_2 noch nicht brauchbar. Ein nachfolgendes Tiefpassfilter unterdrückt die höheren Frequenzanteile weitgehend, so daß V_{ana} nahezu frei von Störanteilen ist.
Den niederfrequenten Spannungsanteil erhält man zu

$$V_{ana} = V_{ref} \cdot (1 - \frac{X}{2^n - 1}).$$

Damit ist es tatsächlich gelungen, den DAC mit hoher Auflösung fast vollständig zu integrieren, lediglich das Tiefpassfilter ist extern.
Ein Nachteil des Pulsweitenmodulators liegt allerdings in der begrenzten Geschwindigkeit. Einfache Überlegungen zeigen, daß die Grenzfrequenz f_g des Tiefpasses

$$f_g \leq \frac{f_{clk}}{2^n}$$

sein muß, wobei f_{clk} die Frequenz des Clocks und n die Auflösung des DACs sind.
Für eine akzeptable Einschwingzeit des DACs sind die Anforderungen an die Clockfrequenz und somit an die Schaltgeschwindigkeit der Komponenten recht hoch.
Ein DAC ähnlich zu Bild 4a wurde in unserem Labor mit einem PLD realisiert. Die Programmierung des PLDs wurde in VHDL vorgenommen. Bei einer Auflösung von 8 Bit und einer Taktfrequenz von 40 MHz wurde die Grenzfrequenz auf 1 kHz gelegt, so daß die Wandelgeschwindigkeit maximal 1kHz beträgt.

Eine wesentliche Verbesserung hinsichtlich der Geschwindigkeit brächte die Anwendung eines Sigma-Delta-Wandlers [3]. Dabei wird, anschaulich ausgedrückt, die high- und low-Zeit in Bild 4b feiner verteilt, so daß die Grenzfrequenz fg wesentlich höher gewählt werden darf. Überlegungen zu einem DAC nach dem Sigma-Delta-Prinzip sind bei uns im Gange.

3. Analog-Digital-Wandler

3.1. Analog-Digital-Wandler nach dem Wägeverfahren

Das wohl naheliegendste Verfahren zur Analog-Digital-Wandlung ist das Parallelverfahren [2] (Bild 5).

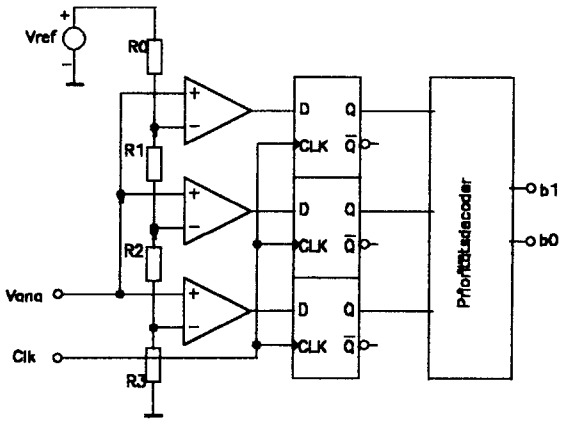


Bild 5: Parallelwandler

Dabei wird das zu digitalisierende Signal V_{ana} in einer Kette von analogen Komparatoren mit Bruchteilen der Referenzspannung V_{ref} verglichen. Die Ergebnisse des Vergleichs werden in Flipflops gespeichert und im Prioritätsdecoder in binäres Format umgewandelt.
Parallelwandler haben zwar den Vorteil sehr hoher Wandelgeschwindigkeit, dem stehen aber die Nachteile hoher Leistungs- und Flächenverbrauch, nur mittlere Auflösung und schlechte Integrierbarkeit in digitale Technologien gegenüber.
Interessanter erscheinen Verfahren, die nach dem Wägeprinzip arbeiten [2] (Bild 6). Hier wird mittels eines Wägeverfahrens eine analoge Spannung U_z so nachgeführt, daß sie der

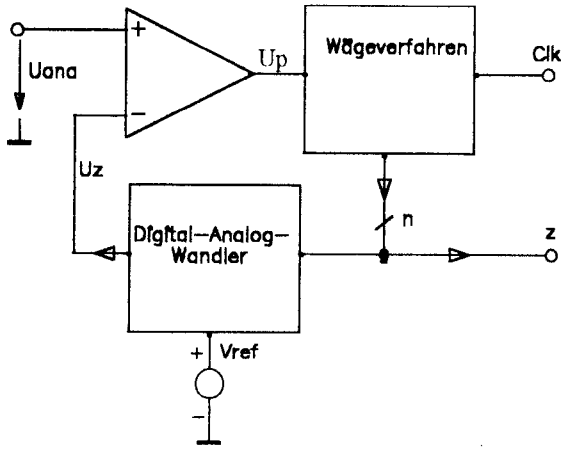


Bild 6: ADC mit Wägeverfahren

zu wandelnden Spannung U_{ana} möglichst ähnlich wird. Je nach verwendetem Wägeverfahren und je nach Typ des Digital-Analog-Wandlers erreicht man mit ADCs nach dem Wägeverfahren eine mäßige bis hohe Umsetzgeschwindigkeit. Der Aufwand ist eher gering und bei konsequenter Ausnutzung aller schaltungstechnischen Möglichkeiten ist dieser Wandlertyp auch in rein digitalen Technologien gut integrierbar.
Es sind verschiedene Wägeverfahren möglich, ein recht leistungsfähiges Verfahren arbeitet mit sukzessiver Approximation. Das Verfahren ist in Bild 7a am Beispiel eines 3 Bit Wandlers dargestellt. Es wird, beginnend mit dem höchstwertigen Bit und dann absteigend in Richtung niederwertigstem Bit, eine Bitstelle nach der anderen probeweise gesetzt. Ist $U_{ana} > U_z$, d.h. Up=1, so bleibt das geprüfte Bit gesetzt, ansonsten wird das geprüfte Bit rückgesetzt.
In unserem Labor wurde ein ADC nach dem Verfahren der sukzessiven Approximation in VHDL entworfen und in ein PLD implementiert. Der DAC in der Rückführung des ADCs arbeitet mir Pulsweitenmodulation, wie in Bild 4a erläutert. Dabei ist das Tiefpassfilter des DAC, wie in Bild 7b gezeigt, mit der Komparatorschaltung verknüpft. Der Inverter übernimmt, zusammen mit den Widerständen R_l und R_g die Funktion des Komparators. Die rückgewandelte Spannung V_z stellt sich so ein, daß

$$V_z = -V_{ana}\frac{R_l}{R_g} - V_{th}\frac{R_l + R_g}{R_g}$$

wird, wobei V_{th} die Entscheiderschwelle des Inverters ist, d.h. V_z stellt ein Abbild von V_{ana} dar.
Somit konnte der gesamte ADC, mit Ausnahme von R_l, R_g und C_g, in eine rein digital ausgerichtete Technologie integriert werden.

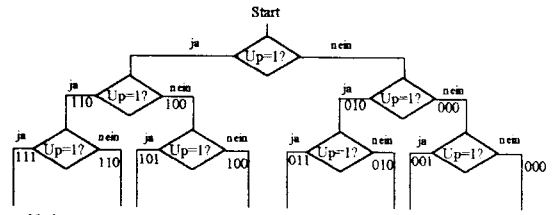


Bild 7a: Wägeverfahren mit sukzessiver Approximation

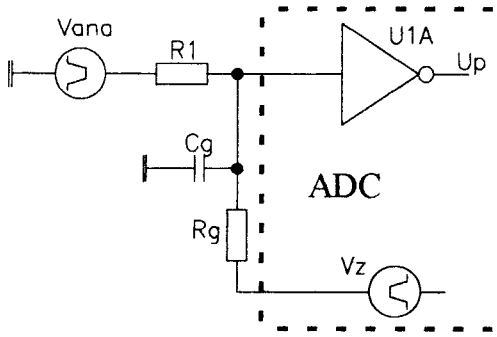


Bild 7b Realisierter ADC

3.2. Analog-Digital-Wandler mit Sigma-Delta-Modulator

ADCs mit Sigma-Delta-Modulator sind in der Verarbeitung von Audiosignalen schon lange im Einsatz, während für Videoanwendungen bisher Parallelwandler nach Bild 5 verwendet werden. Wie schon geschildert, weisen diese Parallelwandler aber einige Nachteile auf, so daß auch hier Wandler mit Sigma-Delta-Wandler eine interessante Alternative darstellen.
Bild 8a zeigt einen Sigma-Delta-Wandler 2. Ordnung. Der digitale Datenstrom d(t) stellt sich so ein, daß er den analogen Momentanwert x(t) möglichst gut nachbildet.

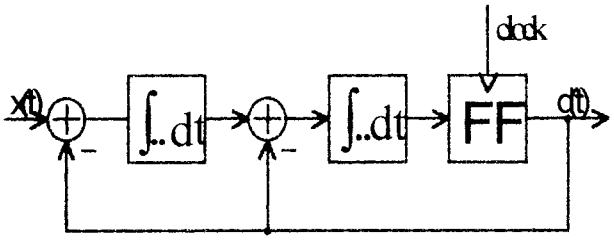


Bild 8a: Sigma-Delta-Wandler 2. Ordnung

Von der Differenz x(t) - d(t) der äusseren Schleife wird das Integral gebildet. Vom Ergebnis der Integration wird in der inneren Schleife d(t) subtrahiert und diese Differenz wird ebenfalls integriert. Ein Flipflop dient zur Speicherung und als 1-Bit-ADC. Dieses Flipflop wir mit einer sehr hohen Taktfrequenz f_{clk} betrieben, die sehr viel größer als die höchste im analogen Signal vorkommende Frequenz f_s ist. Der digitale 1-Bit Datenstrom d(t) wird von einem folgenden Dezimationsfilter in ein paralleles Format gewandelt.

Ein Sigma-Delta-Wandler 1. Ordnung kann ein Signal-Rausch-Verhältnis

$$SNR = 20 \cdot \log\left(\frac{\sqrt{6}}{\pi} \cdot \left(\frac{f_{clk}}{2 \cdot f_s}\right)^{3/2}\right)$$

erzielen, während ein Sigma-Delta-Wandler 2. Ordnung

$$SNR = 20 \cdot \log\left(\frac{\sqrt{60}}{\pi^2} \cdot \left(\frac{f_{clk}}{2 \cdot f_s}\right)^{5/2}\right)$$

erreichen kann [3]. Eine Abschätzung zeigt, daß mit einem Oversampling-Verhältnis

$$\frac{f_{clk}}{2 \cdot f_s} = 64$$

ein Signal-Rausch-Verhältnis besser als 60 dB erreichbar sein sollte.

In Zusammenarbeit mit der Firma Thomson-Multimedia wurde ein experimenteller Sigma-Delta-Wandler entworfen und hergestellt [4], [5], [6].

Zur Realisierung stand eine 20 GHz Silizium Bipolartechnologie zur Verfügung. Schon in der Simulation zeigte sich, daß die Transistoren dieser Technologie nicht so schnell schalten, wie dies idealerweise für einen Sigma-Delta-Wandler 2. Ordnung wünschenswert ist. Wegen der mangelhaften Schaltgeschwindigkeit arbeitet die innere Schleife des Wandler nur unzureichend. Das erreichbare Signal-Rausch-Verhältnis liegt somit näher bei dem für einen Sigma-Delta-Wandler 1. Ordnung erwarteten Wert, als bei dem Wert, der einen Sigma-Delta-Wandler 2. Ordnung erhofft werden kann.

Bild 9a zeigt den in ein 20-Pin LCC-Gehäuse eingebauten Die. Die Gesamtfläche des Chips beträgt 6 mm^2, aber die aktive Fläche des Sigma-Delta-Wandlers nimmt davon nur 0.45 mm^2 in Anspruch. Der größte Teil der Fläche ist für Testschaltungen, ECL-Treiber und für Abblockkondensatoren verwendet. Der Modulator-Kern verbraucht eine Leistung von 250 mW an 5V. Die nominale Taktfrequenz beträgt 1.28 GHz.

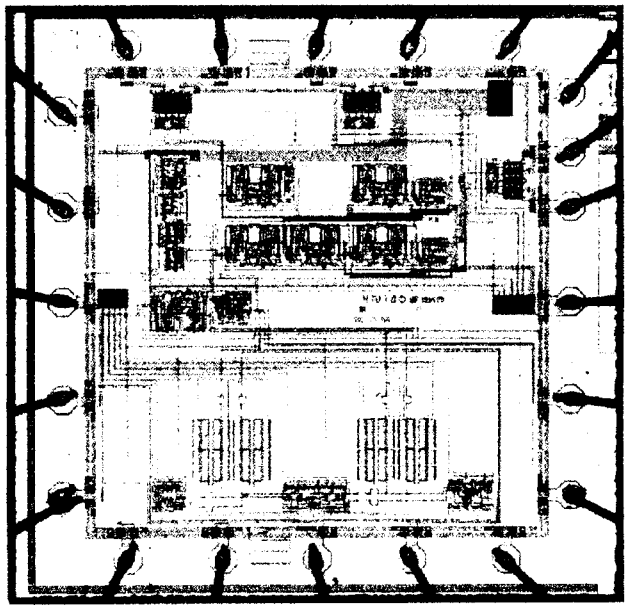


Bild 9a: Chip-Photo des Sigma-Delta-Wandlers

In Bild 9b ist das gemessene Signal-Rausch-Verhältnis in Abhängigkeit von der Clockfrequenz dargestellt. Die Messungen zeigen gute Übereinstimmung mit Spice-Simulationen. Wie aus der Simulation zu erwarten, wird für das Signal-Rausch-Verhältnis etwa 45 dB erreicht. Diese Performanz ist für viele Anwendungen, wie z. B. digitales Fernsehen, ausreichend.

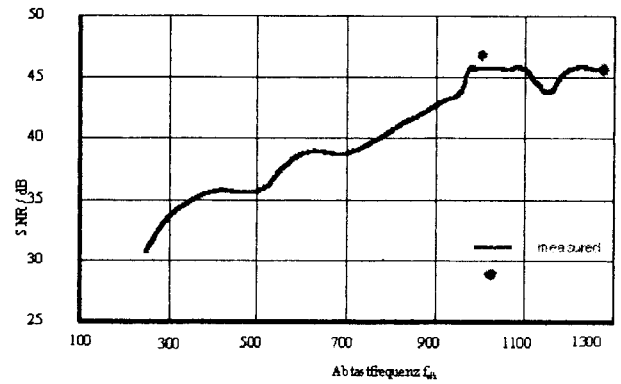


Bild 9b: Signal-Rausch-Verhältnis des Sigma-Delta-Wandlers

4. Zusammenfassung

Es konnte gezeigt werden, wie in rein auf Digitalschaltungen ausgerichteten Technologien, in denen keine Widerstände und Analog-Komparatoren zur Verfügung stehen, Digital-Analog-Wandler und Analog-Digital-Wandler implementiert werden können. Die Entwürfe wurden zusammen mit Studenten ausgeführt, wobei die Erfahrung gewonnen wurde, daß sich diese Art von DACs und ADCs gut für Entwurfsarbeiten im Rahmen von Praktika eignet.

