Zur Generierung von Verhaltensmodellen für gemischt analog/digitale Schaltungen auf Basis der Theorie dynamischer Systeme

Vom Fachbereich Informatik der
Technischen Universität Darmstadt
genehmigte

DISSERTATION

zur Erlangung des Grades eines
Doktors der Ingenieurwissenschaften

von
Dipl.-Inform. Ralf Rosenberger
geb. in Frankfurt am Main

Referenten der Arbeit:
Prof. Dr. S. A. Huss
Prof. Dr. K. Antreich

Tag der Einreichung: 18. Mai 2001
Tag der mündlichen Prüfung: 18. Oktober 2001

D17

Darmstädter Dissertation 2001

Vorwort

Diese Arbeit ist während meiner Tätigkeit als externer wissenschaftlicher Mitarbeiter am Institut für integrierte Schaltungen und Systeme der Technischen Universität Darmstadt entstanden.

Mein herzlicher Dank gilt Prof. Dr.-Ing. S. Huss, der mir nach Beendigung meines Studiums die Möglichkeit bot, mich weiter im Themenfeld Modellierungs-Methodiken zu engagieren und mich während der Dauer der Arbeit stets hervorragend zu motivieren wußte und in zahlreichen Fachgesprächen unterstützte. Herrn Prof. Dr.-Ing. K. Antreich danke ich für die Übernahme des Koreferats und Herrn Prof. Dr. O. von Stryk für den Vorsitz bei der Prüfung.

Besonderer Dank gilt allen Mitarbeitern am Institut für integrierte Schaltungen und Systeme für die sehr gute Zusammenarbeit, konstruktive Beiträge und die technische Unterstützung "vor Ort".

Ich danke der Firma Additive GmbH für das Bereitstellen der für das Gelingen der Arbeit notwendigen Ressourcen: Die richtigen Tools (Origin, Mathematica, Scientific Word und ConceptDraw) und die technische Unterstützung beim Feinschliff.

Mein ganz besonderer Dank gilt meiner Familie für den jederzeit guten Zuspruch und meiner Frau Carina für Standby, viel Geduld und dafür, mir im Alltag stets den Rücken freigehalten zu haben.

Inhaltsverzeichnis

Benutzte Symbole

ω_i	Segment i aus einem Definitionsbereich
Ω	Menge aller Segmente
t	Zeit
$t^{(n)}$	Zeit zu Schritt Nummer n
$\Delta t^{(n)}$	Größe des Zeitschrittes Nummer n
ta	Zeitfunktion
u	Eingangsgröße
x	Zustandsgröße
y	Ausgangsgröße
$u^{(n)}$	Eingangsgröße im Zeitschritt n
$x^{(n)}$	Zustandsgröße im Zeitschritt n
$y^{(n)}$	Ausgangsgröße im Zeitschritt n
$\overrightarrow{u}$	Vektor der Eingangsgrößen
$\overrightarrow{x}$	Vektor der Zustandsgrößen
$\overrightarrow{y}$	Vektor der Ausgangsgrößen
u_i	Element i des Vektors der Eingangsgrößen
x_i	Element i des Vektors der Zustandsgrößen
y_i	Element i des Vektors der Ausgangsgrößen
$u_i^{(n)}$	Eingangsgröße i im Zeitschritt n
$x_i^{(n)}$	Zustandsgröße i im Zeitschritt n
$y_i^{(n)}$	Ausgangsgröße i im Zeitschritt n
$\Delta\overrightarrow{u}^{(n)}$	Veränderung der Eingangsgrößen im Zeitschritt n
$\Delta\overrightarrow{x}^{(n)}$	Veränderung der Zustandsgrößen im Zeitschritt n
$\Delta\overrightarrow{y}^{(n)}$	Veränderung der Ausgangsgrößen im Zeitschritt n
$\widehat{\overrightarrow{x}}$	approximierter Vektor der Eingangsgrößen
$\widehat{\overrightarrow{y}}$	approximierter Vektor der Ausgangsgrößen
f	mathematische Funktion
$\overrightarrow{f}$	Vektor mathematischer Funktionen
$u(t)$	zeitabhängiges Eingangssignal
l_i	Fehlernorm
d_i	Abstandsmaß
z	Differenzfunktion
λ	Ausgangsfunktion
δ	Übergangsfunktion für Ereignisse
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{D}_f$	Definitionsbereich

1 Einleitung

Bei Betrachtung des Marktes der elektronischen Schaltungen nehmen ASSP (Application Specific Signal Processor) und ASIC (Application Specific Integrated Circuit) das größte Segment ein: Ihr Anteil ist mit mehr als 30 Prozent größer als der der Speicher. Diese anwendungsspezifischen Schaltungen kommen speziell in den Bereichen Telekommunikation und Automotive zum Einsatz. Leistungsfähige Schaltungen erfordern hierbei die Integration von analogen und digitalen Komponenten auf einem Chip. Für den Hersteller solcher Schaltungen ist die Entwicklungszeit von zentraler Bedeutung. Werden die Entwicklungskosten für einen ASIC um 50 Prozent überschritten, so beträgt die Gewinneinbuße nach Steuern weniger als 5 Prozent. Wird hingegen die Entwicklungszeit (und damit die Auslieferung des Produktes) um 6 Monate überschritten, beträgt die Gewinneinbuße mehr als 30 Prozent, siehe Abbildung 1 [Weh92]. Vor diesem wirtschaftlichen Hintergrund wird die Bedeutung des rechnergestützten Entwurfs deutlich. Das Nadelöhr bei der Entwicklung einer gemischt analog/digitalen Schaltung ist der analoge Teil. Hier gilt eine klassische 10:90 Beziehung: 10 Prozent der Chipfläche erfordern 90 Prozent der Entwicklungszeit. Während der Entwurf des digitalen Teils weitgehend automatisiert ist, so nimmt der Entwickler und sein Wissen im analogen Designprozeß noch immer eine zentrale Rolle ein.

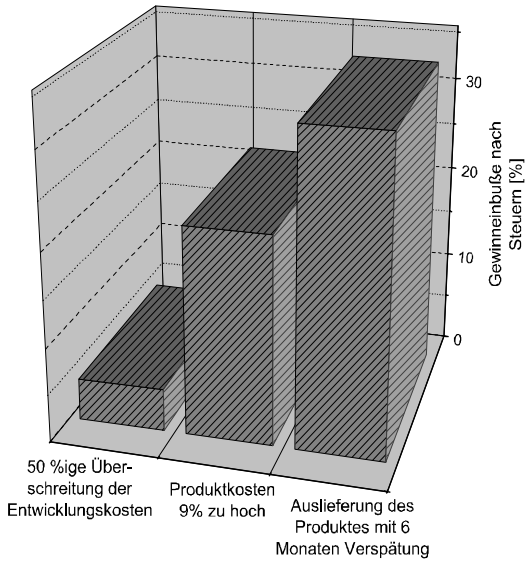

Abbildung 1: Bedeutung der Entwicklungszeit.

Der folgende Abschnitt gibt eine Motivation vor diesem Hintergrund, darauf folgt die Definition der Ziele der Arbeit und eine Übersicht über den Aufbau der Arbeit.

1.1 Motivation

Verhaltensmodelle spielen eine wichtige Rolle im Designprozeß. Sie ermöglichen es, eine neue Schaltung in allen erdenklichen Umgebungen (verschiedene Eingangssignalformen, dynamische Last am Ausgang) zu testen, denn die Simulationszeit ist gegenüber der Spice Netzliste i.a. um mindestens eine Größenordnung geringer. Ein Verhaltensmodell enthält keine Einzelheiten der Schaltungstopologie, sodaß der Hersteller das Modell an Kunden weitergeben kann, ohne Details der Implementierung zu verraten.

Ein kritischer Punkt beim Einsatz von Verhaltensmodellen ist die Genauigkeit. Sie korreliert in der Regel mit der Anzahl von Effekten, welche im Verhaltensmodell berücksichtigt sind. Typische Verhaltensmodelle nähern die Ausgangssignale in Abhängigkeit der Eingangssignale mittels mathematischer Funktionen an. Beispielsweise kann das Ausgangssignal eines Operationsverstärkers durch die Funktion

$$V_{out}(t) = \alpha V_{in}(t + T_D)$$

in Abhängigkeit des Eingangssignals beschrieben werden. In diesem idealen Modell ist α der Verstärkungsfaktor und T_D die Phasenverschiebung (Verzögerung). Diese Gleichung kann in einer einfachen DC-Analyse verwandt werden, für Simulationen mit hohen Anforderungen an die Genauigkeit ist sie jedoch kaum geeignet. Alle Effekte höherer Ordnung, die für die Simulation im Zeitbereich wichtig sind (wie etwa die Flankensteilheit, die Einschwingzeit bzw. das nichtlineare Verhalten des Operationsverstärkers), werden nicht berücksichtigt. Der Modellentwickler kann jetzt weitere Gleichungen für die charakteristischen Eigenschaften hinzufügen, jedoch wird das Modell immer ausschließlich die im Entwurf berücksichtigten Effekte aufweisen.

Diese Überlegungen führen zu dem Schluß, daß der Entwickler eines Verhaltensmodells nach wie vor über sehr detaillierte Kenntnisse der Schaltung und des Anwendungsbezugs verfügen muß. Modelle werden im allgemeinen für eine gleiche Designtechnologie entwickelt. Treten kleine Änderungen auf, so ist es häufig einfacher, das Modell komplett neu aufzubauen, als eine Anpassung eines bestehenden Modells an eine neue Aufgabe vorzunehmen. Die Wiederverwertbarkeit von Verhaltensmodellen ist dementsprechend gering. Am Designprozeß hat sich also nichts geändert: Obwohl mit VHDL-AMS eine effiziente standardisierte Umgebung zur Verfügung steht, können die Modelle nicht automatisch generiert werden, sondern sie basieren auf empirischem Wissen des Entwicklers. Es fehlt eine Methodik, die es einem Anwender ohne Kenntnis der Schaltungsdetails erlaubt - ausgehend von einer strukturellen Beschreibung - ein Verhaltensmodell zu erzeugen, welches die von ihm definierten Anforderungen an die Genauigkeit (maximaler Fehler) einhält und welches Effekte höherer Ordnung implizit enthält.

Das Makromodell für einen Operationsverstärker z.B. nach Boyle [BCP74] ist ein gutes Beispiel für das Fehlen von Effekten, die beim Erzeugen des Modells nicht explizit berücksichtigt werden. Abbildung 7 in Kapitel 2.5.4 zeigt das Modell, das strukturnahe (die Transistoren, die die Stromquelle nachbilden, sind durch ideale Stromquellen ersetzt)

und strukturfremde Elemente (Ein- und Ausgangswiderstand, Transientenverhalten etc. werden durch passive Elemente, Dioden und gesteuerte Quellen ersetzt) enthält. Abbildung 57 in Kapitel 5 zeigt das Simulationsergebnis, wenn das Modell mit einer großen Last beschaltet wird. Da die Lastabhängigkeit im Modell nicht berücksichtigt ist, weichen die Ergebnisse stark von den mit der strukturellen Beschreibung erzeugten Ergebnissen ab. Die Anpassung bzw. Erweiterung kann ausschließlich der Modell*entwickler*, aber nicht der Modell*anwender* vornehmen.

1.2 Ziele der Arbeit

Die Simulation von gemischt analog/digitalen Schaltkreisen erfordert exakte und effiziente Modelle. Diese Arbeit stellt eine neue Methodik zur Erstellung von sogenannten Funktionsblockmodellen vor. Bei der Entwicklung dieser Methodik standen die folgenden Ziele im Mittelpunkt:

- Bereitstellung von Verhaltensmodellen nichtlinearer dynamischer Schaltungen für Simulationen im Zeitbereich.
- Automatische Generierung der Verhaltensmodelle ohne tiefergehende Kenntnis der Schaltungsdetails.
- Hohe Wiederverwendbarkeit der Modelle und Abdeckung der möglichen Betriebsbereiche.
- Berücksichtigung charakteristischer Eigenschaften höherer Ordnung der Schaltung.
- Einfache Integration der Methodik in die Simulator Umgebung beliebiger Hardwarebeschreibungssprachen einschließlich VHDL-AMS.
- Effiziente Ausführung mit bestimmten Aussagen zum Fehler des Modells.
- Skalierbarkeit der Modelle zur Verwendung auf den Abstraktionsebenen "Verhalten" und "Funktional".

Abstrahierend von der konkreten Anwendung zur Modellierung analoger Schaltungen ist die Methodik universell verwendbar zur Modellierung technischer Systeme. Ihr Einsatz in allen Bereichen der Mechanik, wie zum Beispiel Hydraulik oder Thermodynamik, ist möglich. Der Grund hierfür ist, daß reale Systeme - egal aus welchem Bereich - den selben Systemgesetzen unterliegen und ähnliche Verhaltensmuster zeigen, obwohl sie physikalisch völlig unterschiedlich sind [Bos87].

Es ist geplant, durch Charakterisierung und Modellerzeugung die automatische und flexible Generierung von Modellen für Funktionsblöcke zu ermöglichen, die einen hohen Anwendungsbezug haben und keine Kenntnisse des Anwenders über die Schaltungsdetails erfordern. Das Modell basiert nicht auf einer Nachbildung der Eigenschaften des Funktionsblocks mittels R, C Gliedern und gesteuerten Quellen, wie häufig in der Literatur

vorgeschlagen: Es enthält aber alle relevanten charakteristischen Eigenschaften wie zum Beispiel die Settling Time des Operationsverstärkers. Das Verfahren zur Modellerstellung ist eine neue Methodik, die an der TU Darmstadt am Fachgebiet Integrierte Schaltungen und Systeme vorgeschlagen wurde. Dieses Verfahren kombiniert die Vorteile des empirischen Ansatzes mit den Vorteilen der LookUp Tabellen, wobei die Nachteile beider Verfahren wegfallen. Ebenso wie beim Tabellen-Ansatz stellen die Ergebnisse des Simulators die Datenbasis dar. Somit ist die Unabhängigkeit von einer konkreten Technologie gewährleistet, neue Tabellen können bei Bedarf schnell erzeugt werden. Für die Tabellen wird kein umfangreicher Speicherplatz benötigt, da die Methoden-Bibliothek aus den Tabellen Gleichungssysteme erzeugt, die für unterschiedliche Genauigkeitsbereiche gespeichert werden. Diese Gleichungssysteme reflektieren die relevanten physikalischen Eigenschaften des Funktionsblockes. Das Modell schließlich besteht ähnlich wie ein empirische Modell aus abschnittsweise definierten Modellgleichungen, die allerdings automatisch während der Modellerzeugung erzeugt werden. Im Gegensatz zu einem Makromodell, das die durch die Transistoren verursachten Nichtlinearitäten durch lineare Elemente (ideale und gesteuerte Quellen, passive Bauelemente) ersetzt, wird die innere Struktur der Schaltung im Modell nicht berücksichtigt. Stattdessen wird das Verhalten am Ausgang mittels mathematischer Beziehungen zum Eingang beschrieben, unter Beibehaltung der Erhaltungssätze für elektronische Schaltungen (Kirchhoff'sche Gesetze). Das Verhaltensmodell ist effizient berechenbar, da vorzugsweise Polynome eines geringen Grades als Modellgleichungen verwandt werden.

Neben rein zeitdiskreten und rein kontinuierlichen Modellen, wie sie typischerweise für digitale bzw. analoge Modelle verwendet werden, gibt es Modelle, die diskrete und kontinuierliche Elemente enthalten. Diese Modelle sind auf der Ebene der Theorie dynamischer Systeme definiert und stellen die mathematische Basis der neuen Methodik dar. Diese gemischt zeitkontinuierlich/wertdiskreten Modelle haben ihren Anwendungsbezug in der Simulation gemischt analog / digitaler Schaltungen:

1. Die neue Methodik bildet die in der Systemtheorie definierten Modelle auf moderne Simulatorkonzepte ab und stellt so eine praktikable Anwendung der in der Theorie wohlbekannten Konzepte dar.
2. VHDL-AMS hat die Fähigkeit, solche Modelle zu repräsentieren.
3. An die moderne Hardwarebeschreibungssprache VHDL-AMS werden große Erwartungen geknüpft, den Grad der Automatisierung im analogen Schaltungsentwurf bedeutend zu erhöhen.

Die Arbeit gliedert sich in 6 Kapitel. Im 2. Kapitel erfolgt die Einordnung in das Thema Modellierung und Simulation. Neben allgemeinen Grundlagen zur Simulation werden die Design- und Abstraktionsebenen im Schaltungsentwurf vorgestellt. Der aktuelle Stand der Technik wird präsentiert, wobei die vorgeschlagene Methodik mit etablierten Verfahren verglichen wird. Im 3. Kapitel erfolgt die Definition der mathematischen Basis der Methodik. Dabei wird auf Modelle aus der Systemtheorie zurückgegriffen. Die Elemente des dynamischen Systems werden eingehend vorgestellt und diskutiert. Kapitel 4 stellt

die Implementierung der Methodik vor: Alle Schritte von der Charakterisierung der zu modellierenden Schaltung bis zum fertigen Modell werden eingehend vorgestellt und die Genauigkeit der Modelle wird untersucht. Das neue Konzept wird durch unterschiedlich komplexe Beispiele in Kapitel 5 ausführlich demonstriert. Anhand eines linearen Filters wird die Methodik zunächst anschaulich dargestellt. Das nächste Schaltungsbeispiel, ein Operationsverstärker, zeigt, daß die Methodik auf nichtlineare Funktionsblöcke ohne weiteres direkt angewandt werden kann. Als umfangreicheres Beispiel dient ein A/D-Wandler, der zudem die Verwendung eines Modells für eine gemischt analog/digitale Schaltung in der VHDL-AMS Simulationsumgebung zeigt. Abgeschlossen wird die Arbeit durch eine Zusammenfassung und einen Ausblick auf offene Fragestellungen.

2 Simulation und Modellierung

Die zentrale Aufgabe im Schaltungsentwurf ist die Modellbildung und Simulation: Zuverlässige Verhaltensmodelle erlauben eine Optimierung der Systemfunktion mit möglichst wenigen Prototypen. Die Analyse einer elektronischen Schaltung erfolgt durch Simulationen auf dem Computer. Basierend auf einer Spice-Netzliste, die die Topologie der zu analysierenden Schaltung enthält, werden die Ströme und Spannungen innerhalb der Schaltung numerisch berechnet und visualisiert. Diese numerische Berechnung ist in der Regel mit hohem Rechenaufwand verbunden. Mit geeigneten Modellen für die Schaltung wird der Aufwand zur Berechnung reduziert, das Verhalten der Schaltung läßt sich trotzdem beurteilen. Die Modelle stehen auf unterschiedlichen Detaillierungsstufen bzw. Abstraktionsebenen. Dieses Kapitel stellt die Grundlagen der Simulation und Modellierung dar. Ausgehend von der Beschreibung der Schaltung als Differential-Algebraisches Gleichungssystem werden die Grundlagen der Simulation im DC- und Zeitbereich vorgestellt. Es folgt eine Übersicht über die Abstraktionsebenen in der digitalen und analogen Welt. Anschließend erfolgt ein Abriß über die in der Literatur bekannten Modellierungsmethoden, die den derzeitigen Stand der Technik repräsentieren. Der Fokus dabei liegt auf den Verhaltensmodellen für gemischt analog/digitale Schaltungen.

2.1 Differential-Algebraische Gleichungen

Die Kenngrößen der Strukturelemente werden durch Bauelementgleichungen beschrieben. Für jedes Strukturelement muß eine solche Gleichung vorliegen. Beispielsweise wird ein Widerstand durch die Gleichung $i_R = \frac{u_1(t)-u_2(t)}{R}$ und ein Kondensator durch $i_C = \frac{d}{dt}(C * u_1(t) - u_2(t))$ beschrieben. $u_1(t)$ bzw. $u_2(t)$ sind die an den Knoten des Bauteiles anliegenden Spannungen. Zusätzlich zu den Bauelementgleichungen wird die Schaltungstopologie benötigt. Das Aufstellen der Gleichungen erfolgt gemäß den Kirchhoff'schen Gesetzen durch das Maschenstrom- bzw. Knotenpotentialverfahren. Ergebnis ist ein gewöhnliches Differentialgleichungssystem, das in seiner expliziten Form als Differential-Algebraische Gleichung (DAE, Differential-Algebraic Equation) lautet:

$$\dot{\vec{x}} = f(t, \vec{x}, \vec{u}) \tag{2.1a}$$

$$g(t, \overrightarrow{x}, \overrightarrow{y}, \overrightarrow{u}) = 0 \tag{2.1b}$$

Gleichung (2.1 b) ist ein statisches, im allgemeinen nichtlineares, Gleichungssystem für den Ausgangsvektor $\overrightarrow{y}(t)$.

Hierin sind:

$\overrightarrow{u}$:	Eingangsgröße	$\overrightarrow{y}$:	Ausgangsgröße
$\overrightarrow{x}$:	Zustandsvariable	$\dot{\vec{x}}$:	$\frac{d\overrightarrow{x}}{dt}$

Ein lineares Netzwerk ist ein Sonderfall eines allgemeinen Netzwerkes. Die Gleichungssysteme (2.1) lassen sich durch Linearisierung eines allgemeinen Netzwerk in einem Arbeitspunkt in ein lineares Differentialgleichungssystem überführen, die Anfangsbedingungen werden geeignet angenommen. Das Ergebnis läßt sich mit expliziter Zustandsraumdarstellung als Differential-Algebraische Gleichung in Matrizenform wie folgt darstellen [Cel91]:

$$\dot{\overrightarrow{x}} = A * \overrightarrow{x} + B * \overrightarrow{u} \tag{2.2a}$$
$$\overrightarrow{y} = C * \overrightarrow{x} + D * \overrightarrow{u} \tag{2.2b}$$

2.2 Simulation im Zeitbereich

Bei der Schaltungssimulation im Zeitbereich ist die Antwort $\overrightarrow{y}(t)$ der Schaltung auf ein vorgegebenes Eingangssignal $\overrightarrow{u}(t)$ als eine Funktion der Zeit t gesucht. Mathematisch betrachtet, erfordert das Berechnen der Antwort im Zeitbereich die - numerische - Lösung der Gleichungssysteme (2.1). Diese Lösung ist nicht trivial, da viele herkömmliche Verfahren die Lösung algebraischer Gleichungen mit algebraischen Nebenbedingungen nicht unterstützen [Kas00].

Ausgangspunkt sind die linearisierten Netzwerkgleichungen nach (2.2) der zu simulierenden Schaltung. Es ergeben sich symmetrische Matrizen A und B nach der Gleichung $A\overrightarrow{x} + B\dot{\overrightarrow{x}} = \overrightarrow{b}$, deren Elemente nichtlinear oder zeitabhängig sein können. Durch Invertieren ergibt sich die Standardform für Systeme gewöhnlicher Differentialgleichungen:

$$\dot{\overrightarrow{x}} = -B^{-1}A\overrightarrow{x} + B^{-1}\overrightarrow{b(t)} = Tx + g(t) \tag{2.3}$$

Da im allgemeinen B nicht geschlossen invertierbar ist, müssen die Matrizen blockweise zerlegt werden, wobei die Knoten so numeriert werden, daß die Potentiale der Knoten im Vektor vorne stehen, die mit Kapazitäten verbunden sind. Durch Auflösen und Einsetzen kann das resultierende System auf die Standardform gemäß (2.3) reduziert werden. Das Gleichungssystem besteht aus Differentialgleichungen für die Energiespeicher und aus algebraische Gleichungen für die übrigen Elemente. Beim Auflösen für die Transientensimulation stimmt die Anzahl der Unbekannten mit der Anzahl der Energiespeicher im Netzwerk überein. Die zu bestimmenden Größen sind die Zustandsvariablen, die eindeutig den Systemzustand und den Anfangs-Operationspunkt festlegen. Der große Nachteil ist, daß die Notwendigkeit besteht, die Inverse zu bilden (welche mit mathematischen Aufwand verbunden ist bzw. gar nicht berechenbar ist). Deshalb werden numerische Iterationsverfahren auf das Gleichungssystem angewandt. Jedes Iterationsverfahren benötigt einen Startwert, welcher als der Operationspunkt der Schaltung im DC-Bereich bekannt ist. Mit diesem Startwert wird das Gleichungssystem dann iterativ gelöst.

2.2.1 Bestimmung des DC-Operationspunktes

Gesucht ist der Arbeitspunkt $[\overrightarrow{x}_0, \overrightarrow{y}_0]^{\mathrm{T}}$ der im allgemeinen nichtlinearen Schaltung im eingeschwungenen Schaltung mit $t \to \infty$ für eine konstante Eingangsgröße $\overrightarrow{u}_0$. Für eine lineare Schaltung ergibt sich, ausgehend von (2.2), das lineare statische Gleichungssystem

$$\overrightarrow{f}'(\overrightarrow{x}, \overrightarrow{y}) = \begin{pmatrix} \overrightarrow{f}(\overrightarrow{x}) \\ \overrightarrow{g}(\overrightarrow{x}, \overrightarrow{y}) \end{pmatrix}, \tag{2.4}$$

welches z. B. mit dem Newton-Raphson Verfahren gelöst werden kann. Unter Einsetzen einer um f(x) linearisierten Form ergibt sich mit der - dünn besetzten - Jakobi Matrix $J(\overrightarrow{x}) = \frac{\mathrm{d}\overrightarrow{f}(\overrightarrow{x})}{\mathrm{d}\overrightarrow{x}}$ folgende Iterationsvorschrift:

$$\overrightarrow{x}^{(\mu+1)} = \overrightarrow{x}^{(\mu)} - J^{-1}(\overrightarrow{x}^{(\mu)}) * \overrightarrow{f}(\overrightarrow{x}^{(\mu)}) \tag{2.5}$$

In der Praxis wird der Operationspunkt durch den Simulator mittels der die DC Analyse bestimmt.

2.2.2 Bestimmung des Großsignalverhaltens

Das Großsignalverhalten in Abhängigkeit von der Zeit wird mittels numerischer Integrationsverfahren bestimmt. Die Trapezregel ist ein Einzelschrittverfahren zur Integration, in welchem eine Gewichtung für die Ableitung in zwei Punkten verwandt wird, um eine gute Vorhersage zu erzeugen. Es ist definiert als

$$\widehat{\overrightarrow{x}}(\nu+1) = \widehat{\overrightarrow{x}}(\nu) + \frac{\Delta t}{2}(\widehat{\overrightarrow{xs}}(\nu+1) + \widehat{\overrightarrow{xs}}(\nu)) \tag{2.6}$$

mit $\Delta t = t(\nu+1) - t(\nu)$ und $\widehat{\overrightarrow{xs}}(\nu+1) = \widehat{\overrightarrow{f}}(\widehat{\overrightarrow{x}}, \Delta t)$.

Dieses in den Gleichungen (2.5) und (2.6) dargestellte Iterationsverfahren ist die Basis für die von Netzwerksimulatoren wie Spice und Eldo durchgeführten Simulationen. Anstelle von Systemen der Form (2.3), welche für Schaltungen von praxisrelevanter Größe schnell nicht mehr handhabbar werden (in Bezug auf den Rechenaufwand), werden Repräsentationen des Systems auf einer höheren Abstraktionsebene eingesetzt. In diesen Repräsentationen ist die Anzahl der Knoten reduziert und damit auch die Größe des resultierenden Gleichungssystems und der für die Simulation benötigte Rechenaufwand. Der folgende Abschnitt stellt die Abstraktionsebenen für solche Modelle vor, während im darauf folgenden Abschnitt Verfahren zur Erstellung von Modellen beschrieben werden, die den heutigen Stand der Technik markieren.

2.3 Abstraktionsebenen

Die Darstellung einer Schaltung kann unterschiedlich detailliert erfolgen. Der Grad, wie detailliert die Schaltung betrachtet wird, wird mit Abstraktionsgrad bzw. Abstraktionsebene bezeichnet. Ausgehend von der funktionalen Ebene, in der die Funktion einer Schaltung - ohne Rücksicht auf elektrische Details und physikalische Signale - spezifiziert wird, bis hinunter auf die Komponentenebene, in der die Basiselemente Transistoren oder passive Bauteile verwandt werden, sind verschieden detaillierte strukturelle Ansichten der Schaltung möglich. Der Übergang von einer höheren auf eine niedrigere Ebene wird als Implementierung, der Übergang von einer niederen Abstraktionsebene zu einer höheren wird als Abstraktion bezeichnet.

Neben den verschiedenen Abstraktionsebenen kann eine Schaltung auch aus unterschiedlichen Sichten betrachtet werden: Neben der oben definierten strukturellen Ansicht gibt es die physikalische Ansicht (Geometrie der Schaltung) und die funktionelle Sicht (Verhalten der Schaltung). Der Übergang von einer Ansicht in eine andere wird als Synthese bezeichnet. Für Digitalschaltungen hat sich das Y-Diagramm gemäß [GDWL92] etabliert. Die Spezifikation erfolgt dabei in der bevorzugten Ansicht des Schaltungsentwicklers, die anderen Repräsentationen werden daraus automatisch generiert, so kann beispielsweise ausgehend von einer strukturellen Beschreibung das Layout der digitalen Schaltung automatisch generiert werden.

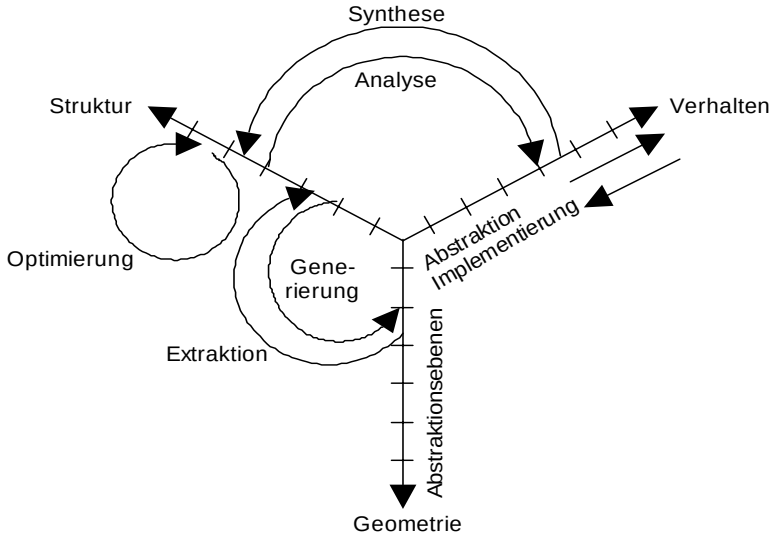

Abbildung 2: Übergange zwischen den Designräumen im Y-Diagramm.

Verhalten Die Beschreibung des Verhaltens eines Objektes hat es zum Ziel, das Ein- / Ausgangsverhalten im Zeitbereich wiederzugeben. Im Mittelpunkt des Interesses steht die Frage, was ein Design-Objekt tut, nicht wie es aufgebaut ist. Die Beschreibung erfolgt oftmals mittels Algorithmen und mathematischen Funktionen.

Struktur Die Beschreibung auf der Strukturebene erfolgt durch Spezifikation eines Objektes mittels einfacher Komponenten, welche miteinander verschaltet sind. Die zur Verfügung stehenden Komponenten hängen vom Grad der Abstraktion ab, siehe Abbildung 3 für die digitale bzw. Abbildung 4 für die analoge Welt.

Geometrie Hier stehen geometrische Objekte zur Verfügung, welche zweidimensional definiert sind. Eine Zuordnung dieser Elemente zur Funktion des Bauteils ist nicht gegeben.

2.3.1 Digitale Schaltungen

In einem lange andauernden Entwicklungsprozeß wurde eine anerkannte und effiziente Ordnung der Abstraktionsebenen für die digitale Designtechnik geschaffen [BGHW96]. Die Abstraktionsebenen enthalten in der Sicht des Hardwareentwicklers die am besten geeigneten Objekte für den Schaltungsentwurf. Abbildung 3 stellt die auf [Arm89] zurückgehenden Abstraktionsebenen für digitale Schaltungen dar.

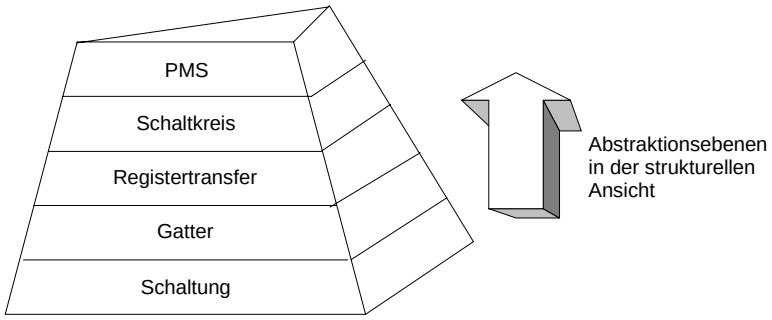

Abbildung 3: Abstraktionsebenen für digitale Schaltungen.

Schaltung Die Schaltung besteht aus verschalteten aktiven und passiven elektrischen Bauelementen. Das Verhalten der Schaltung im Zeitbereich kann durch ein System von nichtlinearen Differentialgleichungen beschrieben werden.

Gatter Auf der Gatter-Ebene sind Schaltnetze und Schaltwerke die verschalteten Elemente. Diese Ebene wird in der Literatur auch als die Logikebene bezeichnet. Die

Boolesche Schaltalgebra wird durch Gatter repräsentiert, dazu kommen einfache Speicherelemente wie Flipflops. Die Spezifikation erfolgt durch Boolesche Gleichungen und Zustandsübergangstabellen. Da die Boolesche Schaltalgebra die in der Praxis wichtigen Verzögerungszeiten der Gatter nicht direkt berücksichtigt, müssen diese durch zusätzliche Modelle in die Simulation einfließen.

Registertransfer Auf der Registertransfer-Ebene stehen komplexe Funktionsblöcke zur Verfügung. Dazu zählen Register, Zähler, Multiplexer und Speichermodule. Die Spezifikation erfolgt ähnlich wie auf der Gatter-Ebene durch erweiterte Boolesche Gleichungen, Wahrheits- oder Zustandsübergangstabellen, zusätzlich stehen Mikrooperationen zur Verfügung.

Schaltkreis Die Schaltkreis-Ebene (oder auch algorithmische Ebene) ist oberhalb der Registertransfer-Ebene angesiedelt. Auf dieser Ebene stehen Mikroprozessoren, Kanalwerke, Speicherbänke und Bussysteme als Strukturelemente zur Verfügung. Die Spezifikation erfolgt als nebenläufig notierte Beschreibung der Ein- / Ausgangsfunktionalität, daraus ergibt sich auch die Bezeichnung algorithmische Ebene. Der Software-Entwickler wird auf dieser Ebene das Verhalten mittels Programmen unter Verwendung der Befehlssätze der Elemente festlegen; die Programmierung auf dieser Ebene wird mit dem Begriff Mikroprogrammierung bezeichnet.

PMS (Processor, Memory, Switch) Die oberste Ebene weist Prozessoren, Speicherbänke und Bussysteme auf. Sie erlaubt den Entwurf von Rechnerarchitekturen und Komponenten und stellt gleichzeitig die Verbindung zu funktionalen Ebenen dar, die keinen Bezug zur Hardware haben. Die Kommunikation der Elemente untereinander und die Befehlssätze müssen genau festgelegt sein.

2.3.2 Analoge Schaltungen

Während die Entwurfsdomänen in der digitalen Welt weitgehend etabliert sind, fehlt eine allgemein anerkannte Hierarchie von Abstraktionsebenen und Transformationen zwischen den Ebenen in der analogen Welt. Abbildung 4 stellt eine mögliche Abstraktionsebenen-Hierarchie dar. Diese Hierarchie ist von [MAP97] abgeleitet; sie bezieht sich auf die strukturelle Sicht einer Design-Einheit. Die Design-Einheit wird mittels "besteht aus" Relationen konstruiert und beschrieben.

Komponente Auf der Komponentenebene sind Transistoren und passive Bauelemente als grundlegende Elemente vorhanden.

Schaltung Eine Stufe höher, auf der Schaltungsebene, sind Operationsverstärker oder Komparatoren die Elemente, aus denen eine Schaltung zusammengesetzt wird.

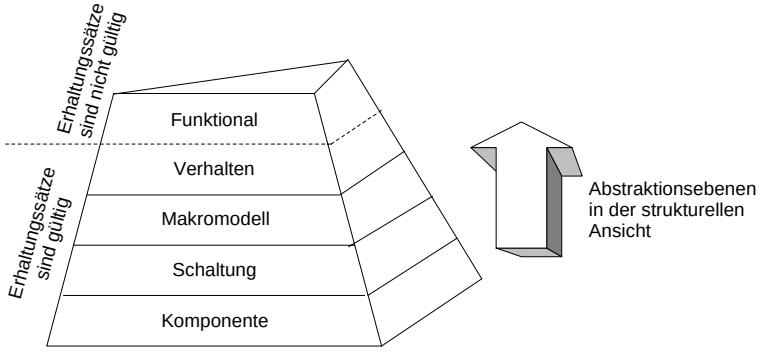

Abbildung 4: Abstraktionsebenen im zeitkontinuierlichen Bereich.

Makromodell Die Makromodellebene entspricht in etwa der Registertransferebene in der digitalen Welt: die Funktion eines Basis-Elementes wird ersetzt durch eine abstraktere Repräsentation einschließlich idealer Bauteile, wie zum Beispiel gesteuerte nichtlineare Strom- und Spannungsquellen.

Verhalten Demgegenüber bestehen Modelle auf der Verhaltensebene aus einer mathematischen Beziehung zwischen den Ein- und Ausgangssignalen, d.h. das Ausgangssignal wird von einer mathematischen Funktionsgleichung beschrieben. Die Ausgänge des Funktionsblockmodells tragen physikalische Signale, die den Erhaltungssätzen unterliegen. In elektronischen Schaltungen sind dies die Kirchhoff'schen Gesetze.

Funktional Die Erhaltungssätze sind schließlich auf der Funktionsebene nicht mehr von Interesse. Auf dieser Ebene stehen komplexe Grundelemente zur Verfügung, wie Wandler, oder sogar Modems. Es gibt keine Signale mehr, die die eine direkte physikalische Interpretation aufweisen.

Schaltungsentwurf Mit Hilfe der in Abbildung 4 dargestellten Abstraktionen kann das Vorgehen beim Entwurf einer analogen Schaltung aufgezeigt werden. Man unterscheidet allgemein zwischen dem Top-Down-Entwurf und dem Bottom-Up Entwurf. Beim Top-Down-Entwurf erfolgt - ausgehend von einer hohen Abstraktionsebene - eine schrittweise Verfeinerung. Demgegenüber werden die Komponenten der Schaltung beim Bottom-Up Entwurf auf der untersten Ebene (Spice Netzliste) entworfen und dann zu immer komplexer werdenden Funktionsblöcken verschaltet, wobei - je nach Abstraktionsebene - die entsprechenden Modelle verwendet werden. Beim Betrachten einer Schaltung, welche einen Operationsverstärker als Bauteil enthält, wird zunächst die Spice Netzliste des Operationsverstärkers für die Simulation auf der Komponenten Ebene benötigt. Dabei treten in der Praxis folgende Probleme auf:

- Die Simulationszeit steigt mit der Anzahl der Bauelemente stark an: für n Knoten werden $n-1$ Gleichungen aufgestellt; daraus ergibt sich eine Matrix der Größe $(n-1)^2$.
- Aus der hohen Anzahl von Nichtlinearitäten der Differentialgleichungen resultieren Konvergenzprobleme: Die Simulation konvergiert deshalb oftmals nicht und muß abgebrochen werden.
- Nicht alle Anfangsbedingungen liefern befriedigende Ergebnisse.

Um diese Probleme behandeln zu können, hat man die Entwurfsmethodik angepaßt. Nachdem ein Funktionsblock auf der Komponenten- oder Schaltungsebene entworfen wurde und die Funktion gemäß Spezifikation validiert wurde, wird ein Verhaltensmodell automatisch generiert. Das Verhaltensmodell wird für die Simulation der gesamten Schaltung auf der Verhaltens Ebene benötigt: Es ist ein mathematisches Modell, dessen Simulationszeit signifikant reduziert ist. Es wird anstelle der Netzliste in die Schaltung eingesetzt. Die Funktion der Schaltung wird nun mit diesem Verhaltensmodell simuliert und überprüft. Die Verhaltensebene ist die höchste Abstraktionsebene, auf der die Erhaltungssätze (Kirchhoff'sche Gesetze) Bedeutung haben. Der Bottom-Up Entwurf ist die Regel beim Entwurf von Analogschaltungen.

2.4 Hardwarebeschreibungssprachen

Die wichtigsten Aufgaben einer modernen HDL (Hardware Description Language, Hardwarebeschreibungssprache) sind:

- Modellierung
- Validierung
- Synthese.

Die Modellierung erfolgt mittels den Elementen der Sprache. Die Sprache muß dem Designer erlauben, seine Ideen und Algorithmen möglichst direkt in einer Computerverständlichen Form zu formulieren, d.h. sie muß widerspiegeln, wie der Designer denkt. Da die Modelle die Schnittstelle zwischen dem Designer (und damit dem Hersteller der Schaltung) und dem Anwender darstellen, sollten auch die Interessen des Anwender berücksichtigt werden: Einfache Verwendung des Modelles, ein breiter Anwendungsbereich und eine möglichst gute Lesbarkeit des Modell-Codes. Geeignete Konzepte wie parallele Abläufe oder die Synchronisation müssen berücksichtigt werden können.

Die Validierung der Schaltung gliedert sich in zwei Schritte. Zunächst wird die Schaltung mit bestimmten Betriebsbedingungen simuliert, anschließend erfolgt die Überprüfung, ob die Ergebnisse korrekt sind. Die Semantik der Sprache muß hierzu sehr genau definiert sein, speziell nebenläufige Abschnitte müssen wohldefiniert und wiederholbar gleich

ablaufen. Hier ist auch die Konsistenz zwischen verschiedenen Abstraktionsebenen wichtig, sodaß eine Simulation auf unterschiedlichen Ebenen (z.B. Komponenten und Verhalten) möglich ist.

Unter Synthese wird der Übergang von einer höheren auf eine niedrigere Abstraktionsebene bei gleichzeitigem Wechsel der Domäne (Verhalten zu Struktur) bezeichnet, siehe Abbildung 2. Dabei muß die Korrektheit der Ergebnisse garantiert sein: Es darf keine Mehrdeutigkeiten geben. Auf der oberen Ebene ist deshalb genau festgelegt, welche Elemente welche Entsprechungen auf dem unteren Level haben, z.B. beim Übergang von der Verhaltens- auf die Strukturebene.

Die HDL stellt ein großes Hilfsmittel im automatisierten Entwurfsprozeß dar. Neben dem Generieren einer formalen Spezifikation, welche direkt als Eingabe für den Simulator dient, ermöglicht sie auch die genaue Dokumentation der Schaltung. Durch die Unterstützung der Entwurfsdomänen Struktur und Verhalten stellt die HDL ein ideales Kommunikationsmedium zwischen den Hardwareentwicklern dar. Darüberhinaus erlaubt sie das effiziente Computer-verständliche Formulieren der Ideen des Entwicklers, wobei in der Regel Spracheinschränkungen notwendig sind, denn nicht jede algorithmische Beschreibung läßt sich auch effizient in Hardware umsetzen. Als Endergebnis wird die Geometrie der Schaltung erzeugt, so wie sie schließlich für die Produktion der Schaltung benötigt wird.

2.4.1 VHDL

VHDL steht für VHSIC Hardware Description Language, wobei VHSIC Very High Speed Integrated Circuits bedeutet. Das VHSIC Projekt wurde im Jahre 1980 vom US Department of Defense gestartet. Es markiert den Versuch, eine Standardisierung der Hardwarebeschreibungssprachen durchzusetzen. Bis dato gab es neben den Booleschen Gleichungen und Spice Netzlisten nur Ansätze zur Modellierung auf der Register Transfer Ebene (DDL) und Standard-Programmiersprachen wie Basic, Pascal oder Lisp. Nach der Festlegung der Anforderungen an VHDL erging im Jahre 1983 der Auftrag zur Implementierung an Texas Instruments, IBM und INTERMETRICS. Nach Abschluß der Sprachspezifikation wurde 1987 ein erster Simulator freigegeben: Im selben Jahr vergab das IEEE den Auftrag zur Implementierung einer eigenen VHDL Version, welche 1988 als IEEE 1076 zum Industriestandard wurde. War Anfang der 90er Jahre der Simulator Verilog das meistbenutzte System, so lief VHDL ihm Mitte der 90er Jahre deutlich den Rang ab.

Als Beispiel für die Modellierungsmöglichkeiten von VHDL soll ein lineares rückgekoppeltes Schieberegister betrachtet werden, welches aus einem Anfangswert ungleich Null eine zyklische Sequenz über alle Bitvektoren mit drei Bits ungleich Null erzeugt. Abbildung 5 links zeigt das aus drei Flipflops bestehende Blockschaltbild.

In VHDL stellt sich die Entity LFSR (Linear Feedback Shift-Register) gemäß Abbildung 5 rechts als "Black Box" mit einem Eingang und drei Ausgängen dar:

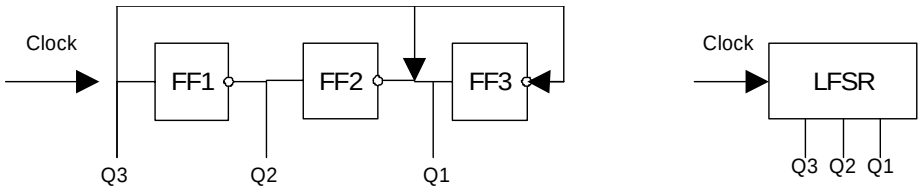

Abbildung 5: Lineares rückgekoppeltes Schieberegister: Block- und Black-Box Schaltbild.

```
entity LFSR is
port
   clock:  in bit;
   Q1, Q2, Q3 : out bit;
end LFSR;
```

Für diese Entity können nun in VHDL verschiedene Architekturen festgelegt werden, wobei jeder Entwickler die aus seiner Sicht am besten geeignete Architektur wählen kann, ohne daß Einschränkungen / Veränderungen der Funktion auftreten.