In unserem Labor wurden verschiedene Entwürfe von DACs und ADCs in PLDs realisiert. Die Tests zeigen, daß die Entwürfe funktionsfähig sind.

Eine Unzulänglichkeit in unserer Simulationsumgebung besteht zur Zeit noch darin, daß Schaltungen mit VHDL-Komponenten und mit Analog-Komponenten nur getrennt nach Analog- und Digitalteil simuliert werden können, was für die ADC-Simulation recht störend ist. Sobald dieser Mangel behoben ist und somit zuverlässige Simulationen möglich sind, werden die DAC und ADC-Entwürfe in einem ASIC realisiert.

In Zusammenarbeit mit Thomson-Multimedia entstand ein experimenteller Sigma-Delta-Wandler für Videosignale. Dieser Ansatz ermöglicht ADCs, die wesentlich kompakter sind als die bei der Videoverarbeitung bisher verwendeten Parallelwandler. Die Schaltung, die in einer Silizium-Technologie mit f_T = 20 GHz realisiert. erwies sich als funktionsfähig. Allerdings wäre eine noch schnellere Technologie wünschenswert, um einen Sigma-Delta-Wandler 2. Ordnung mit hohen Taktfrequenz von 1.2 GHz ideal zu realisieren.

Literaturverzeichnis

[1] The National Technology Roadmap for Semiconductors: Semiconductor Industry Association (SIA) 1997 edition report published by SEMA-TECH, Inc.

[2] A. B. Grebene: "Bipolar and MOS Analog Integrated Circuit Design" John Wiley & Sons 1983.

[3] J.C. Candy, G. C. Temes: "Oversampling Methods for A/D and D/A Conversion". Oversampling delta-sigma data converters: theory, design and simulation, IEEE Press 1992

[4] M. Erbar, M. Rieger, H. Schemmann: A 1.28-GHz Sigma-Delta Modulator for Video A/D Conversion, ICCE 1996 digest of technical papers S. 78,79 ISSN 0-7803-3029-3

[5] M. Erbar, M. Rieger, H. Schemmann: The Fastest Sigma-Delta A/D Converter on Si, Nikkei Electronics 1996.10.7 (no. 672) S. 123 - 130 ISSN 0385-1680

[6] M. Erbar, M. Rieger, H. Schemmann: A 1.28-GHz Sigma-Delta Modulator for Video A/D Conversion, IEEE Transactions on Consumer Electronics, Aug. 1996, vol. 42, No. 3 S. 357 - 361 ISSN 0098-3063

Selbsttest mit Hilfe der Signaturanalyse

G. Stura, A. Führer

Fachhochschule Ulm, Prittwitzstraße 10, 89075 Ulm

Telefon: 0731/50-28338

Im Rahmen einer Diplomarbeit wurde die Signaturanalyse anhand der Literatur untersucht. Die Aufgabe bestand darin, eine Aussage über die Maskierung von Fehlern zu treffen. Ziel war es, ein Signaturregister zu entwickeln, das eine möglichst geringe Fehlermaskierung hat. Die Anwendung der Signaturanalyse wurde anhand eines Beispiels verdeutlicht.

1. Einleitung

1.1 Testen von integrierten Schaltungen

Nach der Herstellung müssen die integrierten Schaltungen getestet werden. Bei der Entwicklung muß der Test der Schaltung berücksichtigt werden.
Der Test kann mit einem externen Testautomaten durchgeführt werden. Beim externen Test ist es nötig, auf dem Chip einen Prüfpfad zu integrieren. Es müssen Testvektoren generiert werden, die an die Schaltung angelegt und sequentiell in den Prüfpfad geschoben werden. Nach jedem Arbeitstakt können die Antworten an den Ausgängen beobachtet werden.
Beim Selbsttest ist kein Testautomat erforderlich, sondern nur eine Testhilfe, die den Chip initialisiert und die Taktfrequenz zur Verfügung stellt. Beim Selbsttest werden Teile der eigenen Logik für den Test der Schaltung verwendet. Das Testergebnis liegt in Form eines Signals an einem Ausgang der Schaltung an.

1.2 Vorteile des Selbsttests

Der Test von Schaltungen mit externen Testgeräten stellt mit zunehmender Komplexität einen enormen Kostenfaktor dar, weil leistungsfähigere und damit kostenintensivere Testautomaten benötigt werden. Für den Selbsttest wird eine Testhilfe verwendet, die preisgünstig ist.
Für die Implementierung des Prüfpfades wird eine große Chipfläche benötigt. Die Logik des Selbsttests nimmt zumeist eine geringere Fläche ein. Die benötigte Chipfläche für den Selbsttest muß nicht unbedingt proportional mit der Schaltungsgröße wachsen.
Die Testqualität ist höher, weil als Taktfrequenz die Betriebsfrequenz des Chips verwendet wird.
Der Test einzelner Baugruppen kann parallelisiert werden.

1.3 Signaturanalyse - ein geeignetes Verfahren für den Selbsttest

Um eine Schaltung zu testen, ist es erforderlich, eine bestimmte Menge von Testmustern zu erzeugen, die an die Eingänge der Schaltung angelegt werden. Man bezeichnet diesen Vorgang als Testmustergenerierung. An den Ausgängen der Schaltung können die Testantworten der Schaltung beobachtet werden. Die Auswertung der Testantworten gibt anschließend Auskunft über die Funktion der Schaltung.
Für den korrekten Ablauf des Tests ist eine Steuerung zuständig.

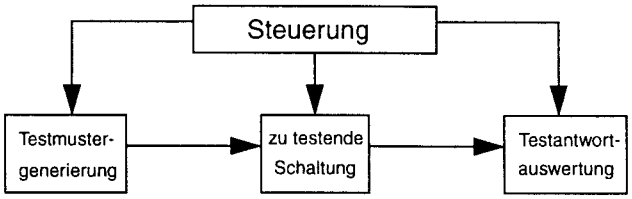


Bild 1 Verschiedene Testfunktionen aus [2]

Beim externen Test übernimmt der Testautomat die drei Testfunktionen (Bild 1).
Für einen vollständigen Selbsttest müssen alle drei Testfunktionen auf dem Chip integriert werden. Um eine Aussage über die Fehlerfreiheit der Schaltung machen zu können, müssen die Testantworten mit den Antworten der fehlerfreien Schaltung verglichen werden.

Die Speicherung der Sollantworten würde einen hohen Speicherplatzbedarf ergeben. Die Signaturanalyse ist ein mögliches Verfahren, bei dem die fehlerfreien Testantworten nicht gespeichert werden müssen. Stattdessen werden die Testantworten zu einem Wort komprimiert und am Testende wird überprüft, ob dieses Wort korrekt ist.

2. Signaturanalyse

Die Signaturanalyse wird mit einem linear rückgekoppelten Schieberegister durchgeführt. Man unterscheidet zwischen einem seriellen und parallelen Schieberegister (Bild 2). Der Unterschied zwischen den beiden Signaturregistern besteht in der Anzahl der Eingänge. Das serielle Signaturregister hat nur einen Eingang, unabhängig von der Anzahl der Registerzellen. Bei einem parallelen Schieberegister werden weitere Eingänge an einzelnen Registerzellen über EXOR-Verknüpfungen hinzugefügt.
Das serielle Signaturregister hat in der Praxis keine Bedeutung, weil die zu testenden Schaltungen meistens über mehrere Ausgänge verfügen. Im folgenden wird jedoch das serielle Signaturregister behandelt, weil die Theorie einfacher zu erläutern ist.
Eine Besonderheit des Signaturregisters ist, daß der Inhalt der letzten Registerzelle mit Hilfe von EXOR-Verknüpfungen auf einzelne Registerzellen zurückgeführt wird. Man spricht von Rückkopplungen.
Die zu komprimierenden Testantworten werden an den Eingang des Signaturregisters angelegt. Sobald eine neue Testantwort verfügbar ist, werden die Inhalte der Registerzellen um eine Stelle nach rechts geschoben und der Wert der letzten Registerzelle wird in das Register zurückgekoppelt. Die Signatur ist das Wort, welches am Ende des Tests im Signaturregister steht.
Das Schieberegister kann durch ein Polynom beschrieben werden. Aus der Anzahl der Register ergibt sich die höchste Potenz des Polynoms. Jede Rückkopplung der letzten Registerzelle entspricht einem Element im Polynom. Beispielweise stellt in Bild 2 das Element x die Rückkopplung an die Registerzelle 3 dar.
Durch die frei wählbaren Rückkopplungen im Schieberegister läßt sich jedes beliebige Polynom vom gegebenen Grad einstellen.
Das Polynom des Schieberegisters wird auch Generatorpolynom $G(x)$ genannt. Die obige Aussage über das Generatorpolynom gilt für die serielle und parallele Signaturanalyse in gleicher Weise.

2.1 Mathematische Grundlage

Das Prinzip der seriellen Signaturanalyse beruht auf der mathematischen Polynomdivision [4].
Die Testantworten lassen sich addieren und als ein Polynom mit einer großen Anzahl von Elementen interpretieren. Dieses Polynom wird auch Eingangspolynom $P(x)$ genannt. Bei der Signaturanalyse wird dieses Polynom durch das Generatorpolynom $G(x)$ des Schieberegisters geteilt. Nach Beendigung der Division entspricht die Signatur dem Restpolynom $R(x)$ der Polynomdivision. Der Quotient der Polynomdivision wird nicht gespeichert. Dadurch ergibt sich eine Datenkompression, die nicht wieder rückführbar ist.

$$\frac{\text{Eingangspolynom } P(x)}{\text{Generatorpolynom } G(x)} = \text{Quotient } Q(x) + \frac{\text{Restpolynom } R(x)}{\text{Generatorpolynom } G(x)}$$

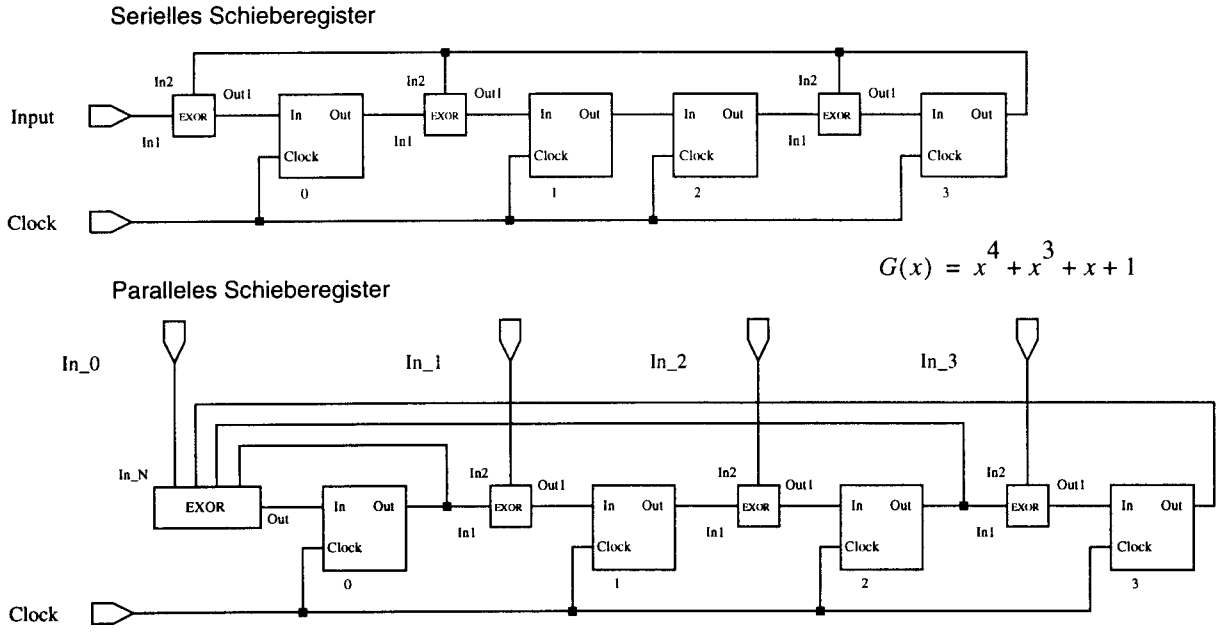


Bild 2 Serielles und paralleles Schieberegister

2.2 Fehlerfreie Schaltung

Am Ende des Schaltungstests steht im Signaturregister die Signatur der getesteten Schaltung. Der Vergleich mit der abgespeicherten Signatur der fehlerfreien Schaltung ergibt das Testergebnis. Entsprechen sich die gebildete und die abgespeicherte Signatur, ist die Schaltung fehlerfrei.

Beispiel für eine fehlerfreie Schaltung

Für dieses Beispiel wird ein Eingangspolynom 7. Ordnung gewählt. In der Praxis würden Polynome mit einer sehr viel höheren Ordnung verwendet werden.

Testantwort $P = 10001010$

Eingangspolynom $P(x) = x^7 + x^3 + x$ (1)

Für die Signaturanalyse wird ein Schieberegister mit 5 Flipflops verwendet.

Generatorpolynom $G(x) = x^5 + x^3 + x + 1$

Das Ergebnis der Polynomdivision zeigt Gleichung (2).

$$\frac{x^7 + x^3 + x}{x^5 + x^3 + x + 1} = (x^2 + 1) + \frac{x^3 + x^2 + 1}{x^5 + x^3 + x + 1} \qquad (2)$$

Die Signatur der fehlerfreien Schaltung beträgt:

$$R(x) = x^3 + x^2 + 1$$

2.3 Fehlerhafte Schaltung

Ein Fehler macht sich dadurch bemerkbar, daß am Ausgang der Schaltung eine andere Folge von Bits auftritt, als im fehlerfreien Fall. Die defekten Bits werden durch ein Fehlerpolynom *E(x)* beschrieben.
Wenn das Polynom *E(x)* nur Nullen beinhaltet, ist die Schaltung fehlerfrei und das Ausgangsmuster entspricht dem korrekten Datenwort. Befindet sich aber eine 1 in dem Fehlerpolynom, so ergibt sich an dieser Position ein fehlerhaftes Ausgangsbit.
Das fehlerbehaftete Eingangspolynom *P'(x)* ergibt sich durch eine Überlagerung des korrekten Datenworts *P(x)* mit dem Fehlerpolynom *E(x)*. Entsprechend Gleichung (3).

$$P'(x) = P(x) \oplus E(x) \qquad (3)$$

Beispiel für eine fehlerhafte Schaltung

Es wird die gleiche Schaltung mit dem selben Signaturregister verwendet wie im vorigen Beispiel:

Eingangspolynom $P(x) = x^7 + x^3 + x$

Generatorpolynom $G(x) = x^5 + x^3 + x + 1$

Ein Fehler macht sich dadurch bemerkbar, daß die Testantworten von der fehlerfreien Schaltung abweichen. Die defekten Bits werden durch das folgende Fehlerpolynom beschrieben:

Fehlerpolynom $E(x) = x^5 + x^3 + 1; \quad E = 00101001$

Das Fehlerpolynom überlagert sich mit dem Eingangspolynom. Es ergibt sich eine fehlerhafte Testantwort:

$$P'(x) = P(x) \oplus E(x)$$

$$P'(x) = (x^7 + x^3 + x) \oplus x^5 + x^3 + 1 = x^7 + x^5 + x + 1$$

$$P' = P \oplus E = 10001010 \oplus 00101001 = 10100011$$

Das Ergebnis der Signaturanalyse bei der fehlerhaften Schaltung entspricht Gleichung (4):

$$\frac{x^7 + x^5 + x + 1}{x^5 + x^3 + x + 1} = (x^2) + \frac{x^3 + x^2 + x + 1}{x^5 + x^3 + x + 1} \qquad (4)$$

Das Restpolynom weicht von der Signatur der fehlerfreien Schaltung (2) ab.
Die defekte Schaltung wird erkannt.

2.4 Maskierung

Wenn der Registerinhalt im Schieberegister nach dem Test nicht der Signatur der fehlerfreien Schaltung entspricht, müssen die Testantworten von der fehlerfreien Schaltung abweichen. Die Schaltung reagiert anders als erwartet und ist aus diesem Grund fehlerbehaftet.
Fehler werden aber dann nicht erkannt, wenn eine fehlerhafte Ausgangsfolge auf die gleiche Signatur abgebildet wird wie die fehlerfreie Ausgangsfolge. Man spricht von einer Fehlermaskierung. Ziel muß es sein, diese möglichst gering zu halten.
Besteht das Fehlerpolynom aus einer Summe von Vielfachen des Generatorpolynoms, so wird der Fehler nicht erkannt. In diesem Fall wird die Division ohne einen Rest berechnet, und die Signatur nicht beeinflußt. Nur der Quotient, der nicht weiter ausgewertet wird, verändert sich.

Beispiel für eine Fehlermaskierung
Es wird die gleiche Schaltung mit dem selben Signaturregister verwendet wie im vorigen Beispiel.

Eingangspolynom $P(x) = x^7 + x^3 + x$

Generatorpolynom $G(x) = x^5 + x^3 + x + 1$

Es wird ein Fehlerpolynom gewählt, das ein Vielfaches vom Generatorpolynom ist. Das Generatorpolynom wird mit x multipliziert, und ergibt so das verwendete Fehlerpolynom.

Fehlerfolge $E = 01010110$

Fehlerpolynom $E(x) = x^6 + x^4 + x^2 + x$

Das Fehlerpolynom überlagert sich mit dem Eingangspolynom. Es ergibt sich eine fehlerhafte Testantwort.

$$P'(x) = P(x) \oplus E(x)$$

$$P'(x) = (x^7 + x^3 + x) \oplus x^6 + x^4 + x^2 + x$$

$$P'(x) = x^7 + x^6 + x^4 + x^3 + x^2$$

$$P' = P \oplus E = 10001010 \oplus 01010110 = 11011100$$

Das Ergebnis der Signaturanalyse bei der fehlerhaften Schaltung entspricht der Gleichung:

$$\frac{x^7 + x^6 + x^4 + x^3 + x^2}{x^5 + x^3 + x + 1} = (x^2 + x + 1) + \frac{x^3 + x^2 + 1}{x^5 + x^3 + x + 1}$$

Das Restpolynom weicht nicht von der Signatur der fehlerfreien Schaltung (2) ab.
Die defekte Schaltung wird nicht erkannt.

2.5 Maskierungswahrscheinlichkeit für die serielle Signaturanalyse

Unter der Voraussetzung, daß alle Fehlerpolynome gleich wahrscheinlich sind, ergibt sich die folgende Maskierungswahrscheinlichkeit nach [3]:

$$Pm = \frac{2^{m-r} - 1}{2^m - 1} \text{ ; für m >> r gilt: } Pm \approx 2^{-r} \qquad (5)$$

Damit ist die Maskierungswahrscheinlichkeit nur von der Breite des Signaturregisters abhängig und geht mit wachsender Registerlänge exponentiell gegen Null. Ein Polynom mit einer einfachen Rückkopplung hat die gleiche Maskierungswahrscheinlichkeit wie ein Polynom mit mehreren Rückkopplungen.

Die Wahrscheinlichkeit, daß ein Fehler maskiert wird, beträgt z. B. bei einem Polynom 8. Ordnung:

$$Pm = 2^{-8} = 0,39\ \%$$

Diese Fehlermaskierungswahrscheinlichkeit ist nur gültig, wenn alle Fehlerpolynome gleich wahrscheinlich sind.

2.6 Maskierungswahrscheinlichkeit für die parallele Signaturanalyse

Auch bei der parallelen Signaturanalyse ist die Maskierungswahrscheinlichkeit nur von der Breite des Signaturregisters abhängig und es gilt Gleichung (5). Diese Aussage wurde durch eine Simulation bestätigt.

Eine besondere Fehlerart bei der parallelen Signaturanalyse ist die Fehlerannulierung, bei der sich zwei Fehler in unterschiedlichen Takten auslöschen. In diesem Fall wird der erste Fehler durch den zweiten Fehler annuliert, bevor eine abweichende Signatur entstehen kann. Die Fehlerannulierung ist in der Fehlermaskierungswahrscheinlichkeit nach Gleichung (5) berücksichtigt.

2.7 Wahl eines geeigneten Generatorpolynoms

In der Praxis kann man nicht von der gleichen Wahrscheinlichkeit aller Fehler ausgehen. Die Fehlerpolynome sind abhängig von der Fehlerart und der Struktur der Schaltung.
Es stellt sich die Frage, ob ein Schieberegister so entworfen werden kann, daß unabhängig davon, welcher Art die Fehlermuster sind, die Maskierungswahrscheinlichkeit immer minimal ist.
In der Grafik (Bild 3) sind Fehlermaskierungen von Polynomen mit verschiedenen Eigenschaften dargestellt. Dabei ist die Anzahl der Register mit $r = 4$ konstant. Nur die Rückkopplungen variieren. Jedes der drei Signaturregister ist mit verschiedenen Mehrfachfehlern untersucht worden.

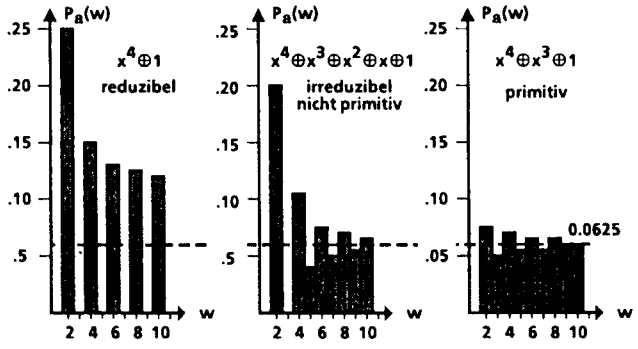


Bild 3 Fehlermaskierung *Pa* bei einem *w*-fachen Fehler im Fehlerpolynom aus [1]

Das Signaturregister mit dem reduziblen Polynom erkennt alle Fehlerpolynome mit einer ungeraden Parität. Dafür werden die Fehlerpolynome mit einer geraden Parität stark maskiert. In der Summe ergibt sich wieder die bekannte Maskierungswahrscheinlichkeit nach Gleichung (5).
Beim Signaturregister mit dem irreduziblen Polynom werden nicht alle Fehlerpolynome mit ungerader Parität erkannt. Die Maskierungswahrscheinlichkeit schwankt aber nicht so stark wie beim zuvor betrachteten Generatorpolynom.
Das primitive Polynom zeigt eine gleichmäßige Maskierungswahrscheinlichkeit mit nur geringen Schwankungen um den stationären Endwert bei allen Mehrfachfehlern.
Bei einer defekten Schaltung kann keine Aussage über die Art und Häufigkeit von Fehlern gemacht werden. Aus diesem Grund hat das primitive Polynom die besten Eigenschaften für ein Signaturregister.
Der Einfluß des charakteristischen Polynoms auf die Maskierung gilt bei der seriellen und parallelen Signaturanalyse in gleicher Weise.

2.8 Möglichkeiten, um die Fehlermaskierung zu reduzieren

Durch die Datenkompression ergibt sich ein Informationsverlust, der nicht vermeidbar ist. Aus diesem Grund ist es nicht möglich, die Maskierungswahrscheinlichkeit gänzlich auszuschließen. Durch folgende Maßnahmen kann sie verringert werden:

a) Die Fehlermaskierung kann reduziert werden, wenn das Signaturregister vergrößert wird. Die Wahrscheinlichkeit, daß ein Fehler nicht erkannt wird, ist abhängig von der Anzahl der Register *r*. Eine Reduzierung um den Faktor 0,5 kann durch Einfügen eines weiteren Registers erreicht werden. Der damit verbundene Hardwareaufwand ist im Bezug auf die Gesamtschaltung sehr gering.

b) Eine andere Möglichkeit wäre, den Test periodisch zu unterbrechen und mehrere Signaturen auszuwerten. Bei dem Vergleich von mehreren Signaturen für einen Testsatz ergibt sich folgende Maskierungswahrscheinlichkeit:

$$Pm \cong \frac{1}{2^{s \cdot r}} \qquad \text{mit s = Anzahl der Signaturen, r = Anzahl der Registerzellen} \qquad (6)$$

c) Eine weitere Methode zur Verringerung der Maskierungswahrscheinlichkeit ist die Wiederholung des Tests mit einem anderen Polynom.

2.9 Zusammenfassung der Eigenschaften von Signaturregistern

Bei der gleichen Wahrscheinlichkeit aller Fehlerpolynome ist die Maskierungswahrscheinlichkeit

- abhängig von der Länge des Signaturregisters
- unabhängig vom verwendeten Generatorpolynom
- unabhängig von der Verwendung eines seriellen oder parallelen Schieberegisters

Unabhängig vom Fehlerpolynom nähert sich die Maskierungswahrscheinlichkeit

- bei einem primitiven Generatorpolynom am schnellsten dem stationären Wert an.

3. Anwendung der Signaturanalyse

3.1 Anwendung der Signaturanalyse zum Testen eines Mikroprozessors

Mikroprozessoren eignen sich besonders gut für den Selbsttest. Die vorhandene "Intelligenz" der Schaltung kann für die Testdurchführung verwendet werden (Bild 4).
Der modulare Aufbau eines Mikroprozessors kann dazu genutzt werden, einzelne Komponenten unabhängig voneinander zu testen .
Die Testvektoren können in dem schon bestehenden ROM abgespeichert werden.
Ein Testprogramm aus dem vorhandenen Befehlssatz kann die Testvektoren an die Teilschaltungen anlegen und den Testablauf steuern.
Mit einem Signaturregister am Bus können die Testantworten der einzelnen Komponenten ausgewertet werden.
Mit dem Rechenwerk kann die Signatur mit der im ROM abgelegten fehlerfreien Signatur verglichen werden.

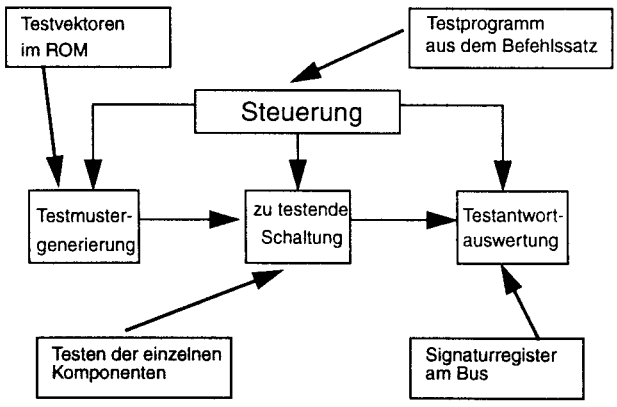


Bild 4 Selbsttest beim Mikroprozessor

Die Fachhochschule Offenburg hat einen Mikroprozessor-Design-Kit entwickelt. Anhand der ALU (Arithmetic Logic Unit) dieses Design-Kits (Bild 5) soll die Anwendung der Signaturanalyse im Zusammenhang mit dem Selbsttest erläutert werden.

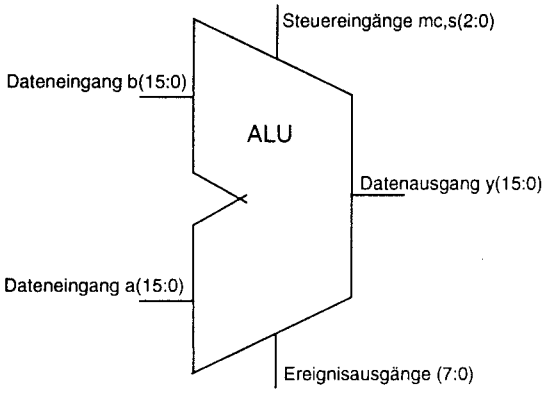


Bild 5 Arithmetische Recheneinheit (ALU)

3.2 Teststrategie

Die ALU in Bild 5 besitzt zwei Datenein- und einen Datenausgang (16 Bit breit), drei Steuereingänge und acht Ereignisausgänge (Flags). Am Datenausgang wird ein 16 Bit breites und an den Ereignisausgängen ein 8 Bit breites Signaturregister vorgesehen. Mit einem automatischen Testmustergenerator werden Testvektoren ermittelt, die eine möglichst hohe Fehlerüberdeckung ergeben.
Diese generierten Testvektoren müssen durch ein Testprogramm an die Eingänge der ALU angelegt werden. Dazu müssen zwei Voraussetzungen erfüllt sein:

1. Anlegen der Steuersignale
Um eine bestimmte Kombination der Steuersignale zu erzeugen, muß ein entsprechender Befehl verwendet werden.
Für jede mögliche Signalkombination der Steuerleitungen muß es mindestens einen Befehl geben, der das entsprechende Muster von Signalen erzeugt, damit alle möglichen Testvektoren an die ALU angelegt werden können.