1. Verhalten Die Spezifikation des Verhaltens erfolgt analog zum Algorithmus, der quasizufällige Bitfolgen erzeugt, indem ein primitives Polynom mit Koeffizienten modulo 2 in einer Endlosschleife berechnet wird:

```
architecture Verhalten of LFSR is
begin
process
   variable zustand:  bit_vector( 3 downto 0) := '0111';
begin
   -- auf steigende clock warten
   wait until clock = '1';

   -- shiften
   zustand := zustand(2 downto 0) & '0';
   if zustand(3)='1'
      then zustand := zustand xor ''1011'';
   end if;

   Q3 <= zustand(2) after 5 ns;
   Q2 <= zustand(1) after 5 ns;
   Q1 <= zustand(0) after 5 ns;
end process;
end Verhalten;
```

2. Datenfluß Soll sich die Spezifikation am Datenfluß im Block orientieren, so wird mittels dreier Signale eine Rückkopplung aufgebaut:

```
architecture Datenfluss of LFSR is
   signal FF1,FF2,FF3 : bit := '1';
begin
   --Warten auf clock
   b1 : block(clock = '1' and not clock'stable)
   begin
      FF3 <= FF2 after 5 ns;
      FF2 <= FF1 xor FF3 after 5 ns;
      FF1 <= FF3 after 5 ns;
   end block;

    --externe Beschaltung
   Q3 <= FF3;
   Q2 <= FF2;
   Q1 <= FF1;
end Datenfluss;
```

3. Struktur Die Architektur Struktur ergibt sich direkt aus dem Schaltbild nach Abbildung 5. Diese Strukturbeschreibung kann direkt umgesetzt werden in VHDL:

```
architecture Struktur of LFSR is
   signal xor_out: bit;
   signal SR1, SR2, SR3: bit := '1';

   component FF
      port (clock, data: in bit;
      Q: out bit);
   end component;

   component XORgate
      port (a,b : in bit;
      x: out bit);
   end component;

begin
   FF1: FF port map (clock, SR3, SR1);
   FF2: FF port map (clock, xor_out, SR2);
   FF3: FF port map (clock, SR2, SR3);
   xor1: XORgate port map (SR1, SR3, xor_out);
   Q3 <= SR3; Q2 <= SR2; Q1 <= SR1;
end Struktur;
```

4. Zusammenhang der Modelle Alle drei Modelle leisten dasselbe und können beliebig miteinander vertauscht werden, d.h. eine Testbench, die einen Stimulus für das LFSR enthält, kann jedes der drei Modelle aufrufen, ohne daß sich das Verhalten verändert. Der folgende VHDL Code zeigt, wie die Datenfluß-orientierte Beschreibung in eine Testschaltung eingesetzt und mit einem Signal stimuliert wird:

```
-- Testbench für das LFSR
entity LFSRstim is
end LFSRstim;

architecture test of LFSRstim is
   component LFSR
      port(clock: in bit;
         Q1,Q2,Q3: out bit);
   end component;
   signal clock: bit := '0';
   signal val: bit_vector(3 downto 1);
begin
-- Instantiierung des LFSR
   L1: LFSR port map(clock,val(1), val(2), val(3));
-- Verhalten von clock:   Taktgenerator
   process
   begin
      for I in 1 to 20 loop
         wait for 1 us;
         clock <= not clock;
      end loop;
      wait;
   end process;
end test;

-- Datenfluß-Konfiguration für das LFSR
configuration DATAconf of LFSRstim is
-- Architektur von LFSRstim:
for test
   -- Datenfluss-Architektur
   for L1: LFSR use entity
            work.LFSR(Datenfluss);
   end for;
end for;
end DATAconf;
```

2.4.2 VHDL-AMS

VHDL-AMS (IEEE Standard 1076.1-1999) ist eine Weiterentwicklung von VHDL nach IEEE Standard 1076. Diese neue Simulationssprache beinhaltet das für die Entwicklung digitaler Schaltungen zum Standard gewordene VHDL und erweitert dessen Konzepte um Beschreibungsmöglichkeiten für zeit- und wertkontinuierliche Entwurfsobjekte. Mit diesen Erweiterungen ist eine Modellierungssprache entstanden, die sowohl zur Beschreibung von digitalen, als auch von analogen und gemischt analog/digitalen Schaltungen auf unterschiedlichen Abstraktionsebenen (siehe Abbildungen 4 und 3) genutzt werden kann. Die wichtigsten Erweiterungen für analoge Objekte sind gemäß [IEEE01]:

Quantities Quantities dienen der Deklaration und Verwendung von wert- und zeitkontinuierlichen Variablen in den umgesetzten Algorithmen. Sie können als Free, Source oder Branch Quantities (definiert durch Across und Through Quantities für Spannungen und Ströme) verwandt werden.

Terminals Terminals sind Knoten, an denen Flußgrößen (Through Quantities) wirken. Für Terminals gelten die Erhaltungssätze, d. h. zwischen zwei Terminals kann stets eine Across Quantity (Potentialdifferenz) definiert werden.

Natures VHDL-AMS soll die interdisziplinäre Verwendung der Simulatoren ermöglichen. Um von den ausschließlich im elektronischen Schaltungsentwurf verwandten Strömen und Spannungen als physikalischen Größen zu abstrahieren, wurde keine feste Zuordnung von physikalischen Größen auf die Potential- und Flußgrößen vorgenommen. Statt dessen wird für jedes Anwendungsgebiet eine sogenannte Nature definiert. In dieser Definition erfolgt die Zuordnung von Potential- und Flußgrößen auf die entsprechenden Quantities und Terminals. In einem Modell können beliebig viele unterschiedliche Natures verwendet werden, um sog. Multi-Nature Systeme zu modellieren. Die für die jeweilige Nature gültigen Erhaltungsregeln (z. B. die Kirchhoff'schen Sätze für elektronische Schaltungen) "kennt" der Simulator, das heißt der Entwickler des Modells braucht sie nicht explizit zu spezifizieren. So ist die Modellierungssprache VHDL-AMS als domänenübergreifender Standard verwendbar.

Im Vergleich zu Modellbeschreibungen als Spice Netzlisten erlaubt VHDL-AMS neben der reinen Strukturbeschreibung auf den Abstaktionsebenen Schaltung und Makromodell strukturfremde Modellnotationen auf den zusätzlichen Ebenen Verhalten und Funktional, siehe Abbildung 4. Die in der Theorie definierten Abstraktionsebenen werden somit in einer konsistenten Modellierungs- und Simulationsumgebung unterstützt. Darüberhinaus wird eine hierarchische Struktur der Modelle ebenso unterstützt wie eine gemeinsame Notation gemischt analog/digitalen Verhaltens. Die Umsetzung konzeptioneller Modelle in die entsprechenden, ausführbaren Modelle wird erst durch derartig mächtige Repräsentationsmöglichkeiten realisierbar. Manche Simulationssysteme, wie z. B. ADVance MS der Fa. Mentor Graphics, erlauben sogar die direkte Einbindung von Teilmodellen, die im Spice Netzlistenformat beschrieben sind. Dadurch kann der Übergang zu detaillierteren Modellen im Zuge des Entwurfsablaufs nahtlos gestaltet werden. Abbildung 6 verdeutlicht die Rolle von Sprachen zur Modellrepräsentation im Entwicklungsablauf gemischt analog/digitaler Schaltungen [HKR99].

Beispiel für ein VHDL-AMS Modell Als Beispiel für die Modellierung mit VHDL-AMS wird ein "Verhaltensmodell" für einen AD-Wandler betrachtet. Das Modell ist ein typisches Beispiel für VHDL-AMS Verhaltensmodelle, es enthält eine mathematische Beschreibung seiner Funktion. Der Bezug zu seiner elektronischen Umgebung wird über die in der Port Definition festgelegten Anschlüsse hergestellt. Das Modell ordnet sich in den Abstraktionsebenen im zeitkontinuierlichen Bereich nach Abbildung 4 auf der Grenze

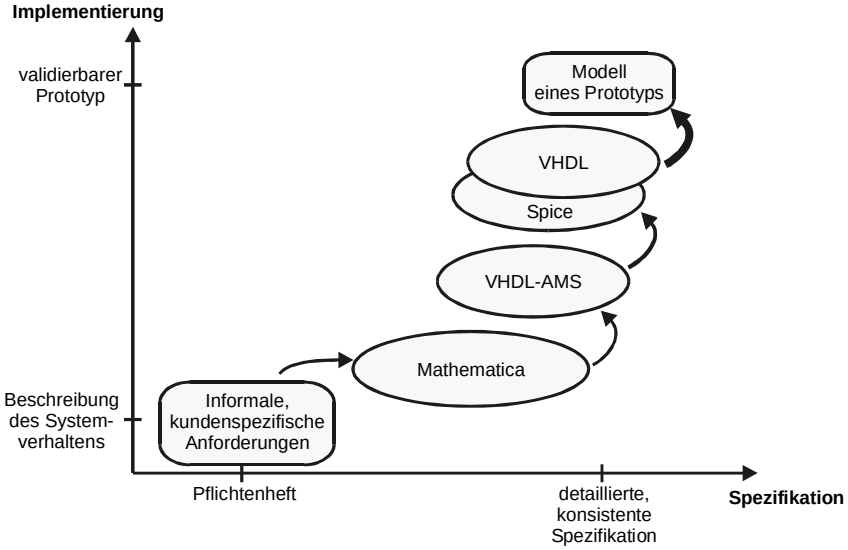

Abbildung 6: Modellierungssprachen im Entwicklungsablauf.

zwischen den Ebenen Verhalten und Funktional ein. Damit die Erhaltungsätze im Modell gültig sind, muß das Modell um die entsprechenden through und accross Quantities erweitert werden, siehe Seite 2.4.2.

Dieses Verhaltensmodell wird im Abschnitt Beispiele zum Vergleich mit dem mittels der in dieser Arbeit entwickelten Methodik erstellten verwandt. Der AD-Wandler ist ein Beispiel für eine gemischt analog/digitale Schaltung: Das Eingangssignal ist analog, mittels Operationsverstärkern und Komparatoren erfolgt das Umsetzen in eine digitalen Bitvektor am Ausgang. Das Programmlisting in VHDL-AMS:

```
library disciplines;
use disciplines.electromagnetic_system.all;
entity my_generic_ADC is
generic (
   MinVoltage : emf := -0.03;
   MaxVoltage : emf := 0.188;
   Res : integer := 12);
port(terminal U_in : Electrical;
   signal Clock, Reset : in std_logic;
   signal Data : out std_logic_vector((Res-1) downto 0));
end my_generic_ADC;

architecture BD of my_generic_ADC is
   quantity U_In_V across U_In;
begin
   P1 : process (Clock, reset)
      variable i : integer;
      variable r, temp : real;
```

```
   begin -- process P1
      if reset = '1' then
         Data <= (others => '0');
      elsif Clock'event and Clock = '1' then
         -- normalisation: input mapping to [0;1]
         r := (U_in_V - MinVoltage ) / (MaxVoltage - MinVoltage);
         -- A/D conversion:
         for i in Data'high downto Data'low loop
            temp := (2.0**(i-Data'length));
            if r > temp then
               r := r - temp;
               Data(i) <= '1';
            else
               Data(i) <= '0';
            end if;
         end loop;
      end if;
   end process P1;
end BD;
```

Zunächst wird die Nature "electromagnetic" festgelegt, damit der Simulator die Zuordnung von Potential- und Flußgrößen auf die entsprechenden Quantities und Terminals kennt. Die Kirchhoff'schen Gesetze kommen so zur Anwendung. Analog zur Modellierung in VHDL werden die Ausgänge als Signal definiert, hier ein Bitvektor. Der Eingang wird als Terminal definiert, somit läßt sich der Potentialunterschied zur Erdung (oder einem beliebigen anderen Terminal) im Modell als Variable verwenden. Dies erfolgt in der Quantity `U_In_V,` die als Across Quantity über den Eingang definiert ist. Der AD-Wandler ist getaktet und kann zurückgesetzt werden, dazu stehen die Signale `clock` und `reset` nach außen zur Verfügung. Die Funktion des AD-Wandlers ist durch mathematische Funktionen definiert. Zunächst erfolgt das Normieren des Eingangssignals auf den Bereich 0...1, sodann werden in einer For-Schleife die Bits des Ausgangs gesetzt, durch Vergleich des normierten Eingangs mit der entsprechenden Zweierpotenz.

Der große Vorteil dieses Modells ist die gleichzeitige Verwendung der Standard-Logik für die Ausgangsbits, die die Schnittstelle zur digitalen Welt darstellen, und des analogen Eingangs in Form einer Across Quantity. Das Verhalten ist als mathematischer Algorithmus definiert, der eher der funktionalen Ebene als der Verhaltensebene entspricht.

2.5 Stand der Technik

Dieser Abschnitt stellt den momentanen Stand der Technik vor. Nach einer kurzen allgemeinen Betrachtung zur Modellierung folgt die Einteilung und Bewertung aktueller Modellierungsansätze im Kontext Transientensimulation gemischt analog/digitaler Schaltungen.

Die Untersuchung eines realen Systems ist mit sehr aufwendigen Methoden der physikalischen Meßtechnik verbunden, in vielen Fällen auch gänzlich unmöglich. Dazu erfordert es die Existenz von zumindest einem Prototypen. Um Einblicke in das Verhalten des realen

Systems zu gewinnen, existieren rechnergestützte Verfahren zur Simulation eines solchen Systems. Um das System simulieren zu können, wird ein Modell benötigt, welches eine vereinfachte Darstellung des Verhaltens einer Komponente darstellt. Die Simulation des Systems mit dem Modell kann ausschließlich Aussagen über die Eigenschaften liefern, die bei der Modellbildung berücksichtigt worden sind. Der Entwickler benötigt Verhaltensmodelle auf verschiedenen Abstraktionsebenen für die einzelnen Phasen des Entwurfs, siehe Abschnitt 2.3 Abstraktionsebenen.

2.5.1 Anforderungen an ein Modell

Das Modell soll die Komplexität eines Systems oder einer Systemkomponente reduzieren. Es muß auf einem Computer berechenbar sein und auf eine Standard-Hardwarebeschreibungssprache abbildbar sein. Die Berechnung soll möglichst effizient in Bezug auf Speicherplatz und Simulationszeit sein. Dazu sollen die folgenden Anforderungen möglichst gut erfüllt werden:

- **Effizienz**: Geringer Bedarf an Systemressourcen (Festplatten- und Hauptspeicherplatz) und kurze Rechenzeit bei der Ausführung des Modells.
- **Wiederverwendbarkeit**: Ein für eine bestimmte Anwendung erzeugtes Modell soll auch in einem anderen Kontext korrekt arbeiten.
- **Modellgenerierung**: Das Generieren des Modells soll geringe Systemressourcen erfordern und automatisch möglich sein.
- **Benötigtes Wissen**: Sowohl Entwickler als auch Anwender des Modells müssen nicht notwendigerweise detailliertes Wissen über den Funktionsblock besitzen, um das Modell zu erzeugen bzw. zu verwenden.
- **Simulator-Abhängigkeit**: Das Migrieren eines Modells von einem Simulator auf einen anderen muß möglich sein, d.h. das Modell sollte keine speziellen Funktionen verwenden, die ausschließlich in einer bestimmten HDL enthalten sind.
- **Mixed-Signal Modelle**: Die nahtlose Integration von analogen und digitalen Elementen muß möglich sein. Im Idealfall sollen in einem Modell sowohl analoge als auch digitale Signale auftreten können.
- **Einsatzbereich**: Der Einsatzbereich sollte möglichst breit sein: Einerseits in Bezug auf die Schaltungsklassen, die modelliert werden können; andererseits auf die durchführbaren Simulationen (Groß- / Kleinsignal).
- **Genauigkeit**: Das Modell muß die bei der Generierung gemachten Anforderungen voll erfüllen. Speziell bei Verhaltensmodellen ist es wichtig, daß überhaupt eine Aussage über die Genauigkeit des Modells getroffen werden kann.
- **Skalierbarkeit**: Das Modell sollte skalierbar sein, damit nur die jeweils benötigten Eigenschaften enthalten sind. Darüberhinaus erlaubt ein flexibler Aufbau die Verwendung auf unterschiedlichen Abstraktionsebenen, z.B. Funktional und Verhalten.

Unter diesen Anforderungen werden im folgenden momentan bekannte Ansätze vorgestellt und bewertet.

2.5.2 Ansätze zur Modellierung

Die folgende Darstellung konzentriert sich auf die Modelle, in denen die Erhaltungssätze gültig sind, siehe die in Abbildung 4 dargestellten Abstraktionsebenen für analoge Schaltungen. Die heute in der Praxis eingesetzten Modelle können grob in vier Klassen eingeteilt werden:

- Spice-Netzlisten.
- Makromodelle.
- Verhaltensmodelle.
- Computer Algebra gestützte Modelle.

Eine Sonderstellung nehmen die mittels Computer-Algebra Systemen generierten Modelle ein: Aufgrund ihrer wachsenden Bedeutung wird dieser Methode ein eigener Abschnitt gewidmet, obwohl sie strenggenommen unter die Rubrik Verhaltensmodelle fällt. Im folgenden werden die in diesen Klassen enthaltenen relevanten Verfahren dargestellt.

2.5.3 Spice

Spice-Modelle sind Netzlisten auf einer der unteren Abstraktionsebenen: Komponente oder Schaltung. Die grundlegenden Elemente in diesen Modellen sind Transistoren und passive Bauelemente. Modelle können hierarchisch aufgebaut sein, das heißt ein Funktionsblock wie ein Operationsverstärker existiert als eigenständiges Modell und kann in größere Schaltungen, wie zum Beispiel einen biquadratischen Filter, eingesetzt werden. Die Schaltungstopologie spiegelt sich in der Netzliste wider.

Spice-Netzlisten nehmen im analogen Schaltungsentwurf nach wie vor eine wichtige Stellung ein. Ausgehend von der Spice-Netzliste eines Elementes werden komplexere Schaltungen aufgebaut, wobei Elemente durch Modelle auf höheren Abstrakionsebenen ersetzt werden (Bottom-Up Entwurf).

Vorteile sind die Genauigkeit, der breite Einsatzbereich, die Wiederverwendbarkeit in beliebigen Schaltungen und die Verwendbarkeit auf jedem Analogsimulator. Der Hauptnachteil ist die ineffiziente Ausführung, d.h. der Zeit- und Ressourcenbedarf der Modelle. Dazu sind Spice-Netzlisten nicht skalierbar und gemischt analog / digitale Schaltungen sind nur über Umwege modellierbar.

2.5.4 Makromodelle

Modelle auf der Makroebene sind ebenfalls weitverbreitet und werden i.d.R. von jedem Bauteilhersteller zusammen mit dem Bauteil ausgeliefert. Modelle auf der Makroebene sind empirische Modelle, das heißt sie erfordern detaillierte Kenntnisse des Modell-Entwicklers. Für die verschiedenen Operationsbereiche des Funktionsblockes werden Gleichungen explizit notiert. Makromodelle zeichnen sich durch eine hohe Genauigkeit und durch hohe Geschwindigkeit bei der Ausführung aus. Das Verfahren ist jedoch technologieabhängig, da ein Modell, das für einen bestimmten technologischen Prozeß erstellt wurde, nicht für einen anderen Prozeß übernommen werden kann [Yoo90]. Die lange Zeit, die Entwickler zum Erstellen und Modifizieren des Makromodelles benötigt, ist ebenfalls ein Nachteil.

Das empirische Makromodell von Boyle, Chon, Pederson und Solomon [BCP74] markiert seit seiner Veröffentlichung im Jahr 1974 eine Standard-Referenz, von der eine Vielzahl von Modellen abgeleitet wurde. Diese Techniken werden heute noch benutzt, ein solches Modell für die Standardkomponente MOPA6 wird im Abschnitt Beispiele zum Vergleich mit dem mittels der hier entwickelten Methodik entworfenen Modell benutzt. Abbildung 7 zeigt die Struktur des Modells nach Boyle. Die Idee des Makromodells nach [BCP74] ist die Trennung der Eingangsstufe des Operationsverstärkers von den restlichen Schaltungsteilen. In der Eingangsstufe eines Differenzverstärkers wird die Stromquelle durch eine ideale konstante Stromquelle ersetzt, das heißt, sie bleibt in der Struktur erhalten. Die übrigen Eigenschaften werden durch gesteuerte Quellen, RC-Glieder und Dioden modelliert. Schon das erste Modell erwies sich als leistungsfähig in Bezug auf Genauigkeit und Geschwindigkeit. Erweiterungen dieses Modells beinhalten neue Techniken (JFET / MOS Eingangstransistoren) und berücksichtigen das Frequenzverhalten sowie Variationen in der Umgebung (Temperatur etc.) des Modells.

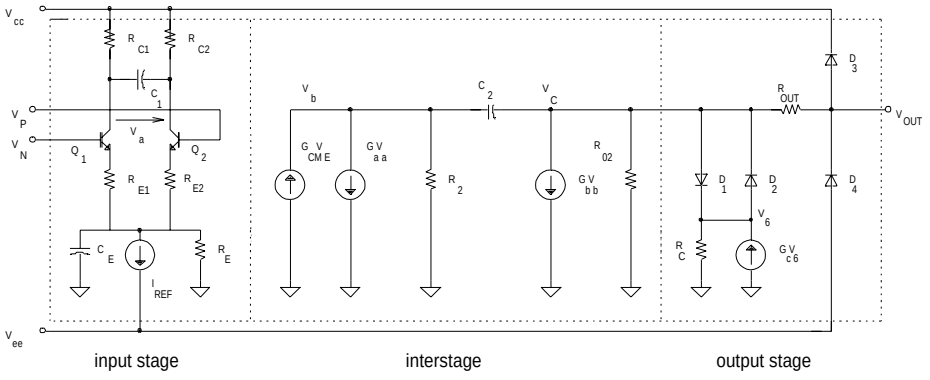

Abbildung 7: Das Makromodell nach Boyle.

Trotz der erreichten Verbesserung in der Simulationsgeschwindigkeit bei guter Genauigkeit hat dieses Modell einige Nachteile, die gegen die allgemeine Verwendbarkeit sprechen:

- Der Entwickler muß über sehr detaillierte Kenntnisse sowohl über die modellierte

Schaltung, als auch über die Einsatzbedingungen des Modells verfügen.

- Im Modell sind nichtlineare Bauelemente, wie Dioden und Transistoren, enthalten.
- Es werden ausschließlich Eigenschaften und Effekte wiedergegeben, die der Entwickler in das Modell eingebaut hat. So kann ein Modell beispielsweise nicht unabhängig von der Last benutzt werden - ein Neudesign für die spezielle Umgebung wäre notwendig, siehe das Beispiel Operationsverstärker im Kapitel Beispiele.
- Da die Modelle nicht parametrierbar sind, können sie weder an neue Aufgaben angepaßt, noch automatisch für neue Schaltungen generiert werden.
- Die im Modell benutzten physikalischen Bauelemente besitzen ihre spezifischen Kennlinien. Über die Auswahl der Bauteile können spezielle Charakteristika der zu modellierenden Schaltung sehr genau modelliert werden, allerdings ausschließlich in einem kleinen Bereich. Über den gesamten Arbeitsbereich betrachtet treten Schwankungen der Genauigkeit auf.

2.5.5 Verhaltensmodelle

Modelle auf der Verhaltensebene können auf unterschiedliche Art und Weise generiert werden. Allen Modellen gemein ist die Beschreibung des Ausgangs in Abhängigkeit des Eingangs mittels mathematischer Funktionen unter Wahrung der Erhaltungssätze. Unterschiedlich ist jedoch, wie diese mathematischen Modelle gewonnen werden und wie die Modelle implementiert werden.

Analytische Modelle Die numerischen Verfahren beruhen auf einer zwei-dimensionalen Analyse des Verhaltens, in welcher die charakteristischen Eigenschaften detailliert wiedergegeben werden. Diese Verfahren erhöhen die Komplexität des Makromodells und sind nicht effizient, weil sie auf numerischen Iterationsverfahren basieren [IMGB82].

Analytische Verfahren beschreiben die zweidimensionale Beziehung zwischen Eingangs- und Ausgangssignal implizit durch eine Menge f nicht linearer Differentialgleichungen und einer nicht linearen Gleichung g auf der Basis eines Zustandsvektors $\overrightarrow{x}$, siehe die Gleichungen (2.1). Die Lösung dieses Gleichungssystems wird numerisch mit Hilfe einer Transientenanalyse des Funktionsblockes gewonnen. Dieses Verfahren erfordert enormen Aufwand sowohl an Platz für die Simulationsergebnisse, als auch an Zeit zur Herleitung des Gleichungssystems. Die Auswertung dieser Modelle während der Simulation ist nicht effizient möglich, da die Gleichungen sehr komplex sind. Die erzeugten Modelle sind abhängig von einer bestimmten Technologie, kleine Veränderungen im Prozeß bewirken, daß das alte Modell unbrauchbar wird. Für einen schnellen Design Zyklus sind analytische Modelle deshalb unbrauchbar [CL75].

Antrim Bibliothek Die Firma Antrim Design Systems entwickelt und vertreibt einen Ansatz zur Charakterisierung und Erzeugung von Modellbibliotheken für gemischt ana-

log/digitale Zellen, siehe [Ant01]. Dieser Ansatz ist auf den ersten Blick der vorgestellten Methodik ähnlich und wird deshalb genauer untersucht. Der Ansatz zur Erstellung eines solchen Verhaltensmodells soll anhand eines Operationsverstärkers aufgezeigt werden, um das Verfahren direkt mit der vorgeschlagenen Methodik vergleichen zu können.

Festlegen der Charakteristika Zunächst müssen die charakteristischen Eigenschaften der zu modellierenden Zelle festgelegt werden. Diese werden sich später im Modell wiederfinden. Die für ein für Simulationen im Großsignalbereich notwendigen Charakteristika des Operationsverstärkers sind die Slewrates der steigenden und fallenden Flanke, die Settling Time des Ausgangs und der Ausgangsstrom.

Charakterization Engine Antrim bietet für jede charakteristische Eigenschaft ein sog. Template an. Das Template besteht aus einer Testschaltung und einer Anregung. Es ist voll parametrierbar. Das Template zum Berechnen der Slewrate des Operationsverstärkers ist eine rückgekoppelte Schaltung mit einem Spannungsimpuls am Eingang. Es wird eine Transientensimulation mit diesem Template durchgeführt, aus dem Ergebnis wird nach einer vorgegebenen Berechnungsvorschrift der gesuchte Wert extrahiert. Widerstände der Rückkoppelung und Last sind parametrierbar, ebenso die Größe des Eingangssprungs und Dauer und Schrittweite der Transientensimulation. Die Antrim Charakterization Engine greift auf das Template zu, setzt die Parameter automatisch und erzeugt so Tabellen als Ergebnisse der Charakterisierung.

Generator für Verhaltensmodelle Ein Modell Generator verwendet die Ergebnisse der Charakterisierung und erzeugt daraus Verilog-A/MS Verhaltensmodelle. Er greift auf Matlab für die notwendigen Berechnungen zurück. Das zugrunde liegende Verfahren ist nicht offengelegt. Die Modelle stehen auf der Abstraktionsebene Verhalten, die Ausgangsgrößen werden mittels mathematischer Funktionen und den Ergebnissen der Charakterisierung beschrieben.

Verifizierung der Verhaltensmodelle Die generierten Modelle werden mit den Modellen auf der Schaltkreis Ebene verglichen, um die Konsistenz zu verifizieren. Dabei ist eine Optimierung in Bezug auf die Genauigkeit möglich.

Die Modelle sind für Simulationen im Klein- und Großsignalbereich geeignet. Das Verfahren ist allgemein verwendbar, die modellierten Funktionsblöcke beinhalten sowohl analoge Zellen wie Verstärker und Oszillatoren als auch gemischt analog/digitale Zellen wie A/D- und D/A-Wandler.

Der Nachteil dieses Ansatzes ist, daß die Modelle empirisch sind. Sie enthalten ausschließlich die bei der Charakterisierung berücksichtigten Charakteristika. Somit ist detailliertes Wissen des Entwicklers notwendig, um ein Modell zu entwickeln. Wird bei einem Operationsverstärker Modell beispielsweise die Settling-Time nicht berücksichtigt, dann schwingt das resultierende Modell nicht korrekt ein. Die Verwendbarkeit der Modelle ist

also auf die Anwendungsfälle beschränkt, welche der Entwickler bei der Modellgenerierung berücksichtigt hat, eine Erweiterung ist nicht ohne weiteres möglich.

Dieses Verfahren kann folgenderweise zusammengefaßt werden:

- Das Antrim Verfahren kann auf beliebige Funktionsblöcke angewandt werden - also auch auf z.B. Oszillatoren. Voraussetzung ist, daß der Modellentwickler den Block gut genug versteht, um die notwendigen charakteristische Eigenschaften herauszuarbeiten.

- Das Verfahren nach Antrim erfordert sehr detaillierte Kenntnisse des Entwicklers über den zu modellierenden Funktionsblock. Im Gegensatz zu der hier vorgeschlagenen Methodik ist die Modellerstellung nicht automatisierbar, der Charakterisierungsaufwand ist jedoch höher.

- Die Antrim Modelle enthalten genau die charakterisierten Eigenschaften. Das Risiko dabei ist, daß Effekte bei den Simulationen nicht korrekt berücksichtigt werden. So kann beispielsweise zwar die Settling Time des Operationsverstärkers modelliert werden, wenn jedoch ein asymptotisches Einschwingen in einer bestimmten Beschaltung erfolgt (siehe das A/D Wandler Beispiel), funktioniert das Modell u.U. nicht korrekt.

Die vorgeschlagene Methodik hat die Vorteile der automatischen Modellgenerierung und allgemeinen Verwendbarkeit, d.h. eine modellierte Teilschaltung kann in jeder Beschaltung eingesetzt werden und enthält implizit Effekte höherer Ordnung.

Tabellen und PWL-Techniken LookUp Tabellen werden zunächst vom Simulator erzeugt. Aus diesen Tabellen extrahiert ein Algorithmus relevante Datenpunkte. Diese Datenpunkte werden während der Simulation interpoliert. Die Tabellen sind einfacher zu erzeugen als die Parameter analytischer Modelle. Das Verfahren ist unabhängig von einer bestimmten Technologie, bei einer Veränderung des Prozesses ist es lediglich notwendig, neue Tabellen zu erzeugen. Interpolation der Daten ist effizienter als die Auswertung der komplexen analytischen Funktionen. Nachteile des Tabellen LookUp Verfahrens sind der große Speicherplatz, der für die Tabellen benötigt wird, und die Tatsache, daß die physikalischen Parameter des Funktionsblockes sich in der Lösung nicht widerspiegeln [YA91], [Eic92].

Eine Weiterentwicklung der LookUp Tabellen stellt die schrittweise Linearisierung (PWL, piece-wise linear) dar [DP99]. Der Ansatz bezweckt die Modellierung und Approximation von Signalen, wobei er die rechenintensiven Algorithmen zur Lösung des Gleichungssystems im analytischen Ansatz umgeht. Alle nichtlinearen Funktionen werden als stückweise lineare Funktionen umgesetzt. Da ausschließlich die grundlegenden algebraischen Operatoren zum Verrechnen benutzt werden, bleiben die Funktionen immer linear. Bestimmte Effekte aus dem Zeitbereich lassen sich in die Modelle einfügen, sie sind jedoch

nicht implizit erhalten. Somit ist, ähnlich wie bei den empirischen Modellen, das Know-How des Entwicklers stark gefragt. Darüberhinaus gilt dasselbe wie für die LookUp Tabellen, nämlich daß großer Speicherplatz benötigt wird und eine nachträgliche Anpassung an bestimmte Genauigkeitsanforderungen nicht möglich ist. Das Verfahren ist jedoch sehr effizient zur Laufzeit und kann ohne Schwierigkeiten auf jedem Simulator implementiert werden, da ausschließlich Grundrechenarten verwandt werden.

Bond-Graphen Bond-Graphen sind ein Verfahren, das aus einem Diagramm von RC-Gliedern und gesteuerter Quellen automatisch eine analytische Lösung generiert. Das System nicht linearer Differentialgleichungen braucht so nicht vom Entwickler von Hand aufgestellt werden, es kann grafisch entworfen werden. Allerdings ist die Handhabung nicht linearer Bauelemente kritisch und für das generierte Modell gelten die selben Nachteile, die für die analytischen Modelle angeführt wurden [Tho90].

Nichtlineare Systemtheorie Innerhalb der nichtlinearen Systemtheorie gibt es verschiedene Ansätze zur Modellierung nichtlinearer dynamischer Systeme. Für Analogschaltungen wurden dazu die sogenannten Hammerstein- bzw. Wiener-Modelle eingeführt [Ise92]. Das Prinzip der Modelle ist es, für eine spezielle Schaltung eine Modellstruktur empirisch zu entwerfen, die Parameter des Modells können dann automatisch mit Regressionsverfahren aus den Simulatordaten generiert werden. Die Modelle bestehen aus einem nichtlinearen statischen Teil und einem linearen dynamischen Teil, welche hintereinander geschaltet sind. Die Modelle sind effizient, da für die Beschreibung des nichtlinearen Teils bevorzugt Polynome verwandt werden (obwohl jede beliebige mathematische Funktion verwendbar ist). Da die nichtlineare Schaltung nur in einem Arbeitspunkt linearisiert wird, wird die Dynamik der Schaltung sehr eingeschränkt berücksichtigt, die Modelle sind also nicht universell zu verwenden. Da die Modelle auf empirischen Ansätzen beruhen, gelten die dort gemachten Einschränkungen auch für diese Modelle.

Das Modellgenerator Konzept nach Hamad Das in [Ham95] entwickelte Konzept zur Verhaltensmodellierung mittels Modellgeneratoren adressiert die Schwächen der empirischen und Computer-Algebra basierten Ansätze. Modelle können automatisch generiert werden mit einer vorgegebenen Genauigkeit, die Beschränkung auf lineare Schaltungen fällt weg. Darüberhinaus läßt sich die Methodik auch auf Schaltkreise mit praxisrelevanter Größe anwenden, wie zum Beispiel einen biquadratischen Filter. Der Ansatz beschreibt das Ein- / Ausgangsverhalten eines analogen Funktionsblocks mittels eines Systems parametrierter mathematischer Modellgleichungen [GHH95]. Der Ausgangspunkt hierzu ist die Erzeugung der Datenbasis mit der in Kapitel 4 vorgestellten Methoden-Bibliothek.

Allerdings ist die Methodik auf die Klasse der rückgekoppelten analogen Funktionsblöcke beschränkt, wie zum Beispiel ein Operationsverstärker im nicht-invertierten Betrieb, siehe Abbildung 20. Die modellierten Teilschaltungen sind nicht allgemein verwendbar: Das Modell des Operationsverstärkers ist nur dann ohne Einschränkung funktionsfähig, wenn es in einer Schaltung mit Rückkopplung verwendet wird, eine Verwendung als Komparator ist nicht ohne weiteres möglich. Der Grund hierfür ist die Rückkopplung. Sie

sorgt dafür, daß die Veränderungen am Eingang korrekt vollzogen werden und die Schaltung nach Spannungssprüngen in einen stabilen Zustand einschwingt. Im Modell wird die Differenzspannung zwischen den beiden Eingangspins betrachtet: Ist der Ausgang jedoch nicht rückgekoppelt, so wird die Differenzspannung nicht Null und das Modell kann nicht bestimmen, zu welchem Zeitpunkt der Ausgang den korrekten Wert angenommen hat. Diese Einschränkung ist für die Praxis relevant, denn

- nur Schaltungen, die rückgekoppelt sind, können so modelliert werden und
- das Verfahren liefert für allgemeine Schaltungen falsche Ergebnisse.

2.5.6 Computer Algebra gestützte Modelle

Mit der zunehmenden Rechenleistung moderner Computer und der stetigen Verbesserung von Computer-Algebra Systemen [Fuc96] gewinnt die CALS (Computer Algebra gestützte Simulation) zunehmend an Bedeutung. Die symbolische Analyse ist eine formale Technik, die das Verhalten eines Schaltkreises mathematisch beschreibt. Dabei sind Zeit oder Frequenz die unabhängigen Variablen, Spannungen und Ströme sind die abhängigen Variablen. Zusätzlich kommen die Elemente des Schaltkreises als Parameter, d.h. symbolische Variablen, vor. Das Computer-Algebra Programm erhält diese Beschreibung als Eingabe und erzeugt daraus automatisch geschlossene Ausdrücke (mathematische Formeln) für die gewünschen Charakteristika. Einen guten Überblick über CALS gibt [Gie96]. Ein Widerstand beispielsweise taucht in einem Ergebnis mit dem Symbol R auf. Damit ist die Formel unabhängig von einer bestimmten Dimensionierung der Schaltung. Der Wert von R kann nun einfach durch Einsetzen in das Ergebnis verändert werden, ohne daß die Schaltung neu simuliert werden muß (wie bei einem numerischen Modell). Somit kann untersucht werden, welche Auswirkung R auf das Ergebnis hat, indem das Ergebnis gezeichnet wird, wobei R in einem bestimmten Bereich iteriert wird. Genauso ist es möglich, den Wert von R unter bestimmten Nebenbedingungen zu optimieren. Diese Verfahren können innerhalb des Computer-Algebra Systems durchgeführt werden.

Zwei grundlegende Verfahren zur CALS sind von Bedeutung:

- Algebraische (Matrix- oder Determinanten-basierte) Verfahren und
- Graphische (Topologie-basierte) Verfahren.

Algebraische Verfahren beschreiben das Verhalten der Schaltung mit einem Gleichungssystem mit symbolischen Koeffizienten. Die symbolische Netzwerkfunktion wird durch Umformung (symbolisches Erweitern der Determinante etc.) und symbolischen Auflösen des Gleichungssystems nach den entsprechenden Variablen automatisch vom Computer generiert. Graphische Verfahren erzeugen aus der Schaltkreis-Topologie einen oder zwei Graphen mit symbolisch gewichteten Knoten. Die Netzwerkfunktion wird durch Operationen auf diesen Graphen (Schleifen auflösen oder minimale spannende Bäume) gewonnen [GW96].

In beiden Fällen schließt sich an das Aufstellen der symbolischen Netzwerkfunktion das Vereinfachen selbiger an. Unter Vorgabe eines tolerierten Fehlers werden die Gleichungen symbolisch vereinfacht, d.h. irrelevante Elemente werden aus der Gleichung eliminiert.

Die Symbolische Analyse ist bis heute weitgehend auf lineare Schaltkreise beschränkt, d.h. Schaltkreise, die aus Widerständen, Kapazitäten, Leitwerten, unabhängigen und gesteuerten Quellen bestehen. Nichtlineare analoge Schaltungen (z.B. Operationsverstärker) müssen in eine äquivalente Kleinsignalschaltung linearisiert werden. Die Kleinsignal-Simulationen erfolgen im Frequenzbereich. Simulationen im Zeitbereich und das Großsignalverhalten werden nur unter Verwendung zusätzlicher numerischer Iterationsverfahren berücksichtigt [STH98].

Die wichtigste Einschränkung ist bis heute die Beschränkung der Größe der Schaltkreise, die mit symbolischen Verfahren modelliert werden können. Da die Größe des generierten Gleichungssystems mit der Anzahl der Elemente des Schaltkreises exponentiell anwächst, kommen selbst modernste Computer schnell an die Grenzen des verfügbaren Speicherplatzes bzw. die Berechnungen sind nicht in einem vernünftigen Zeitrahmen durchführbar. In der Praxis stellen Schaltungen mit mehreren hundert Transistoren die obere Größe des Machbaren dar. Somit ist das wichtigste Forschungsthema die Verbesserung der Effizienz bei der Berechnung der Funktionen. In der Literatur sind hierzu drei Verfahren bekannt [Bor97]:

- Vereinfachung vor dem Erzeugen
- Vereinfachen während des Erzeugens
- Vereinfachen nach dem Erzeugen.

Trotz der Beschränkung der Größe der zu modellierenden Schaltung existieren dennoch wichtige Anwendungen für symbolische Techniken in der analogen Schaltungsentwicklung. Diese sind:

- Lehre: ideales Werkzeug für einen ersten Einblick in das Schaltungsverhalten
- Automatische Auslegung der Schaltkreisparameter
- Erforschen der Struktur des Designraums
- Statistische Analyse
- Fehlerdiagnostik und Test
- Generieren analoger Verhaltensmodelle.

Praxisrelevant ist das auf dem Computer-Algebrasystem Mathematica aufgesetzte Analog Insydes [HH98]. Um mit der Komplexität der Ergebnisse umzugehen, implementiert dieses System Funktionen zum Erzeugen vereinfachter symbolischer Formeln, die klein

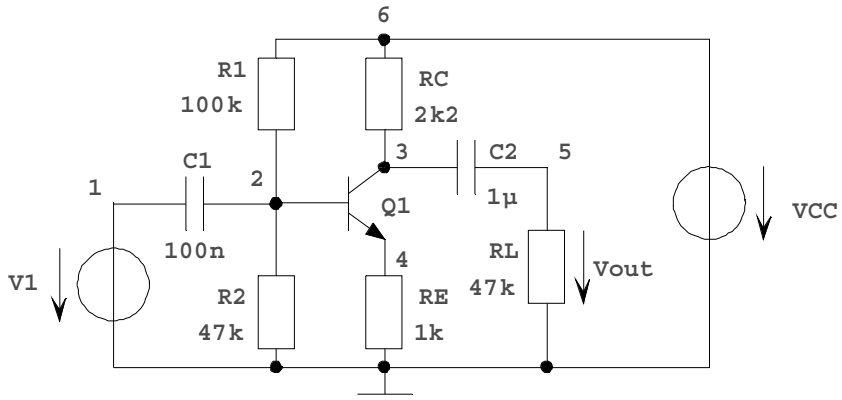

Abbildung 8: Common-Emitter Verstärker.

und einfach zu verstehen sind. Die angenäherten symbolischen Ausdrücke werden erzeugt, indem numerisch unbedeutende Terme aus den Schaltkreis-Gleichungen oder den Übertragungsfunktionen eliminiert werden.

Ein Common-Emitter Kleinsignal Verstärker ist als Beispiel für die Anwendung von Analog Insydes in Abbildung 8 dargestellt. Das resultierende Gleichungssystem hat bereits für diese sehr überschaubare Schaltung mehr als 100 Terme.

Das folgende Listing in Mathematica-Notation zeigt, wie die entsprechende Netzliste, die als Eingabe für Analog Insydes dient, aufgestellt wird (sie kann automatisch aus Spice übernommen werden). Das besondere hieran ist, daß für jedes Bauteil sowohl ein Symbol, wie z.B. C1, als auch ein numerischer Wert vorgegeben werden. So kann ein und dieselbe Netzliste sowohl als Eingabe für die symbolische Vereinfachung bzw. das Lösen des Gleichungssystems verwandt werden, als auch für eine rein numerische Simulation.

```
Netlist[
   {V1, {1, 0}, 1}, (* Unit signal at input *)
   {VCC, {6, 0}, type -> VoltageSource, Value -> 0},
   {C1, {1, 2}, Symbolic -> C1, Value -> 1.*10^-7},
   {R1, {2, 6}, Symbolic -> R1, Value -> 1.*10^5},
   {R2, {2, 0}, Symbolic -> R2, Value -> 4.7*10^4},
   {RC, {6, 3}, Symbolic -> RC, Value -> 2.2*10^3},
   {RE, {4, 0}, Symbolic -> RE, Value -> 1.*10^3},
   {C2, {3, 5}, Symbolic -> C2, Value -> 1.*10^-6},
   {RL, {5, 0}, Symbolic -> RL, Value -> 4.7*10^4},
   {Q1, {2 -> B, 3 -> C, 4 -> E},
   Model-> NPNTransistor, Selector -> ACdynamic,
   RBN1, BetaN -> 200., RON -> 1.*10^4}]
```

Gesucht ist ein einfacher Ausdruck für die Bandpass Spannungsverstärkung. Dazu wird mittels der Vereinfachung vor dem Erzeugen Funktionen das Gleichungssystem für einen

bestimmten Arbeitspunkt (hier: $f = 1kHz$) und mit den numerischen Werten vereinfacht. Wird ein maximaler Fehler von 10% in diesem Punkt erlaubt, so liefert Analog Insydes die Formel

$$Vout = -RC/RE$$

als Ergebnis für den Spannungsgewinn am Ausgang. Dies entspricht den in der Literatur bekannten Näherungsformeln für diesen Typ Verstärker. Bei einem Vergleich zwischen den Original-Ergebnissen und der vereinfachten Gleichung wird deutlich, daß die Formel im spezifizierten Arbeitspunkt innerhalb des Toleranzbereichs liegende Ergebnisse liefert. Das Vereinfachen vor dem Erzeugen liefert für eine genau spezifizierte Aufgabenstellung (hier: Frequenz im Arbeitspunkt und numerische Werte für die Bauteile) sehr gute Ergebnisse, jedoch keine allgemein gültigen Gleichungen.

Durch Hinzufügen von Designpunkten in anderen Frequenzbereichen kann der Bereich, in dem die Lösung gültig ist, erweitert werden. Für die folgenden Berechnungen wurde ein zweiter Designpunkt mit einer Frequenz von 100 Hz zusätzlich berücksichtigt. Die gewonnenen Ergebnisse können durch Durchführen von Vereinfachungen nach dem Erzeugen weiter vereinfacht werden, dabei werden alle für die gesetzten Designpunkte und die konkrete Aufgabenstellung irrelevanten Terme eliminiert. Das Ergebnis

$$Vout = -\frac{C1 * C2 * R1 * R2 * RC * RL * s^2}{RE * (C1 * R2 * s * R1 + R1 + R2) * (C2 * RL * s + 1)}$$

hat weniger als 10 Terme. Abbildung 9 zeigt ein Bode Diagramm mit dem Vergleich der auf der Komponentenebene gewonnenen Originalergebnissen mit der vereinfachten symbolischen Lösung. Das Bode Diagramm zeigt, daß die approximierte Ausgangsfunktion von der Frequenz 0 bis hin zum oberen Ende des Bandes korrekt arbeitet. Effekte bei höheren Frequnzen werden nicht wiedergegeben, da sie bei der Spezifikation des Designpunktes nicht berücksichtigt wurden.