2. Anlegen der Signale an die Dateneingänge
Um an die Dateneingänge entsprechende Signale anlegen zu können, müssen die Register, die bei der Rechenoperation beteiligt sind, vorbelegt werden. Es ist problemlos möglich, ein bestimmtes Bitmuster im Register für einen Testvektor zu erzeugen, weil die dazugehörigen Daten in einem Datenfeld im ROM abgespeichert werden können. Schwierigkeiten können nur dadurch entstehen, daß keine Signalquelle(Register) zur Verfügung steht, die das Datum an die Eingänge anlegt z.B. ist bei einem Schiebebefehl nur ein Dateneingang aktiviert, der zweite bleibt undefiniert.

Somit muß für jeden Testvektor ein Befehl vorhanden sein, der die entsprechenden Signale für die Steuerleitungen zur Verfügung stellt und das Anlegen eines Datums an die Dateneingänge ermöglicht.
Anhand des Mikrocodes wird jeder einzelne Befehl in Bezug auf diese Forderungen untersucht.

Nach der Analyse des gesamten Befehlssatzes steht fest, daß nicht alle Testvektoren mit einem Programm an die ALU angelegt werden können. Die betroffenen Testvektoren im Testsatz müssen soweit verändert werden, daß ein Programm mit den gegebenen Befehlen in der Lage ist, die entsprechenden Testvektoren zu erzeugen.
Anhand des so veränderten Testsatzes kann bestimmt werden, welche Fehlerüberdeckung mit diesen Testvektoren erreicht wird.

3.3 Ergebnisse des vorgeschlagenen Testverfahrens

Die Fehlerüberdeckung des ursprünglichen Testsatzes beträgt 99,78% für alle Fehler, die an den Ausgängen der ALU zu einem falschen Ergebnis führen. Der mit dem Testprogramm erzeugte modifizierte Testsatz ergibt eine Fehlerüberdeckung von 99,18%. Wenn alle Fehler mit der gleichen Wahrscheinlichkeit auftreten können, erhöht sich die Maskierungswahrscheinlichkeit nicht, wenn mehrere unabhängige Signaturregister für die Testauswertung verwendet werden. Die verwendeten Schieberegister mit 8. und 16. Ordnung ergeben insgesamt dieselbe Maskierungswahrscheinlichkeit *Pm*, wie sie bei einem Schieberegister mit 24 Flipflops auftreten würde:

$$Pm = \frac{1}{2^{24}} \approx 5,9604 \cdot 10^{-6}\ \% \qquad (7)$$

Das Ergebnis von Gleichung (7) zeigt deutlich, daß die Maskierungswahrscheinlichkeit bei der Auswertung der Testergebnisse sehr gering ist.
Die Testqualität wird durch die Anwendung der Signaturanalyse so gut wie nicht verschlechtert. Sie hängt praktisch nur von der Fehlererkennungswahrscheinlichkeit des Testprogramms ab.
In diesem Fall wurde der Selbsttest der ALU nachträglich entwickelt. Das hat zu einer Senkung der Fehlerüberdeckung um 0,6 % geführt.
Diese Differenz, wäre nicht aufgetreten, wenn die Testbarkeit bei der Schaltungsentwicklung berücksichtigt worden wäre.

4. Zusammenfassung

Das Ergebnis macht deutlich, daß die Signaturanalyse ein geeignetes Verfahren für die Auswertung der Testvektoren ist. Die Fehlerüberdeckung des Tests der ALU zeigt, daß der Selbsttest eine Alternative zum externen Test ist. Insbesondere dann, wenn bei der Entwicklung die Testbarkeit berücksichtigt wird.

5. Literatur

[1]
M. Gerner, B. Mueller, G. Sandweg: "Selbsttest digitaler Schaltungen", Oldenbourg Verlag, München, 1990

[2]
Birgit Reeb: "Generierung von qualitativ hochwertigen Testmustern für den Selbsttest von digitalen Schaltungen", DDD, Dr. und Verlag, Darmstadt, 1997

[3]
W. Daehn: "Testverfahren in der Mikroelektronik", Springer Verlag, Berlin, 1997

[4]
P.H. Bardell, W.H. McAnney, J. Savir: "Built-In Test for VLSI: Pseudorandom Techniques", John Wiley & Sons, Inc., New York, 1987

Entwicklung eines ASIC für Ethernet-Analysatoren

Jochen Jesinger
Fachhochschule Ravensburg-Weingarten
Tel.: 0751/501-680, E-Mail: jesinger@cae-ws01.fbe.fh-weingarten.de
http://cae-ws10.fbe.fh-weingarten.de/jesinger

1. Thema der Diplomarbeit

Thema der Diplomarbeit ist die Entwicklung eines ASIC für Ethernet-Analysatoren. Entwickelt wird unter der VLSI-Entwicklungsumgebung von Mentor Graphics auf UNIX Workstations von Hewlett Packard. Als Entwicklungswerkzeug stehen die Programme Design Architect, Quicksim II und der VHDL-Compiler AutoLogic zur Verfügung.

Zieltechnologie ist zum einen die FPGA-Familie XC4010 von Xilinx. Der FPGA verhält sich ähnlich wie ein statischer RAM-Baustein. Er läßt sich bequem über ein EPROM mit den Schaltungsdaten laden. Wird er von der Versorgungsspannung getrennt, so verliert er alle gespeicherten Informationen wieder. So lassen sich sehr schnell neue Schaltungen entwickeln und testen.

Die Flexibilität der Xilinx-Technologie hat jedoch ihren Preis: Erstens liegt die maximale Taktfrequenz des FPGA weit unter der eines ASIC. Zweitens ist die Zahl der Logikzellen begrenzt und drittens ist der FPGA in der Massenproduktion deutlich teurer als ein ASIC.

Als zweite Zieltechnologie ist deshalb ein Gate Forest ASIC der Firma IMS vorgesehen. Die auf dem FPGA getesteten Schaltungsteile werden später auf diese Technologie portiert und zu einem fertigen Produkt integriert.

2. Das Ethernet - Konzept

2.1 Historie

Ethernet basiert auf dem ALOAH-Protokoll, das 1970 an der Universität Hawaii entwickelt wurde und zur Rechnervernetzung per Funk diente. ALOAH wurde von Xerox weiterentwickelt und 1976 als Experimental Ethernet mit einer Übertragungsrate von 3MBit/s vorgestellt. Ein Konsortium aus DEC, Intel und Xerox entwickelte auf dieser Basis Ethernet v1.0 (1980) und Ethernet 2.0 (DIX 2.0). Parallel dazu stellte die IEEE ihren Standart 802.3 vor, der Ethernet 2.0 in vielem ähnlich ist.

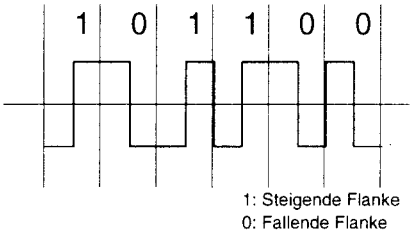
Heute wird der Begriff „Ethernet" oft sowohl für Ethernet 2.0 als auch für IEEE 802.3 Netzwerke verwendet.

2.2 Zugriff auf den Übertragungskanal

Mit Ethernet wurde zum ersten Mal ein CSMA/CD LAN realisiert. CSMA/CD bedeutet „**C**arrier **S**ense **M**ultiple **A**ccess with **C**arrier **D**etect" und beschreibt, wie auf das Übertragungsmedium zugegriffen wird: Der Kanal wird sowohl vor als auch während der Übertragung eines Datenpakets abgehört. Wird dabei eine Kollision erkannt, so wird nach dem Senden eines Jam-Signals die Übertragung abgebrochen. Das Jam-Signal besteht aus 4 bis 6 Byte beliebiger Information.

Erst nachdem eine statistische Wartezeit verstrichen ist, wird erneut auf den Kanal zugegriffen.

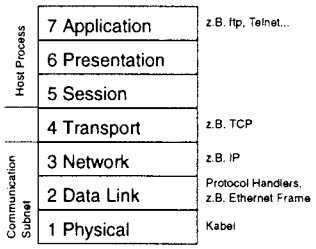
Für die Leitungscodierung wird Manchester-Code verwendet. Dies ermöglicht eine einfache Taktrückgewinnung und Kollisionserkennung.



Zur Verkabelung von 10 Mbps Ethernet wird meist Koaxialkabel mit 50 Ohm Wellenwiderstand oder Twisted Pair (zwei verdrillte Doppeladern) verwendet. Für 100Mbps Ethernet (Fast Ethernet) steht Twisted Pair oder Lichtwellenleiter zur Verfügung.

2.3 Ethernet im OSI Schichtenmodell

Das OSI Schichtenmodell teilt die Kommunikation zwischen Datenstationen in sieben Schichten auf. Die unteren drei Schichten werden „Communication Subnet" genannt. Sie regeln den eigentlichen Datenaustausch zwischen den Stationen. Bei den oberen drei Schichten spricht man vom „Host Process". Die mittlere Schicht, die Transportschicht, stellt einen Puffer zwischen Communication Subnet und Host Process dar.



2.4 Ethernet 2.0 Datagramme

Ethernet 2.0 Frames (Datagramme) beginnen mit einer 7 Byte langen Präambel. Sie besteht aus einer Folge aus 1010... bit und dient zur Synchronisation der empfangenden Stationen.
Der darauffolgende Start Frame Delimiter (SFD) zeigt das Ende der Präambel und den Beginn des Frame Headers an. Der Start Frame Delimiter ist ein Byte lang und unterscheidet sich von den Präambel-Bytes durch ein 1-Bit am Schluß.

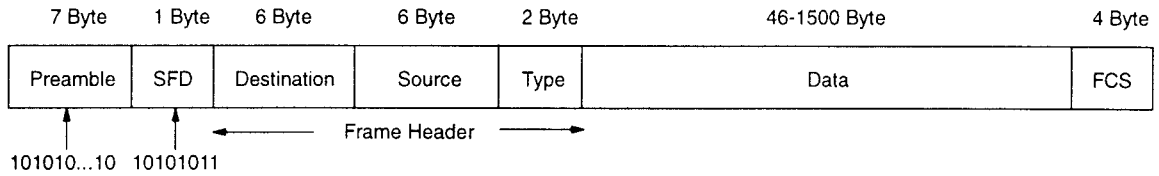
Der Frame Header besteht aus der Zieladresse (Destination), der Quelladresse (Source) und dem Typenfeld. Die beiden Adressfelder sind je 6 Byte lang. Im 2 Byte langen Typenfeld wird definiert, welches übergeordnete Protokoll im Datenteil des Frames verwendet wird (z.B. IPX oder IP).

Der darauffolgende Datenteil muß mindestens 46 Byte lang sein und darf eine Länge von 1500 Byte nicht überschreiten. Er enthält Daten und Protokolle ab Schicht 3 im OSI Schichtenmodell.

Am Ende des Frames befindet sich die 4 Byte lange Frame Check Sequence (FCS). Sie enthält eine CRC Prüfsumme, die von der sendenden Station aus dem Header und dem Datenteil errechnet wird. Die empfangende Station ermittelt die CRC ebenfalls und vergleicht sie mit dem empfangenen Wert. So kann überprüft werden, ob der Frame korrekt übertragen wurde.

Sowohl im Header als auch im Datenteil und in der FCS wird das niederwertigste Bit (LSB, Least Significant Bit) zuerst gesendet. Zwischen zwei Frames muß mindestens eine Wartezeit von 9,6µs verstreichen (Inter-Frame-Gap).

Ethernet 2.0 Frame



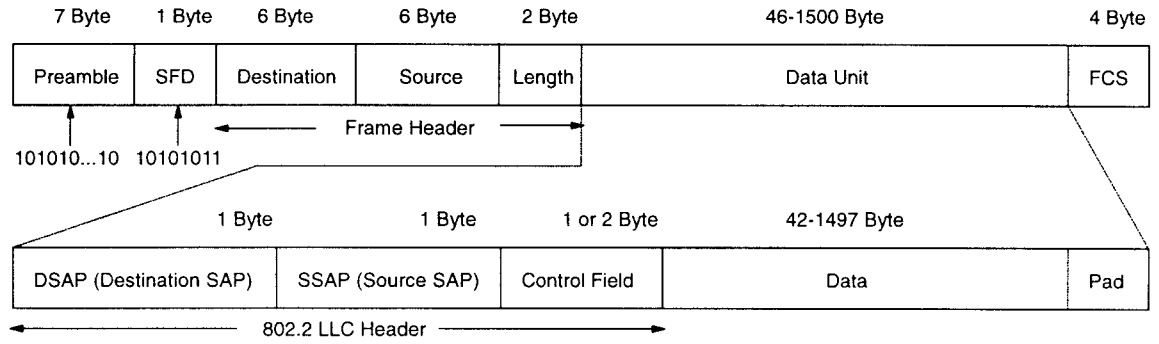
2.5 Unterschiede zwischen Ethernet 2.0 und IEEE 802.3 Datagrammen

Bei 802.3 Frames wird das Datenfeld automatisch mit sogenannten Pad-Zeichen auf die minimale Länge von 46 Byte aufgefüllt. Dies wird bei Ethernet 2.0 auf höherer Protokollebene abgewickelt.

Außerdem wird bei 802.3 Frames statt des Typenfeldes ein 2 Byte langes Längenfeld eingefügt. Es gibt die Anzahl der Bytes im Datenteil (ohne Pad-Zeichen) an.

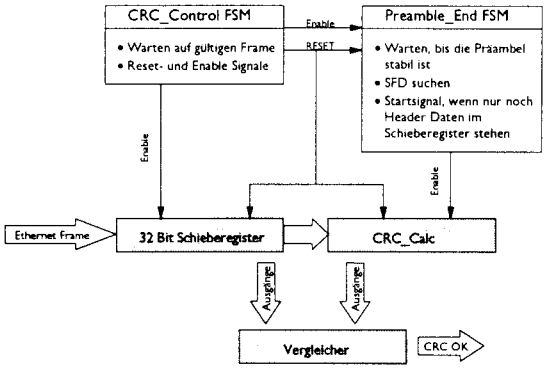
Ethernet 2.0 und IEEE 802.3 Frames haben zwar auf dem Data Link Layer die gleiche Länge, sind jedoch auf dem Network Layer und auf den Schichten darüber inkompatibel, da im Datenteil eines IEEE 802.3 Frames noch ein 3 bzw. 4 Byte langer 802.2 LLC Header verwendet wird. Er ist das Gegenstück zum Typenfeld bei Ethernet 2.0. Über Service Access Points (SAP) wird eine Schnittstelle zu den Protokollen auf dem Network Layer (Schicht 3) bereitgestellt.

IEEE 802.3 Frame



3. Stand der Diplomarbeit

Bisher wurde ein Schaltungsteil realisiert, der überprüft, ob ein Ethernet-Frame eine korrekte Frame Check Sequence besitzt. Er besteht aus einem 32 Bit langen Schieberegister, einem Automaten, der die CRC berechnet (CRC_Calc) und zwei FSMs. Für die Berechnung der FCS ist nur der Header- und Datenteil eines Frames relevant. Die Schaltung funktioniert sowohl bei IEEE 802.3 als auch bei Ethernet 2.0 Frames.



Zur Prüfung der FCS wird der Ethernet Frame zunächst durch das Schieberegister und danach durch CRC_Calc geschoben. Bricht der Empfangstakt am Ende des Frames ab, so bleiben die letzten 32 Bit des Frames - die FCS - im Schieberegister hängen. So ist sichergestellt, daß die FCS nicht durch CRC_Calc läuft.
Die FSM „Preamble_End" prüft das Ende der Präambel ab. CRC_Calc beginnt erst dann mit der Berechnung der CRC, wenn das erste relevante Bit am Ende des Schieberegisters angekommen ist. Die FSM „CRC_Control" liefert Reset- und Enable Signale für die restliche Schaltung und erkennt den Anfang und das Ende eines Frames.

Nachdem ein Frame Schieberegister und CRC_Calc vollständig durchlaufen hat, wird die empfangene mit der berechneten CRC verglichen und angezeigt, ob der Frame korrekt übertragen wurde.

4. Literatur

[1] Miller, M.: *LAN Protocol Handbook,* M&T Publishing

[2] Sweeney, P.: *Codierung zur Fehlererkennung und Fehlerkorrektur,* Hanser-Verlag

[3] Hegering, H.-G., Läpple, A.: *Ethernet, Basis für Kommunikationsstrukturen* Datacom-Verlag

[4] Advancen Micro Devices: *Ethernet/IEEE 802.3 Family, World Network Data Book*

[5] Advancen Micro Devices: *How to calculate packet CRC in software,* http://www.amd.com/products/npd/software/pcnet/utilities/document/exampcrc.html

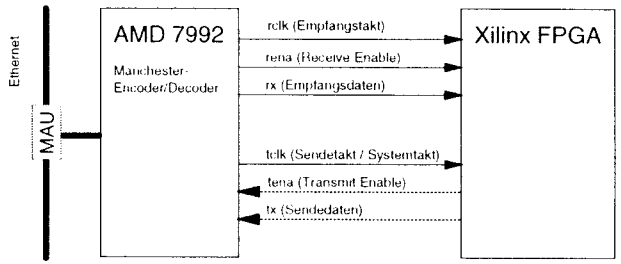
Thema der Diplomarbeit:

Entwicklung eines ASIC für Ethernet-Analysatoren

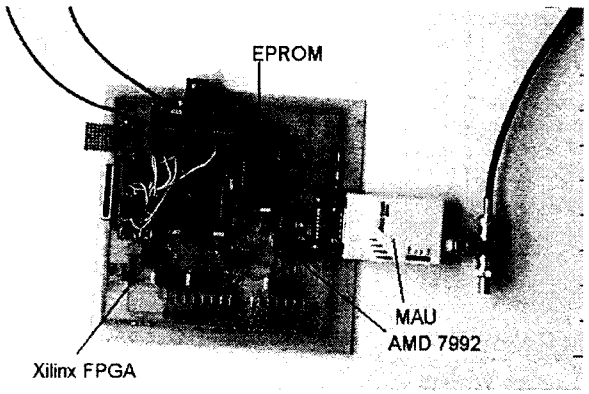
- Fehlererkennung (z.B. Prüfsumme, korrekte Länge...)
- Filterung der Daten für Mikrocontroller oder PC

Testumgebung: Xilinx FPGA
Zieltechnologie: IMS Gate Array

Der Versuchsaufbau

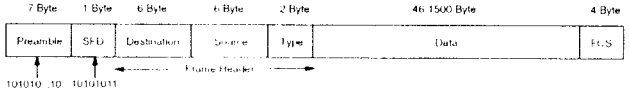


Das Testboard

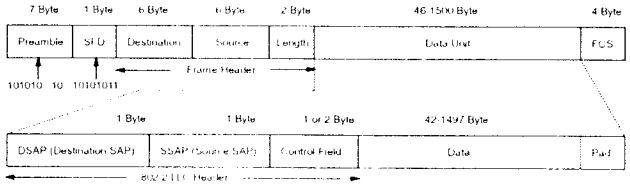


Ethernet Frame-Formate

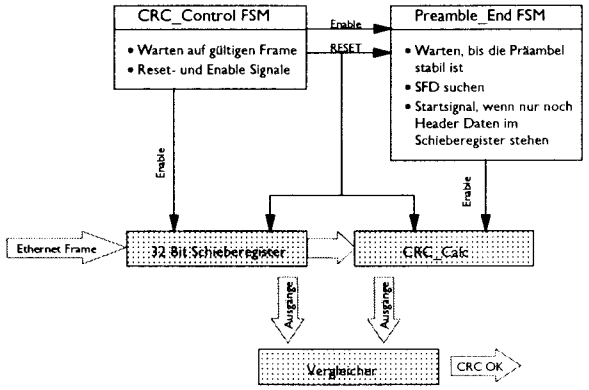
Ethernet 2.0 Frame



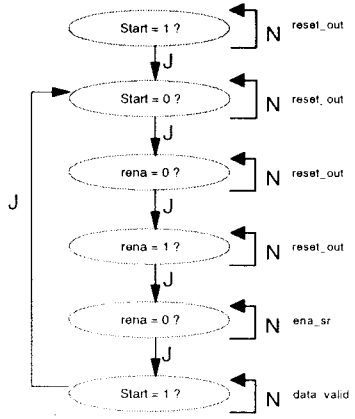
IEEE 802.3 Frame



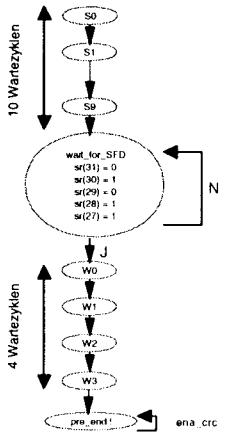
CRC Check



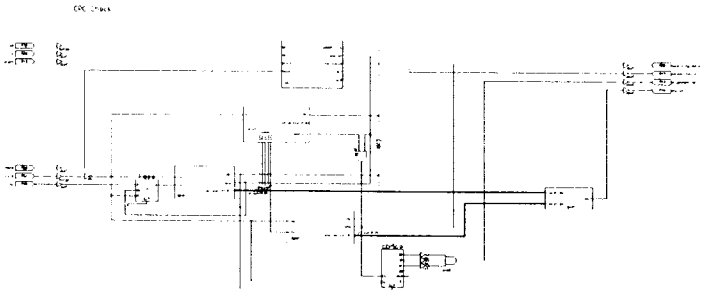
CRC Control FSM



Preamble End FSM

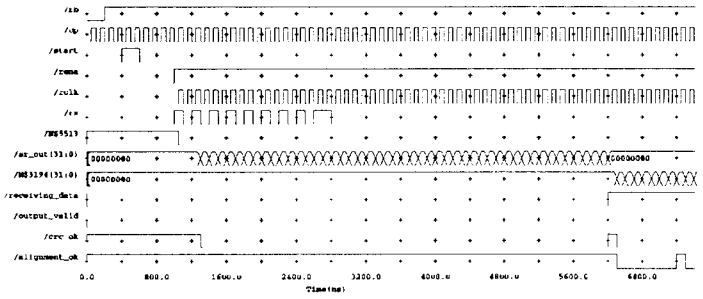


CRC Check (Design)



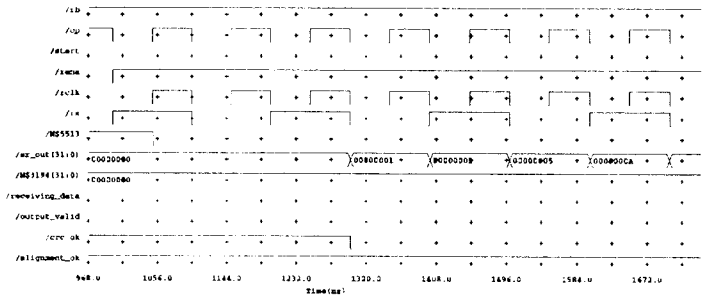
CRC Check:

- Beginn eines empfangenen Frames
- Frame besteht aus kurzer Präambel, 60 "0"-Bytes und der CRC



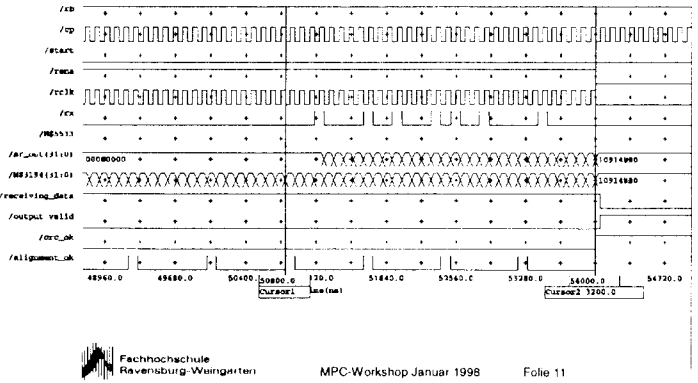
CRC Check (Zoom)

- Präambel wird durch das Schieberegister geschoben
- CRC_Calc ist noch inaktiv



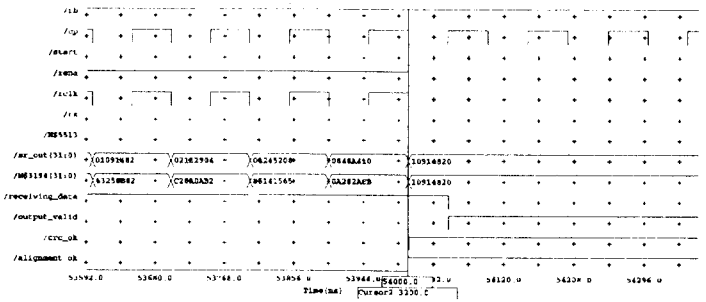
CRC Check:

- Ende des empfangenen Frames
- CRC = "10 91 48 20"



CRC Check (Zoom)

- CRC wird richtig erkannt
- 1 Takt nach Frameende wird "output_valid" gesetzt



Polynomdivision mit 4 Bit Generatorpolynom

Daten → 1010 000 : 1011 ← Generatorpolynom

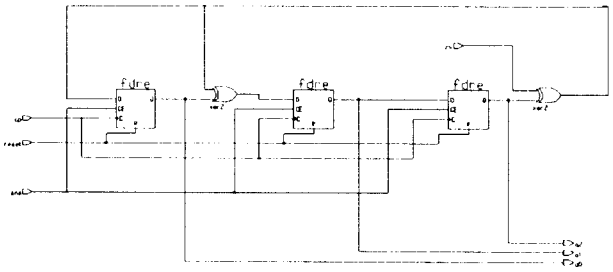
1011

1 000

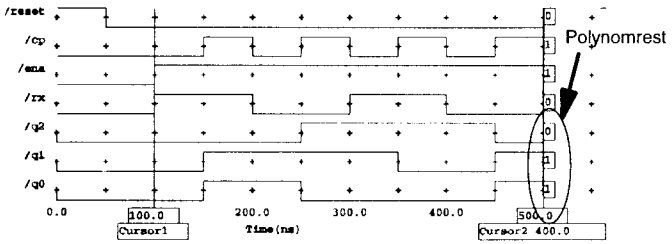
1 011

011 → Polynomrest

Beispielschaltung

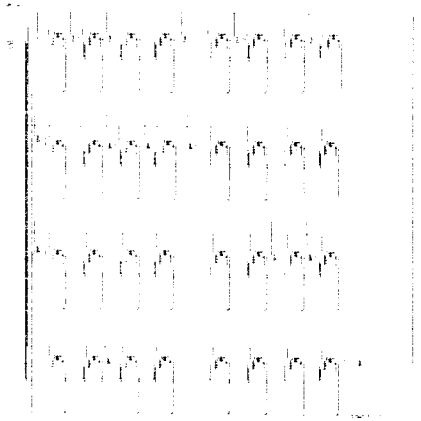


Simulation

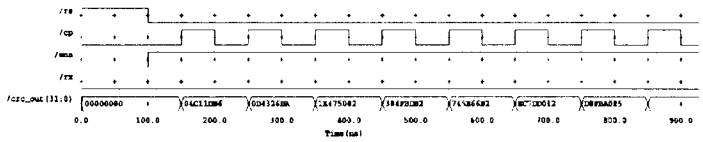


CRC Calc

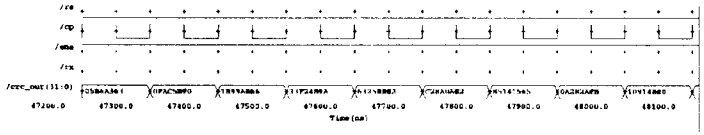
Generatorpolynom: (LSB) 1111 1011 0011 1110 1110 0010 0100 1001 (MSB)



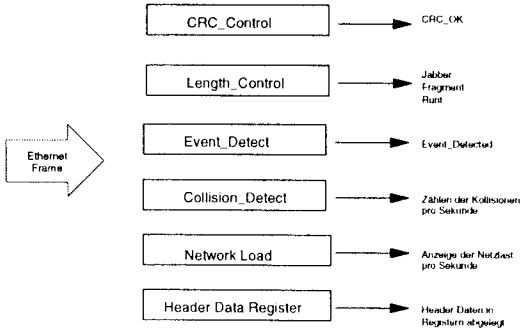
CRC Calc: CRC aus 60 "0" Bytes berechnen (Start)



CRC Calc: CRC aus 60 "0" Bytes berechnen (Ende)



Ausblick: Ethernet Control



Dokumentation auf dem Internet:

http://cae-ws10.fbe.fh-weingarten.de/jesinger

Firmenbesuche der MPC-Verbund Teilnehmer bei der CICC'97 im Mai 97, Santa Clara

K. Schmidt

1 Hewlett - Packard, Microwave Technology Center, Santa Rosa, Cal.

1400 Fountaingrove Parkway, Santa Rosa, CA 95403

Hier werden GaAs Integrated Circuits mit mehreren Prozessen hergestellt. Die Grenzfrequenz liegt zwischen 14 und 50 GHz. Zur Zeit werden 62 verschiedene Bausteine in einer Gesamtstückzahl von etwa 1/4 Million pro Jahr produziert. Zur 'foundry' gehört auch eine Entwicklungsabteilung für die Bausteine. Die gesamte Mitarbeiterzahl liegt bei ca 300 einschliesslich der ca 60 Ingenieure.
Der Besuch war eingeteilt in Fachvorträge und eine Besichtigung der GaAs-Linie (fab). Als Beispiel für einen hier hergestellten Baustein sei der Verstaerker 5026 darstellt (s. Anhang).
Die Vorträge stellten den Stand und die Arbeiten vor zum Gebiet der Microwave Communication, dem High Frequency Design mit den entsprechenden EDA- und CAE-Werkzeugen, der Electromagnetic and Circuit Analysis, der Microwave Simulation z.B. mit der die Schnelligkeit um Grössenordnungen verbessernden 'Envelope Simulation', IC-CAP, Statistics Package zur Prozessverbesserung - hier neu das Boundary Model der Probability Density Function.
Im 'wrap up' kamen zur Sprache 'guest scientist periods' und 'student internships'. Zu letzteren werden im folgenden die Anschriften angegeben, an die sich Studenten in Deutschland bzw. USA wenden können:

Ingeborg Refke
Germany Country Headquarters
Hewlett-Packard GmbH
Herrenberger Strasse 130
71034 Boeblingen
Tel 07031 14 1567

Yolanda Lemus-Page
Corporate College Recruiting
Hewlett-Packard Company
3000 Hanover St., MS: 20AC
Palo Alto, CA 94304
Tel 857-4655, Fax 857-3549

Dank gilt insbesondere Herrn R. Giacometti und Herrn Marc Petersen, ebenso allen, die uns fuehrten und die Themenkreise vortrugen.