Analog Insydes berechnet eine vereinfachte symbolische Übertragungsfunktion, welche ausschließlich die Terme mit dominantem Einfluß auf die Bandpass Spannungsverstärkung enthält. Die Vereinfachungen reduzieren die Anzahl der Terme im Ergebnis von über 100 auf 10.

Trotz dieser relevanten Anwendungen, die die Bedeutung der symbolische Analyse für Schaltkreise praktischer Größe zeigen, bleiben die wichtigsten zu lösenden Themen die Erweiterung der Schaltkreisklassen auf nichtlineare Schaltungen [WPHH99] und die Erweiterung der Simulationen auf den Zeitbereich. In [FRHG98] werden Möglichkeiten aufgezeigt, wie größere analoge Schaltkreise durch geeignete Vereinfachungen modelliert werden können.

Da symbolische Modelle derzeit weder den Transientenbereich noch das Großsignalverhalten abdecken, stehen sie nicht komplementär zu den Funktionsblockmodellen, sondern decken mit dem Kleinsignalverhalten genau den Bereich ab, den die vorgeschlagene Methodik nicht modelliert.

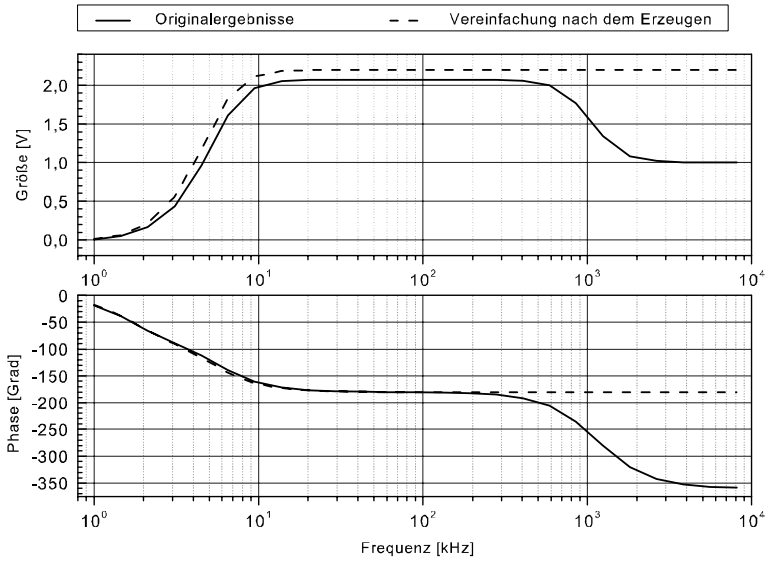

Abbildung 9: Vereinfachung nach dem Erzeugen mit zwei Designpunkten.

2.5.7 Bewertung der Ansätze

Die heue verfügbaren Verfahren zur Verhaltensmodellierung analoger Schaltungen basieren wesentlich auf der Erfahrung und dem Wissen des Schaltungsentwicklers. Die Modelle sind spezifisch für eine bestimmte Schaltungsklasse, oftmals sogar für einen bestimmten Einsatz des Modells. Fehlerabschätzungen bzw. das Setzen bestimmter Toleranzbereiche zur Simulationszeit sind im allgemeinen nicht möglich. Das automatische Erzeugen von numerischen Modellen basiert auf einer riesigen Datenbasis (Tabellen, PWL) und ist extrem technologieabhängig. Automatische Verfahren für den symbolischen Modellentwurf berücksichtigen z.Zt. nur lineares dynamisches und nichtlineares statisches Schaltungsverhalten. Darüberhinaus sind sie auf kleine Schaltungen (kleiner 100 Transistoren) beschränkt. Auf der anderen Seite steht mit VHDL-AMS eine Repräsentations- und Ausführungsmöglichkeit für Modelle zur Verfügung, die den VHDL Standard um die benötigten Konzepte für den Entwurf analoger Schaltungen erweitert. Die Modelle können auf unterschiedlichen Abstraktionsebenen entworfen werden, vom Einbinden einer Spice-Netzliste bis hin zu einer mathematischen Beschreibung der Funktion ist alles möglich und kann miteinander kombiniert werden. VHDL-AMS unterstützt speziell den Entwurf gemischt analog/digitaler Schaltungen und ist dabei, unanhängig von einem bestimmten Hersteller, auch in der Praxis zum Standard zu werden. Die bekannten Verfahren zur Verhaltensmodellierung nutzen jedoch nur Bruchteile der in VHDL-AMS bereitgestellten Funktionalität.

Die in diesem Kapitel vorgestellten Anforderungen an ein Modell werden in den Zeilen

der Tabelle 1 aufgelistet. Die Spalten enthalten die unterschiedlichen Modellansätze, die in den vorhergehenden Abschnitten vorgestellt wurden: SM steht für Spice Netzlisten, MM für Makromodelle, VM für Verhaltensmodelle und CAM für Computer-Algebra Modelle. Zusätzlich existiert als Vorgriff auf die Leistungen der hier vorgeschlagenen Methodik die Spalte FBM (Funktionsblockmodell) mit der Bewertung für derart entwickelte Modelle.

In den Zellen der Tabelle steht eine Bewertung, welche Modelle welchen Anforderungen wie gerecht werden. Als Symbole werden - (nicht vorhanden / nicht möglich), o (erfüllt das Kriterium durchschnittlich), + (erfüllt das Kriterium gut) und ++ (erfüllt das Kriterium hervorragend) benutzt.

Ausgehend von dieser Bewertung werden die Modelle mit der vorgeschlagenen Methodik verglichen. Die Unterschiede der neuen Methodik werden in der Spalte Vorteil dargestellt, hier zeigt das Symbol ++ die Vorteile im Vergleich zu anderen Modellen auf höheren Abstraktionsebenen auf, wie in den folgenden Kapiteln gezeigt wird.

Anforderung	**Bezug auf**	**SM**	**MM**	**VM**	**CAM**	**FBM**	**Vorteil**
Effizienz	SpeedUp	+	++	++	++	++	+
	Ressourcen	+	++	++	++	++	=
Wiederverwendbar		+	+	+	+	++	+
Modellgenerierung	Ressourcen	++	++	++	o	o	=
	Automatisch	-	-	-	+	++	++
Benötigtes Wissen	Entwickler	+	+	+	+	++	++
	Anwender	+	+	+	+	+	+
Simulator abhängig		++	+	+	+	++	++
Mixed Signal		-	-	+	o	++	++
Einsatzbereich	Simulation	++	+	+	+	+	=
	Klassen	++	++	++	+	+	=
Genauigkeit		++	++	+	+	++	=
Skalierbarkeit		+	+	+	++	++	++

Tabelle 1: Anforderungen an ein Modell und Realisierung.

3 Konzepte zur Verhaltensmodellierung

Im Mittelpunkt dieses Kapitels steht die mathematische Definition des neuen Konzepts zur Verhaltensmodellierung. Zunächst wird erläutert, warum und wie das Konzept auf einer systemtheoretischen Basis aufgesetzt wird. Die wichtigsten Modelle der Systemtheorie werden anhand kurzer Beispiele erläutert und eine Motivation für die Verwendung des kombinierten Ansatzes gegeben. Abgeschlossen wird das Kapitel durch Betrachtungen zu den von der Methodik abgedeckten Schaltungsklassen und zu den Anwendungsbereichen der Funktionsblockmodelle.

3.1 Die Theorie dynamischer Systeme

Die Systemtheorie bietet einen interdisziplinären Ansatz zur Modellierung und Simulation dynamischer Systeme. Sie unterscheidet nicht zwischen elektronischen, mechanischen oder chemischen Systemen. Eine für einen konkreten Anwendungsbereich entwickelte Methodik läßt sich direkt auf andere Disziplinen anwenden. Die Systemtheorie stellt ein formales Instrumentarium für die Modellierung auf der Makro- und der Verhaltensebene bereit und ermöglicht die theoretische Spezifikation der Konzepte, die der Implementierung der Modelle zugrunde liegen [Zei89]. Speziell für die Spezifikation hybrider Systeme, das heißt im Zusammenspiel digitaler zeit- bzw. ereignisdiskreter mit analogen zeitkontinuierlichen Komponenten, stehen geeignete Basisinstrumente zur Verfügung.

Nach der Vorstellung eines klassischen dynamischen Systems (mit Ein- und Ausgängen) werden die erweiterten Instrumente für hybride Systeme vorgestellt. Auf deren Basis erfolgt die Ableitung des Konzeptes für die Funktionsblockmodelle.

3.1.1 Modellklassen

Ein allgemeines dynamisches System mit Ein- und Ausgängen wird durch den System-Zustand und die System-Dynamik beschrieben. Der Zustand beschreibt das innere Verhalten zu einem bestimmten Zeitpunkt, die Dynamik legt fest, wie sich der interne Zustand mit der Zeit verändert. Dazu wird eine Übergangsfunktion definiert, die die Zustandsänderung in Abhängigkeit der Eingangswerte und des aktuellen internen Zustandes berechnet. Eine Ausgangsfunktion berechnet die Werte des Ausgangs in Abhängigkeit des internen Zustandes und gegebenenfalls in Abhängigkeit von den Eingangsgrößen.

Im allgemeinen wird das Verhaltensmodell eines nichtlinearen analogen Funktionsblockes als dynamisches System definiert, [Pra91]:

Ein- / Ausgabesystem (allgemein: dynamisches System)

$$IOS = \{t, \overrightarrow{u}, \overrightarrow{y}, \overrightarrow{x}, \delta, \lambda\}. \tag{3.1}$$

Tabelle 2 listet die Elemente des Systems auf.

t	Zeit
$\overrightarrow{u}$	Vektor der Eingangsgrößen
$\overrightarrow{y}$	Vektor der Ausgangsgrößen
$\overrightarrow{x}$	Vektor der Zustandsvariablen
δ	Zustandsübergangsfunktion
λ	Ausgangsfunktion

Tabelle 2: Allgemeines dynamisches Ein- / Ausgabesystem.

Ein *Ein- / Ausgabesystem* beschreibt das Verhalten des Ausgangs $\overrightarrow{y}$ in Abhängigkeit der Eingangsgrößen $\overrightarrow{u}$. Mittels der Zustandsvariablen $\overrightarrow{x}$ und der Übergangsfunktion δ wird die innere Struktur des Systems modelliert. Zur Ausführung der Spezifikation eines Funktionsblocks wird ein Modell benötigt, welches mathematisch den Zusammenhang von Eingang zu Ausgang im Zeitbereich beschreibt. Als Modellklassen wird zwischen zeitkontinuierlichen Modellen (DESS, Differential Equation Specified System), ereignisdiskreten Modellen (DEVS, Discrete Event Specified System) und zeitdiskreten Modellen (DTSS, Discrete Time Specified System) unterschieden [Zei84].

Im folgenden werden diese Modellklassen vorgestellt und, darauf aufbauend, die Anforderungen des Funktionsblockmodells dargestellt.

Zeitkontinuierliches Modell Ein zeitkontinuierliches Modell ist eine Struktur

$$DESS = \{\overrightarrow{u}, \overrightarrow{y}, \overrightarrow{x}, f, \lambda\}. \tag{3.2}$$

$\overrightarrow{u}$	Vektor der Eingangsgrößen
$\overrightarrow{y}$	Vektor der Ausgangsgrößen
$\overrightarrow{x}$	Vektor der Zustandsvariablen
f	Funktion für die zeitliche Änderung
λ	Ausgangsfunktion

Tabelle 3: Elemente des zeitkontinuierlichen Modells.

Tabelle 3 listet die Elemente des Systems auf.

Ein DESS ist gekennzeichnet durch die zeitkontinuierlichen Verläufe der Variablen, d.h. die Zeitbasis ist die Menge der reellen Zahlen. Die Werte der Zustandsvariablen ändern sich innerhalb eines endlichen Zeitrahmens unendlich oft. Die Repräsentation erfolgt in der Regel durch ein System von Differentialgleichungen. Ein einfaches Beispiel für ein lineares Netzwerk lautet:

$$\frac{du_0}{dt} = R_1 * (\frac{di_1}{dt} - \frac{di_2}{dt}) + \frac{1}{C_1}(i_1 - i_2 - i_3). \tag{3.3}$$

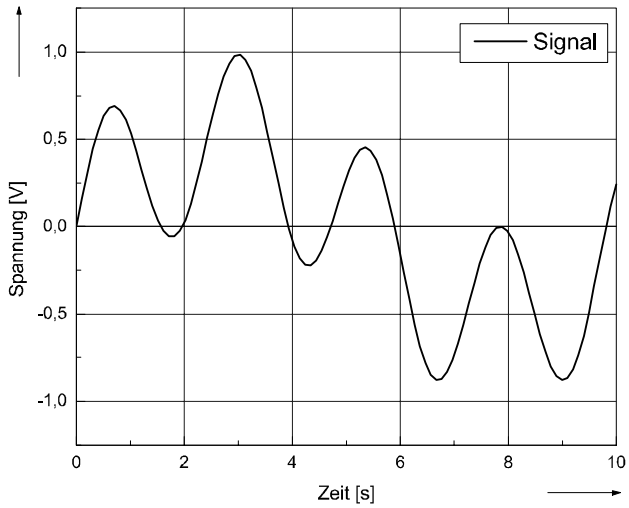

Abbildung 10: Beispiel für zeitkontinuierliches Signal.

Abbildung 10 zeigt ein Beispiel für einen zeitkontinuierlichen Signalverlauf.

Ereignisdiskretes Modell Ein ereignisdiskretes Modell ist eine Struktur

$$DEVS = \{\overrightarrow{u}, \overrightarrow{y}, \overrightarrow{x}, \delta, \lambda, ta\}. \tag{3.4}$$

$\overrightarrow{u}$	Vektor der Eingangsgrößen
$\overrightarrow{y}$	Vektor der Ausgangsgrößen
$\overrightarrow{x}$	Vektor der Zustandsvariablen
δ	Übergangsfunktion für die Ereignisse
λ	Ausgangsfunktion
ta	Zeit-Funktion

Tabelle 4: Elemente des ereignisdiskreten Modells.

Tabelle 4 listet die Elemente des Systems auf.

Ein DEVS ist gekennzeichnet durch kontinuierliche Wertebereiche der Variablen; die Zeitbasis ist die Menge der reellen Zahlen. Die Werte der Zustandsvariablen ändern sich innerhalb eines endlichen Zeitrahmens nur endlich oft: Genau dann, wenn ein Ereignis eintritt. Es treten interne Ereignisse, das heißt vom Simulator erzeugte Ereignisse (Integrationsschritte) und externe von außen getriggerte Ereignisse auf. Ein externes Ereignis wird in der folgenden Form gesetzt

```
schedule_event(t_1,x_i,v);
```

und durch die Übergangsfunktion δ abgearbeitet.

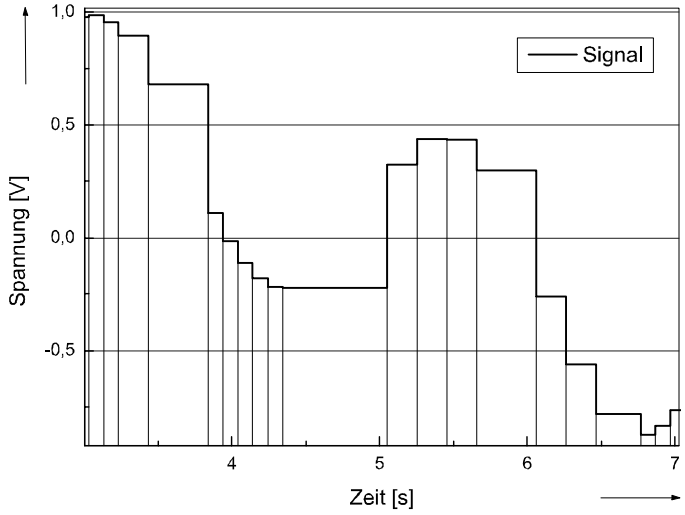

Abbildung 11: Beispiel für ein ereignisdiskretes Signal.

Abbildung 11 zeigt ein Beispiel für einen ereignisdiskreten Signalverlauf.

Zeitdiskretes Modell Ein zeitdiskretes Modell ist eine Struktur

$$DTSS = \{\overrightarrow{u}, \overrightarrow{y}, \overrightarrow{x}, \delta, \lambda\}. \tag{3.5}$$

$\overrightarrow{u}$	Vektor der Eingangsgrößen
$\overrightarrow{y}$	Vektor der Ausgangsgrößen
$\overrightarrow{x}$	Vektor der Zustandsvariablen
δ	Übergangsfunktion für die Ereignisse
λ	Ausgangsfunktion

Tabelle 5: Elemente des ereignisdiskreten Modells.

Tabelle 5 listet die Elemente des Systems auf.

Ein DTSS ist gekennzeichnet durch diskrete Wertebereiche der Variablen; Zeit und Variablen werden zu diskreten Zeitpunkten "abgetastet", wobei die Zeitschritte äquidistant

sind. Die Werte der Zustandsvariablen ändern sich innerhalb eines endlichen Zeitrahmens nur zu den durch die Abtastrate bestimmten diskreten Zeitpunkten. Ein DTSS wird durch Differenzengleichungen implizit definiert:

$$\overrightarrow{y}^{(i+1)} = g[\overrightarrow{u}^{(i)}, \overrightarrow{x}^{(i)}, t^{(i)}] \tag{3.6}$$

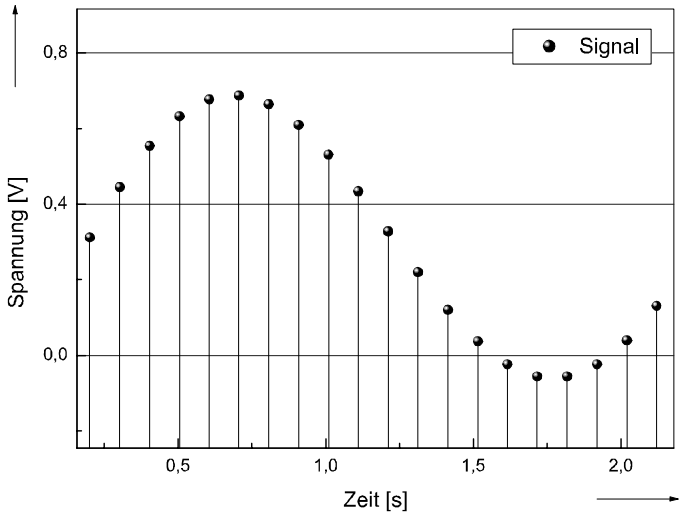

Abbildung 12: Beispiel für ein zeitdiskretes Signal.

Abbildung 12 zeigt ein Beispiel für einen ereignisdiskreten Signalverlauf.

3.1.2 Erweiterter Formalismus für kombinierte Modelle

Die im vorigen Abschnitt vorgestellten Konzepte eignen sich zur Beschreibung analoger Schaltungen (DESS) und von digitalen Schaltungen (DTSS oder DEVS). Elektronische Schaltungen sind jedoch nicht nur zunehmend komplex, sondern insbesondere durch das Zusammenspiel von zeitdiskreten und zeitkontinuierlichen Komponenten gekennzeichnet. Für diese kombinierten Modelle, welche sowohl einen zeitdiskreten, als auch einen zeitkontinuierlichen Teil enthalten, wird die vorgestellte klassische Aufteilung in DESS, DEVS und DTSS in [Pra91] erweitert um Formalismen für kombinierte Modelle. Ein DESS&DEVS (Differential Equation Specified System with Discrete Event Specified System) ist definiert als:

Definition 1 (DESS&DEVS)

$$DESS\&DEVS = \{t, \overrightarrow{u}, \overrightarrow{y}, \overrightarrow{x}, \delta_{x\&s}, \delta_{int}, \lambda, ta, f\} \tag{3.7}$$

$\overrightarrow{u}$	Eingangsgrößen
$\overrightarrow{y}$	Ausgangsgrößen
$\overrightarrow{x}$	Zustandsvariablen
$\delta_{x\&s}$	Übergangsfunktion für externe und Zustands-Ereignisse
δ_{int}	Übergangsfunktion für interne Ereignisse
λ	Ausgangsfunktion
ta	Zeit-Funktion
f	Funktion für die zeitliche Änderung

Tabelle 6: Ein DESS&DEVS besteht aus einem zeitdiskreten und einem zeitkontinuierlichen Teil.

Tabelle 6 listet die Bestandteile des DESS&DEVS auf. Moderne HDL (HDL: Hardware Description Language, Hardware Beschreibungssprachen) unterstützen dieses Konzept nahezu direkt durch die Möglichkeit, Zustandsvariablen zu definieren und Ereignisse für diese Variablen und die Ein- und Ausgangsvariablen zu setzen.

Beispiel 1 *Ein Ball, der aus einer bestimmten Höhe fallen gelassen wird, ist ein typisches Beispiel für ein kombiniertes Modell. Das Modell hat keine Eingänge, $\overrightarrow{u} = \{\}$, die Höhe des Balles ist die Ausgabe $\overrightarrow{y} = \{s\}$, als Zustände werden Höhe und Geschwindigkeit definiert, $\overrightarrow{x} = \{v, s\}$.*

Der freie Fall läßt sich durch eine einfache Differentialgleichung beschreiben. Die Geschwindigkeit v ist als die Ableitung des Weges s nach der Zeit t definiert. Die Funktion f für die zeitliche Änderung der Ausgänge und Zustandsvariablen lautet für dieses Beispiel:

$$\begin{aligned} \frac{dv}{dt} &= -g - v^{2R} \\ \frac{ds}{dt} &= v \end{aligned} \tag{3.8}$$

Die entsprechende VHDL-AMS Implementierung für f lautet für positive Geschwindigkeit v (für negative Geschwindigkeit wirkt der Luftwiderstand entgegen):

```
s'dot == v;
v'dot == -G - v** 2* Air_ Res;
```

Der Aufprall des Balles auf den Boden stellt ein diskretes Ereignis dar. Die Übergangsfunktion für externe und Zustands-Ereignisse $\delta_{x\&s}$ invertiert die Geschwindigkeit, sobald der Weg (die Position des Balles) nicht positiv ist, sie lautet in VHDL-AMS:

```
break v => -v when not s'above( 0.0);
```

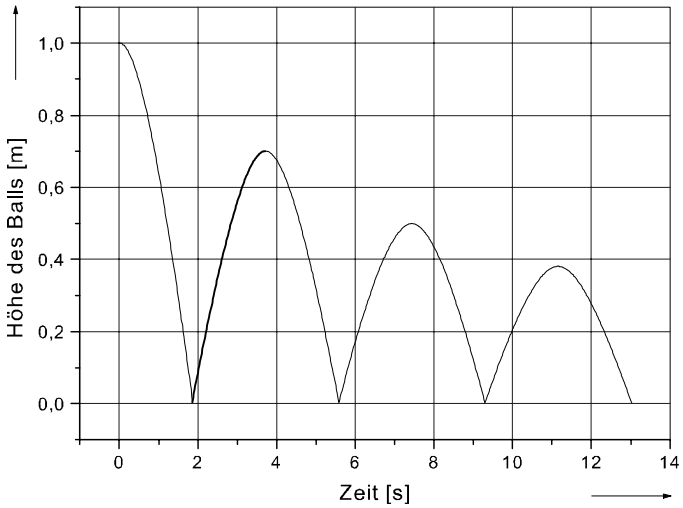

Abbildung 13: Invertieren der Richtung beim Aufprall auf den Boden.

Die Ausgangsfunktion λ gibt die Höhe s des Balles aus. Die Zeit-Funktion ta wird vom Simulator gesteuert.

Abbildung 13 zeigt das Ergebnis einer Simulation des Beispiels „springender Ball". Das System ließe sich weder durch eine geschlossene Differentialgleichung, noch durch reine Ereignisbehandlung effizient beschreiben. Das DESS&DEVS ist eine eins-zu-eins Umsetzung des Systems, welches exakt den physikalischen Grundlagen entspricht.

3.1.3 Anforderungen an den Simulator

Die Funktionsblockmodelle erfordern im allgemeinen die Leistungen der Transientensimulation in Verbindung mit der Fähigkeit, Ereignisse zu verarbeiten. Diese Anforderung ist wohl bekannt von der Simulation gemischt analog/digitaler Schaltungen, bei welcher der zeitkontinuierliche analoge Teil der Schaltung mit dem zeitdiskreten, ereignisorientierten digitalen Teil interagiert. Der kommende IEEE Industriestandard VHDL-AMS [VHDL97] ist eine Erweiterung des für den Design digitaler Schaltungen de fakto Industriestandards VHDL 1076-1993. Die wesentliche Erweiterung in VHDL-AMS ist eben die benötigte Fähigkeit der Simulation gemischt analog/digitaler Schaltungen. Diese Fähigkeit ermöglicht die einfache Implementierung der Modelle in jedem Simulator, welcher VHDL-AMS unterstützt. Allein durch die Anpassung der Syntax kann ein Modell von einem Simulator auf einen anderen portiert werden. Diese Aufgabe kann fast komplett von Codegeneratoren übernommen werden, denn die enthaltene mathematische Grundfunktionalität benötigt die Sprachelemente Bedingungen und Polynome. Diese Elemente sind neben in VHDL-AMS auch in den Simulationsumgebungen ELDO-FAS [MM91] oder Saber MAST [Ana96]

enthalten, welche zu den meistgenutzten Simulatoren gehören.

3.2 Definition des Funktionsblockmodells

Zu Beginn dieses Kapitels wurden die Grundkonzepte der Systemtheorie vorgestellt und ein Formalismus zur Verhaltensmodellierung kombiniert diskret-kontinuierlicher Systeme definiert. In diesem Abschnitt soll nun das Funktionsblockmodell auf den systemtheoretischen Ansatz abgebildet werden. Im Kapitel 5 wird anhand verschiedener Funktionsblöcke gezeigt, wie das Konzept in der Simulationspraxis verwandt wird. Die folgenden Grundlagen und Überlegungen beziehen sich primär auf elektronische Schaltungen [RH98].

Bei der Analyse des Transientenverhaltens eines allgemeinen dynamischen Systems kann in der Regel kein direkter Zusammenhang zwischen Eingangsgrößen und Ausgangsgrößen hergestellt werden. Unendlich viele Eingangsformen stehen unendlich viele Ausgangsformen gegenüber.

Die Berechnung der Ausgangsgrößen in einem beliebigen Zeitschritt ist nicht ausschließlich von den am Eingang anliegenden Größen, sondern immer auch vom bisherigen Verlauf der Simulation abhängig.

Im Transientenbereich wird immer die Veränderung der Ausgangsgrößen betrachtet. Diese Veränderungen können als Differentialgleichungen beschrieben werden. Ein Teil des Modells besteht aus einer oder mehreren kontinuierlich zeitlich Veränderlichen.

Neben diesen kontinuierlichen Größen gibt es Ereignisse, die zu diskreten Zeitpunkten eintreten. Analoge Schaltungen haben in der Regel eine Verzögerungszeit, die den diskreten Zeitpunkt ausdrückt, ab dem eine Veränderung des Eingangs sich am Ausgang bemerkbar macht.

Diese Anforderungen werden vom DESS&DEVS nach Definition (3.7) vollständig abgedeckt. Es stellt deshalb eine optimale Basis für die Definition des neuen Konzepts zur Verhaltensmodellierung dar. Dieses Kapitel definiert zunächst das Modell, die folgenden Abschnitte stellen dann die einzelnen Elemente ausführlich vor.

Definition 2 (Funktionsblockmodell) *Das Funktionsblockmodell läßt sich nach Definition (3.7) als DEVS&DESS darstellen.*

$$FktBlockModell = \{\overrightarrow{u}, \overrightarrow{y}, \overrightarrow{x}, \delta, \lambda, ta, f\} \tag{3.9}$$

Tabelle 7 erläutert die in der Signatur (3.9) aufgelisteten Elemente des Modells. Die einzelnen Elemente werden im folgenden ausführlich beschrieben und ihre Funktion innerhalb der Funktionsblockmodells wird dargestellt.

$\overrightarrow{u}$	Eingangsgrößen
$\overrightarrow{y}$	Ausgangsgrößen
$\overrightarrow{x}$	Zustandsvariablen
δ	Übergangsfunktion für Ereignisse
λ	Ausgangsfunktion
ta	Zeit-Funktion
f	Funktion für die zeitliche Änderung

Tabelle 7: Die Elemente des Funktionsblockmodells.

3.3 Ein- und Ausgangsgrößen

$\overrightarrow{u}$ ist der Vektor der k Eingangsgrößen des Funktionsblocks. $\overrightarrow{u}$ beinhaltet die an den Eingangspins anliegenden Spannungen und die durch die Eingangspins fließenden Ströme.

$$\overrightarrow{u} = \{u_i | u_i \in I\!R, i = 1 \ldots k\}. \tag{3.10}$$

$\overrightarrow{y}$ ist der Vektor der l Ausgangsgrößen des Funktionsblocks. $\overrightarrow{y}$ beinhaltet die an den Ausgangspins anliegenden Spannungen und die durch die Ausgangspins fließenden Ströme.

$$\overrightarrow{y} = \{y_i | y_i \in I\!R, i = 1 \ldots l\}. \tag{3.11}$$

3.4 Zustandsvariablen

In der Struktur des Makromodells werden neben den Ein- und Ausgangsgrößen $\overrightarrow{u}$ und $\overrightarrow{y}$ eine beliebige Anzahl von Variablen zur Beschreibung des Verhaltens des Funktionsblocks benutzt. Diese Variablen können beispielsweise interne Spannungsdifferenzen oder Ströme enthalten. Sie stellen die internen Zustände des Funktionsblockmodells dar. Anhand der Belegung dieser Zustände kann jederzeit das Ausgangssignal berechnet werden, auch wenn die Schaltungssimulation unterbrochen war. Das dynamisches System enthält somit die Menge der m Zustandsvariablen $\overrightarrow{x}$:

$$\overrightarrow{x} = \{x_i | x_i \in I\!R, i = 1 \ldots m\}. \tag{3.12}$$

Die Zustandsvariablen $\overrightarrow{x}$ beinhalten die Energiespeicher des dynamischen Systems, siehe Abschnitt 2.1. Im Gegensatz zu dieser seit langem bekannten Spezifikation zeitkontinuierlicher Systeme, die u.a. der Arbeit [GW98] zugrunde liegen, kommen im Funktionsblockmodell die intern benutzten Variablen (Through oder Across Quantities) hinzu.

Die leistungsfähige Modellierungssprache VHDL-AMS basiert auf dem für die Entwicklung und Simulation digitaler Schaltungen bekannte VHDL. Die übernommenen Konzepte werden um Beschreibungsmöglichkeiten für zeit- und wertkontinuierliche Modelle ergänzt. Als Erweiterung des klassischen Vorgehens mit Energiespeichern ist die Definition von Quantities (Zustandsvariablen) möglich. Diese sind vorgesehen zur Deklaration von wert-

und zeitkontinuierlichen Variablen. Sie können als Free, Source oder Branch Quantities (definiert durch Across und Through Quantities) verwendet werden und erlauben die Abstraktion von einem konkreten Energiespeicher in zu einer allgemeinen zeitkontinuierlichen Variable [HKR99]. Somit stehen im Verhaltensmodell Free Quantities zur Verfügung, welche zum Modellieren von Differenzen oder von Charakteristika verwendet werden können, ohne einen konkreten Bezug zu einem Energiespeicher des Systems zu haben. Folgendes Beispiel für ein VHDL-AMS Verhaltensmodell verwendet die Quantity `qint` um den Ausgang eines Addierer/Integrators zu berechnen - rein mathematisch, ohne Bezug auf Energiespeicher der zugrunde liegenden analogen Schaltung:

```
architecture Sfg of AdderIntegrator is
quantity qint: REAL;
begin
   qint == k1* in1 + k2* in2;
   output == qint'integ;
end architecture Sfg;
```

Diese Erweiterung der Zustände des Systems um die Zustandsvariablen im Funktionsblockmodell ist wichtig. Die Anzahl der Zustandsvariablen des Modells ist nicht notwendigerweise gleich der Anzahl der Energiespeicher in der Strukturbeschreibung des modellierten Systems.

Der dem Funktionsblockmodell zugrunde liegende Zustandsbegriff wird im Kapitel 3.10 ausführlich dargestellt und es wird gezeigt, wie die Energiespeicher des modellierten Systems in das Funktionsblockmodell einfließen.

3.5 Übergangsfunktion für Ereignisse

Die Übergangsfunktion δ legt fest, was mit den Zustandsvariablen des Modells passiert, wenn ein Ereignis eintritt. Sie berechnet die Werte, mit welchen die Zustände belegt sind, mittels mathematischer Funktionen neu. Sie wird repräsentiert durch einen Vektor $\overrightarrow{h}$ von o Funktionen h_j.

$$\delta() = \overrightarrow{h} = \{h_j, j = 1 \ldots o\}$$

Die von δ behandelten Ereignisse werden von der Zeit-Funktion $ta()$ ausgelöst, siehe unten.

Die Funktion h_j berechnet den neuen Wert für eine Zustandsvariable x_i im Zeitschritt t zu:

$$x_i^{(t)} = h_j(\overrightarrow{u}^{(t)}, \overrightarrow{x}^{(t-1)}, \overrightarrow{y}^{(t-1)}). \tag{3.13}$$

Zur Berechnung von $x_i^{(t)}$ kann h_j auf die aktuellen Eingangsgrößen $\overrightarrow{u}^{(t)}$ und auf die bei Bearbeitung des letzten Ereignisses im Zeitschritt $t-1$ berechneten Ausgangsgrößen $\overrightarrow{y}^{(t-1)}$

und Zustandsvariablen $\overrightarrow{x}^{(t-1)}$ zugreifen. h_j ist zunächst eine beliebige mathematische Funktion.

$\overrightarrow{h}$ besteht soweit aus mathematischen Funktionen, die den Zusammenhang zwischen einer (oder mehrerer) Größen des Funktionsblockmodells und den Zustandsvariablen $\overrightarrow{x}^{(t)}$ beschreiben. Diese Funktionen sind die intrinsischen Funktionen des Funktionsblockmodells, im folgenden werden sie auch als *Parameterfunktionen* bezeichnet. Sie enthalten u.a. alle charakteristischen Eigenschaften des Funktionsblockmodells, die für die Simulation wichtig sind. Kapitel 4 stellt dar, wie die Parameterfunktionen gewonnen werden.

3.6 Ausgangsfunktion

Die Ausgangsfunktion legt fest, wie die Ausgangsgrößen bei Eintritt eines Ereignisses neu berechnet und gesetzt werden.

$$\lambda(\overrightarrow{x}) = \overrightarrow{y} \tag{3.14}$$

Die Ausgangsgrößen werden in Abhängigkeit der Zustandsvariablen $\overrightarrow{x}$ wie im folgenden beschrieben berechnet.

Die Eingangsgrößen des Systems sind zeitkontinuierlich. Sie werden vom Simulator in Zeitschritten $\Delta t^{(n)}$ abgetastet, die nicht äquidistant sein müssen. Zu genau diesen Zeitpunkten werden die Ausgangsgrößen durch die Ausgangsfunktion auf neue Werte gesetzt. Im Zeitschritt n kann die Ausgangsveränderung $\Delta\overrightarrow{y}^{(n)}$ ausgehend von $\Delta t^{(n)} = t^{(n)} - t^{(n-1)}$ berechnet werden mit:

$$\Delta\overrightarrow{y}^{(n)} = \overrightarrow{y}^{(n)} - \overrightarrow{y}^{(n-1)}. \tag{3.15}$$

Im Zeitschritt $t^{(n)}$ wird zur Berechnung der neuen Werte für die Ausgangsgrößen die letzte Belegung des Ausgangs $\overrightarrow{y}^{(n-1)}$ benötigt, die um $\Delta\overrightarrow{y}^{(n)}$ inkrementiert wird. So ergibt sich eine rekursive Definition des Ausgangssignals zu

$$\overrightarrow{y}^{(n)} = \overrightarrow{y}^{(n-1)} + \Delta\overrightarrow{y}^{(n)} \tag{3.16}$$

mit der Rekursionsverankerung

$$\overrightarrow{y}^{(0)} = DC(\overrightarrow{u}^{(0)}). \tag{3.17}$$

Der Startwert $\overrightarrow{y}^{(0)}$ in (3.17) wird aus der DC-Analyse des Systems gewonnen, er entspricht dem Operationspunkt (2.4), der zu Beginn einer Transientenanalyse vom Simulator berechnet wird. Die in (3.17) benutzte Funktion DC kann aus einer Tabelle mit den Ergebnissen der DC-Simulation des Systems über den gesamten Definitionsbereich gewonnen werden. Sie ist direkt vom Wert der Eingangsgröße abhängig, somit ist auch die in der

Signatur (3.14) enthaltene Abhängigkeit der Ausgangsfunktion von den Eingangsgrößen gegeben.

Die Formulierung (3.16) entspricht einer stückweisen linearen Beschreibung des Ausgangssignals mit der variablen Schrittweite $\Delta t^{(n)}$ zwischen den Zeitpunkten $t^{(n-1)}$ und $t^{(n)}$.

In (3.16) läßt sich $\Delta\overrightarrow{y}^{(n)}$ auf Grund einer Reihenentwicklung ersetzen durch $\Delta t^{(n)} * f(\overrightarrow{y}, t^{(n)})$; hierin ist $f(\overrightarrow{y}, t^{(n)})$ die Funktion für die zeitliche Veränderung im Funktionsblockmodell gemäß (3.9), welche im Abschnitt „Funktion für die zeitliche Veränderung" in Abschnitt 3.8 definiert wird. Das Ergebnis ist eine analytische Beschreibung des Ausgangssignals:

$$\overrightarrow{y}^{(n)} = \overrightarrow{y}^{(n-1)} + \Delta t^{(n)} * f(\overrightarrow{y}, t)|_{t=t^{(n)}} \tag{3.18}$$

Die Rekursion ist gemäß Gleichung (3.17) verankert. f beschreibt die zeitliche Veränderung der Ausgangsgrößen mit Differentialgleichungen und wird im Kapitel 3.8 ausführlich dargestellt. Abbildung 14 illustriert, wie das Ausgangssignal gemäß Gleichung (3.18) schrittweise zusammengesetzt wird.

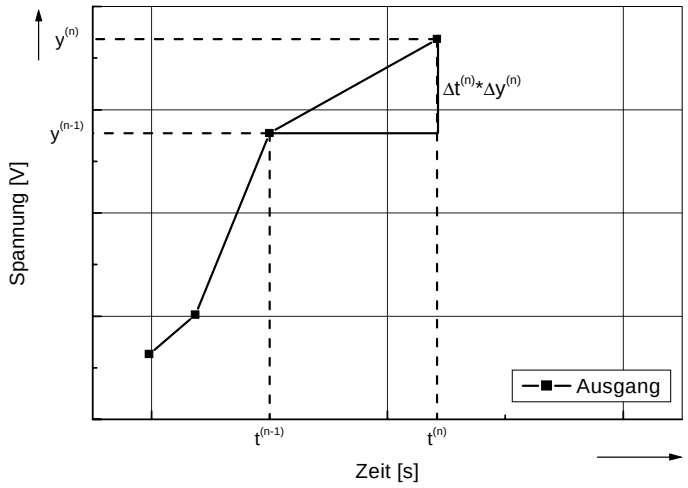

Abbildung 14: Das Ausgangssignal wird schrittweise konstruiert.

3.7 Zeit-Funktion

Die Zeit-Funktion ta des Funktionsblockmodells ist die Vereinigungsmenge der beiden Zeit-Funktionen ta_{intern} und ta_{extern}:

$$ta() = ta_{intern} \cup ta_{extern} \tag{3.19}$$

ta_{intern} wird vom Simulator gesteuert und legt fest, wann das Modell ausgewertet wird, das heißt, zu welchen Zeitpunkten Ereignisse eintreten. Die von ta_{intern} gesetzten Ereignisse werden im folgenden als **interne Ereignisse** bezeichnet. Die Integrationsschritte sind von den Veränderungen der zeitkontinuierlichen Eingangsgrößen abhängig. Sie sind nicht äquidistant und können nicht vorhergesagt werden. Dazu kommt die Zeit-Funktion ta_{extern}, welche die Zeitereignisse des Verhaltensmodells berechnet, die sogenannten **externen Ereignisse**, siehe den Abschnitt 3.7.2. Zur Implementierung dieser diskreten Ereignisse stehen in modernen HDL die entsprechenden Konstrukte bereit. So invertiert der folgende VHDL-AMS Auszug für das auf Seite 42 eingeführte Modell des springenden Balles die Geschwindigkeit nach Gleichung (3.8), sobald der Weg (die Position des Balles) nicht positiv ist:

```
-- announce discontinuity and reset velocity value
break v => -v when not s'above( 0.0);
```

Externe Ereignisse ziehen immer ein internes Ereignis nach sich, d.h. wenn ein Zustand einen neuen Wert annimmt, wird das gesamte System ausgewertet.

3.7.1 Zeit-Funktion für interne Ereignisse

Der Netzwerksimulator wertet das Funktionsblockmodell zu diskreten Zeitpunkten aus. Der Abstand dieser Zeitereignisse ist nicht äquidistant, d.h. die Zeitereignisse werden dynamisch während der Simulation festgelegt und können nicht vorausgesagt werden. Die Analyse im Zeitbereich ist kontinuierlich in dem Sinne, daß der Simulator gemäß des gewählten Integrationsverfahrens während der Simulation selbständig die zur Einhaltung der geforderten Genauigkeit hinreichende Schrittweite wählt. Zu diesen Zeitschritten wird das Modell ausgewertet, d.h. die Eingangsgrößen werden abgetastet und ihre Veränderung wird übernommen, die Zustandsvariablen werden neu berechnet und die aktuelle Belegung der Ausgangsgrößen wird festgelegt. Diese Berechnungen erfolgen mittels mathematischer Funktionen. Die Zeit-Funktion ta_{intern} legt zu jedem Zeitpunkt t fest, zu welchem diskreten Zeitpunkt t_0 das Integrationsverfahren die nächste Auswertung des Modelles vornimmt.

```
-- Integrationsschritt des Simulators
when  (event_on(now)) {
      -- Abtasten des Eingangs
      VD = difference_voltage(V_in);
      -- Übergangsfunktion für Zustandsvariable
      state_var_neu = sin( VD ) + 5.0;
      -- Ausgangsfunktion
      V_out= state_var;
      }
```

Obiger Pseudocode zeigt den Teil eines Funktionsblockmodells, welcher die internen Ereignisse abarbeitet. Die Systemvariable des Simulators `now` enthält die aktuelle Simulationszeit. Tritt eine Änderung ihres Wertes ein, so befindet sich das System in einem

neuen Zeitschritt, ein internes Ereignis ist eingetreten. Im Code wird zunächst die effektive Eingangsdifferenz `VD` für den aktuellen Zeitschritt berechnet. Anhand dieses Wertes wird die Zustandsvariable `state_var_neu` exemplarisch mittels einer Sinusfunktion und einem Offset berechnet. Anschließend wird der neue Wert der Ausgangsgröße `V_out` mittels der Zustandsvariable `state_var`, welche noch nicht auf den neu berechneten Wert aktualisiert wurde, gesetzt. Das Aktualisieren der Zustandsvariable wird im nächsten Abschnitt mittels einem externen Ereignis erfolgen.

3.7.2 Zeit-Funktion für externe Ereignisse

In der Regel vergeht eine bestimmte Zeit, bis sich ein Ereignis, d.h. eine Änderung an einer Eingangsgröße, an den Ausgangsgrößen des Systems bemerkbar macht. Der Funktionsblock gibt die Änderung der Ausgangsgrößen verzögert wieder. Die Zeit-Funktion ta_{extern} des Funktionsblockmodells ermöglicht es, festzulegen, wann eine Änderung des Eingangs sich auf den Ausgang auswirkt. Die Ausgangsgrößen werden nach Gleichung (3.14) in Abhängigkeit der Zustandsvariablen berechnet. In jedem Zeitschritt t, in dem das Modell ausgewertet wird, berechnet die Übergangsfunktion δ nach Gleichung (3.13) neue Werte für die Zustandsvariablen $\overrightarrow{x}$. Um die Verzögerung der Ausgangssignale korrekt im Funktionsblockmodell zu berücksichtigen, können die Zustandsvariablen $\overrightarrow{x}$ zeitverzögert auf neue Werte gesetzt werden. Durch dieses verzögerte Aktualisieren der Zustandsvariablen wird von der Ausgangsfunktion die alte Belegung zur Berechnung benutzt, die Änderung von $\overrightarrow{u}$ zeigt im aktuellen Zeitschritt keine Auswirkung in $\overrightarrow{y}$.

Hat die Verzögerung im Zeitschritt $t^{(n)}$ beispielsweise den konstanten Wert TD und wurde $\overrightarrow{x}^{(n)}$ durch δ neu berechnet, so wird zum Zeitpunkt $t^{(n+1)} = t^{(n)} + TD$ ein externes Ereignis gesetzt. Ab diesem Ereignis werden die neuen Werte von $\overrightarrow{x}^{(n)}$ zur Berechnung des Ausgangs benutzt. TD ist nicht notwendigerweise konstant, sondern kann als Zustandsgröße in das Modell eingehen. Gilt $TD = x_i|_{i=1,\ldots,o}$ und somit $TD \in \overrightarrow{x}$, dann wird die Zustandsgröße TD indirekt von der Übergangsfunktion δ in Abhängigkeit von $\overrightarrow{u}$ berechnet. So wird sichergestellt, daß große Veränderungen länger verzögert werden als kleine Veränderungen.

In der Logiksimulation springt man direkt zum nächsten Ereignis und wertet es aus. Dieses Vorgehen ist hier nicht möglich, denn zwischen $t^{(n)}$ und $t^{(n)} + TD$ können noch Zeitschritte des Simulators liegen, in welchen das System auf Grund von zum Beispiel einer vorgegebenen maximalen Schrittweite ausgewertet werden muß (und zu denen noch nicht die neue Belegung der Zustandsvariablen gilt). Abbildung 15 zeigt exemplarisch, daß zwischen dem Ereignis am Eingangssignal und der Wirkung am Ausgang einige interne Ereignisse, die Integrationsschritte des Simulators, liegen können.

Der folgende Codeauszug verdeutlicht das Setzen eines externen Ereignisses. Der Zeitpunkt für das Ereignis wird durch den aktuellen Zeitschritt `now` plus die Verzögerungszeit `TD` festgelegt. Zu dem Ereignis wird die Zustandsvariable `state_var` auf den neuen Wert `state_var_neu` gesetzt.