2 Altera Corporation, San Jose Cal.

2610 Orchard Parkway, San Jose, CA 95134

Der Besuch war vom Kollegen G. Voland initiiert worden und man darf sagen, wir wurden hervorragend empfangen. Das 'university program' von Altera wurde uns in diversen Vorträgen vorgestellt. Da schon einige Kollegen partizipieren, wurden aus dem MPC-Verbund Arbeiten und Erfahrungen mit den Altera-PLDs einbezogen.
Einige Zahlen zu Altera. 900 Angstellte (300 in R&D) 15000 Kunden, wachsend, Umsatz 1996 $497 Millionen, Gewinn bei 10%. Foundries unter Vertrag, aber auch eigenes Investment z.B. 16% der neuen TSMC Fabrik in Washington-State, Fabrik in Asien.
Markt: 50% USA, 20% Japan, 20% Europa, 10% die restl. Regionen.
Verteilung des Umsatzes:

58%	in Communications
19%	in Computers
14%	Industrial (betont wurde z.B. auch im Audi)
4%	Consumer
4%	Military
1%	andere

Die Programmable Logic Bausteine von Altera gehören zu den High Density komplexen PLDs, CPLD genannt. Von der neuesten Entwicklung wurde der Baustein EPF10k100GC503-3 FLEX Device mit 10 000 000 Transistoren vorgestellt. Kostenpunkt $600. PGA Gehäuse mit 7-reihiger Anschlussanordnung. Die Schaltung ist 'unterlegt' durch ein SRAM zu ihrer Programmierung. Es gibt 24kBits embedded memory für den Benutzer, so dass auch die umfangsreichste State Machine erzeugt werden kann.
Der CMOS PLD Markt ist weltweit ein '2-Billion-Dollar Market' (Anm.: 2000 Millionen $ / Jahr).

Die Anteile sind:

30%	Xilinx
26%	Altera
13%	Vantis (formerly AMD)
11%	Lattice
8%	Actel
5%	Lucent (gehoert zu AT&T)
3%	Cypress
4%	andere

Die Technologie hat natürlich wesentlich zu dieser Entwicklung beigetragen. Hier eine Uebersicht:

Technology parameter \ year	'95	'96	'97	'98
channel length [μm]	0.6	0.5	0.35	0.25
voltage [V]	5.0	5.0	3.3	2.5

Wichtig ist natürlich auch die Entwicklungssoftware, man bewertet sie als 'easy to learn' (contrary to the competition's software, where one establishes a 'priesthood'). Der Student erledigt Übungen im 'lab' in einigen Minuten Rechenzeit (die Simulation von 100 000 'gates' benötigt ca 4 Std. Rechenzeit). Codiert wird in der Max-Plus-2 Altera programming language.
Den Custom Computing Machines war ein Vortrag gewidmet. Das Reconfigurable Computing nimmt zu. Im 30 000 Millionen $ Prozessor Markt gibt es für CPLDs gute Möglichkeiten.
Altera wird Lehrbücher unterstützen und möchte von Veröffentlichungen zu Anwendungen von Altera PLDs Kopien bekommen, dafür aber ist das Mitmachen im University Program kostenfrei. Man wende sich unter diesem Stichwort an die obige Adresse. Ein Antragsformblatt hat jeder Teilnehmer bei den umfangreichen Tischunterlagen mitbekommen.
Dank an das Altera Team, insbesondere Frau Anna Acevedo.

3 Mentor Graphics, San Jose Cal.

1001 Ridder Park Drive, San Jose, CA 95131
Tel 408-436-1500

Dieser Besuch hing am seidenen Faden, wie man so sagt, da der Kontakt über München kurz vor Abflug begonnen hatte. Als Kontaktadresse war das Hotel in SFO angegeben, aber während unserer dortigen Zeit nicht gleich genutzt. Schliesslich wurde die verlorene Fährte ins Westin-Hotel gefunden, wo gerade noch rechtzeitig ein Anruf von MG den richtigen Empfänger fand. Dennoch legte MG sich mächtig ins Zeug, den MPC-Verbund mit seinen Teilnehmern in seinem Zentrum

in San Jose zu empfangen. Ein Vortrag zu der MG Interconnect Synthesis wurde arrangiert und ihm folgte ein 'technical roundtable' mit Videoverbindung zu MG Fachleuten in Oregon.
Mit dem Wissen aus bisheriger Nutzung der Mentor Software und der Möglichkeit Ausbau bzw. Erneuerung im MPC-Verbund vorzunehmen, waren die Diskussionen zu der breiten Palette der Design-Software wertvoll. Ein Rundgang durch das Gebäude, geführt wurden wir von Herrn P. Bridenne, schloss diesen Besuch mit interessanten Eindrücken vom Silicon Valley ab.

A Datenblatt des HMMC-5026 von HP

2-26.5 GHz GaAs MMIC Traveling Wave Amplifier

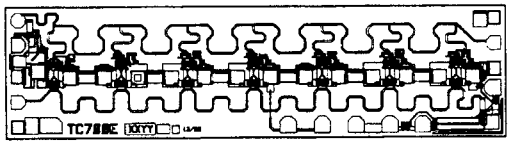
HMMC-5021(2 - 22 GHz)
HMMC-5022(2 - 22 GHz)
HMMC-5026(2 - 26.5 GHz)

Features

- Wide-Frequency Range: 2 - 26.5 GHz
- High Gain: 9.5 dB
- Gain Flatness: ± 0.75 dB
- Return Loss:
  Input: -14 dB
  Output:-13 dB
- Low-Frequency Operation Capability: < 2 GHz
- Gain Control: 35 dB Dynamic Range
- Moderate Power:
  20 GHz: P_{-1dB}:18 dBm
  P_{sat}: 20 dBm
  26.5 GHz: P_{-1dB}:15 dBm
  P_{sat}: 17 dBm

Description

The HMMC-5021/22/26 is a broadband GaAs MMIC Traveling Wave Amplifier designed for high gain and moderate output power over the full 2 to 26.5 GHz frequency range. Seven MESFET cascode stages provide a flat gain response, making the HMMC-5021/22/26 an ideal wideband gain block. Optical lithography is used to produce gate lengths of ≈ 0.4 µm.The HMMC-5021/22/26 incorporates advanced MBE technology, Ti-Pt-Au gate metallization, silicon nitride passivation, and polyimide for scratch protection.



Chip Size: 2980 × 770 µm (117.3 × 30.3 mils)
Chip Size Tolerance: ±10 µm (±0.4 mils)
Chip Thickness: 127 ± 15 µm (5.0 ± 0.6 mils)
Pad Dimensions: 75 × 75 µm (2.95 × 2.95 mils), or larger

Absolute Maximum Ratings†

Symbol	Parameters/Conditions	Min.	Max.	Units
V_{DD}	Positive Drain Voltage		8.0	volts
I_{DD}	Total Drain Current		250	mA
V_{G1}	First Gate Voltage	-5	0	volts
I_{G1}	First Gate Current	-9	+5	mA
V_{G2}††	Second Gate Voltage	-2.5	+3.5	volts
I_{G2}	Second Gate Current	-7		mA
P_{DC}	DC Power Dissipation		2.0	watts
P_{in}	CW Input Power		23	dBm
T_{ch}	Operating Channel Temp.		+150	°C
T_{case}	Operating Case Temp.	-55		°C
T_{stg}	Storage Temperature	-65	+165	°C
T_{max}	Max. Assembly Temp. (for 60 seconds maximum)		300	°C

†Operation in excess of any one of these conditions may result in permanent damage to this device. T_A = 25°C except for T_{ch}, T_{stg}, and T_{max}.
††Minimum voltage on V_{G2} must not violate the following: $V_{G2}(min) > V_{DD} - 9$ volts.

1-3

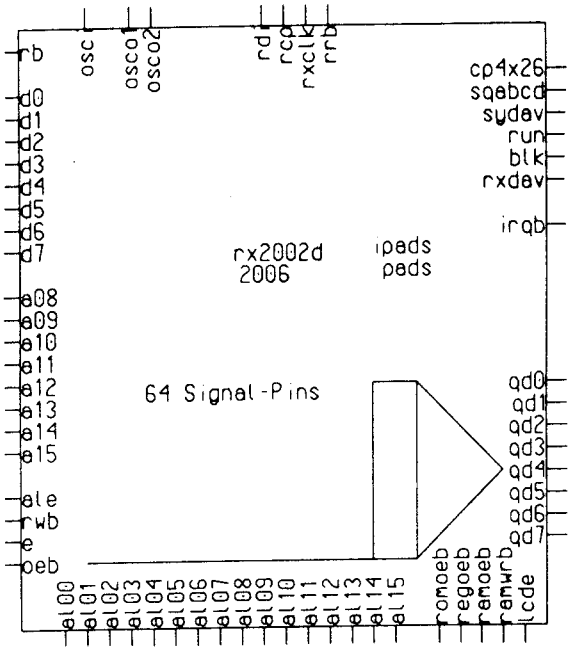
rx2002d - RDS-Empfänger

Funktionsblöcke entworfen
und simuliert im Fach
"CAE Elektronischer Schaltungen",

Entwurfs-Team:
5. Semester ELEKTRONIK

im WS96/97

S.Fischer,D.Frick,
A.Friedel,M.Fuchs,
R.Hecker,A.Heim,
J.Hofherr,H.Jäger,
B.Kapfelsberger,J.Kneissle,
M.Kübler,H.Lauter,
W.Lindner,R.Link,
W.Müller,T.Mühe,
C.Preußer,M.Sassano,
J.Schmollinger,M.Stein,
S.Vogler,



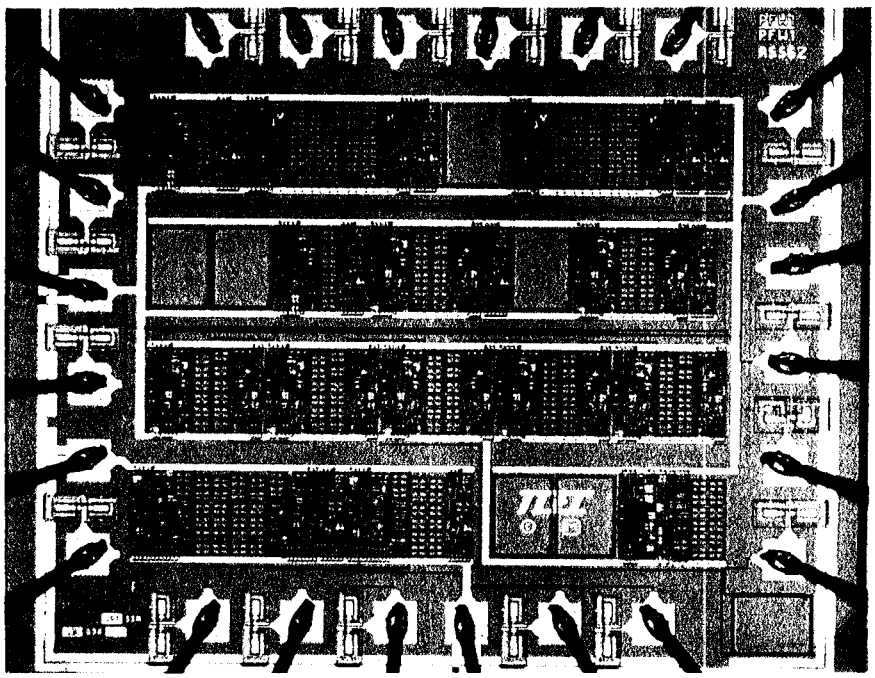
Der ASIC realisiert wichtige Funktionsblöcke eines RDS-Empfängers in hardware. RDS-Empfangsfunktionalität ist ohne Software-Overhead gegeben. Die Anforderungen an einen externen Mikroprozessor bezüglich Echtzeitverhalten und Speicher sind minimal.

rx2002d besteht aus den Baugruppen Synchronisation SYN, Sequenzerkennung SEQ, Empfänger RXI und dem Registerblock REG. Eine Prozessor-Schnittstelle PCI erlaubt die Ankopplung eines externen Mikroprozessors.
SYN stellt fest, ob ein Datensatz mit RDS-typischen Merkmalen (RDS-Block: 16-Bit-Datenwort, 10-Bit-Prüfwort+Offset) vorliegt. Block und Offset-Information werden an SEQ weitergegeben.
SEQ prüft, ob Blöcke im richtigen Zeitraster und in der richtigen Reihenfolge auflaufen. Die Überprüfung des Blocksignals erfolgt durch ein FIR-Filter, die Abfolge der Blocksequenz wird durch einen Zustandsautomaten überwacht.
RXI nimmt den Datenstrom entgegen und führt - angestoßen durch den Mikroprozessor - ggf. eine Fehlerkorrektur durch.
REG erlaubt es, interne Register des ASICs auszulesen.
Die Prozessor-Schnittstelle PCI dient zur Ankopplung eines externen Mikroprozessors. Sie erhält als Eingangsgröße den üblicherweise gemultiplexten Daten- und Addressbus eines uPs und generiert daraus Status- und Address-Signale für externe Komponenten: EPROM, RAM und LCD-Display. Ein Konfigurations-Register in PCI erlaubt das Umschalten oder Mischen von RAM und EPROM. Nach dem Reset startet das System aus einem EPROM-Bereich von 64kByte. Der EPROM-Bereich kann durch 64kByte statisches RAM "beschattet" werden, so daß nach dem Initialisieren des RAMs ein System mit 64kByte-RAM zur Verfügung steht.