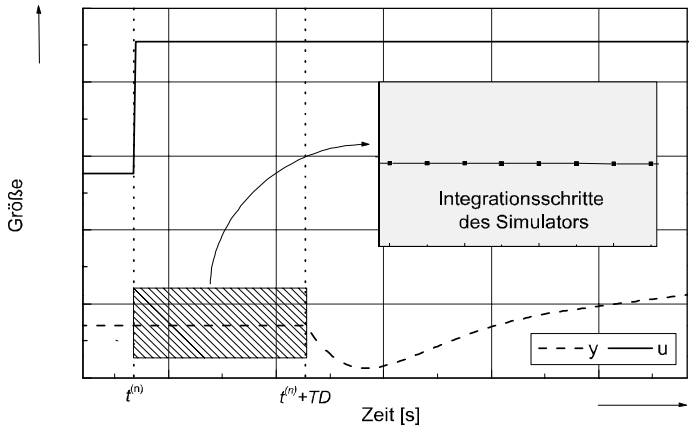

Abbildung 15: Integrationsschritte vor Eintritt des externen Ereignisses.

```
-- Externes Ereignis setzen
 schedule_event(now+TD, state_var, state_var_neu);
```

3.7.3 Die Ereignisstruktur

Das Funktionsblockmodell Konzept enthält durch ta_{intern} und ta_{extern} eine zweistufige Ereignisstruktur. Analog zur Vorgehensweise bei der Logiksimulation wird im folgenden Abschnitt ein Verfahren zum Zusammenfassen der Ereignisse und zum Behandeln gleichzeitiger Ereignisse definiert.

Ausgangspunkt ist das erste interne Ereignis. Wie oben beschrieben werden die Veränderungen an den Eingangsgrößen erfaßt. Die - nach Ablauf der Verzögerungszeit - geltenden neuen Werte der Zustandsvariablen werden errechnet. Die Verzögerungszeit ist entweder bekannt oder wurde als Zustandsvariable mitberechnet. Jetzt ist bekannt, wann eine Änderung stattfindet und welche neuen Werte gesetzt werden, ta_{extern} setzt das externe Ereignis. Abbildung 16 illustriert die zweistufige Ereignisstruktur. Die interne Ereignisschlange enthält die von ta_{intern} gesetzten Ereignisse, für die externen Ereignisse existiert die externe Ereignisschlange.

Ein externes Ereignis kann mit einer Unterbrechung (Interrupt) auf der Hardwareseite verglichen werden. Es wird die Ausführung der Übergangsfunktion für Ereignisse erzwungen, welche die Werte der Zustandsvariablen neu berechnet und somit ein internes Ereignis erzwingt. Der folgende Code zeigt, daß das Abarbeiten der externen Ereignisse keinen Aufwand erfordert. Der Simulator setzt automatisch die Zustandsvariablen auf neue Werte (die Werte, die im vorherigen Abschnitt beim Setzen des Ereignisses übergeben wurden). Das Auslösen eines internen Ereignisses zur selben Zeit bewirkt, daß alle

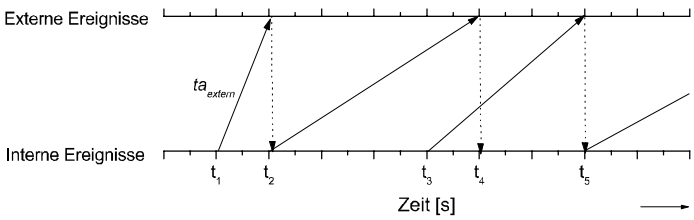

Abbildung 16: Interne und externe Ereignisschlange.

Zustandsvariablen und die Ausgangsgrößen aktualisiert werden, und damit auch ein neues externes Ereignis gesetzt wird.

```
-- Externes Ereignis abarbeiten
when event_on(state_var)) {
    -- state_var wird vom Simulator aktualisiert
    -- Erzwinge einen Integrationsschritt
    schedule_internal_event(now);
    }
```

3.7.4 Überlagerung der Ereignisse

Die Auswertung der externen Ereignisse unterliegt einer bedingten Priorität. Bei einer endlichen Anstiegszeit bewirkt ein großer Sprung der Eingangsgrößen eine länger währende Antwort am Ausgang als ein Sprung von einer geringeren Größe. Dies wird durch die in der Regel begrenzte Flankensteil des Ausgangs verursacht. Ebenso wirkt sich, bedingt durch die nicht notwendigerweise konstante Verzögerung des Funktionsblocks, ein großer Eingangssprung erst nach einer längeren Zeit am Ausgang aus als ein Sprung einer kleineren Amplitude. Fallen beispielsweise zwei externe Ereignis "Veränderung von `state_var` auf 600000" und "Veränderung von `state_var` auf 10" auf die Zeitpunkte t_1 und t_2, welche beide in einem bestimmten kleinen Intervall $\epsilon \geq 0$ liegen, so muß entschieden werden, welchen Wert die Zustandsvariable `state_var` annehmen soll. Abbildung 17 illustriert das Szenario: im Zeitschritt $t = 1$ ändert sich das Eingangssignal stark, ein externes Ereignis wird bei $t = 5$ gesetzt, es soll `state_var`$= 600000$ gelten. Bevor $t = 5$ gilt, tritt ein Integrationsschritt des Simulators ein: Bei $t = 3$. Da zu diesem Zeitpunkt keine Änderung des Eingangs auftritt, wird festgelegt, daß `state_var` ab dem Zeitschritt $t = 5.5$ wieder den Wert `state_var`$= 10$ annehmen soll. Diese Änderung darf aber erst dann aktiv werden, wenn die bei $t = 1$ verursachte Änderung am Ausgang vollzogen wurde. Bei $t = 5.5$ müssen die beiden Ereignisse `state_var`$= 600000$ und `state_var`$= 10$ so überlagert werden, daß der Eingangssprung korrekt vollzogen wird.

Für die Überlagerung der beiden Ereignisse gelten die folgenden Überlegungen:

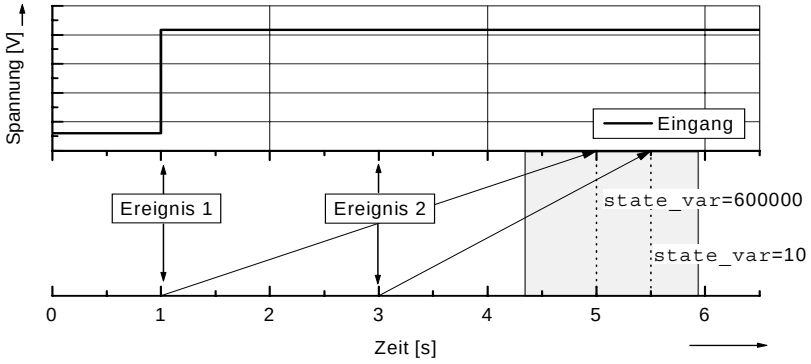

Abbildung 17: Überlagerung von zwei Ereignissen.

- Je größer der neue Wert, desto höher ist die Priorität des Ereignisses, denn es existiert eine proportionale Abhängigkeit zwischen der Größe des verursachenden Eingangssprunges und der resultierenden Zustandsvariable. Dieses Wissen wird für die Definition der Überlagerungsfunktion verwandt, sodaß gilt:

$$\texttt{state\_var} = \max(\texttt{state\_var}_1, \texttt{state\_var}_2).$$

Auf diese Weise ist sichergestellt, daß bedeutende Änderungen immer vollzogen werden.

- Die Zustandsvariablen erlauben es, einen Sollwert für Ausgangsgrößen zu definieren. Bei Eintritt eines Ereignisses wird dieser Sollwert mittels der Übergangsfunktion δ berechnet. Durch Vergleich von der aktuellen Ausgangsgröße mit ihrem Sollwert kann festgestellt werden, inwieweit vergangene Ereignisse schon abgeschlossen wurden. Im Abschnitt 3.8.3 wird das Verfahren diskutiert.

Die Zeit-Funktion kombiniert ta_{intern} und ta_{extern}. Sie ermöglicht so die für die zeitkontinuierliche Simulation notwendige Reaktion auf beliebige, vom Simulator frei zu setzende Zeitpunkte, zu welchen das Modell ausgewertet wird, und eine Vorwärts-Definition von Zeitpunkten, zu denen Änderungen der Zustandsgrößen - vom Modell gesteuert - auftreten. Für jede Zustandsgröße läßt sich diese Änderung individuell setzen.

3.8 Funktion für die zeitliche Veränderung

Die Funktion für die zeitliche Veränderung beschreibt die zeitliche Veränderung der Ausgangsgrößen im Transientenbereich mittels Differentialgleichungen. Sie ist definiert als Ableitung des Ausgangs nach der Zeit:

$$f(\overrightarrow{y}, t) = \frac{d\overrightarrow{y}}{dt} \tag{3.20}$$

f wird in Gleichung (3.18) zur Berechnung der Ausgangsfunktion benutzt. (3.20) kann in (3.18) eingesetzt werden, wodurch sich die neue Rekursionsgleichung für den Ausgang ergibt zu:

$$\overrightarrow{y}^{(n)} = \overrightarrow{y}^{(n-1)} + \Delta t^{(n)} * \frac{d\overrightarrow{y}}{dt}|_{t=t^{(n)}} \tag{3.21}$$

Im Beispiel springender Ball ist die Geschwindigkeit v als die Ableitung des Weges s nach der Zeit t definiert, siehe Gleichung (3.8). Der entsprechende Einsatz im Verhaltensmodell lautet:

```
v'dot == -G - v** 2* Air_ Res;
```

Damit das Funktionsblockmodell erfolgreich eingesetzt werden kann, muß $f(\overrightarrow{y}, t)$ bekannt sein, denn sonst kann der Ausgang nach Gleichung (3.21) nicht berechnet werden. Zum Berechnen von $f(\overrightarrow{y}, t)$ nach (3.20) wird die Ableitung des Ausgangs nach der Zeit benötigt. Diese Ableitung kann nicht berechnet werden, denn der Ausgang im aktuellen Zeitschritt ist ja noch nicht bekannt. Deshalb wird ein Verfahren benötigt, das die zeitliche Änderung in Gleichung (3.21) berechnet.

3.8.1 Berechnung der zeitlichen Veränderung

Der neue Ansatz zur Verhaltensmodellierung verwendet die Funktion für die zeitliche Änderung der Ausgänge, wobei die Änderung nicht durch das Aufstellen der DAE der Schaltung, sondern durch die Modellierung der Flankensteilheit der Schaltung gewonnen wird.

Die Klasse der durch ein Funktionsblockmodell darstellbaren Systeme ist gekennzeichnet durch eine obere Begrenzung der zeitlichen Veränderung der Ausgangswerte bei der Sprungantwort. Für diese Systeme kann beobachtet werden:

Bei Eintritt eines Ereignisses werden die Eingangsgrößen $\overrightarrow{u}$ abgetastet. Analog zu Gleichung (3.15) kann $\Delta\overrightarrow{u}^{(n)}$ in jedem Zeitschritt berechnet werden zu:

$$\Delta\overrightarrow{u}^{(n)} = \overrightarrow{u}^{(n)} - \overrightarrow{u}^{(n-1)}. \tag{3.22}$$

Abbildung 18 zeigt graphisch die Bestimmung von $\Delta\overrightarrow{u}^{(n)}$. In einem konkreten Zeitschritt $\Delta t^{(n)}$ ist die Form des Eingangssignals nicht von Bedeutung, benötigt wird lediglich der letzte Wert des Eingangs $\overrightarrow{u}^{(n-1)}$, um $\Delta\overrightarrow{u}^{(n)}$ zu berechnen. Wird ein Spannungssprung der Größe $\Delta\overrightarrow{u}^{(n)}$ auf einen Eingang des Systems gelegt, so vollzieht der Ausgang diesen Sprung nur verlangsamt nach. Die zeitliche Änderung des Ausgangssignals erfolgt nicht

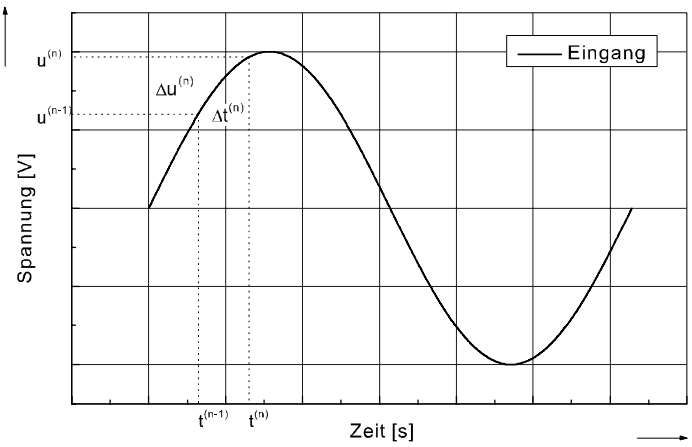

Abbildung 18: Das Eingangssignal wird zu diskreten Zeitpunkten abgetastet.

als Sprung, sondern als stetiger Übergang mit einer Steigung, deren Größe nach oben begrenzt ist.

Diese Steigung kann anhand einer Transientensimulation des Systems berechnet werden, wie in Abbildung 19 dargestellt wird.

Die zeitliche Änderung $\frac{d\overrightarrow{y}}{dt}|_{t=t^{(n)}}$ in (3.21) ist somit von den Eingangsgrößen $\overrightarrow{u}$ abhängig, denn die Größe des Spannungssprunges $\Delta\overrightarrow{u}^{(n)}$ korreliert mit der Flankensteilheit des Ausgangs. Die Flankensteilheit wird als Funktion $\widehat{f}(\overrightarrow{u}, \overrightarrow{y}, t)$ um die Abhängigkeit von den Eingangsgrößen erweitert. Der Wert für $\widehat{f}$ kann durch eine Transientensimulation wie oben dargestellt berechnet werden, oder in Tabellen hinterlegt werden. Die Gleichung (3.18) wird jetzt um einen direkten Zusammenhang zwischen den Änderungen der Eingangsvariablen $\overrightarrow{u}$ und der Belegung der Ausgangsvariablen $\overrightarrow{y}$ erweitert werden zu:

$$\overrightarrow{y}^{(n)} = \overrightarrow{y}^{(n-1)} + \Delta t^{(n)} * \widehat{f}(\overrightarrow{u}, \overrightarrow{y}, t)|_{t=t^{(n)}} \tag{3.23}$$

Formel (3.23) läßt außer Acht, daß eine Veränderung am Eingang sich unterschiedlich lange auf den Ausgang auswirkt. Ein kleiner Spannungssprung am Eingang bewirkt einen zeitlich kurzen Sprung am Ausgang, während ein sehr großer Sprung einen entsprechend längere Wirkung hat. Abbildung 19 zeigt diese Wirkung für einen Sprung der Eingangsgröße. Es ist offensichtlich, daß die Dauer durch die Slewrate bestimmt wird, also durch die Begrenzung der zeitlichen Änderung des Ausgangssignals. Die Wirkung hält genauso lange an, bis sich die Schaltung wieder in einem stabilen Zustand befindet, das heißt, bis alle Einschwingvorgänge abgeklungen sind. Für die Modelle bedeutet dies, daß ein Verfahren notwendig ist, mit welchem festgestellt werden kann, wann die Wirkung einer Veränderung der Eingangsvariablen vollzogen ist. Zudem verändert sich das Eingangssignal dynamisch, somit überlagern sich die Wirkungen: Eine ablaufende Wertentwicklung des Ausgangs

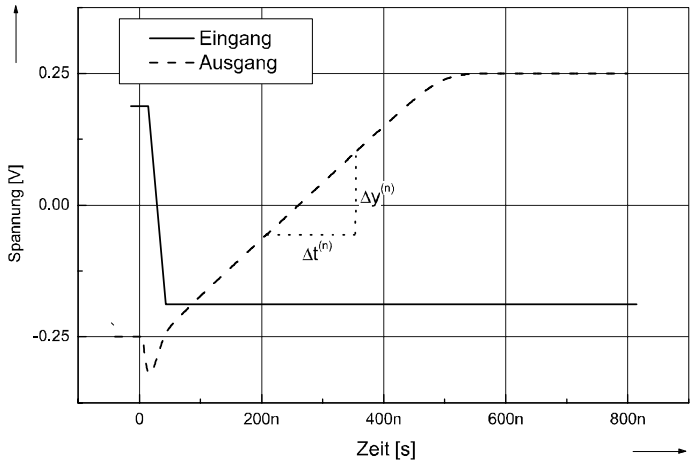

Abbildung 19: Ein Sprung am Eingang wirkt für eine längere Zeit auf den Ausgang ein.

kann durch den weiteren Verlauf des Eingangssignals gedämpft oder verstärkt werden.

Die folgenden Abschnitte stellen das im Funktionsblockmodell eingesetzte Verfahren vor. Dazu wird zunächst die Klasse der rückgekoppelten Systeme betrachtet. Ausgehend von den Auswirkungen, die diese Rückkopplung in der Schaltung hat, wird dann die Signatur für die zeitliche Veränderung erweitert, so daß die Methodik von der Klasse der rückgekoppelten Schaltung auf allgemeine Schaltungen erweitert wird.

3.8.2 Rückgekoppelte Schaltungen

Der Operationsverstärker ist ein Beispiel für ein dynamisches System, welches in vielen Grundschaltungen rückgekoppelt eingesetzt wird. Die Standardbeschaltungen als Spannungsfolger oder als Integrator führen den Ausgang an einen der zwei Eingangspins zurück, je nachdem ob die Stufe invertieren soll oder nicht.

Abbildung 20 zeigt die typische Beschaltung eines Operationsverstärkers als nichtinvertierende Verstärkerstufe. Der Operationsverstärker hat zwei Eingangspins V_{in+} und V_{in-} und einen Ausgangspin V_{out}. Der Ausgangspin ist mit V_{in-} verbunden, das heißt der Operationsverstärker ist rückgekoppelt beschaltet. Die an den Pins anliegenden Eingangsgrößen werden mit u_+ und u_- bezeichnet, am Ausgang liegt y an. Die Differenz zwischen den beiden Eingangspins sei mit $\Delta u = u_+ - u_-$ bezeichnet.

Bei der Betrachtung der Meßschaltung zu einem Zeitschritt $t^{(n)}$ ergibt sich folgender Zusammenhang:

Die Eingangsgröße u_+ springt von $u_+^{(n-1)}$ auf den Wert $u_+^{(n)}$. Damit beträgt $\Delta u_+^{(n)} =$

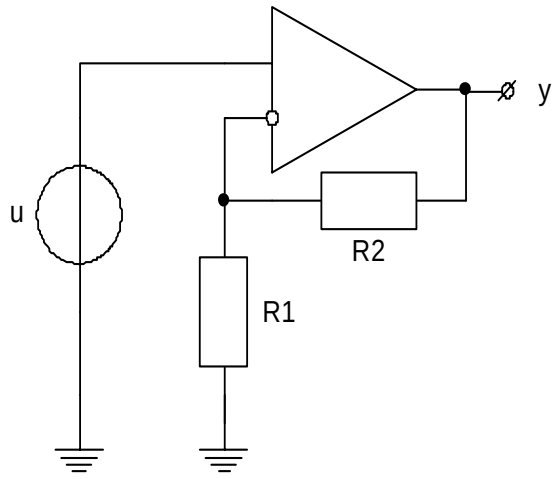

Abbildung 20: Nichtinvertierte Operationsverstärker Testschaltung.

$\left|u_+^{(n)} - u_+^{(n-1)}\right|$. Zum Zeitpunkt $t^{(n-1)}$ lag $y^{(n-1)}$ am Ausgang an. Durch die Rückkopplung gilt $u_-^{(n)} = y^{(n-1)}$. Die Eingangsspannungsdifferenz , die auf den Operationsverstärker im Zeitschritt $t^{(n)}$ einwirkt, ist $\Delta u^{(n)} = u_+^{(n)} - y^{(n-1)}$. Das Ausgangssignal ist einen Schritt zurück - bedingt durch die Verzögerungszeit des Bauteils. Nach genügend langer Zeit sind alle Einschwingvorgänge abgeklungen und es gilt $u_+^{(n)} - y^{(n-1)} = u_+^{(n)} - y^{(n)} = 0$ bzw. $y^{(n)} = A_\nu * \Delta u^{(n)}$. Die Schaltung befindet sich in einem stabilen Zustand. Abbildung 21 zeigt die Abläufe der Spannungsgrößen während des Einschwingens der Schaltung.

Für die Methodik der Modellierung ist wichtig, daß zu jedem Zeitpunkt die Eingangsspannungsdifferenz $\Delta u^{(n)}$ betrachtet wird, in welche die Ausgangsgröße durch die Rückkopplung einfließt. In Abhängigkeit von $\Delta u^{(n)}$ wird die Funktion für die zeitliche Veränderung und damit das Ausgangssignal nach (3.23) berechnet. Auf diese Art ist garantiert, daß die Wirkung des Spannungssprunges korrekt wiedergegeben wird, denn die zeitliche Änderung wird erst dann 0, wenn der Ausgang den Sollwert erreicht hat und $\Delta u^{(n)}$ somit 0 ist. Die Methodik läßt sich also uneingeschränkt auf die Schaltungsklasse der rückgekoppelten Schaltungen anwenden.

Wird die Verbindung von Ausgang zu Eingang in Abbildung 20 getrennt, so gilt $\Delta u = u_+$. Der Effekt daraus ist, daß die Schaltung in die Sättigung gefahren wird und die Simulation kein korrektes Ergebnis liefert. Die Beschaltung des Operationsverstärkers als Komparator ist ein Beispiel für eine nicht rückgekoppelte Operationsverstärkerschaltung.

Im folgenden Abschnitt wird die Signatur von $f(\overrightarrow{y}, t)$ um einige Definitionen erweitert, sodaß sie unabhängig von der Beschaltung funktioniert - und damit ein Funktionsblockmodell in beliebigen Schaltungen eingesetzt werden kann.

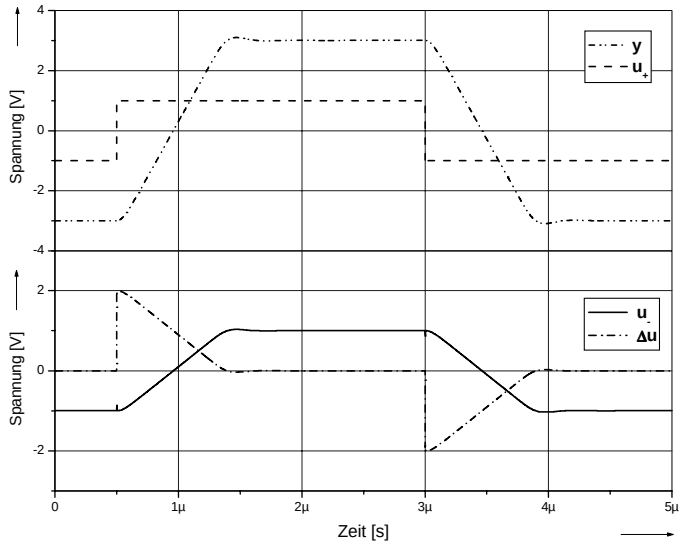

Abbildung 21: Sprungantwort und effektive Eingangsspannungsdifferenz der Testschaltung.

3.8.3 Allgemeine Schaltungen

In einem allgemeinen - nicht rückgekoppelt beschaltetem - System ist eine Definition der Differenzspannung in Abhängigkeit von Ein- und Ausgangsgrößen analog zum vorherigen Abschnitt nicht möglich. Die Werte der Ausgangsgrößen fließen nicht in die Modellierungsmethodik ein, denn die zeitliche Änderung wird wie oben dargestellt, anhand der Differenz der Eingangsgrößen $\Delta\overrightarrow{u}^{(n)}$ im letzten Zeitschritt definiert.

Bei der Modellierung mit $\Delta\overrightarrow{u}^{(n)}$ treten folgende Schwierigkeiten auf:

- Damit sich das System in einem stabilen Zustand befindet, in dem alle Einschwingvorgänge abgeschlossen sind, muß die Funktion für die zeitliche Veränderung den Wert 0 annehmen, damit die Ausgangsgrößen konstant bleiben. Nur so wird das Inkrement in Gleichung (3.23) gleich 0. Dies ist in einem stabilen System gegeben, wenn sich die Eingangsgrößen nicht ändern und damit kein Eingangsspannungssprung auftritt.

- Die Bedingung, daß sich die Eingangsgrößen nicht ändern, ist nicht hinreichend, denn es kann nicht festgestellt werden, ob Einschwingvorgänge, die auf Veränderungen in der Vergangenheit beruhen, vollzogen wurden. Eine Änderung einer Eingangsgröße wirkt länger als der eine Zeitschritt, in dem sie auftrat.

- Wird $\Delta\overrightarrow{u}^{(n)} = \overrightarrow{u}^{(n)}$ gesetzt, so wird das Modell bei einem Sprung in die Sätti-

gung gefahren, denn $\overrightarrow{u}_i$ nimmt nicht den Wert 0 an, welcher anzeigt, daß sich die Schaltung in einem eingeschwungenen stabilen Zustand befindet.

- Gilt $\Delta\overrightarrow{u}^{(n)} = \overrightarrow{u}^{(n)} - \overrightarrow{u}^{(n-1)}$, so wird die Auswirkung des Sprunges nicht korrekt wiedergegeben, da sie nur in einem Zeitschritt n auftritt und $\Delta\overrightarrow{u}^{(n+1)}$ den Wert 0 annimmt, falls $\overrightarrow{u}^{(n+1)} = \overrightarrow{u}^{(n)}$ gilt. Es fehlt ein Mechanismus, der festestellen kann, ob die Wirkung von $\Delta\overrightarrow{u}^{(n)}$ vollständig am Ausgang vollzogen worden ist.
- Gilt $\Delta\overrightarrow{u}^{(n)} = \overrightarrow{u}^{(n)} - \overrightarrow{y}^{(n-1)}$ (analog zum Beispiel Operationsverstärker im vorigen Abschnitt), so ergibt sich nicht notwendigerweise das korrekte Ergebnis, weil im eingeschwungenen Zustand nicht automatisch $\overrightarrow{y}^{(n)} = \overrightarrow{u}^{(n)}$ gelten muß.

Aus diesen Betrachtungen heraus wird deutlich, daß eine **effektive** Eingangsdifferenzspannung $\Delta\widetilde{\overrightarrow{u}}^{(n)}$ benötigt wird. Die Belegung der Ein- und Ausgangsgrößen fließt in die Berechnung von $\Delta\widetilde{\overrightarrow{u}}^{(n)}$ ein:

$$\Delta\widetilde{\overrightarrow{u}}^{(n)} = g(\overrightarrow{u}^{(n)}, \overrightarrow{u}^{(n-1)}, \overrightarrow{y}^{(n-1)}) \tag{3.24}$$

Mit Hilfe einer DC-Analyse der Originalschaltung kann ein Wert $\widetilde{\overrightarrow{y}}^{(n)}$ berechnet werden, den der Ausgang für einen diskreten Eingang $\overrightarrow{u}^{(n)}$ annimmt, sobald alle Einschwingvorgänge abgeklungen sind. Dieser Wert kann als der Sollwert für die Ausgangsgrößen betrachtet werden, er wird mittels der DC-Analyse in jedem Zeitschritt für die aktuelle Belegung der Eingangsgrößen berechnet. Dieser Sollwert kann mit den tatsächlichen im Modell errechneten Ausgangsgrößen des letzten Zeitschrittes verglichen werden:

$$\Delta\widetilde{\overrightarrow{y}}^{(n)} = \widetilde{\overrightarrow{y}}^{(n)} - \overrightarrow{y}^{(n-1)} \tag{3.25}$$

In Abbildung 21 ist $\Delta u = u_+ - u_-$ eingezeichnet, die effektiv wirkende Differenzspannung zwischen den Eingangspins des Operationsverstärkers. Ähnlich sieht der Verlauf für $\Delta\widetilde{\overrightarrow{y}}$ aus, auch hier ist der Sollwert nach einem Sprung der Eingangsgrößen größer als der für den Ausgang berechnete Wert, er nähert sich mit fortlaufender Zeit, wenn der Ausgang näher an seinen Sollwert kommt, immer weiter der Nullinie an.

Die Idee ist es jetzt, den in Gleichung (3.24) notierten funktionalen Zusammenhang um den Sollwert zu erweitern:

$$\Delta\widetilde{\overrightarrow{u}}^{(n)} = g(\overrightarrow{u}^{(n)}, \overrightarrow{u}^{(n-1)}, \overrightarrow{y}^{(n-1)}, \widetilde{\overrightarrow{y}}^{(n)}) \tag{3.26}$$

g erfordert nun eine nähere Betrachtung. Einerseits wirkt die Veränderung $\Delta\overrightarrow{u}^{(n)}$, andererseits kommt die Differenz des Ausgangs $\widetilde{\overrightarrow{y}}^{(n)} - \overrightarrow{y}^{(n-1)}$ hinzu. Diese beiden Werte müssen so verknüpft werden, daß $\Delta\widetilde{\overrightarrow{u}}^{(n)}$ korrekt wird und die zeitliche Änderung für den Zeitschritt n korrekt berechnet wird. Mit den Gleichungen (3.22) und (3.25) ergibt sich aus Gleichung (3.26):

$$\Delta\widetilde{\overrightarrow{u}}^{(n)} = \Delta\overrightarrow{u}^{(n)} \circ \Delta\widetilde{\overrightarrow{y}}^{(n)}. \tag{3.27}$$

Wird in der Verstärkerschaltung aus Abbildung 20 die Rückkopplung, d.h. die Verbindung vom Ausgang zum Eingang, getrennt, so kann sie im Funktionsblockmodell nachgebildet werden. In diesem Falle kann die Verknüpfung in (3.27) mathematisch mit Hilfe des Verstärkungsfaktors des Operationsverstärker A_ν berechnet werden zu:

$$\Delta\widetilde{\overrightarrow{u}}_i = \Delta u_+^{(n)} + \frac{\widetilde{y}^{(n)} - y^{(n-1)}}{A_\nu}.$$

Hier entspricht $\Delta\widetilde{\overrightarrow{u}}$ exakt dem Verlauf von Δu aus Abbildung 21, die Simulation der Schaltung erfolgt korrekt, obwohl die Rückkopplung getrennt ist.

Für beliebige Systeme ist die Suche nach der Verknüpfungsfunktion in (3.27) nicht so naheliegend wie beim Operationsverstärker. Eine Idee ist es, einen Faktor $k = \widetilde{\overrightarrow{y}}^{(n)} * \frac{1}{\overrightarrow{u}^{(n)}}$ zwischen dem Wert der Eingänge und der Ausgänge zu berechnen, ähnlich wie es der Verstärkungsfaktor A_ν beim Operationsverstärker tut. Dazu muß die Schaltung im stabilen Zustand betrachtet werden. Da k nicht notwendigerweise konstant ist, kann es auch als Zustandsvariable $k \in \overrightarrow{x}$ in das Funktionsblockmodell einfließen, wobei es durch δ berechnet wird. Jetzt kann die Verknüpfung allgemein mathematisch erfaßt werden in:

$$\Delta\widetilde{\overrightarrow{u}}^{(n)} = \Delta\overrightarrow{u}^{(n)} + \frac{\widetilde{\overrightarrow{y}}^{(n)} - \overrightarrow{y}^{(n-1)}}{k} \tag{3.28}$$

$\Delta\widetilde{\overrightarrow{u}}^{(n)}$ ist die effektive Differenz der Eingangsgrößen, anhand derer der Wert für die zeitliche Änderung des Ausgangs berechnet wird. Für die Signatur des Funktionsblockmodells bedeutet dies, daß die Funktion für die zeitliche Änderung um Abhängigkeiten von den Eingangsgrößen und von den Zustandsvariablen erweitert werden muß. $\widetilde{\overrightarrow{y}}^{(n)}$ ist eine Zustandsgröße, die während der Simulation berechnet wird. Somit ergibt sich:

$$\widehat{f}(\overrightarrow{u}, \overrightarrow{x}, \overrightarrow{y}, t) \simeq \frac{d\overrightarrow{y}}{dt}$$

und für den Ausgang im Zeitschritt n als Erweiterung von (3.23):

$$\overrightarrow{y}^{(n)} = \overrightarrow{y}^{(n-1)} + \Delta t^{(n)} * \widehat{f}(\overrightarrow{u}, \overrightarrow{x}, \overrightarrow{y}, t)|_{t=t^{(n)}} \tag{3.29}$$

Folgendes Listing zeigt einen Ausschnitt aus dem Modellcode in ELDO-FAS, der zur Modellierung des Operationsverstärkers in der nichtinvertierenden Verstärkerstufe in Abbildung 20 verwandt wurde:

```
-- Vout_Soll ist eine Funktion von Vin, d.h. der SOLL-WERT
make VD = (Vout_Soll - Vout_Last)
-- Verzögern des Eingangs um einen Zeitschritt
make DeltaVin = Vin - STATE.LAST_VALUE(Vin, TSTEP)
-- Ueberlagern durch Addition
make VD_eff = (VD + DeltaVin)
```

Im Listing entspricht `Vout_Soll` dem $\widetilde{\overrightarrow{y}}^{(n)}$ aus Gleichung (3.28), `Vout_Last` entspricht $\overrightarrow{y}^{(n-1)}$ und `DeltaVin` entspricht $\Delta\overrightarrow{u}^{(n)}$. `VD_eff` ist die nach (3.28) berechnete effektive Differenz der Eingangsgrößen $\Delta\widetilde{\overrightarrow{u}}^{(n)}$, wobei in diesem Beispiel $k = 1$ gilt.

3.9 System-Dynamik des Funktionsblockmodells

Bei der Formulierung der Ziele der Arbeit wurde festgelegt, daß die Funktionsblockmodelle für Simulationen auf der Verhaltensebene gemäß Abbildung 4 verwandt werden sollen. Die Verhaltensebene ist die oberste Ebene, auf der die Erhaltungssätze gelten. Damit ist gefordert, daß das Modell nicht nur Ausgänge hat und diese auf diskrete Werte setzt, sondern daß sowohl Potentiale als auch Flußgrößen am Ausgang korrekt berücksichtigt werden. Erst dadurch wird gewährleistet, daß das Funktionsblockmodell beliebig beschaltet werden kann und dennoch korrekte Ergebnisse liefert.

Die Definition des Funktionsblockmodells ermöglicht es, ein Modell je nach Anwendungsfall für den Einsatz auf der Funktionalen Ebene oder für den Einsatz auf der Verhaltensebene zu skalieren. Um ein auf der Funktionalen Ebene entworfenes Modell dementsprechend zu erweitern, muß die Menge der Zustandsvariablen um die Innenwiderstände der Schaltung erweitert werden, welche durch δ zu jedem Zeitpunkt berechnet werden. So kann sichergestellt werden, daß das Modell auf unterschiedliche Beschaltungen reagiert und korrekte Ströme und Spannungen am Ausgang bereitstellt.

Dieser Abschnitt stellt die System-Dynamik für Simulationen auf der Funktionalen Ebene dar, das heißt wie die Elemente des Funktionsblockmodells bei der Simulation zusammenarbeiten. Die Darstellung wird sodann um die notwendigen Zustandsvariablen erweitert, um es zu einem echten Modell auf der Verhaltensebene zu machen. Dabei wird speziell die Situation mit unterschiedlichen Lasten am Ausgang untersucht.

3.9.1 Simulationen auf der Funktionalen Ebene

Alle Funktionsblockmodelle enthalten eine gemeinsame minimale Modell-Struktur, die zur Beschreibung des Verhaltens nach Definition (3.9) notwendig ist. Ausgehend vom Verhalten des Modells erfolgt in diesem Abschnitt die Beschreibung des Zusammenspiels der Elemente aus Definition (3.9) im fertigen Modell.

Dazu kann das dynamische Verhalten eines Funktionsblockmodells analog zu [Pra91], Kapitel 2.2.3 beschrieben werden mit:

Zum Zeitpunkt $t^{(n)}$ befindenden sich die Zustände des Funktionsblockmodells in $\overrightarrow{x}^{(n)}$, an den Eingängen liegt $\overrightarrow{u}^{(n)}$ an und die Ausgangsgrößen sind auf $\overrightarrow{y}^{(n)}$ gesetzt. Das nächste interne Ereignis wird mittels der Zeitfunktion ta auf den Zeitpunkt $t^{(n+1)} = t^{(n)} + ta(\overrightarrow{x}^{(n)})$ gesetzt, bis dahin gelten $\overrightarrow{x}^{(n)}$. Das Modell befindet sich für $e = ta(\overrightarrow{x}^{(n)})$ Zeiteinheiten im Gesamtzustand $FktBlockModell^{(n)} = \{\overrightarrow{u}^{(n)}, \overrightarrow{y}^{(n)}, \overrightarrow{x}^{(n)}, \delta, \lambda, ta, f\}$.

Zum Zeitpunkt $t^{(n+1)}$ sind die folgenden Aktionen durchzuführen:

1. Der Eingangsvektor $\overrightarrow{u}^{(n+1)}$ wird aktualisiert (die aktuellen Werte, die am Eingang anliegen, werden in das Modell übernommen).
2. Die neue Belegung der Zustandsvariablen $\overrightarrow{x}^{(n+1)} = \delta(\overrightarrow{u}^{(n+1)}, \overrightarrow{x}^{(n)})$ wird mittels der Übergangsfunktion δ anhand des Eingangsvektors und der vorhergehenden Werte berechnet.
3. Die Funktion für die zeitliche Veränderung berechnet $f = \frac{d\overrightarrow{y}}{dt} \simeq \widehat{f}(\overrightarrow{u}, \overrightarrow{x}, \overrightarrow{y}, t)$. Damit ist die Flankensteilheit der Ausgangsgrößen bekannt.
4. Die Zeitfunktion ta setzt mittels der Verzögerungszeit (welche in Schritt 2 mittels δ berechnet wurde) ein externes Ereignis, zu dem der aktualisierte Wert für die zeitliche Veränderung f und für die Zustandsvariablen $\overrightarrow{x}^{(n+1)}$ aktiv wird.
5. Die Ausgangsgrößen werden auf den Wert $\overrightarrow{y}^{(n+1)} = \lambda(\overrightarrow{x}^{(n)})$ gesetzt. Damit liegen am Ausgang die aktualisierten Werte an.

Die Übergangsfunktion δ in Schritt 2 berechnet im Minimum den Soll-Wert des Ausgangs mittels der DC-Analyse (dies wird in Gleichung (3.25) benötigt), Verzögerungszeit zum Setzen der Ereignisse, und die Flankensteilheit des Ausgangs.

Das Beispiel lineare Filterschaltung in Kapitel 5 zeigt auf, wie ein konkretes Funktionsblockmodell aufgebaut wird und wie die Elemente zusammenspielen. Weitere Parameterfunktionen können hinzugefügt werden, um das Funktionsblockmodell für bestimmte Anwendungen zu skalieren, z.B. Abhängigkeiten von der Umgebung des Funktionsblocks wie der Temperatur.

3.9.2 Simulationen auf der Verhaltensebene

Simulationen mit dem Funktionsblockmodell auf der Verhaltensebene erfordern, daß an den Pins des Modells die Erhaltungssätze gelten. Im vorigen Abschnitt wurden ausschließlich Aussagen über die an den Ausgangspins anliegende Spannung getroffen, der durch die Pins fließende Strom ist außen vor geblieben. In diesem Abschnitt wird das Funktionsblockmodell zu einem echten Verhaltensmodell erweitert. Das Vorgehen hierzu wird anhand eines Operationsverstärkers illustriert. Das Funktionsblockmodell auf der Funktionalen Ebene entspricht einem idealen Operationsverstärker, während das Funktionsblockmodell auf der Verhaltensebene einem realen Operationsverstärker entspricht.

Das Verhaltensmodell eines nichtlinearen Elementes muß korrekt sein für eine beliebige Lastsituation am Ausgang, sonst kann ein vorab erzeugtes Modell nicht in beliebigen Schaltungen eingesetzt werden. Der am Ausgang anliegende Lastwiderstand R_L liegt grundsätzlich im Bereich zwischen dem Kurzschluß und dem Leerlauf, das heißt $R_L \in \{0, \ldots, \infty\}$. Für jede Lastsituation ergibt sich ein konkretes Strom-Spannungsdiagramm, die Kurve beginnt bei $V_{out} = 0$ - also kurzgeschlossenen Ausgang - und endet bei $I_{out} = 0$ - also im Leerlauf. Bei der zeitdiskreten Behandlung dynamischer Elemente mittels der Rückwärts Euler oder Trapez Integration wird ein lineares Ersatzschaltbild erzeugt und der Zusammenhang zwischen Spannung $U^{(n)}$ und Strom $I^{(n-1)}$ im Zeitpunkt $t^{(n)}$ mit der Iterationsformel $I^{(n-1)} = Y^{(n-1)} * U^{(n)}$ bestimmt. Abbildung 22 zeigt ein mögliches Strom-Spannungsdiagramm mit einem ausgezeichneten Arbeitspunkt. Dargelegt wird der kontinuierliche Übergang zwischen einer Spannungs- und einer Stromquelle mit jeweils nahezu konstanten Werten.

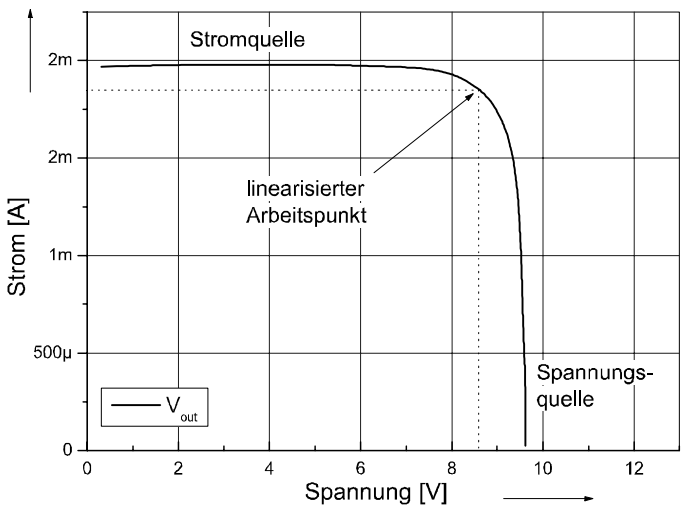

Abbildung 22: Strom-Spannungskennlinie eines nichtlinearen Bauelementes.

Das ideale Ersatzschaltbild eines Operationsverstärkers modelliert die Ausgangsstufe durch eine Spannungsquelle. Es entspricht direkt einem Funktionsblockmodell auf der Funktionalen Ebene, welches ebenso keine Aussage über die Ströme der Ausgangsstufe trifft. Die Beschaltung mit einem Lastwiderstand führt zu einem direkten Widerspruch, wenn mittels der Kirchhoff'schen Erhaltungssätze Knoten- und Maschengleichungen aufgestellt werden. Für die Praxis wird daher ein um einen Innenwiderstand erweitertes Ersatzschaltbild benötigt.

Abbildung 23 zeigt die Ausgangsstufe des um den Innenwiderstand erweiterten Operationsverstärkermodells. Für die Betrachtung im Leerlauf ist der Innenwiderstand R_I nicht relevant, da $V_{out} = V_{in}$ gilt. Für beliebige Lastsituationen muß der Ausgangsstrom begrenzt werden: Der maximale Ausgangsstrom $I_{out(\max)}$ fließt bei Kurzschluß des Ausgangs.

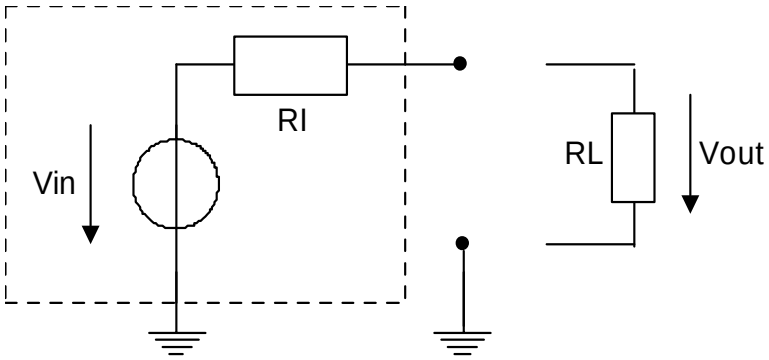

Abbildung 23: Erweitertes Ersatzschaltbild für einen Operationsverstärker.

Die Idee ist nun, R_I so zu dimensionieren, daß $0 \leq I_{out} \leq I_{out(\max)}$ immer gewährleistet ist.

Die Spannungs-Strom-Kennlinie des Innenwiderstand z.B. eines Operationsverstärkers ist im Allgemeinen nicht linear. Abbildung 24 zeigt die idealisierte Verbindung zwischen Leerlauf und Kurzschluß im Vergleich zu realen Simulationsergebnissen mit diskreten Lastpunkten. Der Innenwiderstand ergibt sich zu $R_I = -\frac{U}{I}$. Die alleinige Betrachtung der Grenzfälle (Kurzschluß und open loop) ist offensichtlich nicht hinreichend, um die Größe des Innenwiderstandes zu errechnen. Aus dieser Erkenntnis folgt, daß die Last R_L im Modell bekannt sein muß, damit eine darauf angepaßte Modellierung des Innenwiderstandes möglich wird.

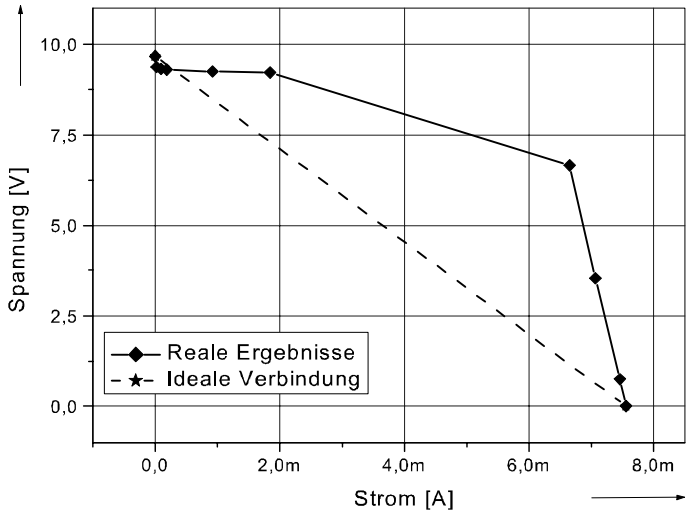

Abbildung 24: Strom-Spannungskennlinie eines Operationsverstärkers.