Entwurf:	FH Ravensburg-Weingarten, CAE-Projekt 5.Semester Elektronik (WS96/97)
betreut durch:	Dipl.-Ing. (FH) F.Förster, Prof. Dr.-Ing. W.Ludescher
CAE-Software:	Mentor V8
Chipfertigung:	IMS Stuttgart, im Rahmen des MPC-Verbundes Baden-Württembergs
Komplexität:	ca. 55000 Transistoren
Technologie:	0.8um CMOS Gate-Forest

BiCMOS Allpass Delay-Line Continuous-Time Filter Testchip

Rudolf Koblitz[1] and Kurt Schmidt[2]



Scope: In video signal processing applications, reducing area usage and power consumption by different structures in delay filter circuits in comparison to those used (purely bipolar) to date of study.

Design: Two types of biquad delay filter circuits implemented on-chip.
1. Novel[3] transconductance structure BiCMOS with bipolar input stage and NMOS in triode region, tuning via R_{ON} by the gate voltage.
2. Transconductance structure with double unsymmetrical bipolar differential pair with additional diode, tuning via g_m by current source.

Nine different filter circuits (delays from 45 ns to 380 ns), a tuning circuit for the BiCMOS types, plus output buffers implemented on-chip.

CAE: Cadence Design Software (ELDO Simulator and Virtuoso Layout Editor)

Chipsize: 2 mm x 2.3 mm

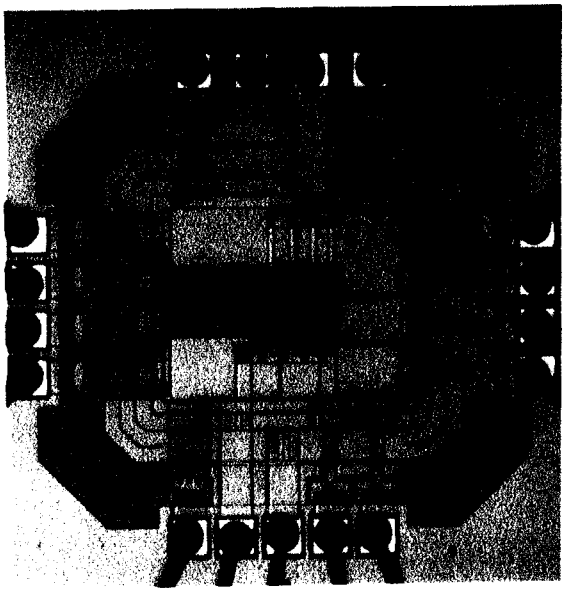
Technology: HF3CMOS, 1.2 μm BiCMOS, manufactured by SGS Thomson, in 1995

Testing: Functional in all parts, performance as expected.

Results: The novel analog circuits can be recommended for further investigations. They offer substantial reductions in area and power. The BiCMOS filter types seem to be most promising.

1 was with Thomson Consumer Electronic Components (TCEC) Grenoble, now Professor with FH Karlsruhe
2 during guest period 3-8/1994 with TCEC, Professor with FH Furtwangen
3 unpublished

tm_entprell



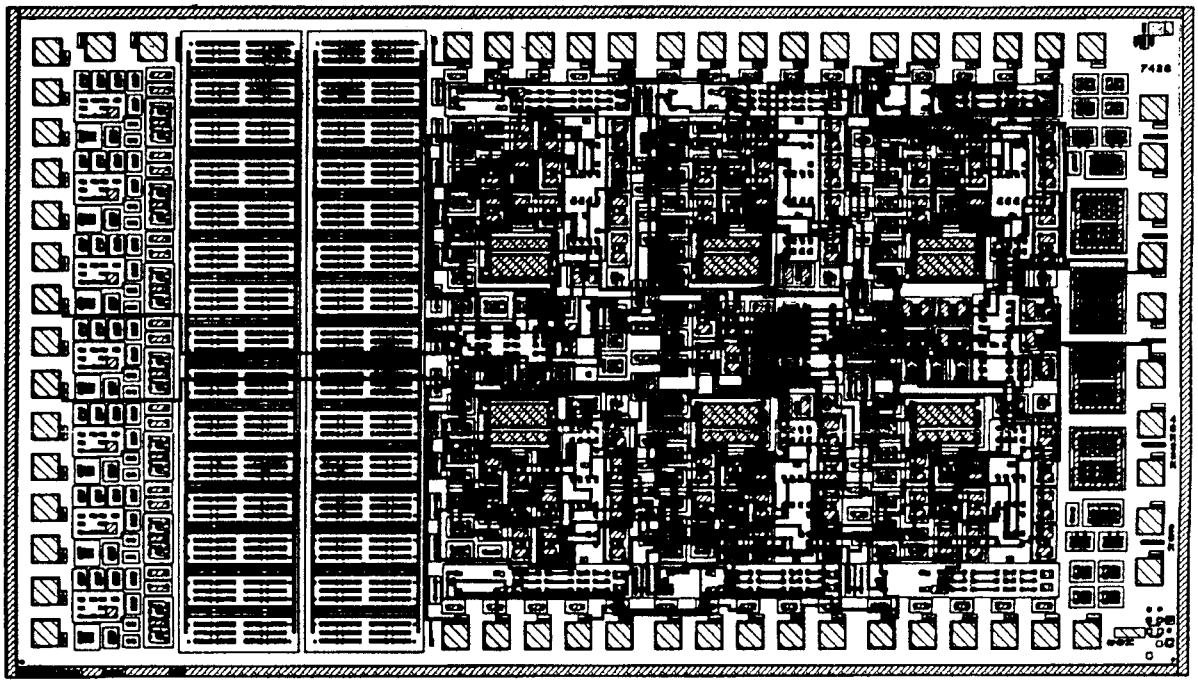
Entwurf:	Fachhochschule Furtwangen Bearbeiter : Dipl. -Ing. A. Gauckler Betreuer: Prof. Dr. W. Rülling
Layouterstellung:	Fachhochschule Furtwangen (Standardzellenentwurf)
Technologie:	ES2 0,7µm (ecpd07)
Chipfertigung:	EuroPractice Run 110
Herstelldatum:	I. Quartal 1997
Kostenträger:	MPC-Mittel FH-Verbund Baden-Württemberg
Chipdaten:	Chipgröße 1,4 mm x 1,3 mm Gehäuse: DIP 16
Funktion:	Dieser Chip ist als „Begleitchip" zur Lehrveranstaltung „ASIC-Entwurf" entstanden. Dabei geht es um den kompletten „Designflow" vom VHDL Entwurf zum getesteten Chip. Die Schaltung beinhaltet einen einfachen Zähler mit Tastenentprellung. Die Schaltung wird in VHDL beschrieben, simuliert und synthetisiert. Mit dem Synopsys Test Compiler werden Verilog Testpattern generiert und auf hohe Fehleraufdeckung getrimmt. Der synthetisierte Entwurf wird mittels EDIF Netzliste in die Cadence Entwurfsumgebung überführt, dort nach plazieren, verdrahten und verifizieren zur Fertigung freigegeben. Die generierten Verilog Testpattern dienen in Cadence zur Funktionskontrolle unter Berücksichtigung der Schalltungseinflüsse und auf dem Tester als Funktionspattern.
Testergebnisse.	Die gefertigten Chips bestehen sowohl den Funktionstest auf dem hp82000 Tester als auch den Test in einer Demoplatine.

Untersuchung und Weiterentwicklung von OP-Schaltungen für das Transistor-Array B500D

(momentan kein Bild verfügbar)

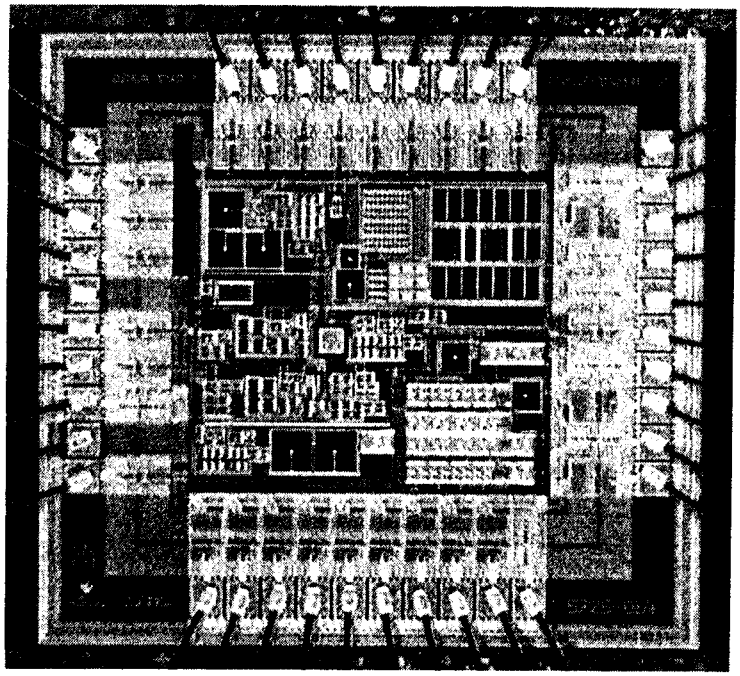
Entwurf:	Fachhochschule Heilbronn Bearbeitung: J. Storch Betreuer: Prof. Dr.-Ing. H. Clauss (2 unterschiedliche Bausteine)
Bausteinfertigung:	TEMIC Heilbronn, Grundlage Transistor-Array B500D
Herstelldatum:	IV/97
Kostenträger:	MPC-Gruppe
Gehäuse:	keramisch, DIL, 64 pol.
Testergebnisse:	Test in Vorbereitung
Funktion:	Untersuchung verschiedener Operationsverstärkerschaltungen (s. Seminararbeit von J. Storch, Oktober 1994)

Schwimmende Spannungsquelle



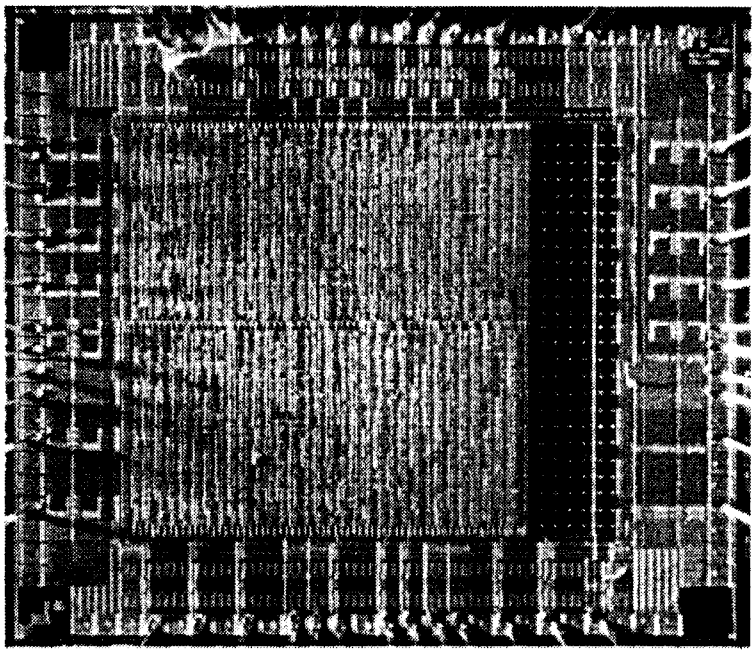
Entwurf:	Fachhochschule Heilbronn Bearbeitung: G. Plappert (Diplomarbeit) Betreuer: Prof. Dr.-Ing. H. Clauss
Bausteinfertigung:	TEMIC Heilbronn, Grundlage Transistor-Array B500D
Herstelldatum:	IV/97
Kostenträger:	MPC-Gruppe
Gehäuse:	keramisch, DIL, 64 pol.
Testergebnisse:	Test in Vorbereitung
Funktion:	Generierung einer „massefreien“ Spannungsquelle (s. MPC Workshop Feb. 97, S. 61ff)

Empfänger-ASIC für ein Datenträgersystem



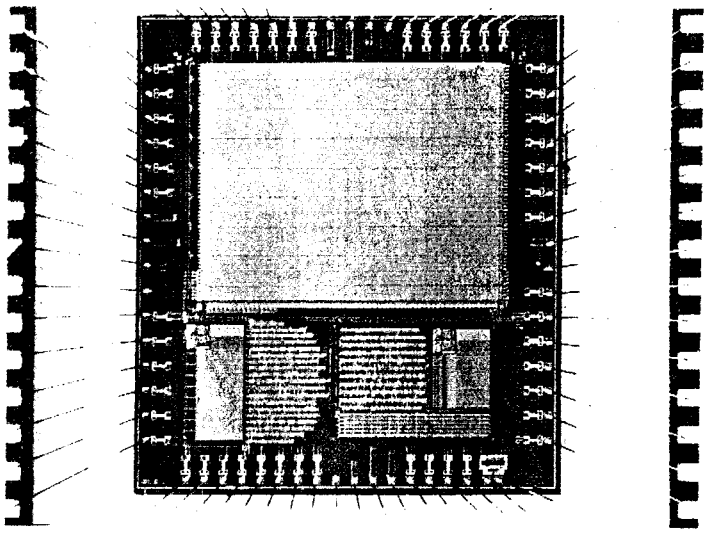
Entwurf:	Fachhochschule Ulm/Daimler-Benz Aerospace AG Bearbeiter: Karlheinz Hartner Betreuer: Prof. Dipl.-Phys. Gerhard Forster		
Layouterstellung:	Fachhochschule Ulm (Mixed Signal-Entwurf) Analog-Teil: Full Custom Design Digital-Teil: Standardzellen		
Technologie:	CAE 1,2 µm CMOS A/D, Fa. AMS		
Chipfertigung:	Fa. AMS, Österreich, über Europractice		
Herstelldatum:	II. Quartal 1997		
Kostenträger:	Daimler-Benz Aerospace AG		
Chipdaten:	Chipfläche:	2,0 x 1,8 mm^2	
	Gehäuse	LCC 44	
	Funktionsblöcke:	Analogteil:	Bandgap-Referenz, 2 Verstärker, 2 Komparatoren
		Digitalteil:	ca. 150 Gatter
Funktion:	Das ASIC ist als Teil eines Datenträgers („Tag") für ein Identifizierungssystem vorgesehen. Eine Beschreibung findet sich im Tagungsband zum MPC-Workshop Februar 1997		
Testergebnisse:	Das Empfänger-Chip wurde erfolgreich getestet und ist als Prototyp zur Systementwicklung einsetzbar.		