Im Funktionsblockmodell wird $R_I \in \overrightarrow{x}$ eingeführt, das heißt der Innenwiderstand geht als eine zusätzliche Zustandsvariable in das Modell ein. Der Zusammenhang R_I zu R_L kann entsprechend (3.13) funktional notiert werden mit

$$R_I = h(R_L) \tag{3.30}$$

und mittels der Übergangsfunktion für die Zustände δ im Modell gesetzt werden. Das Ergebnis für das Beispiel Operationsverstärker illustriert Abbildung 25. Die Beziehung ist in einem weiten Bereich linear, nimmt die Last Werte gegen Null bzw. Unendlich an, ist die Beziehung analog zu Abb.24 jedoch nichtlinear.

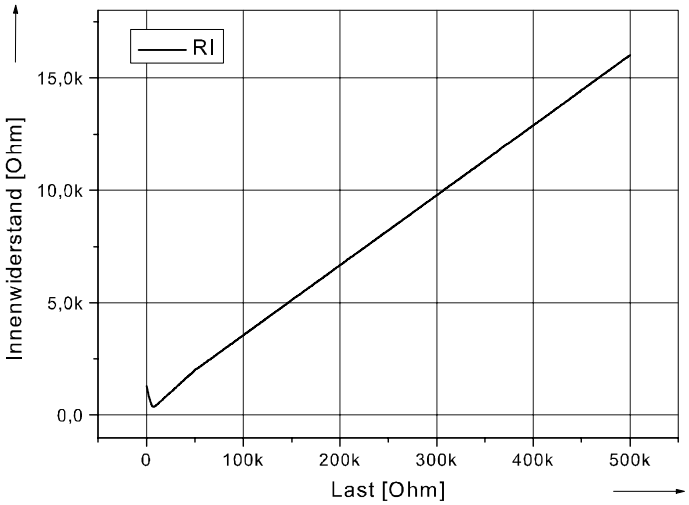

Abbildung 25: Größe des Innenwiderstandes in Abhängigkeit der anliegenden Last.

Aus Gleichung (3.30) und dem allgemeinen Spannungsteiler ergibt sich folgende Situation am Ausgang:

$$V_{out} = \frac{V_{in} * R_L}{R_i + R_L} = \frac{V_{in} * R_L}{h(R_L) + R_L} \tag{3.31}$$

Die Methodik wird im Beispiel Operationsverstärker MOPA1 im Kapitel 5 demonstriert. Die Ergebnisse zeigen, daß es herkömmlichen Modellen auf der Makroebene deutlich überlegen ist.

Mit diesen Erweiterungen wird das Funktionsblockmodell zu einem echten Verhaltensmodell. Ein großer Vorteil des neuen Verfahrens ist die einfache Skalierbarkeit: Für Simulationen, in welchen ein funktionales Modell hinreichend ist, kann die Modellierung der Ausgangsstufe mittels dem Innenwiderstand abgeschaltet werden, ohne Veränderungen am

Modell selber vorzunehmen. Die für die Ausgangsstufe gemachten Aussage lassen sich für alle Pins des Funktionsblocks, insbesondere auch für die Eingänge, verallgemeinern.

3.10 Zustandsbegriff im Funktionsblockmodell

Dieser Abschnitt stellt einen Vergleich her zwischen der mit Differential-Algebraischen Gleichungen beschriebenen Schaltung und dem für diese Schaltung generierten Funktionsblockmodell. Hierbei wird speziell der Begriff des Zustands betrachtet, es wird gezeigt warum das Funktionsblockmodell in der Regel mit einer wesentlich reduzierten Anzahl von Zuständen auskommt.

Dynamische Systeme enthalten besondere Elemente, die Energie speichern können. Diese Elemente, in elektronischen Schaltungen sind dies Kapazitäten (Kondensator) oder Induktivitäten (Spule) werden ge- und entladen. Für jedes dieser Bauteile kann zu jedem Zeitpunkt der Simulation auf der Komponenten-Ebene die im Bauteil gespeicherte Energie berechnet werden. Diese Eigenschaft einer Kapazität oder Induktivität fließt als Zustandsvariable in die Differential-Algebraischen Gleichungen (DAE) nach (2.1) des Systems ein. Herkömmliche Ansätze für strukturnahe Modelle (siehe auch Abschnitt 2.5) basieren darauf, die DAE für die Schaltung vollständig aufzubauen und dann geeignete Vereinfachungen vorzunehmen.

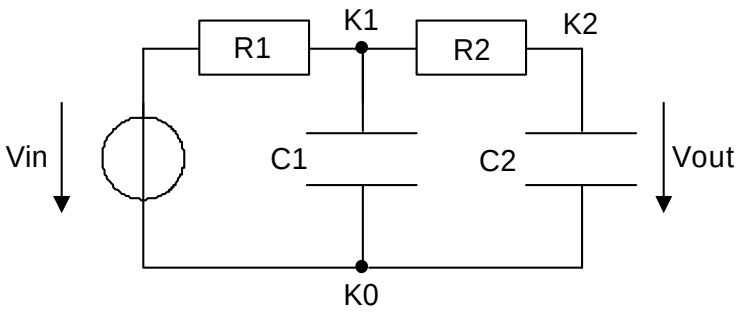

Abbildung 26: RC-Filterschaltung zweiter Ordnung.

Die im folgenden Beispiel dargestellten Gleichungen für die Ströme in den Knoten enthalten zwei Kondensatoren als Energiespeicher. Die Gleichungen stammen aus der in Abbildung 26 dargestellten RC-Filterschaltung, welche im Kapitel 5 modelliert wird:

$$i_{C_2} = C_2 \frac{dU_{out}}{dt} \tag{3.32a}$$

$$i_{C_1} = C_1 \frac{d(R_2 * i_{C_2} + U_{out})}{dt} \tag{3.32b}$$

Der Zustandsvektor einer nach Gleichung (2.2) spezifizierten Schaltung sei im folgen-

den mit $\widehat{\overrightarrow{x}}$ bezeichnet. Das Funktionsblockmodell der selben Schaltung enthält $\overrightarrow{x}$, die Menge der internen Zustände des Funktionsblockmodells. $\widehat{\overrightarrow{x}}$ geht direkt aus der Schaltungstopologie hervor, somit ist die Dimension des Vektors identisch mit den Anzahl der in der Schaltung enthaltenen Energiespeicher. Demgegenüber enthält $\overrightarrow{x}$ die internen Zustandsvariablen des Funktionsblockmodells und es gilt i. Allg. $|\overrightarrow{x}| \neq \left|\widehat{\overrightarrow{x}}\right|$: Die Anzahl der Zustandsvariablen im Modell ist ungleich der Anzahl der Energiespeicher in der Schaltung.

Im Gleichungssystem (3.32) sind zwei Zustandsvariablen enthalten, es gilt also $\left|\widehat{\overrightarrow{x}}\right| = 2$. Demgegenüber gilt $|\overrightarrow{x}| = 3$, denn das Funktionsblockmodell benötigt 3 Zustandsvariablen, welche die Eingangsspannungsdifferenz, den Sollwert des Ausgangs und den alten Ausgangswert speichern. Diese Zustandsvariablen stehen in keinem Zusammenhang mit den Energiespeichern in der Schaltung. Ihre Anzahl bleibt konstant, auch wenn wesentlich komplexere Schaltungen mit einer hohen Zahl an Energiespeichern modelliert werden.

Nun stellt sich die Frage, wieso das Funktionsblockmodell mit einer gegenüber der Schaltungstopologie reduzierten Menge interner Zustände auskommt, ohne daß hierdurch Einbußen an Genauigkeit auftreten. In $\widehat{\overrightarrow{x}}$ ist die Vergangenheit des gesamten Systems gespeichert, es dient dazu, daß die an einer beliebigen Stelle die Simulation unterbrochen werden kann. Beim Fortsetzen zu einem späteren Zeitpunkt läuft die Simulation exakt genau so weiter, als wäre sie nie angehalten worden. Daß dies auch für die neue Methodik gilt, wird mit der Herleitung ausgehend von der als Integrationsverfahren verwandten Trapezintegration in Gleichung (2.6) gezeigt.

Bestimmung des Arbeitspunktes Die Verankerung der Rekursion in Gleichung (2.6) ist die Iteration (2.5), welche den Arbeitspunkt der Schaltung zu Beginn der Simulation bestimmt. Der Arbeitspunkt ist definiert als stabiler Zustand der Schaltung für ein bestimmtes $\widehat{\overrightarrow{x}}$, das heißt alle Einschwingvorgänge sind abgeklungen. Für das Funktionsblockmodell ergibt sich der Ausgang $\overrightarrow{y}$ mit einer DC-Analyse zu

$$\overrightarrow{y} = DC(\overrightarrow{u}). \tag{3.33}$$

In (3.33) wird nun die direkte Beziehung von Eingang zu Ausgang mathematisch beschrieben. Die Richtigkeit folgt aus dem Verfahren zum Erzeugen der Werte, das ja auf den vom Netzwerksimulator bereitgestellten Ergebnisse beruht.

Bei Bedarf kann für jeden inneren Zustand der Schaltung eine Zustandsvariable in das Funktionsblockmodell eingefügt werden und mittels δ berechnet werden. Für das Zusammensetzen des Ausgangssignals im Zeitbereich sind diese Informationen allerdings nicht notwendig. An dieser Stelle ist die Richtigkeit des Ausgangssignals per Definition sichergestellt. $\overrightarrow{x}$ enthält alle notwendigen Initialisierungen für das System, durch die Abbildung $\widehat{\overrightarrow{x}} \rightarrow \overrightarrow{x}$ gehen bei der DC Analyse keine Informationen verloren, die Belegung von jedem $x_i \in \widehat{\overrightarrow{x}}$ ist korrekt in $\overrightarrow{y}$ berücksichtigt, wenngleich an dieser Stelle nicht explizit bekannt (es sei denn, es ist ausdrücklich als Zustandsvariable definiert).

Integrationsschritte des Simulators Ausgehend von (3.33) ist für die korrekte Modellierung des Ausgangssignals die Betrachtung der Systemdynamik, also die Änderungen der inneren Zustände der Schaltung bzw. des Modells, notwendig. Für den n-ten Zeitschritt $t^{(n)}$ führt der Simulator bei der Simulation der Schaltung die Iteration gemäß (2.6) durch. $\widehat{\overrightarrow{x}}^{(n-1)}$ ist bekannt, der Schätzer für $\overrightarrow{xs}$ berechnet anhand der inneren Zustände der Schaltung die zeitliche Veränderung der Ausgänge. Er subtrahiert den aktuellen Wert $n-1$ vom mit dem Schätzer für den Zeitschritt n gewonnenen Wert, um die Steigung des Ausgangssignals zu erhalten. Diese Steigung ist abhängig vom Eingangssignal und von der Belegung der internen Zustandsvariablen der Schaltung.

Die Belegung $\widehat{\overrightarrow{x}}^{(n)}$ der internen Zustandsvariablen legt nun fest, inwieweit die vergangenen Änderungen des Eingangssignals sich auf das Ausgangssignal auswirken, also ob die Einschwingvorgänge abgeklungen sind oder nicht. Hieraus folgt, daß sich Veränderungen des Eingangs unter Umständen gar nicht direkt auf den Ausgang auswirken und daß in jedem Fall eine zeitliche Verzögerung berücksichtigt werden muß. Aus diesen Überlegungen ergibt sich folgende allgemein bekannte Anforderung:

Anforderung 3 *Die zeitliche Änderung des Ausgangs eines dynamischen Systems kann im Allgemeinen nicht direkt anhand des Eingangssignals bestimmt werden, vielmehr ist es notwendig, die Belegung der internen Zustandsvariablen (und damit die Vergangenheit der Simulation) zu berücksichtigen.*

Die betrachtete Schaltung befinde sich zum Zeitpunkt $t^{(n-1)}$ in einem bestimmten Gesamtzustand $\widehat{\overrightarrow{x}}^{(n-1)}$, das Funktionsblockmodell in $\overrightarrow{x}^{(n-1)}$. Er geht mit ta über in den neuen Zustand $\widehat{\overrightarrow{x}}^{(n)}$ bzw. $\overrightarrow{x}^{(n)}$. An dieser Stelle greift die Übergangsfunktion für die Zustandsvariablen im Funktionsblockmodell.

Die zum Parametrieren der Zustandsvariablen und der Funktion für die zeitliche Veränderung durchgeführten Läufe des Netzwerksimulators haben den Übergang von $\overrightarrow{x}^{(n-1)}$ in $\overrightarrow{x}^{(n)}$ geliefert, und zwar mittels der Sprungantwortkurve im Zeitbereich auf $\Delta\overrightarrow{u}^{(n)} = \overrightarrow{u}^{(n)} - \overrightarrow{u}^{(n-1)}$. In dieser Kurve sind die Belegungen der Zustände $\widehat{\overrightarrow{x}}^{(n)}$ der Schaltung enthaltenen, denn die Simulation erfolgte auf der Komponentenebene. Sie stecken implizit in $\overrightarrow{x}^{(n)} = \delta(\overrightarrow{u}^{(n)}, \overrightarrow{x}^{(n-1)})$, also dem Übergang von $\overrightarrow{x}^{(n-1)}$ nach $\overrightarrow{x}^{(n)}$.

Die Tatsache, daß in $\overrightarrow{x}^{(n-1)}$ nicht notwendigerweise alle Einschwingvorgänge abgeklungen sind, wird im Modell dadurch wiedergegeben, daß anhand (3.25) festgestellt werden kann und muß, ob alle Vorgänge abgeschlossen sind. An dieser Stelle greift die erweiterte Ausgangsfunktion nach Gleichung (3.29).

Während in der Schaltung den Zustandsgrößen Energiespeicher entsprechen, welche jeweils in einem festen Wertebereich $0 \leq \widehat{x}_i \leq \widehat{x}_{i(MAX)}$ definiert sind und in ihrer Gesamtheit den Zustand der Schaltung ergeben, werden diese internen Variablen bei der Durchführung der Simulationen zum Errechnen von δ und von f zu einem Gesamtzustand des Systems reduziert, d.h. sie sind in $\overrightarrow{x}$, den Zustandsvariablen des Funktionsblockmodells, implizit enthalten.

Mit diesen Überlegungen kann für den Zeitpunkt $t^{(n)}$ gefolgert werden, daß der aus $\widehat{\overrightarrow{x}}^{(n-1)}$ nach $\widehat{\overrightarrow{x}}^{(n)}$ resultierende Ausgangsvektor $\widehat{\overrightarrow{y}}^{(n)}$ dem aus $\overrightarrow{x}^{(n-1)}$ nach $\overrightarrow{x}^{(n)}$ resultierendem $\overrightarrow{y}^{(n)}$ entspricht, es gilt also:

$$\overrightarrow{y} \Leftrightarrow \widehat{\overrightarrow{y}} \tag{3.34}$$

Mit der Rekursionsverankerung folgt daraus die Übereinstimmung der Verfahren.

3.11 Anwendungsbereiche der Funktionsblockmodelle

Funktionsblockmodelle sind auf alle Schaltungen anwendbar, welche über mindestens einen analogen Ausgang verfügen, der ein Sprungantwortverhalten zeigt. Die Sprungantworten sind notwendig, um während der Charakterisierung die Methode für die Funktion für die zeitliche Veränderung zu erzeugen. Der Anwendungsbereich wurde gegenüber dem von Hamad in [Ham95] eingeführten Modellgeneratorkonzept von der Klasse der rückgekoppelten Schaltungen auf allgemeine Schaltungen erweitert.

Zu den Schaltungsklassen, die mit dem vorgestellten Verfahren nicht modelliert werden können, gehören Schaltungen mit oszillierenden Ausgangssignalen. Ebenso führen Nichtstetigkeiten im Ausgangssignal dazu, daß die Methodik nicht direkt anwendbar ist, wie die A/D-Wandlerschaltung im Kapitel 5 zeigt. Für Klassen, die nicht direkt modelliert werden können, ist es notwendig, eine Partitionierung durchzuführen in Teilschaltungen, welche das benötigte Verhalten zeigen. Wenn dies gelingt, dann können die Teilschaltungen modelliert werden und ohne weiteres in eine Gesamtschaltung kombiniert werden.

Funktionsblockmodelle sind für die Modellierung für Simulationen im DC- und Transientenbereich vorgesehen, das heißt das Kleinsignalverhalten der Schaltungen wird nicht im Modell berücksichtigt.

Die Definition der Funktionsblockmodelle als DESS&DEVS ermöglicht speziell die Modellierung gemischt analog/digitaler Schaltungen. Die Ausgangsgrößen des Modells können sowohl der analogen als auch der digitalen Domäne angehören, wobei die analogen Ausgangsgrößen zeitkontinuierliches Verhalten aufweisen und die digitalen Ausgangsgrößen durch den ereignisdiskreten Teil des Modells abgedeckt sind, d.h. sie können direkt über Ereignisse mit Werten belegt werden.

Ein Funktionsblockmodell läßt sich sowohl für Simulationen auf der Funktionalen als auch auf der Verhaltensebene einsetzen. Die neue Methodik deckt beide Ebenen ab, ohne daß separate Modelle benötigt werden.

In Kapitel 4 wird vorgestellt, wie die neue Methodik in der Praxis eingesetzt wird. Es wird skizziert, wie die Datenbasis während der Charakterisierung gewonnen wird und aus dieser Datenbasis das auf dem Simulator lauffähige Modell generiert wird.

4 Realisierung der Methodik

Kapitel 3 stellte die theoretische Spezifikation des Funktionsblockmodells und die darin enthaltenen Elemente dar. In Definition 3 wurde die Signatur des Funktionsblockmodells eingeführt. Die Signatur ist ein formales Konstrukt, das auf den Grundlagen der Systemtheorie basiert. Damit das Modell für Einsätze in der Praxis verwandt werden kann, wird eine Repräsentation des Funktionsblockmodells in einer HDL benötigt. Dieses Kapitel zeigt wie, vom theoretischen Modell ausgehend, ausführbarer HDL Code generiert wird. Dazu wird zunächst die Struktur des Funktionsblockmodells entwickelt und Möglichkeiten für die Umsetzung der Theorie in die Praxis gezeigt. Die Realisierung der zur Anwendung der vorgeschlagenen Methodik notwendigen Verfahren wird beschrieben. Im Mittelpunkt steht dabei das Erzeugen der Funktionsblockmodelle, das sich in zwei Schritte gliedert:

1. Charakterisierung und
2. Modellerzeugung.

Abbildung 27 zeigt die Schritte zur Realisierung der Methodik und die im folgenden Abschnitt vorgestellten Stufen. Die Charakterisierung und die Modellerzeugung werden ausführlich diskutiert, der Bezug zu Kapitel 3 wird durch die Definition einer minimalen Menge von Parameterfunktionen hergestellt. Abgeschlossen wird das Kapitel durch eine Untersuchung der Genauigkeit der Funktionsblockmodelle.

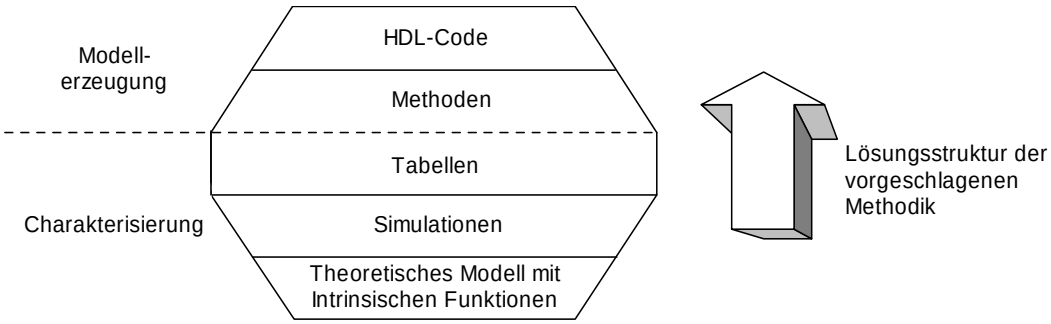

Abbildung 27: Realisierung der Methodik: Vom theoretischen Modell zum ausführbaren Code.

4.1 Struktur der Funktionsblockmodelle

Tabelle 8 stellt eine Einordnung der neuen Methodik dar. Die Funktionsblockmodelle lassen sich auf unterschiedlichen Stufen darstellen und verwenden. Die erste Spalte der Tabelle definiert die Stufe, auf der das Modell steht. Die Übergänge erfolgen von der

Theorie hin zur Praxis. In der zweiten Spalte wird definiert, wie die Elemente des Funktionsblockmodells auf der jeweiligen Stufe repräsentiert werden. Die der Repräsentation entsprechende Signatur wird in Spalte 3 dargestellt.

Modell	**Repräsentation**	**Signatur**
Formal	Theoretisch	$h_i(\overrightarrow{u}_i, \overrightarrow{x}_{i-1}, \overrightarrow{y}_{i-1})$
Simulator	Simulatorläufe	$DC(\overrightarrow{u}), TRAN(\overrightarrow{u})$
Tabelle	Datenstruktur	$T_{\overrightarrow{u}}, T_{\overrightarrow{y}}$
Mathematisch	Methoden	$\{\Omega, F\}$
Praktisch	HDL-Code	`function slewrate(...)`

Tabelle 8: Begriffsbestimmung innerhalb der Lösungsstruktur.

Ausgangspunkt ist das in Definition 3 eingeführte systemtheoretische Funktionsblockmodell, welches die Signaturen für die darin enthaltenen Elemente festlegt.

Die formale Spezifikation des Funktionsblockmodells enthält intrinsische Funktionen, wie zum Beispiel die Verzögerungszeit. Diese intrinsischen Funktionen entsprechen der Übergangsfunktion für Ereignisse $\delta()$ und der Funktion für die zeitliche Änderung f. In jedem einzelnen Integrationsschritt, in dem das Funktionsblockmodell ausgewertet wird, muß für jede intrinsische Funktion der zu den Eingangsgrößen bzw. Zustandsgrößen korrelierende Wert gefunden werden. Dazu ist es zunächst einmal möglich, einen Simulatorlauf der Referenzschaltung mit den entsprechend gesetzten Eingängen durchzuführen und aus der Ergebniskurve die benötigten Werte zu extrahieren. Dies sind die Werte, die die intrinsischen Funktionen in diesem Schritt annehmen. Sie können direkt in das Modell eingesetzt werden, um den Ausgang im aktuellen Integrationsschritt gemäß der Definition zu errechnen. Die Simulationen können auf der DC- oder Transientenebene erfolgen, entsprechend enthält die Signatur in Tabelle 8 diese Simulatorläufe.

Abbildung 28 zeigt, wie dies funktioniert. Das angenäherte Ausgangssignal wird stückweise zusammengesetzt: In jedem Integrationsschritt wird ein Stück aus der Sprungantwort extrahiert, welche anhand eines Simulatorlaufes mit einem der Änderung des Eingangssignals entsprechenden Sprung am Eingang der Referenzschaltung gewonnen wird.

Dieses Verfahren setzt die Methodik direkt um, hat jedoch einen extrem hohen Zeitaufwand, da permanent Simulationen erfolgen müssten. Die Idee ist es deshalb, die Simulationen vorab, d.h. vor dem Verwenden des Modells, durchzuführen. Zur Laufzeit des Modells wird auf die vorab gespeicherten Tabellen zurückgegriffen. Dieses Vorgehen wird als Charakterisierung der Schaltung bezeichnet. Die Charakterisierung definiert sich wie folgt:

Definition 4 *Charakterisierung: Extrahieren der relevanten Daten aus den Ergebnissen, die durch die Ausführung einer Modellrepräsentation, die auf einer niedrigeren Abtraktionsebene und / oder in einer anderen Sicht des Blocks notiert ist, erzeugt werden.*

Beispielsweise kann eine Spice-Netzliste, die den Block auf der Komponentenebene beschreibt, als Referenz der Modellrepräsentation dienen. Simulationen erfolgen im DC-

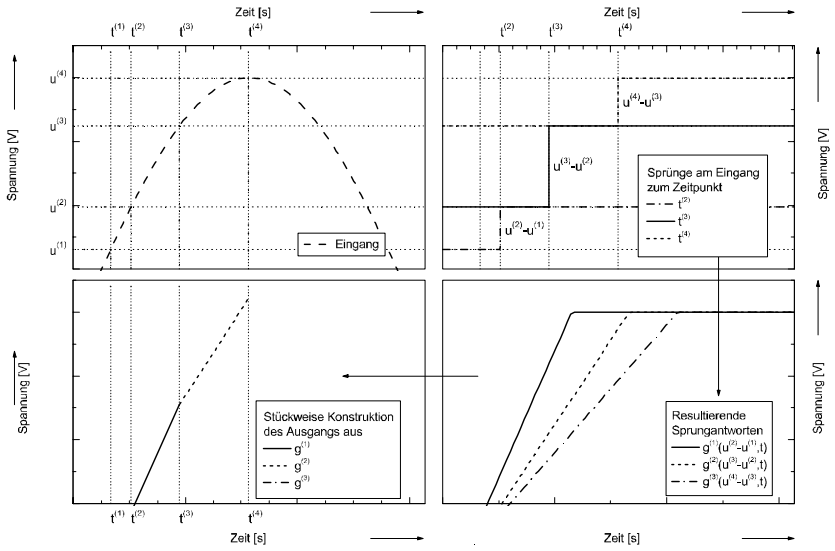

Abbildung 28: Konstruktion des angenäherten Ausgangssignals.

und Transientenbereich, aus den Ergebnistabellen werden die für die Modellerzeugung benötigten Werte extrahiert, das heißt mathematisch berechnet.

Aus den Tabellen werden mittels Verfahren der Regression mathematische Modellfunktionen erzeugt, die effizienter auswertbar und speicherbar sind als Tabellen. Die Modellerzeugung ist wie folgt definiert:

Definition 5 *Modellerzeugung: Die im Funktionsblockmodell benötigten intrinsischen Funktionen (Übergangsfunktionen für Zustandsvariablen und die Funktion für die zeitliche Veränderung) werden anhand der Ergebnisse der Charakterisierung erzeugt. Die intrinsischen Funktionen werden als mathematische Modelle unter Berücksichtigung der Anforderungen an die Genauigkeit realisiert, aus ihnen wird automatisch der Simulatorcode des Funktionsblockmodells generiert.*

Diese Modellfunktionen werden in Tabelle 8 als Methoden bezeichnet. Der wichtigste Schritt ist das Generieren der mathematischen Modelle. Ausgehend von der Vorstellung der in der Literatur bekannten Verfahren zur mathematischen Modellbildung wird die Mathematik vorgestellt, mit welcher die Modelle in dieser Arbeit implementiert werden. Die Anforderungen an die Genauigkeit beim Erstellen der mathematischen Modelle erfolgen über die Vorgabe der Fehlernorm und des Abstandsmasses. Die implementierten Verfahren werden vorgestellt, gefolgt von allgemeinen Modellen zur Fehlerfortpflanzung. Die drei für die Grundfunktionalität essentiellen Methoden Sollwert des Ausgangs, Verzögerungszeit und Slewrate werden mathematisch definiert. Ihre Auswirkung auf die Genauigkeit des

Gesamtmodells wird mittels Fehlerfortpflanzungsmodellen untersucht. Daraus lassen sich Erkenntnisse über die Anforderungen gewinnen, welche beim Erzeugen der Modelle an die Genauigkeit gestellt werden müssen.

Der letzte Schritt zum ausführbaren Modell ist die Umwandlung der Ergebnisse der Methoden in HDL-Code, den der Simulator ausführen kann. An dieser Stelle steht das Funktionsblockmodell in der gewünschten Simulatorumgebung bereit. Dazu werden die mathematischen Funktionen in die Syntax der HDL umgesetzt.

4.2 Charakterisierung

Dieser Abschnitt beschreibt, wie die für das Funktionsblockmodell notwendige Datenbasis gewonnen wird. Die Signatur des Funktionsblockmodells enthält Funktionen, das heißt mathematische Abhängigkeiten. Diese Abhängigkeiten der allgemeinen Form $y = f(u)$ werden während der Charakterisierung in Form von Tabellen (T_u, T_y) erzeugt. Dabei wird zwischen direkten und indirekten Abhängigkeiten unterschieden. Die beiden ersten Teile dieses Abschnittes beschreiben, wie die direkten und indirekten Abhängigkeiten mathematisch modelliert werden und definieren die grundlegenden Parameterfunktionen. Den Abschluß bildet eine Diskussion des Charakterisierungsplanes, so wie er in der Praxis benutzt werden sollte. Die Beispiele in Kapitel 6 basieren auf diesen Grundlagen.

4.2.1 Direkte Abhängigkeiten

Direkte Abhängigkeiten beschreiben einen funktionalen Zusammenhang $y = f(u)$, welcher anhand einer einzigen DC-Analyse gewonnen werden kann. Die unabhängige Variable wird dabei auf einen Eingang gelegt und über ihren gesamten Definitionsbereich mit einer diskreten Abtastrate aufgespannt. Zu jedem Punkt (mit vorgegebener Schrittweite) kann der unabhängigen Variable direkt der korrelierende Wert der abhängigen Variable zugeordnet werden.

Ausgangspunkt für die Modellierung dieser direkten Abhängigkeiten ist eine Repräsentation der zu modellierenden Schaltung auf einer niedrigeren Abstraktionsebene, zum Beispiel eine Spice-Netzliste auf der Komponentenebene. Jetzt soll der Zusammenhang zwischen Eingangsgrößen $\overrightarrow{u}$ und der resultierenden Belegung der Ausgangsvariablen $\overrightarrow{y}$ für die Schaltung nach Ausklang aller Einschwingvorgänge ($t \rightarrow \infty$) hergestellt werden. Die Eingänge der Schaltung haben einen vorgegebenen Definitionsbereich, wobei jeder Eingangsgröße $u_i \in \overrightarrow{u}$ ist ein fest bestimmtes Segment

$$\omega_i : (u_{i1}, u_{i2}) = \{u_i | u_i \in I\!R, u_{i1} \leq u_i \leq u_{i2}\} \tag{4.1}$$

zugeordnet ist. Das Ergebnis dieser Simulationen ist eine Tabelle mit einer Zuordnung der diskreten Belegung der Eingänge $T_{\overrightarrow{u}}$ mit den resultierenden Werten der Ausgänge $T_{\overrightarrow{y}}$. Im Falle einer Schaltung mit einem Eingang und einem Ausgang enthält die Tabelle Wertepaare der Form (T_u, T_y) mit

$$T_y = DC(T_u). \tag{4.2}$$

In (4.2) steht *DC* für die durchgeführte DC-Analyse, d.h. es ist die vom Simulator erzeugte Tabelle mit den Simulationsergebnissen.

4.2.2 Indirekte Abhängigkeiten

Der direkte Zusammenhang nach (4.2) läßt sich für das Ergebnis einer DC Analyse aufstellen und im Verhaltensmodell für die Bestimmung des Operationspunktes verwenden. Für eine Transientenanalyse ist dieses Vorgehen jedoch nicht hinreichend: Hier wird die Dynamik des Systems betrachtet, nicht das Ergebnis im eingeschwungenen Zustand. Wird beispielsweise ein Sprung von u_1 auf u_2 auf den Eingang einer Testschaltung gegeben, so kann mittels (4.2) berechnet werden, welchen Wert der Ausgang y nach dem Abklingen aller Einschwingvorgänge annimmt. Das Einschwingen selber, die Einschwingzeit oder auch die Flankensteilheit bzw. Verzögerungszeit sind daraus nicht herleitbar. Abbildung 19 zeigt eine solche Sprungantwort.

Um diese Signalverläufe berechnen zu können, muß das Ergebnis der Transientenanalyse für den Sprung von u_1 auf u_2 betrachtet werden. Aus der Tabelle mit den Simulationsergebnissen können die relevanten Größen, das sind die Funktionswerte der intrinsischen Funktionen, mittels mathematischer Modellfunktionen berechnet werden.

Während der Charakterisierung der Schaltung wird der Definitionsbereich der möglichen Veränderung der Eingangsgröße mit hinreichender Abtastrate abgedeckt. Jede Veränderung der Eingangsgröße kann als Spannungssprung betrachtet werden. Für jeden Spannungssprung liefert eine Transientenanalyse die resultierende Sprungantwort, es erfolgt die Zuordnung des skalaren Wert des Spannungssprung zu einer Signalform. Das Ergebnis ist eine Kurvenschar.

Abbildung 29 zeigt einen Auszug aus einer Kurvenschar: Für jeden diskreten Wert des Eingangsspannungssprungs liefert die Simulation eine Kurve. Diese Kurvenschar dient als Datenbasis für die Modellierung der indirekten Abhängigkeiten.

Die Grundidee ist es, die kontinuierliche Änderung an der Eingangsgrößen in diskreten Zeitpunkten zu betrachten. Dies entspricht den dynamisch bestimmten Integrationsschritten, die der Simulator in der Transientenanalyse verwendet. In jedem Zeitschritt $t^{(n)}$ ist die Änderung der Eingangssignale $\Delta\overrightarrow{u}^{(n)}$ bekannt. Aus (4.1) folgt, daß die Größe der Änderung eines Eingangssignal Δu_i in jedem Zeitschritt n begrenzt ist. Es gilt:

$$0 \leq \Delta u_i^{(n)} \leq u_{i2} - u_{i1}. \tag{4.3}$$

Um jetzt $\overrightarrow{y}^{(n)}$ gemäß z.B. der Trapezintegration (2.6) zu berechnen, wird $\Delta\overrightarrow{y}^{(n)} = f(\Delta\overrightarrow{u}^{(n)})$ benötigt, die zeitliche Änderung der Ausgangssignale im aktuellen Zeitschritt. Dies kann nicht direkt gemäß (4.2) als direkte Abhängigkeit ausgedrückt werden. Vielmehr wird für die Änderung des Eingangssignals $\Delta\overrightarrow{u}^{(n)}$ eine Transientenanalyse durchgeführt,

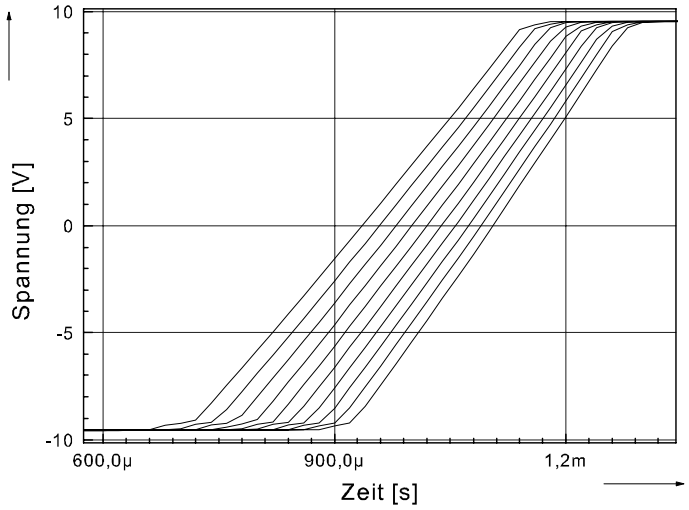

Abbildung 29: Jede Kurve der Schar beschreibt eine Sprungantwort.

d.h. die Modellrepräsentation wird mit einem Eingangs-Spannungssprung mit einem Betrag von $\Delta\overrightarrow{u}^{(n)}$ angeregt. Das Ergebnis ist kein diskreter Vektor $\Delta\overrightarrow{y}^{(n)}$, sondern ein zeitkontinuierlicher funktionaler Zusammenhang $g(t)$. $g(t)$ wird also mit einem Simulationslauf der zu simulierenden Schaltung gewonnen und enthält die Antwort der Schaltung auf einen bestimmten Eingangsverlauf. Diese Simulation kann nun für die Menge $W = \{w_j \mid w_j = \Delta u_i^{(n)}, j = 1...m\}$ durchgeführt werden, die den gesamten Definitionsbereich (4.3) der möglichen Eingangssprünge abdeckt. Ergebnis dieser Simulationsläufe ist eine Kurvenschar $G = \{g_j(t), j = 1...m\}$ nach Abbildung 29, d.h. jedem Element aus W ist die resultierende Antwort $g_j(t)$ zugeordnet.

Die inneren Zustände der Originalschaltung sind in den Ergebnissen der DC-Analyse implizit enthalten - und sind somit auch im Funktionsblockmodell vorhanden. Dieser Zusammenhang ist für die indirekten Abhängigkeiten nicht so deutlich, denn $(\delta, f)^{\mathrm{T}}$ entspricht nicht den Energiespeichern, siehe die Definition in Kapitel 3. Mit $\widehat{\overrightarrow{x}}$ werden im folgenden die Werte der inneren Zustände der Modellrepräsentation auf der niedrigeren Abstraktionsebene bezeichnet. In der Kurve $g(t)$ sind die Werte von $\widehat{\overrightarrow{x}}$ (welche sich während der Durchführung der Transientensimulation mit dem Eingangssprung am Eingang verändern) enthalten, und damit auch die aus dem Eingangsimpuls resultierenden Änderungen der inneren Zustände $\Delta\widehat{\overrightarrow{x}}$. Es ist also möglich, einen Zusammenhang der Form

$$\Delta\widehat{\overrightarrow{x}}^{(n)} = \overrightarrow{h}\,(g_j(t)) \tag{4.4}$$

herzustellen, welcher beschreibt, wie sich eine Veränderung am Eingang der Schaltung

$\Delta\overrightarrow{u}$ auf die inneren Zustände des Systems auswirkt. Wichtig hierbei ist, daß $\overrightarrow{h}$ keine Abhängigkeit von der Zeit hat, sondern daß $\overrightarrow{h}$ aus dem Ergebnis einer Transientenanalyse die Werte der Zustände berechnet. Anhand dieser Änderung kann der resultierende Ausgangsvektor bestimmt werden. Mit den in der Datenbasis G enthaltenen Informationen kann jetzt eine vollständige Simulation im Zeitbereich nach (2.6) erfolgen. Dieses Vorgehen hat noch keine entscheidenden Vorteile gegenüber der numerischen Simulation im Netzwerkanalyseprogramm, da durch die große Anzahl der inneren Zustände eine riesige Datenbasis anfällt und die Berechnung anhand der Schaltungstopologie sehr aufwendig wird. Deshalb wird an dieser Stelle der Zustandsraum transformiert.

Abbildung 19 zeigt eine Sprungantwort $g_j(t)$ für ein bestimmtes $\Delta\overrightarrow{u}^{(n)}$. Folgende Annahmen bilden die Basis der neuen Methodik: Die Grundform der Antwortkurve ist für jedes $g \in G$ identisch. Die Unterschiede lassen sich anhand der intrinsischen Funkionen des Funktionsblockmodells $(\delta, f)^{\mathrm{T}}$ gemäß Definition 3 festmachen, wie zum Beispiel der

- Verzögerungszeit, nach welcher die Ausgänge der Schaltung auf die Veränderung des Eingangsvektors reagieren, oder der
- Flankensteilheit, mit welcher sich das Ausgangssignal ändert.

Diese charakteristischen Größen lassen sich mathematisch aus der Kurve $g_j(t)$ berechnen, wobei folgender Zusammenhang gilt:

$$(\delta, f)^{\mathrm{T}} = \overrightarrow{h}(g_j(t), \vec{u}). \tag{4.5}$$

Mit (4.4) und (4.5) folgt die Abhängigkeit $(\delta, f)^{\mathrm{T}} = \overrightarrow{h}(\widehat{\overrightarrow{x}}, \vec{u})$, d.h. die Belegung der Zustandsvariablen fließt in die Berechnung der Verzögerungszeit und der Flankensteilheit mit ein. Diese Abhängigkeit ist implizit gegeben, sie steckt in der Definition von G. Tatsächlich findet an dieser Stelle die Transformation statt von der allgemeinen Beschreibung des dynamischen Systems mit der Zustandsdarstellung zu der abstrahierten Darstellung in der neuen Methodik, wobei der Zustandsraum $\widehat{\overrightarrow{x}}$ auf $(\delta, f)^{\mathrm{T}}$ abgebildet wurde.

Ausgangsbasis zur Erzeugung der Modelle ist eine Repräsentation des zu simulierenden Funktionsblocks für den Netzwerksimulator. Der Definitionsbereich der durch das Modell beschriebenen diskreten Eingangssprünge muß mit hinreichender Genauigkeit abgedeckt werden; die Anzahl der resultierenden Simulationen im Transientenbereich ist dementsprechend sehr groß. Aus der während der Simulationsläufe erzeugten Datenbasis werden - wie später gezeigt - bei der Modellerzeugung mathematische Modelle generiert.

4.2.3 Parameterfunktionen

Ein minimales Funktionsblockmodell nach Definition 3, das auf der Funktionalen Ebene eingesetzt werden soll, enthält in jedem Fall die Übergangsfunktion für

- den Sollwert der Ausgangsgrößen,
- die Zustandsvariable Verzögerungszeit und
- die Funktion für die zeitliche Veränderung, die Slewrate.

Diese für das Funktionsblockmodell grundlegenden Parameterfunktionen werden im folgenden dargestellt. Dazu wird definiert, wie die Werte während der Charakterisierung gewonnen werden.

Sollwert des Ausgangs Die an einem Ausgang des Funktionsblocks im eingeschwungenen Zustand anliegende Spannung ist eine direkte Abhängigkeit nach Gleichung (4.2), die jeweils einer diskreten Eingangsspannung zugeordnet ist. Sie wird durch eine DC-Analyse als Parameterfunktion `VoutDC` erzeugt als Tabelle $(T_{\overrightarrow{u}}, T_{\overrightarrow{x}})$ mit $T_{\overrightarrow{x}} =$`VoutDC`$(T_{\overrightarrow{u}})$.

Der Vergleich mit dem Ausgangsgleichungssystem (2.2) zeigt, daß die Berechnung des DC Operationspunktes nach (2.4) allein abhängig von den Eingangsgrößen erfolgt. Die Tabellen mit den direkten Abhängigkeiten werden anhand von Simulationsläufen gemäß (2.5) gewonnen und enthalten implizit $\overrightarrow{x}$, die korrekte Belegung aller Energiespeicher der Schaltung. Bei Bedarf können diese auch explizit sichtbar gemacht werden und im Modell benutzt werden. Dazu wird die Belegung der Zustandsgrößen als zusätzliche Information während der Simulation erzeugt und in der Ergebnistabelle abgelegt. Dies dient allein dem Zweck, Einblick in Details der Schaltung zu gewinnen: Für die Funktionsblockmodelle sind diese zusätzlichen Informationen nicht notwendig, denn der Ausgangsvektor wird anhand der Eingänge direkt berechnet. Die Energiespeicher, welche bei der Simulation auf der Komponentenebene in der Netzliste enthalten sind, treten im Verhaltensmodell nicht direkt auf, deshalb ist ihre Modellierung optional. Im oben skizzierten Fall einer Schaltung mit einem Eingang und einem Ausgang enthält die erweiterte Tabelle Einträge der Form $(u^{(n)}, \overrightarrow{x}^{(n)}, y^{(n)})$.

Mit diesen Grundlagen ist sichergestellt, daß die Simulation im Zeitbereich mit dem korrekten DC Operationspunkt gestartet wird. Diese Methodik hat gegenüber den Simulationsläufen den Vorteil, daß die Rechenzeit signifikant reduziert wird. Das Aufstellen der Jakobi-Matrix und die Lösung der Gleichungssysteme hat eine exponentielle Rechenzeit $\Theta(n^{\frac{3}{2}})$ (in Abhängigkeit der Schaltungstopologie, d.h. der Anzahl n der Zustandsvariablen); das Auswerten der Parameterfunktion für einen Arbeitspunkt dagegen hat konstanten Aufwand $\Theta(1)$, siehe [VS83].

Die generierten Tabellen dienen nicht nur zur Bestimmung des Operationspunktes beim Start der Transientenanalyse, sondern mit Hilfe dieser Tabellen kann zu jedem Zeitpunkt der Sollzustand des Systems bestimmt werden. Anhand eines vorgegebenen diskreten $\overrightarrow{u}^{(n)}$ kann errechnet werden, wie die Ausgangsvariablen (und optional die Zustandsvariablen) nach Ausklang aller Einschwingvorgänge belegt sein müssen. Diese Eigenschaft wird zur mathematischen Modellierung der Rückkoppelung angewandt.

Verzögerungszeit Wird ein Spannungssprung an den Eingang eines Funktionsblocks angelegt, so ist zu beobachten, daß dieser Spannungssprung erst nach einer bestimmten Zeit eine Wirkung am Ausgang verursacht. Diese Zeit ist die Verzögerungszeit des Funktionsblocks, sie wird wie folgt definiert:

Definition 6 (Verzögerungszeit) *Die Eingangsgröße u_i springt von u_{i1} auf u_{i2}; der Ausgang ändert sich in genügend langer Zeit von y_{i1} auf y_{i2}. Jetzt sei $t_{in50} = t|u_i(t) = \frac{u_{i2}-u_{i1}}{2}$; analog dazu $t_{out50} = t|y_i(t) = \frac{y_{i2}-y_{i1}}{2}$. Die Verzögerungszeit* TD *ist:*

$$TD = t_{out50} - t_{in50} \tag{4.6}$$

Alle zur Berechnung von TD notwendigen Größen stehen nach einer Transientenanalyse zur Verfügung. Zu beachten ist, daß T_D nicht immer konstant ist, sondern daß TD von der Größe des Spannungssprunges am Eingang abhängen kann. Daraus ergibt sich die intrinsische Funktion in Form einer Tabelle $(T_{u_i}, T_{\mathtt{TD}})$. Darin ist u_i die unabhängige Variable und $\mathtt{TD} \in \delta$ wird mit Gleichung 4.6 berechnet.

Flankensteilheit Für die Schaltungsklasse der analogen Funktionsblöcke, welche Energiespeicher enthalten, kann folgendes beobachtet werden: Wird ein Spannungssprung an den Eingang des Funktionsblocks angelegt, so vollzieht der Ausgang diesen Sprung verlangsamt nach. Die zeitliche Änderung des Ausgangssignals erfolgt nicht als Sprung, sondern als stetiger Übergang mit einer Steigung, deren Größe nach oben begrenzt ist. Diese Steigung wird als Slewrate bezeichnet und wie folgt definiert:

Definition 7 (Slewrate) *Die Eingangsspannung springt von u_{i1} auf u_{i2}; der Ausgang ändert sich in genügend langer Zeit von y_{i1} auf y_{i2}. Nun gelte $y_{i10} = \frac{y_{i2}-y_{i1}}{10}$ und $y_{i90} = \frac{9(y_{i2}-y_{i1})}{10}$ sowie die Zeiten $t_{out10} = t|y_i(t) = y_{i10}$ und $t_{out90} = t|y_i(t) = y_{i90}$. Die Definition der Slewrate ist durch*

$$slewrate = \frac{y_{i90} - y_{i10}}{t_{out90} - t_{out10}} \tag{4.7}$$

gegeben.

Die so berechnete Slewrate entspricht der Steigung der Ausgangsspannung zwischen 10 und 90 Prozent der Spannungsdifferenz zwischen Start- und Endwert. Alle zur Berechnung notwendigen Größen werden durch die Transientenanalyse des Funktionsblocks mit dem entsprechenden Sprung der Eingangsspannung gewonnen. Zu beachten ist, daß die Slewrate in der Regel nicht konstant ist, sondern daß ihr Wert von der Größe des Spannungssprunges am Eingang abhängt. Deshalb ergibt sich die Parameterfunktion `slewrate`, welche den Wert der Flankensteilheit in Abhängigkeit des Eingangssignals u errechnet, als Tabelle $(\Delta u, \mathtt{slewrate}(\Delta u))$. Abbildung 30 zeigt als Beispiel die Flankensteilheit des Ausgangs des Operationsverstärkers, aufgetragen über der jeweiligen Spannungsdifferenz an den Eingangspins. In diesem Beispiel wird deutlich, daß speziell der Bereich des Nulldurchgangs besonders wichtig ist. Abschnitt 4.2.4 zeigt auf, wie bestimmte Bereiche genauer als andere charakterisiert werden können.

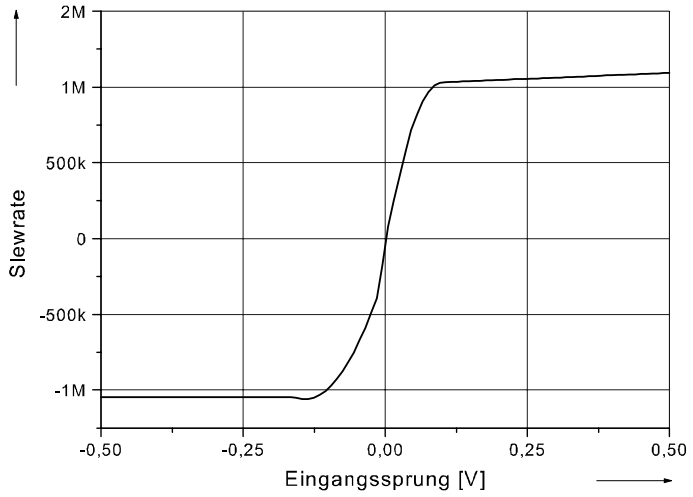

Abbildung 30: Methode Slewrate für MOPA1.

Die allgemeine Form der Funktion für die zeitliche Veränderung nach Gleichung (3.21) enthält die Ableitung des Ausgangs nach der Zeit. Diese wird jetzt ersetzt durch die nach (4.7) berechnete Flankensteilheit. Im folgenden die mathematische Herleitung für diese Ersetzung, der übersichtlicheren Darstellung halber wird ein skalarer Ein- und Ausgang $u^{(n)}$ bzw. $y^{(n)}$ im Zeitschritt n betrachtet.

Behauptung: $\frac{\mathrm{d}y}{\mathrm{d}t} = \texttt{slewrate}(\Delta u^{(n)}) \mid t = t^{(n)}$

Herleitung: Zum Zeitpunkt $t^{(n-1)}$ befindet sich die Schaltung in einem fest definiertem Zustand, Ein- und Ausgangsgrößen haben die festen Werte $u^{(n-1)}$ bzw. $y^{(n-1)}$. $\Delta u^{(n)} = u^{(n)} - u^{(n-1)}$ verursacht am Ausgang eine Wirkung: $\Delta y^{(n)} = y^{(n)} - y^{(n-1)}$ ist die resultierende Änderung des Ausgangs im Zeitschritt n, welche eine Dauer von $\Delta t^{(n)} = t^{(n)} - t^{(n-1)}$ hat. Es gilt: $\frac{\mathrm{d}y}{\mathrm{d}t} = \frac{\Delta y^{(n)}}{\Delta t^{(n)}} \mid t = t^{(n)}$, dies ist die Steigung der Geraden,

die die Punkte $(t^{(n-1)}, y^{(n-1)})$ und $(t^{(n)}, y^{(n)})$ miteinander verbindet. $\texttt{slewrate}(\Delta u^{(n)})$ beschreibt gemäß Definition 7 die Flankensteilheit des Ausgangssignals bei einem Sprung der Eingangsgröße mit dem Betrag $\Delta u^{(n)}$. Mit Gleichung (4.7) gilt $\frac{y_{i90} - y_{i10}}{t_{out90} - t_{out10}} = \frac{\Delta y^{(n)}}{\Delta t^{(n)}}$, denn per Definition liegen die Punkte (t_{out10}, y_{i10}) und (t_{out90}, y_{i90}) auf der Geraden, die $(t^{(n-1)}, y^{(n-1)})$ und $(t^{(n)}, y^{(n)})$ verbindet. Somit entspricht der Wert von $\texttt{slewrate}(\Delta u^{(n)})$ der Ableitung $\frac{\mathrm{d}y}{\mathrm{d}t} \mid t = t^{(n)}$.

Wichtig ist, daß ausschließlich qualitative Aussagen der Sprungantwort (wie Flankensteilheit oder Verzögerungszeit) und keine quantitativen Werte (wie der absolute Betrag des Sprunges) zur Modellierung verwandt werden.

Mittels der Parameterfunktionen ergibt sich die Rekursionsgleichung für den Ausgang nun gemäß (3.16) und (3.17) insgesamt zu:

$$\overrightarrow{y}^{(n)} = \overrightarrow{y}^{(n-1)} + \Delta t^{(n)} * \texttt{slewrate}(\Delta \overrightarrow{u}^{(n)}) \quad (4.8a)$$

$$\overrightarrow{y}^{(0)} = \texttt{VoutDC}(\overrightarrow{u}^{(0)}) \quad (4.8b)$$

4.2.4 Charakterisierungsplan

Im vorigen Abschnitt wurde die Charakterisierung einer Schaltung vorgestellt. Ein *Charakterisierungsplan* legt exakt fest, für welche Eingangssignalformen die Simulationen im DC- und Transientenbereich durchgeführt werden, wie die mathematischen Zusammenhänge zwischen den Eingangsgrößen und den zu modellierenden Eigenschaften sind, und mit welchen Anforderungen an die Genauigkeit die Methoden erzeugt werden sollen. Tabelle 9 zeigt den Ablaufplan einer typischen Charakterisierung.

Ergebnis-Datei	**Startwert V_{in}**	**Endwert V_{in}**	**Schrittweite Δ**
remote1.asc	-10	-5	0.05
remote2.asc	-5	-0.1	0.01
remote3.asc	-0.1	-1E-6	1E-4
remote4.asc	10	5	0.05
remote5.asc	5	0.1	0.01
remote6.asc	0.1	1E-6	1E-4

Tabelle 9: Beispiel für einen Charakterisierungsplan.

In Tabelle 9 ist genau festgelegt, wie die Datenbasis für das Funktionsblockmodell generiert wird. Da bestimmte Bereiche des Definitionsbereichs eine genauere Betrachtung erfordern als andere, wird hier spezifiziert, in welchem Bereich mit welcher Schrittweite simuliert wird. Für die Modellierung der Flankensteilheit ist es beispielsweise notwendig, den Bereich um Null möglichst exakt im Modell wiederzugeben, da von diesem die korrekte Wiedergabe des Überschwingens abhängt. Deshalb ist im Ablaufplan festgelegt, daß dieser Bereich mit größtmöglicher Genauigkeit simuliert wird. Andere Bereiche, in welchen die Slewrate sich nicht signifikant verändert, werden mit entsprechend großen Schrittweiten abgearbeitet.

Über die konkreten Eingangssignale hinaus enthält der Charakterisierungsplan die Formeln für die mathematischen Berechnungen, die auf den vom Simulator erzeugten Tabellen ausgeführt werden. Das folgende Programmfragment (aus dem Charakterisierungsprogramm, in C-ähnlicher Notation) zeigt die Formeln für die Berechnung der Flankensteilheit und der Verzögerungszeit:

```
/* Berechnen der Slewrate */
v10 = voutmin + 0.1 * (voutmax - voutmin);
v90 = voutmin + 0.9 * (voutmax - voutmin);
xofy (TIM, UOUT, v10, &t10 );
xofy (TIM, UOUT, v90, &t90 );
```

```
sl_1090 = abs(v90 - v10) / (t90 - t10);

/* Verzoegerungszeit */
td = t90 - (v90 - voutmin) / sl_1090;
```

Der Charakterisierungsplan muß unter bestmöglicher Ausnutzung der vorhandendenen Rechnerkapazitäten automatisch ausgeführt werden [HGT91]. Dazu steht z.B. die Charakterisierungssprache Clang zur Verfügung [Ger90]. Clang wurde um eine Verbindung zu Mathematica erweitert, um innerhalb einer Umgebung alle notwendigen Rechenschritte durchführen zu können [Ros94b]. Somit können beliebige Berechnungen - sogar symbolische - durchgeführt werden, die Zwischenergebnisse können schnell visualisiert werden, um eine optische Kontrolle zu haben und Tendenzen früh erkennen zu können. Befehle für die Steuerung des Netzwerksimulators und für die Berechnung der charakteristischen Eigenschaften stehen zur Verfügung.

Das folgende Listing zeigt die Hauptschleife eines typischen Clang Programmes, welches die Start- und Endwerte eines Spannungssprunges sowie die Schrittweite der Simulation für einen Teil des Ablaufplanes festlegt und die parametrisierte Funktion *simslew* aufruft, welche den Netzwerksimulator steuert und die Berechnungen der charakteristischen Größen durchführt.

```
program
real vx_min, vx_max, delta;
vx_min = -10;
vx_max = 10;
delta = 0.00001;
while (vx_max > 0) do
   simslew(vx_max, vx_min, vx_step);
   vx_max = vx_max - delta;
   vx_min = vx_max * (-1);
enddo;
end
```

Nachteil dieser Lösung ist die fehlende Möglichkeit zum Festlegen nebenläufiger Aufgaben, d.h. für jeden zur Verfügung stehenden Rechner muß ein individueller Charakterisierungsplan erstellt werden, die Ergebnisse müssen später zusammengefaßt werden. Darüberhinaus ist das Erlernen einer neuen Sprache notwendig, welche keine visuellen Hilfswerkzeuge unterstützt. Clang unterstützt darüberhinaus ausschließlich die Sun Rechnerplattform.

Die Einschränkungen der Clang Charakterisierungsumgebung, insbesondere die Forderung nach einer visuellen Unterstützung beim Entwurfsprozeß des Charakterisierungsplanes, führten zur Entwicklung der visuellen Simulationsumgebung ViCE [Goe01]. Hier fließen einer hoher Grad an visuellen Elementen zusammen mit flexibler Handhabung der erzeugten Daten, ein transparenter Zugang zu den mathematischen Funktionen und ein direkter Zugriff auf die Methodenbibliothek zusammen. Die zahlreichen Simulationen im

Zeitbereich können einfach durch das Verteilen auf mehrere Rechner parallelisiert werden. ViCE enthält einen effizienten Algorithmus, der automatisch die anfallenden Aufgaben auf die im Netzwerk zur Verfügung stehenden Ressourcen verteilt. Aufgaben, die auf einer einzelnen Workstation Stunden bis Tage erfordern, können so ohne zusätzlichen Aufwand des Entwicklers auf Minuten bis Stunden reduziert werden. Auch umfangreiche Schaltungen können so charakterisiert werden. Abbildung 31 zeigt den zur Erzeugung der Methode *slewrate* im Beispiel Operationsverstärker verwandten Charakterisierungsplan. Die Elemente werden per Mausklick plaziert und mittels der "Drag&Drop" Technologie verbunden, wobei diese Verbindungen der Datenfluß festlegen. So dient das Ergebnis der Transientensimulation als Eingabe für die mathematische Funktion, welche die zur Eingangsgröße gehörende Slewrate berechnet. Diese Berechnungen erfolgen in einer großen Hauptschleife, die den Definitionsbereich der Eingangsgröße abdeckt. Das Ergebnis dieser Schleife ist eine Tabelle gemäß (4.11), welche der Methodenbibliothek als Eingabe dient. Die Hauptschleife wird parallel ausgeführt, der Anwender braucht lediglich festlegen, welche der zur Verfügung stehenden Workstations belegt werden dürfen [GH94].

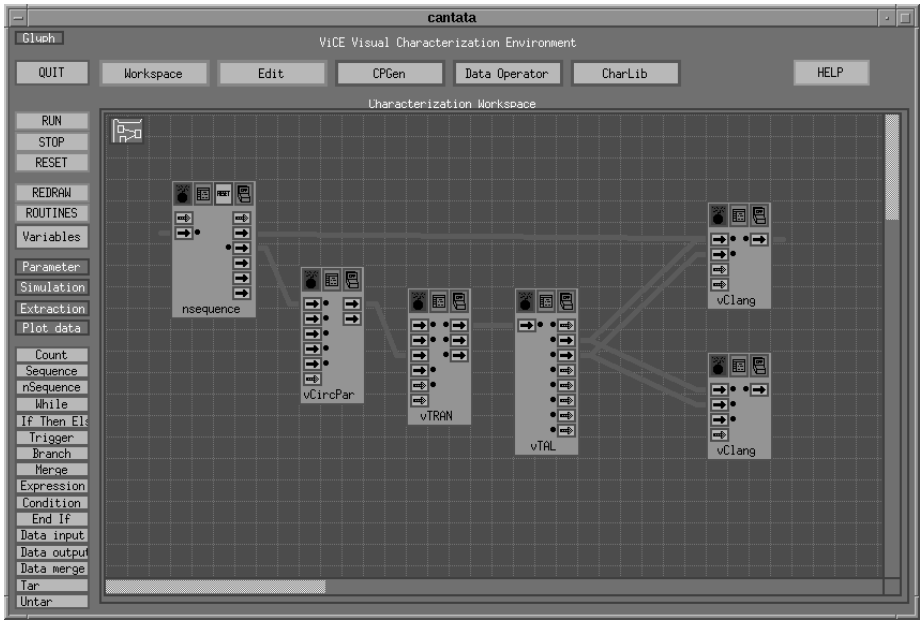

Abbildung 31: Visuelle Unterstützung beim Entwurf des Charakterisierungsplans.

4.3 Modellerzeugung

Nach Abschluß der Charakterisierung stehen alle Daten zur Verfügung, welche für ein Funktionsblockmodell nach Definition 3 benötigt werden. Die Parameterabhängigkeiten liegen zunächst als Tabellen vor. Tabellen benötigen jedoch viel Speicherplatz, dazu dauert das Nachschauen lange, denn die Parameterfunktionen können nicht im Modell codiert werden, sondern sind immer auf externe Speichermedien (Festplatte etc.) angewiesen.

Zur Steigerung der Effizienz werden aus der Datenbasis mathematische Modelle für die Parameterfunktionen erzeugt, welche in HDL darstellbar sind.

Ausgehend von allgemeinen Verfahren zur mathematischen Modellbildung wird in diesem Abschnitt der Begriff der Methode eingeführt und der Leistungsumfang der Methodenbibliothek vorgestellt, einschließlich dem Codegenerator. Das Ergebnis ist die Repräsentation des Funktionsblockmodells in einer HDL.

4.3.1 Mathematische Modellbildung

Die Grundaufgabe besteht darin, einen Datensatz mit einem mathematischen Modell zu beschreiben. Folgende Verfahren sind allgemein bekannt, um Datensätze mathematisch zu modellieren:

Look-Up Tabellen Die Tabellen werden bei der Charakterisierung erzeugt, siehe oben. Aus diesen Tabellen extrahiert ein Algorithmus relevante Datenpunkte und speichert diese. Um den Wert einer abhängigen Variable zu berechnen, wird in der Tabelle "nachgeschlagen". Hauptnachteile sind der große Speicherplatz, der für die Tabellen benötigt wird und die Beschränkung der Genauigkeit auf die in der Tabelle vorliegenden Daten. Dazu kommt, daß keine effiziente Implementierung in HDL möglich ist, sondern die Tabellen immer extern auf Speichermedien ausgelagert sind.

Interpolation Interpolationsverfahren erzeugen ein Polynom vom Grad n, welches an den n Stützstellen $x_1 \dots x_n$ exakt die Werte des Datensatzes $y_1 \dots y_n$ annimmt. Dieses Verfahren ist für die Parameterfunktionen nicht geeignet, da bei einer großen Anzahl von Datenpunkten ein Polynom des entsprechenden Grades entsteht, das nicht effizient auswertbar ist.

Spline Splines reduzieren den Datensatz auf relevante Stützstellen und liefern abschnittsweise definierte Polynome [Bli94]. Folgende Probleme treten bei der Verwendung von Splines für die Parameterfunktionen auf:

- Bei der Kompression des Datensatzes wird ein Approximationsfehler vorgegeben, höhere Genauigkeiten und andere Fehlernormen sind danach nicht realisierbar, da der ursprüngliche Datensatz gelöscht wird.
- Das Generieren der Splines erfolgt während der Simulation und ist ineffizient.
- Es werden Tabellen als Datenbasis benötigt, der Speicherplatz dafür ist trotz Reduktion der Datenpunkte beträchtlich.

Response Surfaces Response Surface Methoden dienen zur Untersuchung des Zusammenhangs zwischen einer oder mehrerer abhängiger Variablen (den Antworten) und

einer Menge von unabhängigen Variablen. Im Kontext der Modellierung elektronischer Schaltungen entsprechen die Antworten den Ausgangsgrößen des Blocks, während die unabhängigen Variablen die Eingangsgrößen und Betriebsparameter (Versorgungsspannung, Gleichtaktspannung, Temperatur etc.) enthalten. Die Hauptaufgabe der Response Surface Methode besteht darin, das Verhalten der Antworten in Abhängigkeit der unabhängigen Variablen zu modellieren, siehe [BD87]. Im Gegensatz zu den vorgestellten Methoden Interpolation und Spline ist die Anwendung nicht auf die Modellierung eines direkten 1:1 Verhältnisses wie Eingangsspannung zu Ausgangsspannung einer DC-Analyse beschränkt, sondern erlaubt eine dreidimensionale Darstellung: Mit dem Faktor Eingangsspannung auf der Abszisse und beispielsweise der Gleichtaktspannung auf der Ordinate läßt sich die Antwort Ausgangsspannung als 3D-Oberfläche darstellen. Graphisch können über Farbkodierungen oder Animationen weitere unabhängige Variablen hinzugefügt werden. Abbildung 32 zeigt ein Beispiel für ein Surface Response Modell: Links eine 3D-Oberfläche, rechts eine Kontur- bzw. Isoliniengraphik. Für das Wertepaar Eingangsspannung Gleichtaktspannung ist der resultierende Wert des Ausgangs dargestellt, basierend auf den Ergebnissen der entsprechenden DC-Analysen.

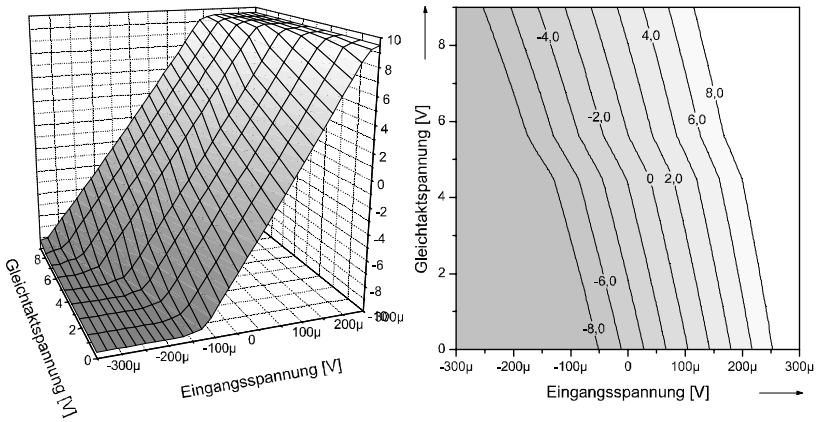

Abbildung 32: Oberfläche und Kontur des Surface-Response Modells.

Zur Anwendung der Response Surface Methoden sind die folgenden Schritte notwendig [Mon91]:

- Festlegen, welches die unabhängigen und abhängigen Variablen sind und welche Faktoren Einfluß auf die Antworten haben.
- Auswahl eines passenden Response Surface Designs - Standarddesigns sind Central Composite Designs und Box-Behnken Designs [KC87].
- Durchführen der Charakterisierung, um die notwendige Datenbasis zu schaffen.

- Anpassen des gewählten Designs an die während der Charakterisierung gewonnenen Daten.
- Umwandeln der mathematischen Modelle in Simulatorcode.

Die Mathematik zum Erzeugen der Modelle basiert auf einer Regressionsanalyse. Das Modell enthält lineare Terme, quadratische Terme und wechselseitige Beziehungen. Existieren vier abhängige Variablen $\{A, B, C, D\}$, so enthält das Modell die in Tabelle 10 gelisteten 14 Terme.

Modelltyp	**Resultierende Terme im Modell**
linear	$\{A, B, C, D\}$
quadratisch	$\{A*A, B*B, C*C, D*D\}$
wechselseitig	$\{A*B, A*C, A*D, B*C, B*D, C*D\}$

Tabelle 10: Zusammensetzung eines Response Surface Modells.

Response Surface Methoden bieten als großen Vorteil die Möglichkeit, direkte Beziehungen zwischen mehr als einer abhängigen Variable herzustellen. Nachteile sind ein großer Charakterisierungsaufwand, denn die Anzahl der notwendigen Simulatorläufe wächst exponentiell mit der Anzahl der abhängigen Variablen. Die Modelle sind effizient ausführbar, jedoch nicht an geänderte Bedingungen anzupassen: Für jede Änderung ist das Neuerstellen notwendig. Das Modellieren von indirekten Abhängigkeiten ist nicht vorgesehen.

Regression Gegeben ist eine parametrisierte Modellfunktion und der Datensatz, gesucht sind Werte für die Parameter, sodaß der Fehler zwischen Modellfunktion und Datensatz minimiert wird. Die Berechnung des Fehlers kann mit unterschiedlichen Fehlernormen erfolgen. Das Anpassen der Parameter der Funktion an den Datensatz wird in der Mathematik Regression genannt, wobei zwischen der linearen und nichtlinearen Regression unterschieden wird. In einem lineares Regressionsmodell wirken die Parameter linear auf die Funktionen ein. Ein Beispiel für ein lineares Regressionsmodelle ist

$$y = a + bu + cu^2 + de^u. \tag{4.9}$$

In (4.9) sind a, b, c und d die unbekannten Parameter der Modellgleichung, u ist die unabhängige Variable (oft auch Regressor genannt), y ist die abhängige Variable. (4.9) ist ein lineares Regressionsmodell, da der Parameter d linear auf die Exponentialfunktion einwirkt. In nichtlinearen Regressionsmodellen erscheinen die Parameter in einem nichtlinearen Zusammenhang. Ein Beispiel für ein nichtlineares Regressionsmodell ist

$$y = \sigma_1 + \sigma_2 e^{-\sigma_3 u}. \tag{4.10}$$

Der Vergleich zwischen (4.9) und (4.10) macht den Unterschied deutlich. In (4.9) wirkt d linear auf die Exponentialfunktion ein, in (4.10) steht der Parameter σ_3 im Exponent der

Funktion und nimmt exponentiellen, und damit nichtlinearen Einfluß auf das Ergebnis der Funktion.

Der Regression liegt ein Datensatz der Form (T_u, T_y) zugrunde. Die Parameterfunktion T_y ist die abhängige Variable, die in Abhängigkeit von dem Eingang T_u beschrieben wird.

Die folgenden Abschnitte stellen die mathematischen Grundlagen für die Anwendung der Regressionsmodelle vor.

Least Squares Fit-Algorithmen Diese Algorithmen minimieren den geringsten quadratischen Fehler zwischen der Tabelle und einer parametrisierten mathematischen Funktion. Über den *maximalen* prozentualen Fehler der erzeugten Funktion kann jedoch keine Aussage gemacht werden.

Min-Max Fit-Algorithmen Diese Algorithmen minimieren den maximalen Fehler. Andere Abstandsmaße können nicht vorgegeben werden, der minimale maximale Fehler wird als absoluter Wert geliefert, der *prozentuale* maximale Fehler kann jedoch an einer anderen Stelle des Datensatzes auftreten.

Da keines der Verfahren unmodifiziert anwendbar ist, wurde ein Vorgehen entwickelt, das auf den üblichen Fit-Algorithmen basiert, aber Vorgaben hinsichtlich des Fehlers (Größe, Fehlernorm, Abstandsmaß) einhält, indem es den Datensatz in geeignete Segmente zerlegt. Im folgenden Abschnitt wird dieses Verfahren vorgestellt.

4.3.2 Die Methodenbibliothek

Die Methodenbibliothek [Ros94a] wird dann verwandt, wenn zwischen den abhängigen Größen (Ausgangsspannung, Slewrate, Parameter eines Regressionsmodells, etc.) und den unabhängigen Größen (Eingangsspannung, Umgebungstemperatur, etc.) ein funktioneller Zusammenhang folgender Form existiert:

$$y = f(x) \tag{4.11}$$

Ein solcher Zusammenhang wird mittels einer **Methode** beschrieben. Sie wird mathematisch wie folgt definiert:

Definition 8 (Methode) *Der Definitionsbereich der Eingangsgröße ist in Segmente aufgeteilt. Segmente sind definiert:*

$$\omega_i : (x_{i1}, x_{i2}] = \{x | x \in I\!R, x_{i1} < x \leq x_{i2}\} \tag{4.12}$$

Ein Segment wird auf eine mathematische Modellfunktion abgebildet:

$$\varphi_i : \omega_i \rightarrow f_i(x) \tag{4.13}$$

Eine Methode $M = \{\Omega, F\}$ *ist eine Menge bestehend aus Segmenten* $\Omega = \{\omega_j : 1 \leq j \leq n\}$ *und Funktionen* $F = \{\varphi_k : 1 \leq k \leq m\}$ *mit den folgenden Eigenschaften:*

Vollständigkeit: *$\mathbb{D}_f$ sei der Definitionsbereich der zu modellierenden Eingangsgröße. Es gilt: $x \in \mathbb{D}_f \Rightarrow \exists i : x \in \omega_i$*

Eindeutigkeit: *Für jedes Paar (i, j) von Segmenten mit $i \neq j$ gilt: $x \in \omega_i \Rightarrow x \notin \omega_j$*

C_0-Stetigkeit: *Für zwei aufeinanderfolgende Segmente $\omega_1 : (x_i, x_j] \rightarrow f_1(x)$ und $\omega_2 : (x_j, x_k] \rightarrow f_2(x)$ gilt $\lim_{x \rightarrow x_j} f_2(x) = f_1(x_j)$. Diese Eigenschaft ist wichtig, um die Konvergenz der Simulationsläufe zu sichern.*

Genauigkeit: *Für jeden Wert x_i der Eingangsgröße, den entsprechenden Wert der Ausgangsgröße y_i und einen maximalen Fehler ϵ gilt: $f(x_i) - y_i \leq \epsilon$. Unterschiedliche Fehlernormen (l_1, l_2 und l_∞) [Box78] und Abstandsmaße (absolut und Euklidische Distanz) [JRS91] können zur Berechnung von ϵ gewählt werden.*

Als Ausgang der Charakterisierung nach (4.2) steht die Datenbasis der Parameterfunktionen in der Form (T_u, T_y) zur Verfügung. Die Methodenbibliothek erzeugt aus diesen Tabellen eine Methode, die den oben genannten Bedingungen genügt. Die mathematische Hauptaufgabe beim Erzeugen der Methoden ist die Kurvenanpassung [Rat83]. Standardmäßig werden lineare und polynomielle Modelle verwandt, damit die Methoden möglichst schnell in der Ausführung sind und einfach auf die Modellierungssprache des Simulators abgebildet werden können. Der Anwender kann jedoch auch beliebige nichtlineare Regressionsmodelle zur Anpassung verwenden, um spezielle Verhaltensmuster nachzubilden. Dies ist ein besonderes Merkmal der Methoden-Bibliothek, wie sie hier implementiert wurde.

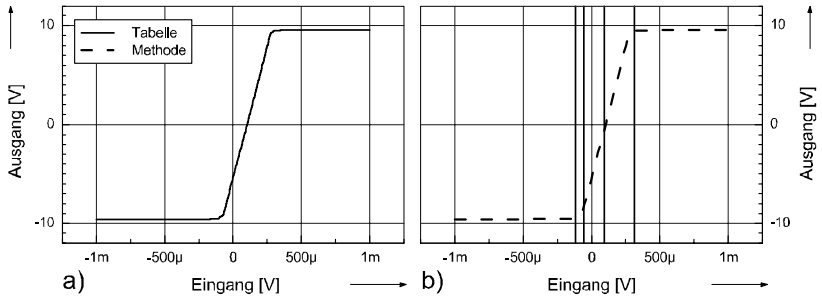

Abbildung 33: a) Tabellarisches Ergebnis der DC-Analyse. b) Mittels der Methodenbibliothek erzeugte Methode DC.

Die DC-Analyse ist ein gutes Beispiel für eine solche direkte Abhängigkeit nach Gleichung (4.2). Abbildung 33 a) zeigt die graphische Repräsentation des vom Simulator erzeugten Ergebnis der DC-Analyse eines Operationsverstärker. Aus diesem Datensatz erzeugt die Methodenbibliothek eine Methode, welche hier aus fünf Segmenten, deren Größen den vorgegebenen Anforderungen an die Genauigkeit entsprechen, besteht. Abbildung 33 b) zeigt das Ergebnis, welche für jeden Wert im Definitionsbereich des Eingangssignals den Wert der DC-Analyse als Ergebnis liefert.

4.3.3 Codegenerator

Der Anwender möchte das erzeugte Modell in seiner gewohnten Simulator-Umgebung verwenden. Der Codegenerator konvertiert dazu die Modellgleichungen in gängige HDL. Die Methode wird von der Methodenbibliothek als Liste erzeugt, die Informationen über den Fehler enthält und die Segmente mit den entsprechenden Modellgleichungen. Der Codegenerator kann Quellcode für die folgenden HDL erzeugen:

- **ELDO-FAS** Es werden Codefragmente erzeugt, die direkt in eine Datei des Netzwerksimulators Eldo integriert werden können.
- **VHDL-AMS** Es werden Codefragmente erzeugt, die direkt in VHDL-AMS integriert werden können.
- **C** C-Code ist universell einsetzbar. Er kann zum Netzwerksimulator gebunden werden, wodurch die Methode als externe Funktion angesprochen und sehr schnell wird. Der Code kann aber auch als externe Funktion in Mathematica installiert werden, oder von beliebigen C-Programmen aufgerufen werden.
- **Mathematica** Erzeugt wird eine Funktion im Mathematica Format. Die Methode kann so in Mathematica direkt visualisiert und mit dem Ausgangsdatensatz optisch verglichen werden.

Da das Erzeugen der Methoden in Mathematica erfolgt, wurde der Codegenerator als Komponente der Methodenbibliothek implementiert. Mathematica wird dazu mittels dem Package [Sof94] Ausgabefunktionen erweitert, die eine Umwandlung der Methoden in effizienten Programmcode unterstützen:

- **ANSI C Code.** Es wird überprüft, ob die Funktionen des Mathematica Ausdruckes dem ANSI Standard entsprechen, wenn nicht, wird eine entsprechende Warnung ausgegeben. Entsprechen Sie dem ANSI Standard, so werden sie korrekt übersetzt. Insbesondere alle Arten von logischen Ausdrücken werden korrekt übersetzt [Str97].
- **Korrekte Handhabung von Potenzen**. Potenzen der Form $x^{\frac{1}{2}}$ und $x^{-\frac{1}{2}}$ werden erkannt und in Ausdrücke der Form $\sqrt{x}$ und $\frac{1}{\sqrt{x}}$ umgewandelt, alle ganzzahligen Exponenten und rationalen Zahlen werden in den C Datentyp double konvertiert.
- **Direkte Zuweisungen.** Der Name der Variablen kann übergeben werden, welcher im C Programm der Wert des Ausdruckes zugewiesen werden soll.
- **Konvertierung der Segmente**. Mathematica Listen werden automatisch in C Felder konvertiert.

Mit Hilfe dieser Funktionen können jetzt die Mathematica Ausdrücke, die eine Methode beschreiben, in die HDL des Simulators übersetzt werden.

4.4 Genauigkeit des Funktionsblockmodells

Um die Genauigkeit des Funktionsblockmodells zu verifizieren, ist ein Vergleich mit den - vom Simulator erzeugten - Originaldaten notwendig. $\widehat{y}(t)$ ist im folgenden die Referenzkurve über dem Intervall $t \in [a, b]$, $y(t)$ ist das entsprechende Signal des Funktionsblockmodells.

Als Maß für den Abstand der Kurven wird d_p^m definiert:

$$d_p^m(y, \widehat{y}) = \|z_m\|_p \tag{4.14}$$

In (4.14) fließen mit z_m unterschiedliche Abstandsmaße und mit $\|...\|_p$ verschiedene Fehlernormen ein. Das Abstandsmaß legt fest, wie die Differenz zwischen einem beliebigen Punkt der Modellkurve $y^{(n)}$ und der Referenzkurve $\widehat{y}$ berechnet wird. Die Fehlernorm legt fest, wie die mit dem Abstandsmaß berechneten Differenzen zu einer Aussage über die Größe des Fehlers, und damit zur Genauigkeit des Funktionsblockmodells, verrechnet werden. Sie dienen als Maß für die Güte des Funktionsblockmodells. Die folgenden Abstandsmasse und Fehlernormen sind in der Methodenbibliothek implementiert.

4.4.1 Fehlernormen

Die Fehlernorm legt fest, wie die Differenzen zwischen Referenzkurve und Modellkurve des Funktionsblockmodells zu einer Aussage über die Größe des Fehlers, und damit zur Genauigkeit des Funktionsblockmodells, verrechnet werden. Sie dienen als Maß für die Güte des Funktionsblockmodells. Die folgenden Fehlernormen sind in der Methodenbibliothek implementiert.

l_1 Fehlernorm Die Idee der l_1 Fehlernorm ist es, die Differenz zwischen allen Datenpunkten und den entspechenden Funktionswerten zu summieren. Sie wird auf die Differenzfunktion z angewandt, über den gesamten Definitionsbereich:

$$\|z\|_1 = \int_{t=a}^{b} |z(t)|\, dt \tag{4.15}$$

Bei der Anwendung dieser Fehlernorm wirkt der Fehlers jedes Punktes gleich stark auf das Ergebnis, unabhängig vom jeweiligen Betrag.

l_2 Fehlernorm Dieses Verfahren bildet die Grundlage für Methoden, die auf dem geringstenquadratischen (least squares, LS) Fehler beruhen. Die vertikale Differenz zwischen Referenz- und Modellkurve wird zunächst quadriert. Anschließend wird das Integral über die quadrierte Differenz gebildet. Der Fehler ist die Quadratwurzel des Integrals:

$$\|z\|_2 = \sqrt{\int_{t=a}^{b} z(t)^2 dt} \tag{4.16}$$

Im Gegensatz zu der l_1 Norm machen sich hierbei in der Summe große Abweichungen an einzelnen Punkten durch die Quadrierung sehr deutlich bemerkbar.

Auf der LS-Abstandsnorm beruhen viele in der Praxis angewandte Minimierungsverfahren zum Erzeugen von Fit-Funktionen. Die lineare Regression arbeitet mit diesem Verfahren, in der nichtlinearen Regression wird es als Basis der Verfahren Gauss-Newton und Chi^2 angewandt. Die prinzipielle Idee ist es, das Fitten als Optimierungsproblem zu definieren. Die zu minimierende Funktion ist die LS- Fehlerfunktion.

l_∞ Fehlernorm Die Idee der l_∞ Fehlernorm ist es, den maximalen vertikalen Abstand zwischen zwei Kurven zu betrachten:

$$\|z\|_\infty = \max_{t\in[a,b]} |z(t)| \tag{4.17}$$

Auf der l_∞ Fehlernorm beruhen *MinMax* Fit Algorithmen, die den maximalen Fehler zwischen zwei Kurven minimieren.

Die Methoden-Bibliothek wendet die l_∞ Fehlernorm an, wenn der Anwender keine Fehlernorm als Option angibt. Allerdings tritt dabei das Problem auf, daß die gewünschte Genauigkeit der Modellgleichungen als maximale Abweichung in Prozent spezifiziert wird. Wird der maximale Fehler nach (4.17) berechnet, so sagt dieser Wert nichts über die prozentuale Abweichung aus. Insbesondere bei einem steilen Übergang zwischen positiven und negativen Werten treten prozentual große Abweichungen um den Nulldurchgang der Kurve auf. Die l_∞ Fehlernorm wird deshalb modifiziert zu:

$$\|\tilde{z}\|_\infty = \max_{t\in[a,b]} \left|\frac{z(t)}{x(t)}\right| \tag{4.18}$$

Im Bereich $x(t) \in [-\varepsilon, \varepsilon], \varepsilon \to 0$ kann (4.18) nicht angewandt werden, da der Netzwerksimulator normalerweise keine Ergebnisse mit der notwendigen Genauigkeit liefert. Damit diese gerundeten Werte das Ergebnis nicht verfälschen, wird ein Schwellwert ε als Ergänzung zu (4.18) definiert: $\frac{z(t)}{x(t)} = 0, z(t) \in [-\varepsilon, \varepsilon], \varepsilon \to 0 \vee x(t) = 0$. Mit dieser Ergänzung wird auch der Fall $x(t) = 0$ abgedeckt, in dem es nicht möglich ist, die prozentuale Abweichung zu berechnen. Den Schwellwert ε gibt der Anwender in Abhängigkeit der maximalen Genauigkeit der Simulationsergebnisse explizit vor.

Die Berechnung der l_∞ Fehlernorm ist effizient, da lediglich Grundrechenarten und Vergleiche notwendig sind.

Abbildung 34 enthält eine Referenzkurve $\widehat{y}$ mit der Modellkurve y. $\widehat{y}$ gibt einen Sprung am Ausgang wieder, wobei sich $\widehat{y}$ innerhalb einer bestimmten Zeit auf diesen Wert ein-

schwingt. Dies entspricht einer gedämpften Schwingung. In der Praxis sind solche Einschwingvorgänge häufig, zum Beispiel werden sie beim Operationsverstärker durch die Settling-Time charakterisiert. y gibt dieses Einschwingen nicht wieder, es ist eine idealisierte Darstellung von $\widehat{y}$.

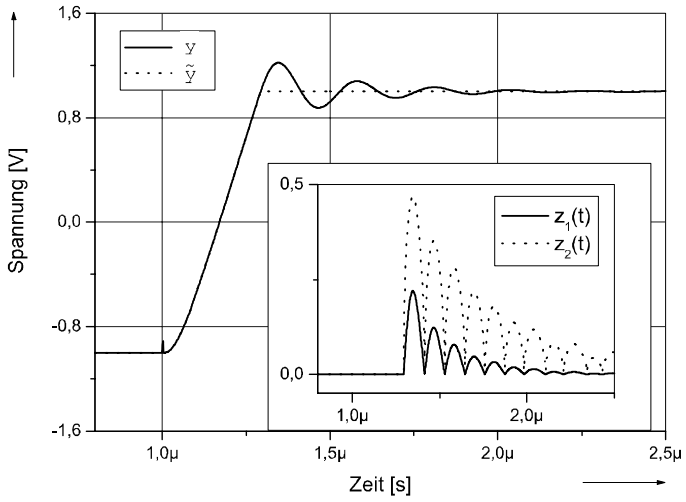

Abbildung 34: Vergleich der Fehler zwischen Datensatz und Modellfunktion.

Es soll jetzt mit den Fehlernormen überprüft werden, ob y eine geeignete Modellgleichung für $\widehat{y}$ darstellt. Dazu wird die Differenzfunktion z nach Gleichung (4.19) gebildet. Sie ist in Abbildung 34 in der eingefügten Graphik aufgetragen, wobei gilt: $z_1 = z$ und $z_2 = z^2$.

Da die $l_1$und l_2 Fehlernormen auf einer Summierung von z_1 bzw. z_2 beruhen, wird anhand der Kurven deutlich, daß diese Integrale einen großen Wert annehmen, der impliziert, daß y eine ungeeignete Modellgleichung für $\widehat{y}$ ist. Der Grund hierfür liegt im Einschwingvorgang: Da der Betrag der Amplitude für die Differenz verwandt wird, summiert sich der Fehler auf einen hohen Wert. Abhilfe schafft die Erweiterung der Formeln zum Bestimmen des Fehlers um die im folgenden Abschnitt vorgestellten Abstandsmaße.

4.4.2 Abstandsmaße

Gegeben seien zwei Kurven $y(t)$ und $\widehat{y}(t)$ über dem Intervall $t \in [a, b]$. Dabei bezeichnet $\widehat{y}$ die Referenzkurve, die während der Charakterisierung generiert wird, und y die entsprechende Kurve des Funktionsblockmodells. Zunächst wird die Differenzfunktion $z(t)$ gebildet. Dies erfolgt mit einem Abstandsmaß.

Vertikaler Abstand Das gängigste Abstandsmaß ist der absolute Abstand der Ordinaten, das den Abstand wie folgt berechnet:

$$z(t) = y(t) - \widehat{y}(t) \tag{4.19}$$

Diese Differenzfunktion mißt den vertikalen Abstand zwischen der Referenzkurve und der Modellkurve und wird in die Formel der Fehlernorm eingesetzt.

Euklidische Distanz als Abstandsmaß Die Euklidische Distanz ersetzt die Funktion z in (4.19), die bisher zur Berechnung der Differenzkurve verwandt wurde. Sie bezeichnet den kürzesten Abstand zwischen einem Punkt der Referenzkurve und der Modellkurve, z.B. [JRS91]. Der neue Operator z_E sei für einen Punkt t_i wie folgt definiert:

$$z_E(t_i) = \min_{t \in [a,b]} \sqrt{z^2(t) + s^2(t - t_i)^2} \tag{4.20}$$

Abbildung 35 zeigt ein Beispiel für die Berechnung des Operators im Punkt t_0. Der Radius des Kreises ist dabei die Euklidische Distanz.

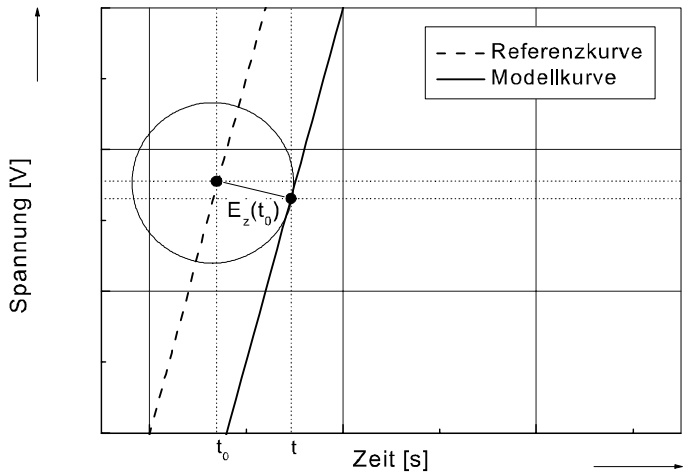

Abbildung 35: Berechnung des Fehlers mittels der Euklidischen Distanz.

Sie ist jedoch nicht geeignet, wenn das Verhalten des Modells im Hochfrequenzbereich betrachtet werden soll, da sie als Tiefpaß-Filter wirkt.

Für alle $t \in [a, b]$ gilt: $0 \leq E_z(t) \leq z(t)$. Die Euklidische Distanz wird mit einer der drei Fehlernormen zu einem Abstandsmaß d_1^E, d_2^E, d_∞^E oder $\widetilde{d}_\infty^E$ kombiniert. Die Definition der Abstandsmaße kann jetzt um die Familie der Maße mit Euklidischer Distanz erweitert werden:

$$d_p^E(y, \widehat{y}) = \left\| E_{y-\widehat{y}}(t) \right\|_p = \|E_z\|_p \tag{4.21}$$

In [JRS91] wird gezeigt, daß $E_z(t)$ immer stetig ist, auch wenn $z_E(t)$ nicht stetig ist. $E_z(t)$ ist ein nichtlinearer Tiefpaß-Filter, der unerwünschte Hochfrequenzeffekte, die in $z_E(t)$ auftreten, filtert. $E_z(t)$ ist nichtlinear, da im allgemeinen $E_{\alpha z} \neq |\alpha E_z|$ gilt. Eine Verschiebung entlang der Ordinate bewirkt lediglich einen Impuls mit geringer Amplitude, eine Differenz zwischen den Kurven fließt unvermindert in den Abstand ein.

Der Nachteil der Euklidischen Distanz ist der im Vergleich zur Differenzfunktion z nach (4.19) sehr hohe Rechenaufwand. Das Bestimmen von $E_z(t)$ ist ein Optimierungsproblem, das mit bekannten Linear Programming Algorithmen gelöst werden kann. Die Lösung ist mit dem entsprechenden Aufwand für jeden einzelnen Punkt verbunden. Insbesondere wenn sehr umfangreiche Datensätze behandelt werden, fällt dieser Aufwand negativ ins Gewicht. Für die Laufzeit einer Simulation ist dieser Aufwand unerheblich, da die Differenzfunktionen nur während der Charakterisierung benötigt werden.