LICHT-SEND-BAUSTEIN



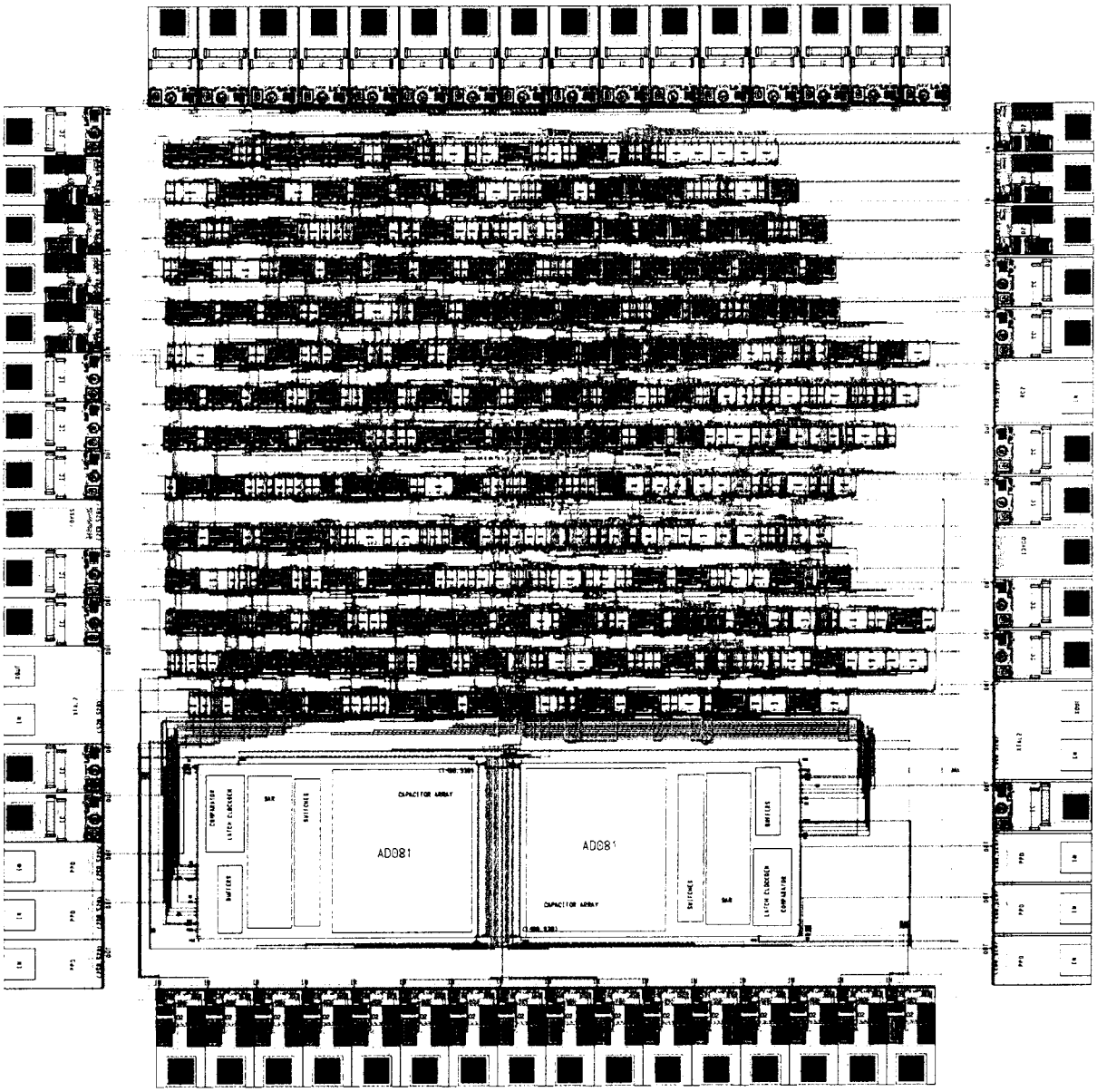
Entwurf:	Fachhochschule Ulm Bearbeiter: Prof. Dipl.-Ing. Arnold Führer
Technologie:	0,8 µm (gfn012) Gate Array Master
Chipfertigung:	IMS
Herstelldatum:	I. Quartal 1997
Kostenträger:	MPC-Mittel, FH-Verbund Baden-Württemberg
Chipdaten:	Chipgröße: 3,5 x 3 mm (Pads: 34) Gatter: 3514 (12258 Transistoren) Gehäuse: CLCC44
Funktion:	Der IC enthält die gesamte Steuerung des Senders eines Lichtszenenschalters. Die Funktionsweise des verwendeten Datenübertragungssystems für Elektroinstallationen im Wechselstromnetz ist im Workshopband Februar 1996 Karlsruhe beschrieben. Die Funktion der Schaltung wurde dahingehend erweitert, daß jetzt zusätzlich 10 Lichtszenen speicher- und abrufbar sind.
Testergebnisse:	Der IC wurde mit dem Textronix Digital-Testgerät LV500 erfolgreich getestet und arbeitet auf der eingesetzen Platine einwandfrei.

Thermologger V2



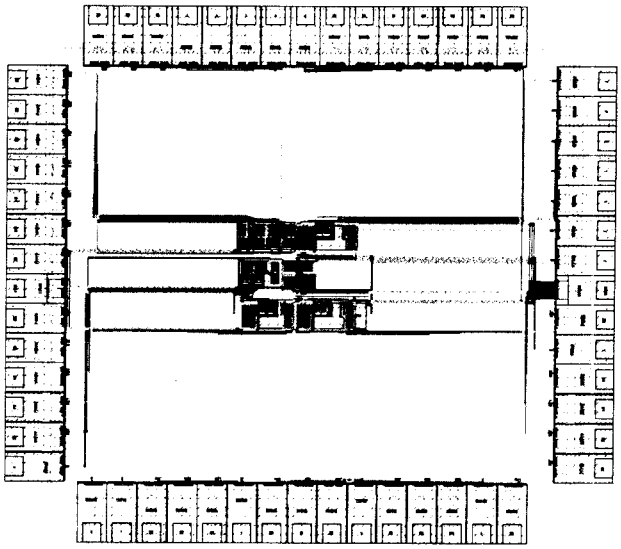
Entwurf:	Fachhochschule Offenburg Bearbeiter: Thomas Klumpp Betreuer: Prof. Dr.-Ing. Dirk Jansen
Layouterstellung:	Fachhochschule Offenburg (Standardzellenentwurf)
Technologie:	ES2 0,7µm (ecpd07)
Chipfertigung:	Europractice, Run 160
Herstelldatum:	August 1997
Kostenträger:	MPC-Mittel FH-Verbund Baden-Württemberg
Chipdaten:	Chipgröße: 6,5 x 5,8 mm^2 Gehäuse: JLCC 44 Komplexität: ca. 50000 Transistoren
Funktion:	Der Chip ist eine Weiterentwicklung des Thermologgers: Er enthält einen 8 KB RAM-Speicher, einen Mikroprozessorkern (FHOP) eine serielle Schnittstelle nach dem Telefonkartenstandard, eine Power-Down-Unit mit programmierbarem Timer sowie ein ROM für die Betriebssystemsoftware. Der Chip arbeitet mit einem Temperatursensor zusammen, soll in regelmässigen Abständen die Temperatur aufnehmen, komprimieren und im RAM ablegen. Die Daten können bei Bedarf über die serielle Schnittstelle ausgelesen und als Temperaturprofil mit Hilfe eines PC-Programms dargestellt werden. Durch das Redesign wurde ein Betriebssystem im ROM integriert sowie eine IIC-Schnittstelle zur Kommunikation mit der Thermozelle und eine 8 Bit breite parallele Schnittstelle ergänzt.
Testergebnisse:	Alle Funktionseinheiten arbeiten wie entworfen und sind voll funktionsfähig. Durch Downloadfähigkeit ist eine Erweiterung bzw. Anpassung des Betriebsystems jederzeit möglich.

ASIC – Solar



Entwurf:	FH- Aalen, EDA- Zentrum Bearbeiter: Andreas Herb Betreuer: Prof. Dr. B. Kohlhammer
Chipgröße:	5,7mm x 5,9mm
Transistoren:	4314
Chiphersteller:	Mietec-Alcatel, Belgien
Eigenschaft:	Chip zur Optimierung von Photovoltaik-Anlagen; mittels MPP-Tracking und Pulsweitenmodulation soll eine maximale Leistungsabgabe der Photovoltaikanlagen erreicht werden. Der Chip soll in jedes Modul eingebaut werden, dessen Leistung optimieren und per getrennter Steuerung eine bessere Anpassung an die Bedürfnisse des Verbrauchers ermöglichen.

Steuerbaustein zur Diversityumschaltung in einem Funkmikrofonempfänger



Entwurf:	Fachhochschule Aalen Bearbeiter: Diplomand A. Lang Betreuung: Prof. Dr. B. Kohlhammer, Dipl.-Ing(FH) G. Busch,
Layouterstellung: und Chipfertigung	Institut für Mikroelektronik (IMS), Stuttgart
Herstelldatum:	I. Quartal 1997
Kostenträger:	MPC-Mittel FH Verbund Baden-Württemberg
Technologie:	IMS 0,8µm Gate Forest

Chipdaten

Master:	GFN004
Max. nutzbare digitale Gatter:	2200
Analoge Tiles:	8
Gehäuse:	PLCC20
Analoge Signalpins :	2
Digitale Signalpins:	12

Funktion:

Der Schaltkreis DAISY ist ein Steuerbaustein der analoge Schaltungsteile (zwei A/D-Wandler) und digitale Schaltungsteile (Auswertung, Steuerung) enthält. Er steuert die Umschaltung zweier Antennen in einem Funkmikrofonempfänger.

Eigenschaften:

- 6 Bit A/D-Wandler nach dem $\Sigma\Delta$-Prinzip
- 16 KHz Wandelrate
- Eingangsspannungsbereich 0..5V
- Taktversorgung 1MHz

1. Vorbetrachtung

In Zusammenarbeit mit einer Firma sollte ein Integrierter Schaltkreis für ein Funkmikrofonsystem entworfen und gefertigt werden.
Ziel ist es den Schaltkreis in Serie zu fertigen.
Partner bei dieser Arbeit ist die Firma MAINTRONIC in Schweinfurt.

2. Aufgabenstellung

Die Aufgabe des Schaltkreises ist es die Umschaltung von zwei Antennen in einem Diversityempfänger steuern.
Für beide Antennen ist nur eine Empfangsschaltung vorhanden.
Von der Empfangsschaltung werden zwei analoge Spannungen ausgegeben. Beide haben einen Wert zwischen 0V und 3V. Eine der Spannungen ist proportional dem Rauschanteil des empfangenen Signals (U_NOISE), die Andere ist proportional der Feldstärke (U_FELDST). Von diesen Spannungen wird auf die Empfangsqualität geschlossen.
Anhand dieser beiden Spannungen nimmt der Chip die Antennenumschaltung vor.

3. Das Diversityprinzip

Fast alle Antennen besitzen eine Richtcharakteristik. D.h. sie empfangen ein Signal aus unterschiedlichen Richtungen unterschiedlich stark. Bei Sendesystemen in denen Sender und Empfänger einen festen Standort haben, werden die Sende- und Empfangsantenne so ausgerichtet, daß die Empfangsqualität optimal wird.
In Sendesystemen, bei denen sich Sender und bzw. oder Empfänger in Bewegung befinden werden zur Verbesserung der Übertragungsqualität oft Diversityempfänger verwendet. Dabei werden zwei oder mehr Antennen unterschiedlich ausgerichtet oder nur räumlich getrennt. Eine Auswerteschaltung bestimmt dann welche Antenne momentan die besten Empfangseigenschaften besitzt. Das Signal dieser Antenne wird dann weiterverarbeitet.

4. Umschaltkriterium

Da immer nur die Empfangsqualität einer Antenne bekannt ist, muß ein Umschaltprinzip gewählt werden, das eine häufige Umschaltung zwischen den Antennen bewirkt. So kann immer die momentan 'bessere' Antenne bestimmt werden.
Umschaltkriterium ist die 'Verschlechterung' des Empfangs. D.h. wenn die beiden Spannungen U_NOISE und U_FELDST auf einen schlechteren Empfang schließen lassen werden die Antennen umgeschaltet. Ist der Empfang nach dem Umschalten noch schlechter als vorher wird sofort wieder zurückgeschaltet.

5. Realisierung

Realisiert wurde der Schaltkreis mit einem Gate Forrest GFN004 des IMS Stuttgart. Auf diesem Master können sowohl analoge als auch digitale Schaltungsteile implementiert werden. Damit wurde es erst möglich eine eigentlich analoge Schaltung digital zu realisieren.
Der Analogteil beschränkt sich nun auf die Komparatoren in den zwei $\Sigma\Delta$-A/D-Wandlern. Der digitale Schaltungsteil beinhaltet die Steuerung des A/D-Wandlers und die Auswerteschaltung.
Die Auswerteschaltung wurde im Rahmen der Diplomarbeit entworfen. Die A/D-Wandler werden vom IMS mit unterschiedlichen Auflösungen und Taktraten zur Verfügung gestellt.

6. Funktion des integrierten Schaltkreises

Die beiden analogen Eingangsspannungen werden von den A/D-Wandlern digitalisiert. Die beiden Werte werden addiert und in einem Register zwischengespeichert. Nach jeder A/D-Wandlung wird der aktuelle Wert mit dem Vorherigen verglichen. Ist der aktuelle Wert kleiner bedeutet dies eine Verschlechterung des Empfangs. Eine Umschaltung der Antennen erfolgt.
Ist die Empfangsqualität minimal und der aktuelle Wert ist null, wird die Umschaltung der Antennen durch einen Oszillator erzwungen.

7. Vorteile gegenüber der bisherigen Schaltung

Die bisherige Schaltung ist diskret aufgebaut. Sie besteht aus Operationsverstärkern und Flipflops. Ausgewertet wird nur die Spannung U_NOISE, die proportional dem Rauschanteil ist. Die Auswerteschaltung besteht im wesentlichen aus drei Komparatoren, die als A/D-Wandler mit 2 Bit Auflösung betrachtet werden können. Daraus ergeben sich die Vorteile des Schaltkreises:

- höhere Auflösung
- Auswertung von zwei Spannungen
- Platz und Bauteilersparniss

8. Schlußbetrachtung

Der Chip befindet sich momentan noch in der Prototypenfertigung. Erfüllt er die in ihn gesetzten Erwartungen wird eine Serienfertigung ins Auge gefaßt.