Die Funktion als Tiefpaß-Filter und das Tolerieren eines geringen Fehlers entlang der Ordinate in der Modellgleichung sind die Vorteile der Euklidischen Distanz. Wird ein $\widetilde{d}_p^E$ Maß bei der Erzeugung einer Methode verwandt, so haben die generierten Modellgleichungen eine wesentlich kompaktere Form und sind während der Simulation effizienter zu berechnen.

Einfluß des Skalierungsfaktors Der Skalierungsfaktor s in (4.20) bewirkt ein Ausgleich zwischen den Einheiten der x- und y-Achsen. Ist beispielsweise eine Spannung über der Zeit aufgetragen, so hat die Spannung typischerweise die Einheit Volt, während die Zeit in Nanosekundenbereich liegen kann. Wird die Zeit ohne Skalierung verwandt, so ist die kürzeste Entfernung zwischen Punkt und Kurve in der Regel der Abstand parallel zur x-Achse, da die Zeit hier mit einer um den Faktor 10^{-9} kleineren Einheit einfließt. Um den gleichmäßigen Einfluß beider Größen zu gewährleisten, kann $s = \frac{\Delta x}{\Delta y}$ gesetzt werden, Δx und Δy sind die Einheiten der Achsen. Für das obige Beispiel Volt über Nanosekunden ergäbe sich $s = \frac{1V}{1\eta s} = 1 * 10^8 \frac{V}{s}$. Für $s \to \infty$ verhalten sich E_z und z identisch.

Horizontales Abstandsmaß Dieses Abstandsmaß berechnet die Verschiebung entlang der Abszisse. Sie wird dann der Berechnung des Fehlers zugrunde gelegt, wenn die Verschiebung ohne Einfluß der Ordinate berechnet werden soll. Der Wert für diese Verschiebung ergibt sich aus der Auflösung der Gleichung $f(x_1) = y_0$ nach x_1. Der Abstand ist die absolute Differenz zwischen x_0 und x_1. Aus dieser Herleitung ergibt sich der neue Operator H wie folgt:

$$z_H(t_0) = s\,|t_1 - t_0|\,, \text{wobei } f(t_1) = y_0. \tag{4.22}$$

Der horizontale Abstand wird mit einer der Fehlernormen zu einem Abstandsmaß kombiniert. Daraus ergeben sich folgende neue Abstandsmaße:

$$d_p^H(y, \widehat{y}) = \left\| H_{y-\widehat{y}}(t) \right\|_p \tag{4.23}$$

Die Abstandsmaße d_1^H, d_2^H, d_∞^H und $\widetilde{d}_\infty^H$, können durch Setzen der Optionen der Methodenbibliothek ausgewählt werden.

Der Skalierungsfaktor s in (4.22) hat dieselbe Bedeutung, wie der oben beschriebene Skalierungsfaktor der Euklidischen Distanz. Wird er auf einen konstanten Wert gesetzt, so bezieht sich das Abstandsmaß ausschließlich auf die x-Werte, die Dimension der y-Werte hat keinerlei Einfluß. Durch geeignete Wahl von s werden sie berücksichtigt.

Zusammenfassung Im vorigen Abschnitt wurde die Funktion $z(t)$ als Differenzfunktion verwandt. Sie wurde in (4.19) als Abstand zwischen den Werten der Referenz- und der Modellkurve an einem bestimmten Punkt definiert. Diese Berechnung des Abstandes entspricht dem Abstand zwischen den Kurven. Von der Methodenbibliothek werden alle Kombinationen der vorgestellten Abstandsmaße und Fehlernormen unterstützt, $\widetilde{d}_\infty$ ist die Standardeinstellung.

4.4.3 Fehlerfortpflanzung in mathematischen Modellen

Als Fehler eines mathematischen Modelles wird die Differenz zwischen dem berechneten Wert und dem tatsächlichen Wert bezeichnet. Als tatsächlichen Wert (Referenz) kann das während der Charakterisierung erzeugte Ergebnis betrachtet werden, das der Modellerzeugung zugrunde liegt. In der Definition der Methode auf Seite 8 wurden unter der Rubrik Genauigkeit die Anforderungen an den Fehler ϵ für eine Methode des Verhaltensmodells erfasst. Beim Erzeugen dieser Methode kann genau festgelegt werden, nach welchen Fehlernormen und Abstandsmassen ϵ berechnet wird und der maximale Fehler $\epsilon_{\max}$ kann festgelegt werden.

Allein diese Aussage ist jedoch nicht hinreichend, um zu einer Aussage über die Genauigkeit des Gesamtmodells zu gelangen, denn

- im Funktionsblockmodell nach (3.9) besteht $\delta()$ aus mehreren Methoden - die Fehler aus allen Methoden fließen in das Modell ein, und
- es können mehrere Instanzen eines Modells zu einem Gesamtsystem verschaltet werden, wobei die Fehler aus allen Instanzen in das Gesamtsystem einfließen.

Für den Schaltungsentwickler ist eine Aussage über den Gesamtfehler im System wichtig. Die Fehler der einzelnen Komponenten können additiv wirken und das Gesamtsystem unbrauchbar machen, können sich jedoch auch gegenseitig weitgehend aufheben. Die folgenden Abschnitte behandeln eine statistische Abschätzung für den Fehler des Gesamtsystems basierend auf Fehlerfortpflanzungsmodellen [Bev69] [Wei01] [Wol98].

Die Genauigkeit einer Methode $f \in \delta()$ in einem Zeitschritt t ist $d_p(f,g,t) = \|f(\overrightarrow{u}_i,t) - g(\Delta\overrightarrow{u}_i,t)_p\|$, die Differenz zwischen der Methode f, welche fehlerbehaftet ist, und dem Wert der Kurve g aus der Datenbasis, dem Referenzwert. d bezeichnet die Fehlernorm, siehe [Ros94a].

Für die statistische Betrachtung des Fehlers ist zunächst der mittlere Fehler, d.h. das Mittel $\overline{d}$ der Abweichung der beiden Kurven, relevant. f enthält die endliche Anzahl N von Datenpunkten. Somit ergibt sich

$$\overline{d} := \frac{1}{N}\sum_{t=1}^{N} d_p(f,g,t) \tag{4.24}$$

und daraus die Varianz s^2 des Fehlers zu

$$s^2 = \frac{1}{N-1}\sum_{t=1}^{N}(d_p(f,g,t) - \overline{d})^2. \tag{4.25}$$

Für die l_2 Fehlernorm ergibt sich beispielsweise die Varianz als $s^2 = \frac{N\left(z-\sqrt{N\,z^2}\right)^2}{-2+N}$ mit $z(t) = f(\overrightarrow{u}_i,t) - g(\Delta\overrightarrow{u}_i,t)$.

Generell ist jedoch der genaue Fehler in den Parametern nicht bekannt. Bekannt ist lediglich die Varianz für jeden Parameter. Mittels dieser Information muß die Ungewißheit des Modells des Gesamtsystems bestimmt werden.

Das Funktionsblockmodell FBM setzt sich aus mindestens zwei Methoden $M_1, M_2, ...$ zusammen. Die Charakteristik des Systems ergibt sich nun als funktionaler Zusammenhang

$FBM = h(M_1, M_2, ...)$.

Auch wenn das Mittel nicht immer korrekt sein muß, soll angenommen werden, daß der wahrscheinlichste Wert für FBM durch die Mittel der Methoden bestimmt werden kann zu

$\overline{FBM} = h(\overline{M}_1, \overline{M}_2, ...)$.

Die Ungewißheit im Ergebnis kann durch die Kombination der Einzelergebnisse in individuelle Werte für FBM_i ausgedrückt werden.

$FBM_i = h(M_{1(i)}, M_{2(i)}, ...)$.

Unter der Annahme, das unendlich viele Werte existieren, kann die Standardabweichung des Gesamtsystems bestimmt werden zu:

$$\sigma^2_{FBM} = \lim_{N\to\infty}\frac{1}{N}\sum(FBM_i - \overline{FBM})^2 \tag{4.26}$$

Die Abweichungen $FBM_i - \overline{FBM}$ lassen sich jetzt in Abhängigkeit der Abweichungen der enthaltenen Methoden $M_1 - \overline{M}_1$, $M_2 - \overline{M}_2$,...ausdrücken mit

$$FBM_i - \overline{FBM} \simeq (M_1 - \overline{M}_1)\frac{\partial FBM}{\partial M_1} + (M_2 - \overline{M}_2)\frac{\partial FBM}{\partial M_2} + ... \tag{4.27}$$

wobei die partiellen Ableitungen unter der Voraussetzung gebildet werden, daß alle anderen Methoden auf ihr Mittel $\overline{M}_i$ festgesetzt sind.

Mit den Gleichungen für die Standardabweichung der einzelnen Methoden

$$\sigma^2_{M_1} = \lim_{N\to\infty}\frac{1}{N}\sum(M_{1(i)} - \overline{M}_1)^2 \qquad \sigma^2_{M_2} = \lim_{N\to\infty}\frac{1}{N}\sum(M_{2(i)} - \overline{M}_2)^2$$

und der Kovarianz $\sigma^2_{M_1M_2} \equiv \lim_{N\to\infty}\frac{1}{N}\sum[(M_{1(i)} - \overline{M}_1)(M_{2(i)} - \overline{M}_2)]$ zwischen den Methoden ergibt sich als Näherung für die Standardabweichung durch Kombination von (4.26) und (4.27):

$$\begin{aligned}\sigma^2_{FBM} \simeq\ & \sigma^2_{M_1}\left(\frac{\partial FBM}{\partial M_1}\right)^2 + \sigma^2_{M_2}\left(\frac{\partial FBM}{\partial M_2}\right)^2 \\ & + 2\sigma^2_{M_1M_2}\frac{\partial FBM}{\partial M_1}\frac{\partial FBM}{\partial M_2} + ...\end{aligned} \tag{4.28}$$

Weitere Methoden außer M_1 und M_2 fließen analog dazu in das System ein. Unter der Annahme, daß M_1 und M_2 unkorreliert sind, so kann Gleichung (4.28) auf die dominanten Terme reduziert werden:

$$\sigma^2_{FBM} \simeq \sigma^2_{M_1}\left(\frac{\partial FBM}{\partial M_1}\right)^2 + \sigma^2_{M_2}\left(\frac{\partial FBM}{\partial M_2}\right)^2 \tag{4.29}$$

Gleichung (4.29) ist die allgemeine Form für die Fehlerfortpflanzung für eine beliebige verkettete Operation. Im folgenden werden die Grundrechenarten betrachtet.

Addition Unter Berücksichtigung einer Gewichtung a, b der Methoden M_1, M_2 ergibt sich $FBM = am_1 \pm bm_2$. Die partiellen Ableitungen aus (4.29) sind in diesem Fall die Gewichte a, b und die Gleichung kann umgeschrieben werden zu

$$\sigma^2_{FBM} = a^2\sigma^2_{M_1} + b^2\sigma^2_{M_2}. \tag{4.30}$$

Sind beide Methoden gleich gewichtet, d.h. $a = b = 1$, so kann Gleichung (4.30) weiter vereinfacht werden zu

$$\sigma^2_{FBM} = \sigma^2_{M_1} + \sigma^2_{M_2}. \tag{4.31}$$

Werden zwei Methoden addiert oder subtrahiert, so ergibt sich die Standardabweichung des Ergebnisses aus der Quadratwurzel der Summer der einzelnen Standardabweichungen zum Quadrat.

Beispiel: Verschaltung zweier Funktionsblöcke Die Funktionsblockmodelle zweier elektrischer Funktionsblöcke Block 1 und Block 2 werden in Serie geschaltet. Beide Funktionsblöcke haben einen Innenwiderstand R_1 bzw. R_2. Die Spannung, die an den beiden in Serie geschalteten Blöcken abfällt, ist $U = U_1 + U_2$. Existieren jetzt Methoden $DC1(U_{in})$ und $DC2(U_{in})$ für U_1 und U_2, welche mit einer Ungenauigkeit $\sigma_{U_1} = 0,05$ bzw. $\sigma_{U_2} = 0,025$ erzeugt worden sind, so kann die Ungenauigkeit der Gesamtschaltung mit Gleichung (4.31) bestimmt werden zu:

$$\sigma^2_U = 1\sigma^2_{U_1} + 1\sigma^2_{U_2} = 0,003125 \qquad \sigma_U = 0,0559017.$$

Die mittels der Methoden errechnete Spannung hat also eine maximale Ungenauigkeit von 5,6 %.

Soll andersherum garantiert werden, daß die Gesamtschaltung eine bestimmte Anforderung an die Genauigkeit erfüllt, so muß rückwärts gerechnet werden. Ausgehend von $\sigma_U = 0,1$ ergibt sich $\sigma^2_{U_1} + \sigma^2_{U_2} = 0.01$ und $\sigma_{U_1} = \sqrt{0.01 - \sigma^2_{U_2}}$. Mit $\sigma_{U_2} = 0.05$ ergibt sich die Anforderung, bei der Erzeugung von $DC1(U_{in})$ mit einer Genauigkeit von $\sigma_{U_1} = 0,0866$ zu arbeiten, also einen maximalen Fehler von 8,66 % zu erlauben.

Multiplikation Unter Berücksichtigung einer Gewichtung a der Methoden M_1und M_2 ergibt sich $FBM = aM_1M_2$. Die partiellen Ableitungen aus (4.29) enthalten die jeweils andere Methode und die Gleichung kann umgeschrieben werden zu

$$\sigma^2_{FBM} = a^2M^2_2\sigma^2_{M_1} + a^2M^2_1\sigma^2_{M_2}. \tag{4.32}$$

Analog dazu kann die Division berechnet werden mit

$$\sigma^2_{FBM} = \frac{a^2\sigma^2_{M_1}}{M^2_2} + \frac{a^2M^2_1}{\sigma^4_{M_1}}. \tag{4.33}$$

Beispiel: Spannungsteiler Die Formel für den Spannungsteiler lautet: $U_2 = \frac{U*R_1}{R_1+R_2}$. Sind in diesem Beispiel die Widerstände als Methoden $R1(\overrightarrow{u})$ und $R2(\overrightarrow{u})$ in Abhängigkeit der Eingangsspannung definiert, wie zum Beispiel der lastabhängige Innenwiderstand eines

Operationsverstärkers, so ergibt sich mit den Standardabweichungen σ_{R1} bzw. σ_{R2} für die Ungenauigkeit der Formel mittels (4.33) $\sigma_R^2 = \frac{\sigma_{R1}^2}{R1^2} + \frac{(R1+R2)^2}{(\sigma_{R1}^2+\sigma_{R2}^2)^2}$.

In diese Gleichung können die konkreten Werte eingesetzt werden. Haben $R1(\overrightarrow{u})$ und $R2(\overrightarrow{u})$ eine maximale Ungenauigkeit von 10%, so ergibt sich als Ungenauigkeit für die Formel mit $R1(\overrightarrow{u}) = 1K\Omega$ und $R2(\overrightarrow{u}) = 200\Omega$ als Ungenauigkeit $\sigma_R = \sqrt{\frac{0.1^2}{1^2} + \frac{0.012^2}{0.0004}} = 0.11669$, also über 11 %. Entsprechend kann bei spezifizierten Anforderungen an die Genauigkeit rückwärts gerechnet werden.

4.4.4 Funktionsblockmodell

Im folgenden werden die im Funktionsblockmodell enthaltenen Methoden `VoutDC`, `slewrate` und `TD` untersucht, um die Auswirkungen der einzelnen Elemente des Funktionsblockmodells auf die Genauigkeit des Gesamtsystems zu erkennen und die entsprechenden Konsequenzen für die Charakterisierung festzustellen.

Auswirkung der Methode VoutDC Die Methode `VoutDC` bewirkt im Funktionsblockmodell, daß die Ausgangsgrößen die richtigen Werte annehmen. Sie wird für DC-Analysen, das Bestimmen des Operationspunktes beim Start einer Transientensimulation und für das Errechnen der Sollwerte während der Transientensimulation verwandt. Sie bewirkt das Verschieben der Ausgangskurve entlang der Ordinate. Der Fehler ε in der Methode `VoutDC` verschiebt die Ergebniskurve des Funktionsblockmodells um genau diesen Betrag ε(`VoutDC`).

Wichtig beim Erzeugen der Methode `VoutDC` ist die Verwendung des relativen, d.h. prozentualen Fehlers, damit in allen Bereichen die entsprechende Genauigkeit garantiert ist. Der prozentuale Fehler der Methode fließt direkt in das Funktionsblockmodell ein. So bewirkt beispielsweise ein maximal erlaubter Fehler von 5% dafür, daß die Ergebniskurve um bis zu 5% entlang der Ordinate verschoben ist.

Auswirkung der Verzögerungszeit In Abbildung 36 a) sind zwei Kurven dargestellt. Die durchgezogene Kurve ist die Referenzkurve, die das Ausgangssignal des Blocks über der Zeit wiedergibt. Im dargestellten Bereich findet ein Sprung des Ausgangssignals von $VOut_{min} = 0$ auf $VOut_{max} = 7$ statt. Dazu wurde eine Modellkurve eingezeichnet. Die gestrichelt gezeichnete Modellkurve entspricht der Referenzkurve bis auf eine Verschiebung $\varepsilon(T_D)$ entlang der Ordinate, verursacht durch den Fehler im Modell der Verzögerungszeit: Die Veränderung am Ausgang findet zu spät statt.

Die Differenzkurve $z(t) = x(t) - y(t)$ nach Gleichung (4.19) ist in Abbildung 36 b) dargestellt. Wird die Modellgleichungen nach der gängigen l_∞-Fehlernorm (4.17) beurteilt, so schneidet sie sehr schlecht ab, da die Verschiebung für einen Impuls mit hoher Amplitude in der Fehlerkurve sorgt. Der Skalierungsfaktor s zwischen dem Zeitsignal im ns Bereich auf der x-Achse und der Spannung auf der y-Achse berechnet sich zu $s = \frac{1V}{1\eta s} = 1 * 10^8 \frac{V}{s}$.

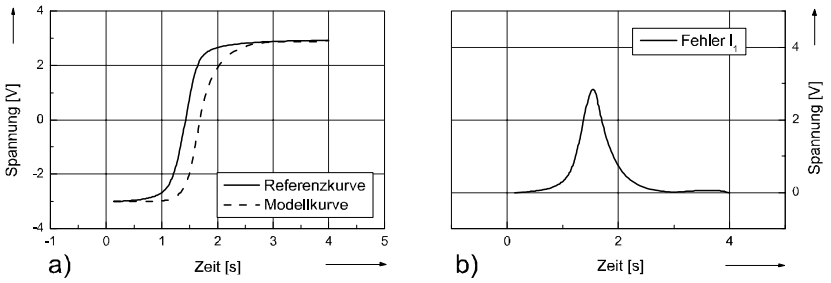

Abbildung 36: Auswirkung der Verzögerungszeit auf den Fehler.

Ein typischer Wert für die Verzögerungszeit ist 58 ns, unter der Annahme einer geforderten maximalen Abweichung von 5% gilt für den maximalen Fehler $\varepsilon \leq 2,9\ ns$. Mit der l_∞-Fehlernorm und dem horizontalem Abstandsmaß hat die Verschiebung einen dominanten Einfluß auf den Fehler: $d_\infty(x,y) \leq s * \varepsilon = 0,29V$. Da dieser Abstand auf der gesamten Flanke konstant ist, wird der relative Fehler $\widetilde{\varepsilon}$ für kleine y sehr groß, z.B. $y_0 = 0,1V \Rightarrow \widetilde{\varepsilon} = \frac{0,29V}{0,1V} = 290\%$.

Wird die Euklidische Distanz der Berechnung zugrunde gelegt, so ist der Fehler sehr gering, da in jedem Punkt der kleinste Abstand zwischen den Kurven betrachtet wird. Der mittels der Euklidischen Distanz berechnete Fehler ist nach oben durch die Phasenverschiebung begrenzt, d.h. es gilt:

$$z(t) < \varepsilon(T_D) \tag{4.34}$$

Abbildung 35 zeigt graphisch, wie die Euklidische Distanz zur Berechnung der Verschiebung entlang der Ordinate benutzt wird: anstelle des vertikalen, hier sehr großen Abstandes zwischen den beiden Kurven im Punkt t_0 wird der Abstand $E_z(t_0)$ benutzt, der minimale Abstand zwischen den beiden Kurven.

In der Praxis bedeutet dies, daß die Berechnung des Fehlers für die gesamte Simulation mit der Euklidischen Distanz erfolgen sollte, weil sonst durch kleine Abweichungen entlang der Ordinate ein großer maximaler Fehler auftreten kann.

Der bei der Charakterisierung erlaubte maximale Fehler findet sich als Verschiebung des Ausgangssignals entlang der Ordinate wieder.

Auswirkung der Slewrate Die Slewrate ist die Flankensteilheit, mit welcher ein Spannungssprung am Ausgang vollzogen wird, wie in Gleichung (4.7) definiert. Während der Charakterisierung wird eine Tabelle $(T_{\Delta u}, T_{slewrate})$ erzeugt, aus welcher während der Modellerzeugung eine Methode $slewrate(\Delta u)$ generiert wird. Diese ersetzt im folgenden in Gleichung (3.18) die Funktion für die zeitliche Veränderung, wie in Kapitel 3.8 dargestellt.

Zur Untersuchung ihrer Auswirkung auf die Genauigkeit des Funktionsblockmodells ist es hilfreich, einen einzelnen Zeitschritt t_0 zu betrachten. In t_0 beträgt der Fehler $\varepsilon_0 = |y(t_0) - \widetilde{y}(t_0)|$. Wird im nächsten Zeitschritt mit der Rekursionsgleichung $\widetilde{y}(t_1) = \widetilde{y}(t_0) + \Delta t_1 * slewrate(\Delta u_1)$ berechnet, so steckt in der Methode $slewrate(\Delta u)$ ein maximaler Fehler $\varepsilon_{\max}$, je nachdem, welche Abstandsmaße und Fehlernormen für die Charakterisierung verwandt wurden. Somit gilt $\varepsilon_1 = \varepsilon_0 + \varepsilon_{\max}\Delta t * slewrate(\Delta u)$ und im Zeitschritt n: $\varepsilon_n = \sum_{i=0}^{n} \varepsilon_{\max}\Delta t_i * slewrate(\Delta u_i)$ bzw. unter der Annahme, daß die Slewrate für den größten Bereich der steigenden Flanke konstant ist, gilt $\varepsilon_n = \varepsilon_{\max}(t_n - t_0)slewrate(\Delta u_1)$. Der Fehler steigt also linear mit der Zeit an. Graphisch kann dies als ein stetiges Auseinanderdriften der beiden Kurven gedeutet werden, wie in Abbildung 37 dargestellt wird. Auch wenn die Slewrate in jedem Zeitschritt nur unwesentlich ungenau ist, summiert sich diese Ungenauigkeit im Verlauf der Simulation zu beachtlichen Abweichungen. Deshalb ist es wichtig, daß die Slewrate mit möglichst geringem Fehler, z.B. $\varepsilon_{\max} < 1\%$, erzeugt wird. Da die Methode $slewrate(\Delta u)$ typischerweise in großen Bereichen linear verläuft, ist diese Anforderung ohne extreme Auswirkungen auf die Komplexität des Modelles und damit der Simulationszeit zu erreichen. Ein maximaler Fehler von 5% bei einer Slewrate in der Größenordnung von $1 * 10^6$ bewirkt nach $5\mu s$ eine Abweichung im Ausgangssignal von $0,25V$ bzw. 25% bei $y(5\mu s) = 1V$. Die Anforderung $\varepsilon_{\max} = 1\%$ reduziert die Abweichung an derselben Stelle im Ausgangssignal auf $0,05V$ bzw. 5%.

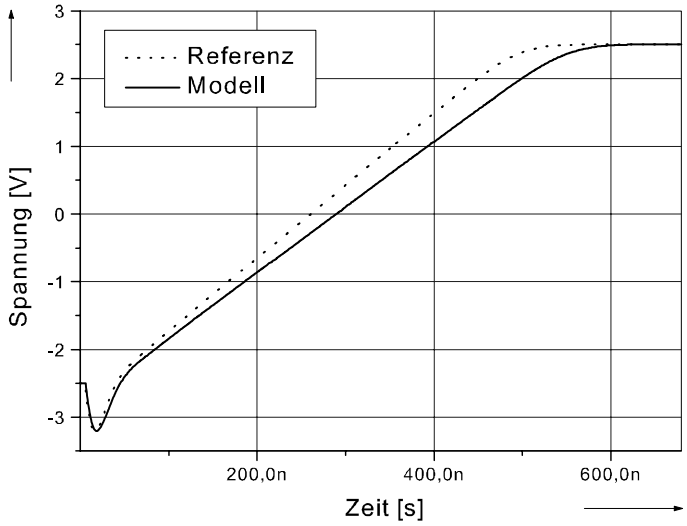

Abbildung 37: Fehler der Slewrate wirkt im Gesamtsystem additiv.

Für die Charakterisierung bedeutet dies, daß die Methode für die Slewrate mit einem möglichst kleinen erlaubten Fehler erzeugt werden muß.

Zusammenfassung Die Fehler der verschiedenen Methoden des Gesamtmodells fließen, wie oben dargestellt, nicht additiv / multiplikativ ein, sondern komplizierter:

- Der Fehler in der Methode $VoutDC$ bewirkt eine Verschiebung in Richtung der Ordinate.
- Der Fehler der Verzögerungszeit TD bewirkt eine Verschiebung in Richtung der Abszisse.
- Der Fehler (flacher / steiler Verlauf) in der Flankensteilheit *slewrate* bewirkt eine Verschiebung sowohl in Richtung der Abszisse als auch in Richtung der Ordinate.

Die Einheiten der Fehler (Zeit, Spannung, Ableitung der Spannung nach der Zeit) sind unterschiedlich und erfordern eine mathematisch exakte Möglichkeit zur Formulierung der verketteten Operation, welche bislang nicht vorliegt.

Aus den Untersuchungen folgt, daß sowohl die Verzögerungszeit als auch die Slewrate mit hoher Genauigkeit und mit geeigneten Fehlernormen modelliert werden müssen. Die Verzögerungszeit ist wichtig, damit das korrekte Einschwingen der Schaltung im Modell möglich ist. Die Slewrate muß stimmen, besonders für kleine Spannungsdifferenzen, also Sprünge in der Umgebung von $0 \pm \varepsilon$. Nur so ist gewährleistet, daß das Einschwingen in Form einer gedämpften Schwingung stattfindet. Andernfalls schaukelt sich die Schaltung bis in die Sättigung auf. Die Methode für die DC-Werte des Ausgangs schließlich sorgt dafür, daß die Differenzspannung im Modell richtig berechnet werden kann und somit auch die Slewrate.

5 Anwendungsbeispiele

Drei Beispiele zeigen Anwendungen der neuen Methodik. Das erste Beispiel ist eine lineare RC-Filterschaltung. Das Beispiel wurde bewußt einfach gewählt, damit die Schritte zur Charakterisierung und Modellierung leicht nachvollziehbar sind und somit ein Vergleich mit herkömmlichen Verfahren möglich ist. Besonderer Wert ist in der Darstellung auf die Charakterisierung, das heißt das Erzeugen der Elemente des Funktionsblockmodells, und dem Vergleich der Originalschaltung mit dem Funktionsblockmodell gelegt. Die Abstraktion von der Komponentenebene auf die Verhaltensebene nach Abbildung 4 während der Charakterisierung wird hier sehr deutlich, gerade in Bezug auf den Zustandsbegriff.

Für das zweite durchgehende Beispiel wurde der Operationsverstärker MOPA1 gewählt. In der Darstellung wurde besonderer Wert auf eine konsistente Definition der Elemente des Funktionsblockmodells gemäß der Definition in Kapitel 3 gelegt. Für die modellierte Schaltung stehen zum Vergleich nicht nur das Transistormodell, sondern zusätzlich ein vom Hersteller ausgeliefertes Makromodell nach Boyle zur Verfügung. Somit sind direkte Vergleiche der Genauigkeit und der Laufzeit möglich, sowohl des charakterisierten Funktionsblocks, als auch einer größereren Schaltung, in welcher mehrere Blöcke verschaltet werden. Besonderer Augenmerk liegt auf der Untersuchung der Lastsituation.

Während die ersten Beispiele rein analoge Schaltungen sind, wurde als drittes Beispiel ein AD-Wandler herangezogen. Der AD-Wandler als gemischt analog/digitale Schaltung ist ein gutes Beispiel für die Wiederverwendbarkeit eines Funktionsblockmodells: Die Konverterstufe läßt sich $n-1$ mal zu einem n-Bit Wandler verschalten. Ein Schwerpunkt hierbei ist die Partitionierung der Schaltung. Darüberhinaus sind hier Vergleiche mit einem üblichen VHDL-AMS Verhaltensmodell möglich. Die nahtlose Integration in VHDL-AMS tritt als besonderer Vorteil der Methodik hervor und wird in diesem Beispiel demonstriert.

Darüber hinaus wird gezeigt, daß die vorgeschlagene Methodik auf unterschiedliche Modellrepräsentationen (Mathematica, ELDO-FAS, VHDL-AMS) angewendet werden kann.

Das Kapitel schließt ab mit einer Gegenüberstellung der in den Beispielen benutzten Modelle und einer zusammenfassenden Darstellung der zur Modellgenerierung notwendigen Schritte.

5.1 Lineare Filterschaltung

Dieses erste durchgehende Beispiel soll anhand einer einfachen RC-Filterschaltung die Anwendung der Methodik veranschaulichen. Ausgangspunkt ist die lineare Filterschaltung aus Abbildung 38.

Unter Anwendung der Kirchhoff'schen Gesetze ergeben sich folgende Gleichungen für die Ströme in den Knoten der Schaltung, die Gleichung des Trivialknoten K2 wird implizit berücksichtigt:

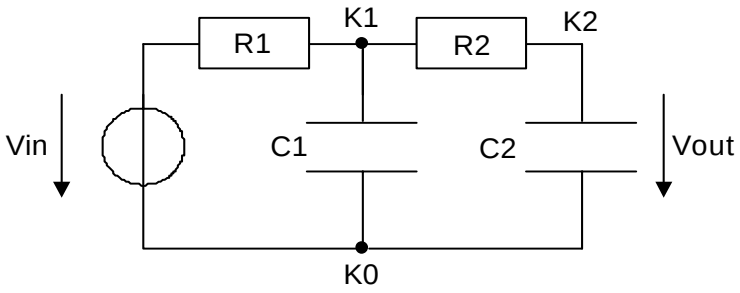

Abbildung 38: RC-Filterschaltung zweiter Ordnung.

$$\begin{aligned} \text{K0} &: \; i_{R_1} - i_{C_1} - i_{C_2} = 0 \\ \text{K1} &: \; -i_{R_1} + i_{C_1} + i_{R_2} = 0 \end{aligned}$$

Die Knotenpotentiale sind V0, V1 und V2; V0 ist mit V0=0V das Bezugspotential (Masse).

Die Systemvariablen $\overrightarrow{x}$ der Schaltung ergeben sich zu:

Zweigströme: $i_{R_1}, i_{R_2}, i_{C_1}, i_{C_2}$

Zweigspannungen: $u_{C_1}, u_{C_2}, u_{R_1}, u_{R_2}$

Zur Berechnung der Belegung der Systemvariablen muß das Gleichungssystem

$$A.\overrightarrow{x} - \overrightarrow{u} = \overrightarrow{0} \tag{5.1}$$

gelöst werden, siehe Abschnitt 3.1. Aufstellen des Gleichungssystems für die Beispielschaltung ergibt:

$$\begin{bmatrix} 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -C_1\frac{d}{dt} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -C_2\frac{d}{dt} \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \\ -R_1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -R_2 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} i_{R_1} \\ i_{R_2} \\ i_{C_1} \\ i_{C_2} \\ u_{R_1} \\ u_{R_2} \\ u_{C_1} \\ u_{C_2} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ u_{in} \\ 0 \\ 0 \\ 0 \end{bmatrix} = \overrightarrow{0} \tag{5.2}$$

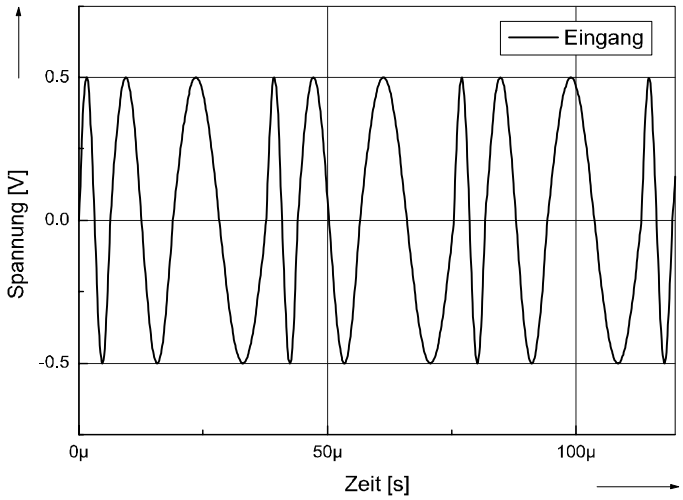

Abbildung 39: Eingangssignal zur Anregung der Beispielschaltung.

Um die Beispielschaltung mit einer nicht trivialen Funktion zu testen, wurde die in Abbildung 39 gezeichnete aperiodische Sinusschwingung als Anregung für die Tests im Zeitbereich verwandt.

Die Beispielschaltung wurde mit den folgenden Verfahren modelliert und simuliert:

- Um ein aussagekräftiges Referenzergebnis zu erhalten, wurde eine Netzwerksimulation mit Eldo durchgeführt. Die Ergebnisse werden zum Vergleich mit den anderen Verfahren herangezogen.
- Mathematica mit Analog Insydes wurde verwandt, um ausgehend von der Netzliste zu einer Approximation des Ausgangssignals zu gelangen.
- Das Modell wurde mittels der neuen Methodik erstellt.

  Den Abschluß dieses Beispiels bildet ein ausführlicher Vergleich dieser sehr unterschiedlichen Modellierungsmethoden.

5.1.1 Referenzlösung

Der Netzwerk-Simulator Eldo erzeugt bei der Beschaltung des linearen Filters in Abbildung 38 mit dem in Abbildung 39 dargestellten Eingangssignals das in Abbildung 40 gezeigte Ergebnis. Folgende Werte für Widerstände und Kapazitäten wurden in die Schaltung eingesetzt: $R_1 = 100\Omega, R_2 = 1K\Omega, C_1 = 3nF$ und $C_2 = 1nF$. Dieses Ergebnis dient als Referenz zum Vergleich mit den anderen Verfahren.

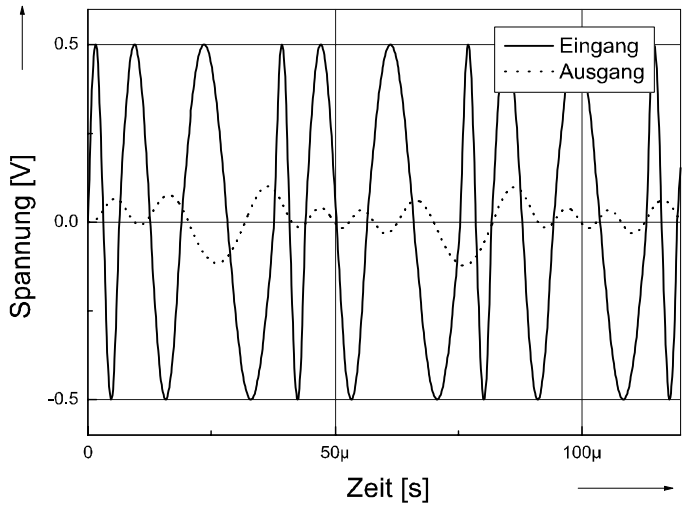

Abbildung 40: Referenzlösung für die lineare Filterschaltung.

5.1.2 Modellierung mittels Analog Insydes

Ausgangsbasis für die Modellierung mit Analog Insydes ist eine Spice Netzliste der Beispielschaltung:

```
LinearerFilter =
   Netlist[ {Vin, {1,0}, Vin[t]}, {R1, {1, 2},100}, {C1,{2, 0},
   3*10^(-9)}, {R2, {2, 3}, 1000}, {C2, {3, 0}, 1*10^(-9)}]
```

Der Analog Insydes Befehl

```
tsol = CircuitEquations[ LinearerFilter, TimeDomain -> True];
```

erzeugt aus der Netzliste die Gleichungen (5.3) für die Zustände V_1, V_2, V_3 (Knotenspannungen) und I_{Vin}, wobei der Schalter `TimeDomain` festlegt, daß das Ergebnis für die Großsignal bzw. Transientenanalyse genutzt werden soll.

$$\begin{gathered} I\$Vin[t] + 1/100 * (V\$1[t] - V\$2[t]) == 0 \\ 1/100 * (-V\$1[t] + V\$2[t]) + (V\$2[t] - V\$3[t])/1E4 + (3 * V\$2[t])/1E9 == 0 \\ (-V\$2[t] + V\$3[t])/1000 + \mathrm{D}[1][V\$3][t]/1E9 == 0 \\ V\$1[t] == Vin[t] \end{gathered} \tag{5.3}$$

(5.3) entspricht System (5.2), nur daß es vollständig automatisch aus der Spice-Netzliste generiert wurde. Das Zeichen $ ist ein von Mathematica intern genutzter Bezeichner für lokale Variablen, das doppelte Gleichheitszeichen ist die logische Relation "Gleichheit".

Analog Insydes kann das System numerisch lösen. Zur Lösung enthält Analog Insydes einen erweiterten numerischen Solver für nichtlineare Gleichungssysteme. Der folgende Befehl löst das Gleichungssystem (5.3), wobei die in Abbildung 39 dargestellte Funktion für das Eingangssignal eingesetzt wird. Der zu untersuchende Zeitbereich wird auf 6ns festgesetzt.

```
sols = NDAESolve[ tsol /. Vin[t] -> u[t], {t, 0., 6us}];
```

Das Ergebnis hat die Form sogenannter "InterpolatingFunctions", welche Mathematica Objekte sind, die eine numerische Näherung für eine symbolisch nicht darstellbare Funktion enthalten. Mittels Stützstellen und Interpolation erfolgt die Auflösung dieser Objekte innerhalb von Mathematica. Dabei ergibt sich folgendes Ergebnis für die vier Zustandsgrößen:

```
{{V$2 -> InterpolatingFunction[],
V$3 -> InterpolatingFunction[],
V$1 -> InterpolatingFunction[],
I$Vin -> InterpolatingFunction[]}}
```

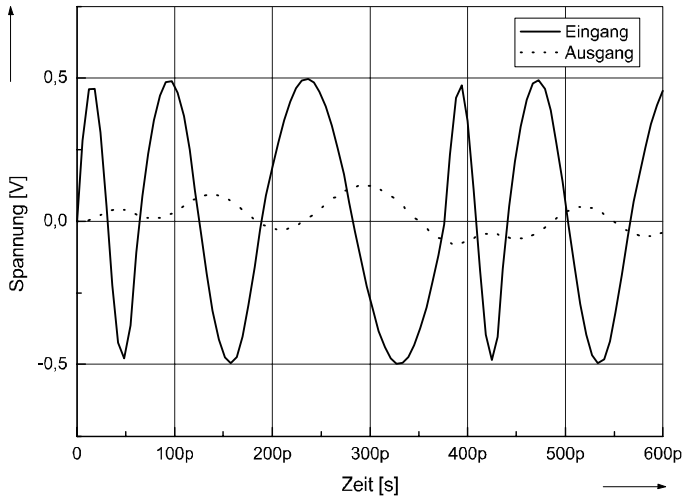

Abbildung 41: Ergebnis der Simulation mit Analog Insydes.

Abbildung 41 zeigt das mittels der Analog Insydes Funktion `TransientPlot` erzeugte Ergebnis für das Ausgangssignal sowie die Eingangsfunktion. Da Mathematica keine

Funktionen für Simulationen im Zeitbereich bereitstellt, ist es schwierig, den Zeitbereich richtig zu skalieren. Somit ist auch nur ein rein quantitativer Vergleich mit den Referenzergebnissen des Netzwerksimulators möglich, der jedoch korrekt wirkt. Leider gibt es zur Zeit noch keinen Code Generator, der aus diesen InterpolatingFunctions C/C++ Code erzeugen kann. Der Anwender ist somit auf Mathematica als Simulationsumgebung angewiesen. Diese fehlende Integrationsmöglichkeit in die Simulationsumgebung und die nicht vorhandene Echtzeitfähigkeit schränken die praktische Verwendung der Modelle stark ein. Die Stärke dieses Ansatzes ist jedoch die automatische Generierung der Gleichungen und Modelle aus der Spice-Netzliste.

5.1.3 Funktionsblockmodell

Der erste Schritt ist das Erzeugen der für die Modellbildung notwendigen Datenbasis. Hierzu wird die Testschaltung aus Abbildung 38 nach dem in Tabelle 11 aufgezeigten Charakterisierungsplan simuliert. Der Bereich ist in zwei Abschnitte aufgeteilt, damit der wichtige Bereich um Null mit hinreichender Genauigkeit modelliert wird, im übrigen Bereich läßt sich durch eine größere Schrittweite Simulationszeit sparen. Die Charakterisierung wird mittels Clang [HGT91] gesteuert, der verwandte Netzwerksimulator ist Eldo.

Ergebnisdatei	**Startwert** V_{in}	**Endwert** V_{in}	**Schrittweite** Δ
remote1.asc	-10	10	0.01
remote2.asc	-0.01	0.01	0.00001

Tabelle 11: Charakterisierungsplan für Anwendungsbeispiel lineare Filterschaltung.

Die Charakterisierung besteht aus zwei Schritten:

1. DC-Analyse

2. Transientenanalyse gemäß Tabelle 11

Die Ergebnisse der beiden Schritte werden im folgenden dargestellt, gefolgt von den resultierenden Methoden und dem Aufbau des gesamten Modells. Den Abschluß bildet die Beschaltung des Modells mit dem Test-Eingangssignal und der Vergleich mit den anderen Verfahren.

Die Ergebnisse der DC-Simulation Das Ergebnis der DC-Analyse ist trivial: Nach dem Abklingen aller Einschwingvorgänge, d.h. wenn die Kondensatoren aufgeladen sind, gilt $V_{out} = V_{in}$. Es besteht also ein direkter linearer Zusammenhang zwischen Eingang und Ausgang, welcher in Abbildung 42 illustriert wird.

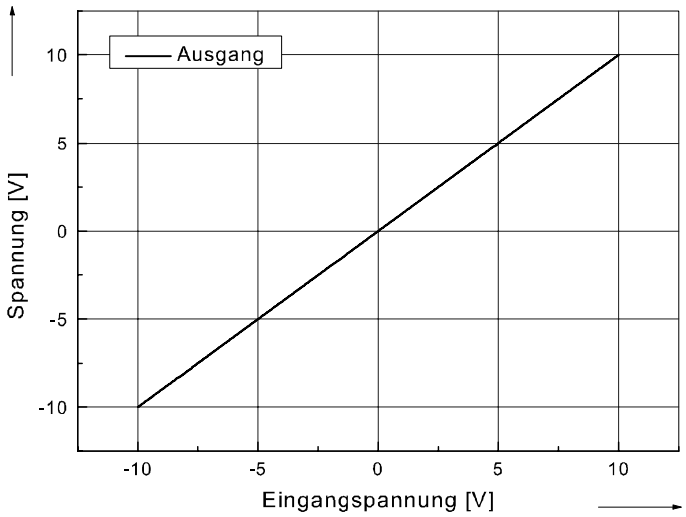

Abbildung 42: Ergebnis der DC-Analyse.

Die Ergebnisse der Simulation im Zeitbereich Die Simulationen im Zeitbereich erfordern, im Gegensatz zu der DC-Simulation, nicht einen einzigen Lauf des Simulators, sondern einen Lauf für jeden diskreten Wert, welchen die Größe des am Eingang wirkenden Spannungssprunges annimmt. Für das Beispiel wurde der Charakterisierungsplan nach Tabelle 11 mittels Clang umgesetzt. Aus den Ergebnissen wurden Tabellen für die Slewrate und für die Verzögerungszeit erzeugt. Die beiden folgenden Abschnitte beschreiben, wie aus den Tabellen mittels der Methodenbibliothek lauffähiger ELDO-FAS Code generiert wird.

Die Methode Slewrate Das Ergebnis ist eine Tabelle mit der direkten Zuordnung Größe Eingangssprung zu Slewrate. Abbildung 43 illustriert den Zusammenhang.

Diese Tabelle wird mittels folgendem Mathematica-Aufruf in eine Methode `slewrate` umgewandelt, der maximal erlaubte Fehler (prozentualer Abstand mathematische Modellgleichung zu Datensatz) wird für das Modell auf ein Prozent festgelegt:

```
slewrate = BuildMethod[ daten, MaxError -> 1];
```

Das Ergebnis sind in diesem Fall drei lineare Abschnitte, d.h. drei lineare Funktionen. Zurückgegeben werden Fehlernorm, Abstandsmaß und maximaler Fehler {{"L-∞", YAxis, 1}, dazu kommen für jeden Bereich Startwert, Endwert und Anzahl der Datenpunkte (z.B. 20., 1.64304, 17750) und schließlich die eigentliche Funktion als eine "Pure Function", ein Mathematica Objekt zur effizienten Repräsentation von Funktionen (z.B. 0.22429 -

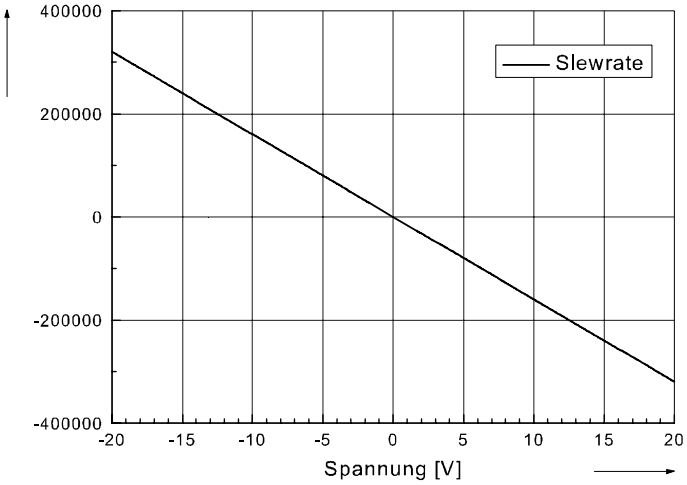

Abbildung 43: Zuordnung Größe Sprung der Eingangsspannung zur berechneten Slewrate.

31996.94121*#1 &). Die Mathematica Ausgabe, welche die Methode für die Slewrate darstellt, ergibt sich zu:

```
{{''L-Infinity'', YAxis, 1},
   {{-20, -0.01994, 17753, -15.4851 - 31997.7 #1 &},
   {-0.01994, 0.01994, 1920, -0.000106598 - 31221.1 #1 &},
   {0.01994, 20, 17753, 15.4851 - 31997.7 #1 &}}}
```

Die Werkzeuge der Methodenbibliothek ermöglichen die Visualisierung der Ergebnisse und die Berechnung der Fehlers, der zwischen der Tabelle mit den Simulationsdaten und der erzeugten Methode auftritt. Abbildung 44 zeigt die Methode `slewrate`. In der Darstellung wird deutlich, daß der Zusammenhang weitgehend linear ist - bis auf einen kleinen Bereich im Nulldurchgang, welcher zwar ebenfalls linear ist, aber mit einer anderen Funktion beschrieben wird als der Rest.

Abbildung 45 zeigt auszugsweise die Residuen, d.h. die Abweichung, zwischen den Tabellenwerten und den Ergebnissen der Methode in einer Graphik. Hier wird ersichtlich, daß der Fehler in der Tat sehr klein ist, was durch die entsprechende Berechnung in Mathematica unterstützt wird:

```
In[74]:=    CalculateDifference[{#[[1]], #[[2]]} & /@ data, slf]
Out[74]=   {0.0281689, {{17691}}}
```

Das heißt, der maximale Fehler im Modell liegt bei 0,28 %.

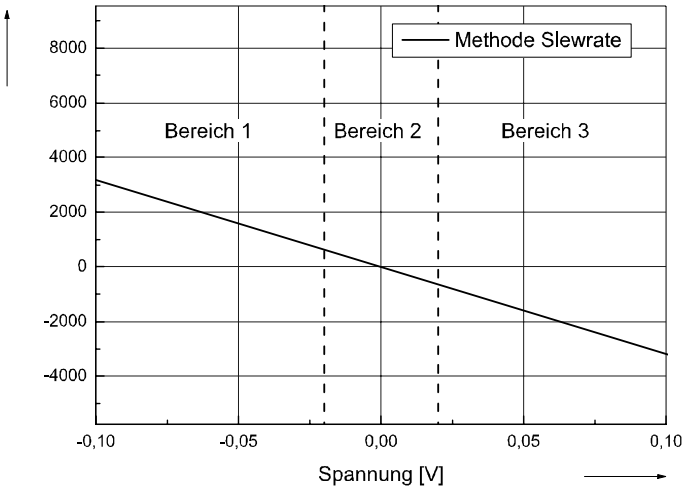

Abbildung 44: Die Methode Slewrate, der mittlere Bereich ist sehr klein.

Der letzte Schritt ist die Umwandlung der Mathematica Methode in Code, der vom Simulator ausführbar ist. Der folgende Mathematica Befehl wandelt die Methode um in ELDO-FAS Code und speichert diesen auf der Festplatte ab:

```
EldoFASFunction[ slm, WriteToFile -> True,
    FunctionName -> ''slewrate'',
   VarName -> {''VI'', ''sr_neu''}]
```

Das Ergebnis dieses Aufrufs ist der folgende Programmcode, der der ELDO-FAS Syntax entspricht und direkt in das Modell für den linearen Filter übernommen werden kann:

```
if ( -2.0e1< VI and VI <= -1.994e-2 ) then
   make sr_new = -1.5485123e1 + -3.1997704e4 * VI
endif
if ( -1.994e-2 < VI and VI <= 1.994e-2 ) then
   make sr_new = -1.0659833e-4 + -3.1221118e4 * VI
endif
if ( 1.994e-2 < VI and VI <= 2.0e1 ) then
   make sr_new = 1.5485123e1 + -3.1997704e4 * VI
endif
```

Die Methode Verzögerungszeit Analog zu dem Vorgehen beim Erzeugen der Methode Slewrate zeigt Abbildung 46 den Zusammenhang zwischen der Größe des Eingangssprunges und der Verzögerungszeit. Auch hier wird deutlich, daß der Nulldurchgang besondere Betrachtung erfordert (d.h. besonders genaue Charakterisierung, entsprechend

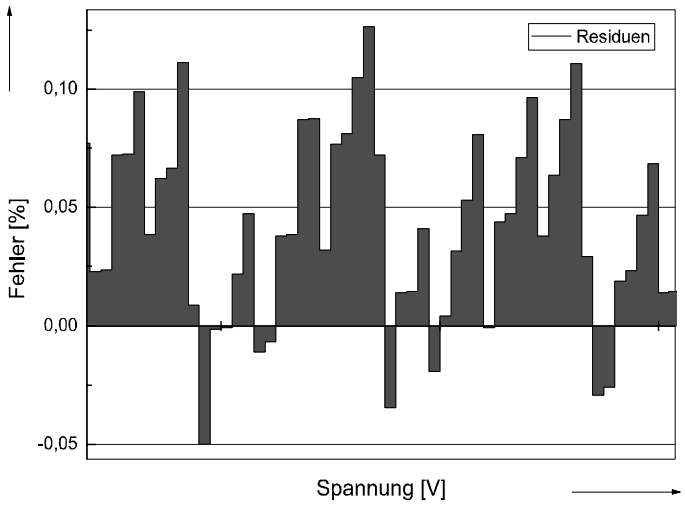

Abbildung 45: Der Fehler zwischen Daten und Methode Slewrate.

dem Charakterisierungsplan aus Tabelle 11). Der Hintergrund ist klar: Kleine Veränderungen am Eingang wirken sich sehr schnell aus, während größere Veränderungen länger brauchen.

Dieser Zusammenhang wird in der Darstellung der Methode in Abbildung 47 deutlich. Der maximal erlaubte Fehler wurde erneut auf 1 % gesetzt, Mathematica wählt für den Bereich des Nulldurchgangs selbstständig ein Polynom zweiten Grades, welches die Daten hinreichend genau beschreibt und effizienter ist als das Aufteilen in viele lineare Abschnitte.

Abbildung 48 zeigt den Unterschied zwischen den Tabellenwerten und dem Methodenansatz. Die Berechnung des maximalen Fehlers ergibt einen Wert von 0,11 %, d.h. das Modell ist hervorragend.

Der folgende Code ist die ELDO-FAS Repräsentation der Ergebnisse der Methode Verzögerungszeit:

```
if ( -2.0e1 < VI and VI <= -1.64422e0 ) then
   make TD = 5.5312937e-5 + -3.8135127e-11 * VI
endif
if ( -1.64422e0 < VI and VI <= 1.64422e0 ) then
   make TD = 5.714e-5 + -6.75986e-7 * VI power 2.0e0 + -7.01852e-21 * VI
endif
if ( 1.64422e0 < VI and VI <= 2.0e1 ) then
   make TD = 5.5312937e-5 + 3.8135127e-11 * VI
endif
```

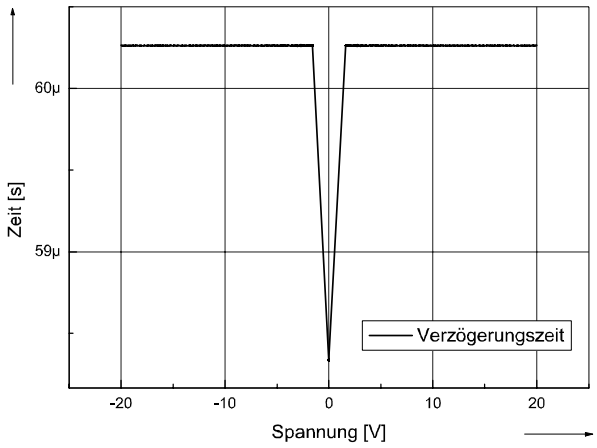

Abbildung 46: Zusammenhang zwischen Eingangssprung und Verzögerungszeit.

Architektur des Verhaltensmodells Der erste Schritt besteht darin, die am Eingang im aktuellen Zeitschritt anliegende Spannung zu berechnen. Dazu wird u_{n-1} von u_n subtrahiert. Alle folgenden Beispiele mit Programmcode sind in einer Pascal-ähnlichen Notation aufgeführt, die gegenüber realer Simulatorsprachen einige Vereinfachungen vornimmt.

```
-- Spannungssprung am Eingang des aktuellen Zeitschritts
make Vin = VOLT.Value(in)
make VI = Vin - STATE.LAST_VALUE(Vin, TSTEP)
```

Als zweite Größe kann jetzt der Unterschied zwischen dem aktuell am Ausgang anliegenden Wert und dem Sollwert am Ausgang ermittelt werden. `Vout` wurde mit der Methode aus Abbildung 42 berechnet und stellt somit eine Funktion des aktuell am Eingang anliegenden Wertes dar. `VL` ist eine Zustandsvariable, die den im letzten Zeitschritt berechneten Wert der Ausgangsspannung trägt. `VL` ist somit die Spannung, die am Ausgang anliegt, deshalb die Bezeichnung als "Ist" Wert.

```
-- Differenz zwischen IST- und SOLL-Wert am Ausgang
make VD = Vout - VL
```

Nun gilt es, diese beiden Werte miteinander zu verrechnen, sodaß der korrekte Wert zum Festlegen der Slewrate verwandt wird. Die folgende Anweisung bildet das Maximum der beiden Werte, d.h. bei der Überlagerung wird der größere der beiden Werte zum Berechnen der Slewrate verwandt.

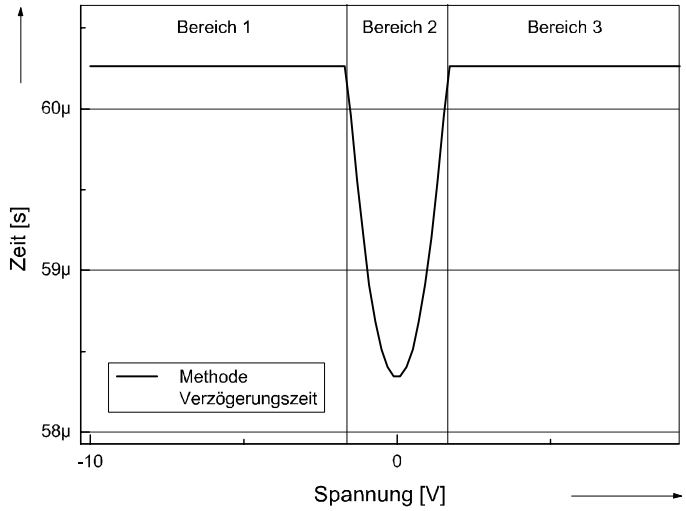

Abbildung 47: Methode Verzögerungszeit.

```
-- Überlagern von Eingang und Ausgang: Maximum verwenden
make Veff = VI >VD?VI:VD;
```

Mittels diesem `Veff` werden jetzt die neue Slewrate `sr_neu` und die Verzögerungszeit `TD` berechnet. Der entsprechende HDL-Code wurde bei der Einführung der Methoden dargestellt. Anschließend wird ein Ereignis gesetzt, welches festlegt, wann die neue Slewrate zum Fortschreiben der Ausgangsspannung verwandt werden kann:

```
-- Setzen eines externen Ereignis
schedule_event(time+TD, sr, sr_neu);
```

Als letztes muß jetzt noch die Veränderung des Ausgangssignals im aktuellen Zeitschritt berechnet werden. Hierzu wird nicht die neue Slewrate `sr_neu` (die erst nach `TD` Zeiteinheiten Auswirkungen zeigt) verwandt, sondern die alte Slewrate `sr` (zu einem früheren Zeitpunkt berechnete und im aktuellen Ereignis übernommene).

```
-- Ausgangsspannung
make VL = VL + sr * TSTEP
make VOLT.across(out) = VL
```

Diese Berechnung entspricht der im Kapitel 3 definierten Rekursionsgleichung nach Gleichung (3.21).

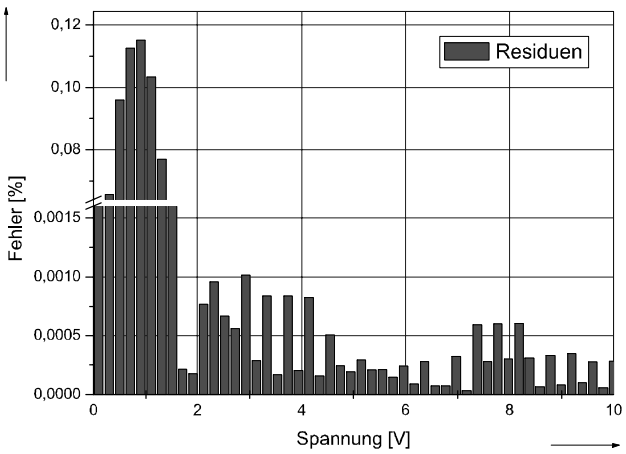

Abbildung 48: Differenz zwischen Tabelle und Methode TD.

Ausführung des Verhaltensmodells Abbildung 49 zeigt das Ergebnis eines Simulationslaufes mit dem neuen Verhaltensmodell unter Anwendung der Referenzsignalsform am Eingang der Schaltung. Die Signalform entspricht dem Ergebnis des Netzwerksimulators, es existieren Abweichungen in der Größenordnung bis 4 % an den Extremwerten der Ausgangskurve.

5.1.4 Vergleich der Ansätze

Numerische Integrationsverfahren Um die Antwort eines Netzwerkes im Großsignalbereich zu berechnen, verwenden Netzwerksimulatoren lineare Iterationsverfahren (LMS, linear multistep). Allen Verfahren gemeinsam ist die Betrachtung eines Zeitschrittes $\Delta t^{(n)} = t^{(n+1)} - t^{(n)}$. Dazu wird ein Startwert $\overrightarrow{x}^{(0)}$ benötigt, der festlegt, in welchem Zustand sich das Netzwerk zum Zeitpunkt $t = t^{(0)}$ befindet. Da $\overrightarrow{x}^{(0)}$ bekannt ist, kann $\overrightarrow{x}'^{(0)}$ aus der Gleichung (5.2) des Netzwerks gewonnen werden. Im einfachsten Fall, der Vorwärts-Euler-Formel, ergibt sich aus diesen Größen der neue Zustand der Schaltung zu

$$\overrightarrow{x}^{(n+1)} = \overrightarrow{x}^{(n)} + \Delta t^{(n)} \overrightarrow{x}'^{(n)}. \tag{5.4}$$

Diese Formel arbeitet gut, wenn die Zeitschritte $\Delta t^{(n)}$ sehr klein gewählt sind. Da der Fehler zwischen dem nach (5.4) berechneten $\overrightarrow{x}^{(n+1)}$und dem richtigen Wert mit einer Vergrößerung des Zeitschrittes stark ansteigen kann, ist die Qualität der Simulationsergebnisse nicht optimal. Die Idee ist deshalb, nicht die Steigung im Punkt $\overrightarrow{x}^{(n)}$, sondern die Steigung im Punkt $\overrightarrow{x}^{(n+1)}$ zu verwenden. Aus (5.4) kann $\overrightarrow{x}'^{(n+1)}$ nicht direkt berechnet werden, es kann aber ein geschätzter Wert für $\overrightarrow{x}^{(n+1)}$ als Startwert für ein Iterationsverfahren verwandt werden. Dieser geschätzte Wert kann beispielsweise mittels (5.4) berechnet

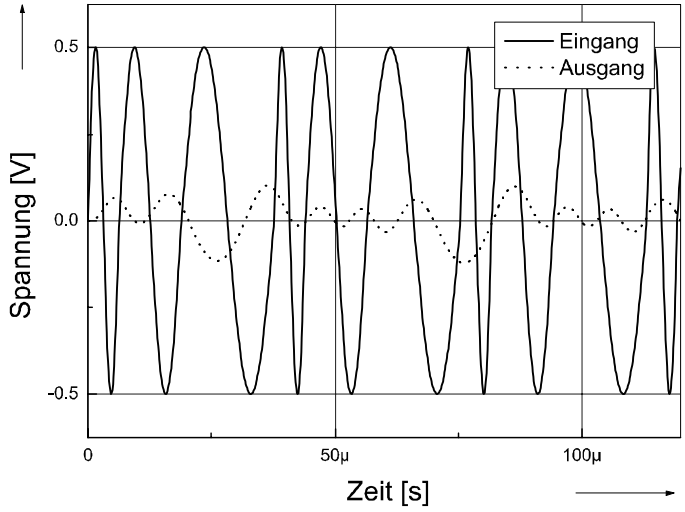

Abbildung 49: Ausführung und Bewertung des Funktionsblockmodells.

werden. Mit dem geeigneten Schätzer $\overrightarrow{x}^{(n+1)}$ und der Ableitung $\overrightarrow{x}'^{(n+1)}$ ergibt sich die Rückwärts-Euler-Formel zu

$$\overrightarrow{x}^{(n+1)} = \overrightarrow{x}^{(n)} + \Delta t^{(n)} \overrightarrow{x}'^{(n+1)}. \tag{5.5}$$

Die dritte Möglichkeit liegt in einer Kombination der beiden Ansätze. $\frac{\overrightarrow{x}'^{(n)} + \overrightarrow{x}'^{(n+1)}}{2}$ verwendet die (berechnete) Ableitung in $\overrightarrow{x}'^{(n)}$ und die (geschätzte) Ableitung in $\overrightarrow{x}'^{(n+1)}$ und führt eine gleichmäßige Gewichtung durch. Daraus ergibt sich direkt die Trapezregel zu

$$\overrightarrow{x}^{(n+1)} = \overrightarrow{x}^{(n)} + \frac{\Delta t^{(n)}}{2}(\overrightarrow{x}'^{(n)} + \overrightarrow{x}^{(n+1)}). \tag{5.6}$$

Bei der Verwendung numerischer Integrationsverfahren gilt allgemein:

- Der Simulator legt selbständig die Zeitschritte fest.
- Die Formen der Eingangssignale werden nicht berücksichtigt, sondern ausschließlich die aktuellen Werte der Eingänge.
- Die Iterationen zum Berechnen eines Schätzers für $\overrightarrow{x}^{(n+1)}$ sind recheninternsiv und dauern lange.

Im folgenden wird zunächst die Trapezregel (5.6) auf die Beispielschaltung 38 angewandt. Anschließend wird die neue Methodik manuell darauf angewendet, um im direkten Vergleich der Verfahren das Funktionsprinzip darzustellen.

Beispielberechnung der Filterschaltung mit dem Funktionsblockmodell Zum Start der Simulation wird $\overrightarrow{y}^{(0)}$ zum Zeitpunkt $t^{(0)}$ benötigt. Abbildung 42 zeigt den Zusammenhang zwischen $\overrightarrow{u}$ und $\overrightarrow{y}$. Mittels der Methode `VoutDC` kann der Startwert $\overrightarrow{y}^{(0)}$ bestimmt werden, da $\overrightarrow{u}^{(0)}$ bekannt ist. Weil die Methode `VoutDC` anhand der Simulationsdaten der Schaltung während der Charakterisierung gewonnen wurde, ist dieser Wert per Definition identisch mit dem vom Netzwerksimulator berechneten DC-Operationspunkt. Der Aufwand wird vom Lösen einer Iteration auf das Einsetzen eines Wertes in eine mathematische Formel reduziert. Im Gegensatz zu der DC-Operationspunktanalyse liefert die Methodik keine Aussagen über die Zustandsvariablen der Schaltung, d.h. darüber, in welchem Zustand sich die Energiespeicher, hier die Kapazitäten C_1 und C_2, befinden. Diese Information ist für die weitere Simulation mit dem Funktionsblockmodell nicht erforderlich.

Die Methoden `slewrate` und `TD` liefern als Startwerte jeweils Null, da zum Zeitpunkt $t^{(0)}$ keine Veränderung am Eingang stattgefunden hat, sondern der Startwert (nach Abklingen aller eventuellen Einschwingvorgänge) berechnet wird.

Zum Zeitpunkt $t^{(1)}$ kann $\Delta t^{(0)} = t^{(1)} - t^{(0)}$ und $\Delta u^{(0)} = u^{(1)} - u^{(0)}$ berechnet werden. Um die Wirkung dieser Veränderung der Eingangsspannung nachzuvollziehen, wird folgendes Vorgehen benutzt:

1. Der Eingang der Referenzschaltung wird mit einem Spannungssprung der Größe $\Delta u^{(0)}$ angeregt. Die entsprechende Transientensimulation wird mit dem Netzwerksimulator durchgeführt, das Ergebnis ist ein funktionaler Zusammenhang. Abbildung 50 zeigt das Ergebnis.

2. Die Analyse des Eingangs-Ausgangs-Zusammenhanges soll folgende Fragen beantworten:

   - Wie steil ist die Flanke des Ausgangssignals?
   - Nach welcher Zeit reagiert der Ausgang auf den Sprung des Eingangssignals?

3. Gemäß den in Kapitel 3 definierten Verfahren werden aus der Kurve die Slewrate `sr_neu` und die Verzögerungszeit `TD` bestimmt. Abbildung 19 zeigt, wie diese Größen aus der Kurve mit der Sprungantwort berechnet werden.

4. Jetzt wird mittels der im Schritt 3 gewonnenen Werte ein externes Ereignis gesetzt.

```
-- Setzen eines externen Ereignis
schedule_event(time+TD, sr, sr_neu);
```

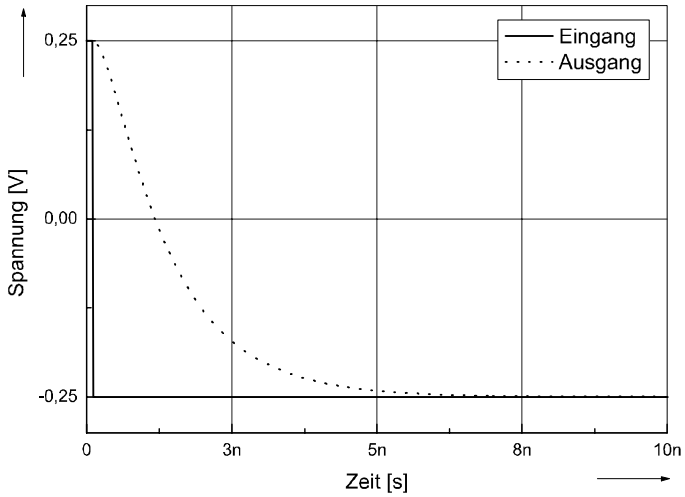

Abbildung 50: Ergebnis der Transientenanalyse für einen Spannungssprung am Eingang.

Für den Simulationslauf bedeutet dies, daß zunächst keine Veränderung am Ausgang passiert, zumindest keine, welche mit der Änderung $\Delta u^{(0)}$ zusammenhängt. Bis die Verzögerungszeit vorbei ist, arbeitet der Simulator mit den alten Werten der Zustandsvariablen weiter, d.h. in diesem Beispiel ändert sich die Slewrate nicht.

Zum Zeitpunkt $t^{(2)} = t^{(1)}+$`TD` tritt das Ereignis ein. Der Simulator setzt jetzt die Zustandsvariable `sr` auf den neuen Wert `sr_neu`. Die Rekursionsgleichung (3.23) wird ausgeführt und der Ausgangsverlauf wird neu berechnet (allerdings noch mit der alten Slewrate, da der neue Wert ab $t^{(2)}$ gilt, nicht für den gerade vergangenen Zeitschritt). Das Vorgehen in den ersten drei Zeitschritten wird in Abbildung 51 illustriert: erst im dritten Zeitschritt erfolgt die Änderung des Ausgangs, die um `TD` verzögerte Sprungantwort.

Zum Zeitpunkt $t^{(3)}$, welcher ein internes, d.h. vom Simulator gesetztes Ereignis darstellt, wird die neue Slewrate zum Berechnen des Ausgangs in der Rekursionsgleichung eingesetzt: $y^{(3)} = y^{(2)} + \Delta t^{(2)} *$ `sr_neu` .

Die unter 1. und 2. durchgeführte Simulation und Berechnung der Zustandsgrößen ist zeitintensiv und kann nicht zur Laufzeit des Simulators erfolgen. Deshalb erfolgen sie beim Erzeugen des Modells während der Charakterisierung. Das Ergebnis der Charakterisierung gemäß Tabelle 11 sind die Methoden `slewrate` und `TD`, welche in Abbildung 44 bzw. 47 dargestellt sind. Durch diese vorab generierten Methoden reduziert sich der Aufwand in einem Simulationsschritt von einem Simulationslauf (Punkt 1) und der Extraktion der benötigten Größen (Punkt 2) auf das Auswerten zweier mathematischer Funktionen.

Die bisherige Betrachtung hat einen wichtigen Punkt außer Acht gelassen, nämlich die Frage, wie das Aufladen der Kapazitäten im Modell berücksichtigt wird. Die Steilheit der

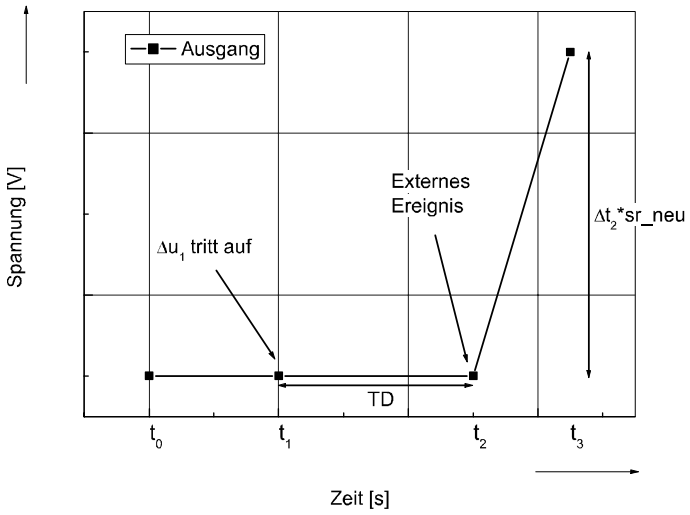

Abbildung 51: Konstruktion des Ausgangssignals für die ersten drei Zeitschritte.

Flanke in einem beliebigen Zeitschritt hängt nicht ausschließlich von der zeitlichen Änderung des Eingangssignals ab (wie bisher in der Darstellung dieses Beispiels angenommen), sondern wesentlich auch von dem Zustand, in dem sich die Schaltung befindet, wenn die Änderung am Eingang auftritt. An dieser Stelle greifen die im Kapitel 3.8 beschriebenen Grundlagen über die mathematische Modellierung der Rückkoppelung.

Abbildung 42 zeigt das Ergebnis der DC-Analyse. Dieser Zusammenhang zwischen Ein- und Ausgang der Schaltung nach dem Abklingen aller Einschwingvorgänge erlaubt es, zu jedem Zeitschritt $t^{(n)}$ genau festzustellen, welcher Wert am Ausgang anliegen sollte, nämlich `VoutDC`. Der Vergleich dieses Wertes mit dem im aktuellen Zeitschritt berechneten Ausgangswert liefert eine Aussage über den Zustand der Schaltung und erlaubt es, die Slewrate zu korrigieren:

- Die Steigung verringert sich, wenn der Ausgang sehr nah am Sollwert liegt. Hierin steckt in diesem Beispiel implizit die bekannte Exponentialfunktion aus den Bauteilgleichungen der Kondensatoren.
- Die Steigung muß erhöht werden, wenn im aktuellen Zeitschritt keine oder eine kleine Änderung des Eingangssignals auftritt (welche die Slewrate auf einen entsprechend geringen Wert setzt), aber eine große Veränderung aus einem früheren Zeitschritt nur teilweise vollzogen wurde.

In Abbildung 52 ist dieser Zusammenhang dargestellt. Nach dem Sprung fand keine Veränderung mehr am Eingang statt, Δu ist somit konstant 0 und dementsprechend liefern die Methoden `slewrate`= 0 und `TD`= 0. Diese Werte sind offenbar falsch und würden

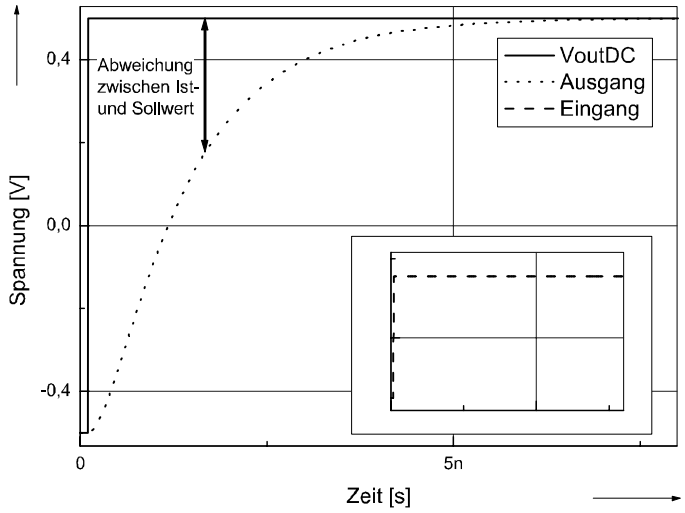

Abbildung 52: Abweichung zwischen aktuellem Simulationsergebnis und dem Sollwert.

dazu folgen, daß der zurückliegende Eingangssprung nicht bis zum Erreichen des richtigen Ausgangswertes vollzogen würde. Die in der Abbildung dargestellte Berechnung der Abweichung zum Sollwert in diesem Zeitschritt ermöglicht es, dies zu erkennen und die Methoden korrekt zuzuweisen.

Die effektiv wirkende Eingangsspannung wird also - wie im Abschnitt Zusammenbau des Verhaltensmodells beschrieben - anhand der Größe des Eingangsspannungssprungs und der Abweichung Ist- zu Sollwert der Ausgangsspannung berechnet. Auf diese Art und Weise ist sichergestellt, daß zu jedem Zeitpunkt der Simulation die Steilheit der Flanke korrekt im Modell eingesetzt wird.

In dieser internen Zustandsvariablen, die den Unterschied von Ist- zu Sollwert der Ausgangsspannung enthält, stecken implizit die Zustandsvariablen des Originalmodells. Eine Kapazität kann zu einem diskreten Wert des Eingangssignals eine genau berechenbare maximale Spannung annehmen, den Sollwert, der nach dem Abklingen der Einschwingvorgänge anliegt bzw. sich ergeben würde, wenn keine weiteren Anregungen am Eingang anliegen würden. Ist dieser Wert noch nicht erreicht, verändern sich die Zustände weiterhin dynamisch, d.h. die Kapazität wird aufgeladen. Die Summe all dieser Zustände im elektronischen Modell wird in der neuen Methodik auf einen Gesamtzustand der Schaltung abstrahiert, welcher von der Wirkung her dieselbe Rolle spielt: Er legt fest, zu welchem Grad die innere Struktur der Schaltung die durch Anregungen am Eingang notwendigen Veränderungen vollzogen hat.

5.2 Operationsverstärker

In diesem Abschnitt wird ein Operationsverstärker vom Typ MOPA1 als Anwendungsbeispiel betrachtet [Ray90]. Für diesen Operationsverstärker stehen drei Modelle zur Verfügung:

- Transistormodell.
- Modell auf der Makro Ebene nach Boyle.
- Mittels der vorgeschlagenen Methodik erstelltes Verhaltensmodell.

Die beiden ersten Modelle werden vom Hersteller zur Verfügung gestellt, Abbildung 53 zeigt das Transistormodell. Das Makromodell wird in Kapitel 2.5.4 in Abbildung 7 gezeigt. Die Erstellung des dritten Modells wird im folgenden Abschnitt ausführlich dargelegt. Die drei Modelle werden miteinander verglichen, wobei sie in unterschiedlichen Testschaltungen eingesetzt werden.

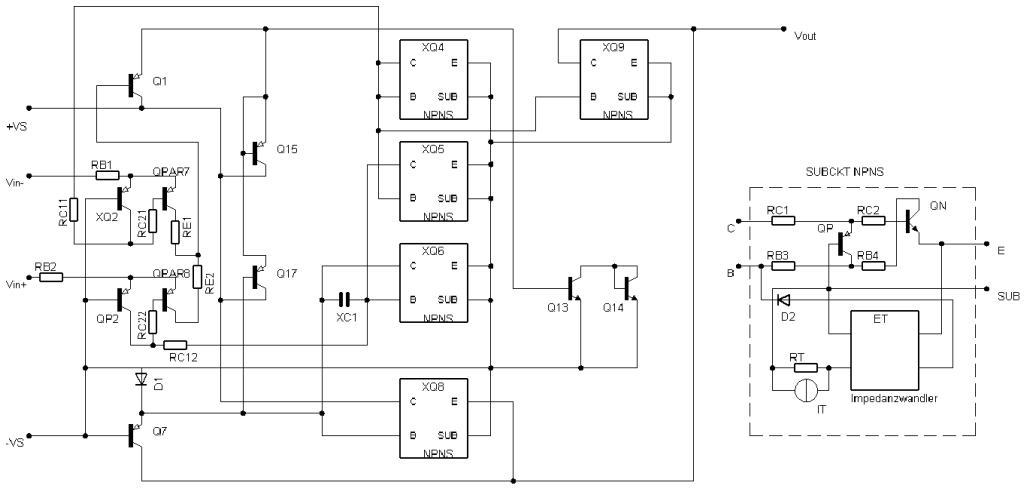

Abbildung 53: Transistormodell des Operationsverstärker MOPA1.

5.2.1 Funktionsblockmodell

Der Operationsverstärker MOPA1 aus Abbildung XXX läßt sich nach Definition (3.9) als Funktionsblockmodell gemäß der vorgeschlagenen neuen Methodik darstellen:

$$MOPA1 = \{\overrightarrow{u}, \overrightarrow{y}, \overrightarrow{x}, \delta, \lambda, ta, f\} \tag{5.7}$$

Die Elemente dieses Funktionsblockmodells werden im folgenden einzeln beschrieben, gleichzeitig wird die Implementierung hier anhand von VHDL-AMS Codefragmenten illustriert.

Eingangsgrößen des Funktionsblocks $\overrightarrow{u} = \{P_{in+}, P_{in-}\}$

Der Operationsverstärker hat die Anschlüsse P_{in+} und P_{in-} mit den resultierenden Eingangsgrößen im VHDL-AMS Modell:

```
port(
   terminal P_in+ : Electrical;
   terminal P_in- : Electrical);
```

Ausgangsgrößen des Funktionsblocks $\overrightarrow{y} = \{P_{out\_V}, P_{out\_I}\}$

Der Operationsverstärker hat den Anschluß P_{out}.

```
port(
   terminal P_out : Electrical);
```

Dem Anschluß P_{out} werden die beiden Ausgangsgrößen Strom P_{out_I} und Spannung P_{out_V} im VHDL-AMS Modell zugeordnet:

```
quantity
   P_out_V across
   P_out_I through
   P_out
```

Interne Zustände des Funktionsblocks $\overrightarrow{x} = \{V_L, V_{in}\}$

Das Modell des Operationsverstärkers verwendet zwei interne Zustände. V_L ist der im letzten Zeitschritt am Ausgang anliegende Wert. Dieser Wert wird für die rekursive Definition des Ausgangssignals benötigt. V_{in} ist die Eingangsspannungsdifferenz zwischen den Anschlüssen P_{in+} und P_{in-}.

Bei der Modellierung in VHDL-AMS entsprechen diese internen Zustände genau den Zustandsvariablen (Quantities), anhand derer das Ausgangsverhalten berechnet wird. Diese Quantities erlauben die Deklaration von Potentialdifferenzen (Across Quantities) und Flußgrößen (Through Quantities).

```
quantity V_in across P_in+ to P_in-;
quantity VL : emf;
```

Übergangsfunktionen für Ereignisse: $\delta() = \{$`VoutDC,TD,slewrate`$\}$

Für den Operationsverstärker werden drei Parameterfunktionen benötigt, die als Methoden in das Modell eingehen.

(1) `VoutDC` berechnet den Spannungswert, der bei einer vorgegebenen Eingangsspannungsdifferenz am Ausgang anliegen soll, das heißt wenn alle Einschwingvorgänge abgeklungen sind und die Schaltung sich in einem stabilen Zustand befindet. Bild 54 zeigt das Ergebnis der DC Analyse der Grundschaltung. Das Ergebnis der Modellierung mit einer Genauigkeit von 1% wird in Tabelle 12 gezeigt. Mathematica hat den Definitionsbereich in fünf Bereiche aufgeteilt, welche mit Polynomen bis zum Grad 5 beschrieben werden. Die Tabelle zeigt Start- und Endwert des jeweiligen Segments, die Anzahl der Punkte, die im Segment liegen, und die mathematische Funktion, die das Segment beschreibt.

Segment		**Anzahl**	**Das Segment**
von	**bis**	**Punkte**	**approximierende Funktion**
-0.0010	-0.0001	443	$68.099x - 9.506$
-0.0001	-0.00058	30	$4.182E8x\hat{}2 + 90908.679x - 4.544$
-0.00058	0.00009	75	$51141.866x - 5.387$
0.00009	0.00031	113	$-3.8E9x\hat{}5 + 3.E16x\hat{}4 - 9.17E12x\hat{}3$ $+1.338E9x\hat{}2 - 40894.964x - 3.024$
0.00031	0.001	343	$78.801x + 9.501$

Tabelle 12: Segmente der Methode `VoutDC`.

Bild 54 visualisiert die Ergebnisse: Die erzeugte Methode `VoutDC` wird mit den Ausgangsdaten verglichen.

(2) `TD` berechnet, nach welcher Zeit sich eine Veränderung des Einganges am Ausgang auswirkt.

(3) `slewrate` berechnet die Flankensteilheit des Ausgangssignals.

In VHDL-AMS Repräsentationen wird eine Methode (hier: `Vout_DC`) im allgemeinen als Funktion realisiert:

```
function Vout_DC(V_diff: real) return real is
   variable VDC: real;
   begin
      if V_diff > 4.45688e-1 then
         VDC := 1.17854e6;
      ...
   return VDC;
end;
```

Die Verzögerungszeit aus (2) weist in diesem Modell aufgrund der vorliegenden Simulationsergebnisse einen konstanten Wert auf. Dieser Wert fließt als Konstante `TD` in

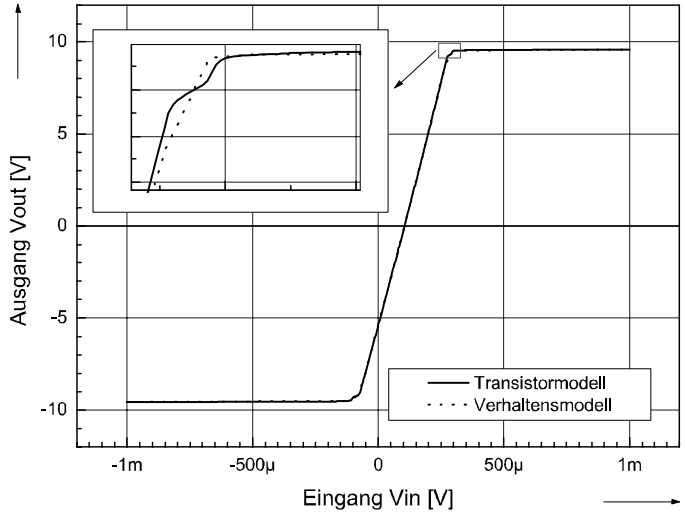

Abbildung 54: Vergleich der Simulationsergebnisse auf Transistorebene mit der Methode `VoutDC`.

das Funktionsblockmodell ein. Eine Konstante ist eine Sonderform einer Methode, die in VHDL-AMS entsprechend nicht als (konstante) Funktion realisiert werden muß.

Die Methode für die Slewrate wird über einen Charakterisierungsplan erzeugt. Simulationen im Zeitbereich für unterschiedliche Größen des Spannungssprungs am Eingang der Schaltung liefern nach Gleichung (4.7) eine Tabelle mit der Zuordnung Größe der Eingangsspannungsdifferenz zum Wert der Slewrate. Die Werte rund um den Nulldurchgang sind besonders kritisch (da sie für das korrekte Einschwingen verantwortlich sind), deswegen wird dieser Bereich mit einer sehr kleinen Auflösung simuliert. Die so entstandene Tabelle ist die Eingabe für die Methoden-Bibliothek. Mit einer maximalen Abweichung von 2% ergibt sich der in Abbildung 55 dargestellte Verlauf der Slewrate für eine Eingangsspannungsdifferenz von -0.5V bis 0.5V. Im Bild wird deutlich, daß besonders im Bereich um Null viele Bereiche gewählt wurden, um die Genauigkeit zu garantieren. Die Methode `slewrate` wird als Funktion für die zeitliche Veränderung in das Funktionsblockmodell eingefügt und wie unten dargestellt, aufgerufen. Implementiert wird sie in VHDL-AMS ebenfalls als Funktion:

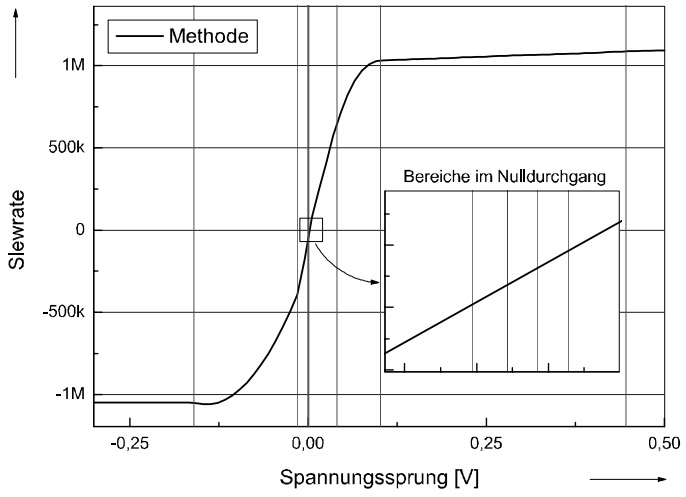

Abbildung 55: Methode Slewrate für MOPA1.

```
function slewrate(V_diff: real) return real is
   variable SR: real;
   begin
      if V_diff < -0.1702 then
         SR := -155077.86 * V_diff - 1.01541e6;
      ...
   return SR;
end;
```

Ausgangsfunktion $\lambda(\overrightarrow{u}) := y$

Folgendes VHDL-AMS Codefragment zeigt, wie beim Start der Transientensimulation der Operationspunkt mit Hilfe der Methode `Vout_DC` gesetzt wird. Es wird der zur Eingangsspannung korrelierende Wert des Ausganges mit der Methode berechnet und die Potentialdifferenz der Quantity wird auf genau diesen Wert gesetzt.

```
if (now < TD) use
   U_out == Vout_DC( V_in );
```

Zeit-Funktion $ta() := ta_{intern} \cup ta_{extern}$

Das Modell wird zu den Integrationsschritten des Simulators berechnet. In VHDL-AMS enthält die Systemvariable now den aktuellen Zeitschritt. Mittels der "Delayed" Methode der Quantities wird die Verzögerungszeit `TD` eingefügt, die hier eine Konstante ist.

```
slewrate( U_in'Delayed( TD ));
```

Funktion für die zeitliche Veränderung $f : \frac{dV}{dt} := slewrate$

Die Veränderung der Ausgangsspannung P_out_V (die am) im Transientenbereich wird üblicherweise mittels Differentialgleichungen berechnet. Die Methode `slewrate` modelliert die Funktion für die zeitliche Veränderung, wie im Abschnitt Übergangsfunktion für interne und externe Ereignisse dargestellt. An dieser Stelle wird der Unterschied zu mittels Differentialgleichungen notierten Modellen besonders deutlich.

```
P_out_V'dot == slewrate( U_in'Delayed( TD ));
```

Obiges Codefragment zeigt, wie elegant die Methodik in VHDL-AMS umgesetzt wird: Hier kann die Steigung des Ausgangs direkt über die Eigenschaft `dot` der across quantity `P_out_V` gesetzt werden. Im Beispiel Lineare Filterschaltung mußte die Veränderung im Zeitschritt berechnet werden und der Ausgang um diesen Wert inkrementiert werden, denn dieses Beispiel wurde in EldoFAS implementiert. In EldoFAS kann die Steigung nicht direkt gesetzt werden.

5.2.2 Spannungsfolgerschaltung

Das im vorigen Abschnitt dargestellte Funktionsblockmodell soll in unterschiedlichen Beschaltungen mit unterschiedlichen Eingangssignalen seine allgemeine Verwendbarkeit zeigen. Zunächst wird in einer Spannungsfolgerschaltung überprüft, wie die Schaltung auf einen Spannungssprung am Eingang reagiert. Diese Operationsverstärker-Anwendung gehört zu den kritischsten Schaltungen, denn das Einschwingen muß korrekt vollzogen werden. Daraufhin wird das Modell erweitert um eine lastabhängige Ausgangsstrombegrenzung. Bei der DC-Simulation mit entsprechender Beschaltung soll das Modell seine Tauglichkeit für unterschiedliche Lastsituationen unter Beweis stellen. Abschließend wird dasselbe Modell in eine größere Schaltung eingesetzt, einen biquadratischen Filter. Hier soll die Wiederverwendbarkeit des Modells in unterschiedlichen Beschaltungen demonstriert werden.

Das Ausgangssignal der Testschaltung nach einem Sprung am Eingang ist in Abbildung 56 dargestellt. Der Vergleich des Funktionsblockmodells mit der Simulation auf der Transistorebene zeigt, daß die Schaltung nicht nur auf den richtigen Wert einschwingt, sondern daß auch die Settling-Time vom Funktionsblockmodell korrekt wiedergegeben wird.

Untersuchung der Lastsituation In Gleichung (3.31) wird das Standard-Funktionsblockmodell um eine Definition für den Innenwiderstand erweitert, um unterschiedliche Lastsituationen zu berücksichtigen. Damit wird die Forderung, daß ein Modell in beliebigen Teilschaltungen verwendbar sein muß, erfüllt. Der Ausgang des MOPA1 kann in einer Testschaltung mit unterschiedlichen Lasten beschaltet sein. Da die Spannungs-Strom-Kennlinie des Innenwiderstand des MOPA1 nicht linear ist, ist die Betrachtung der

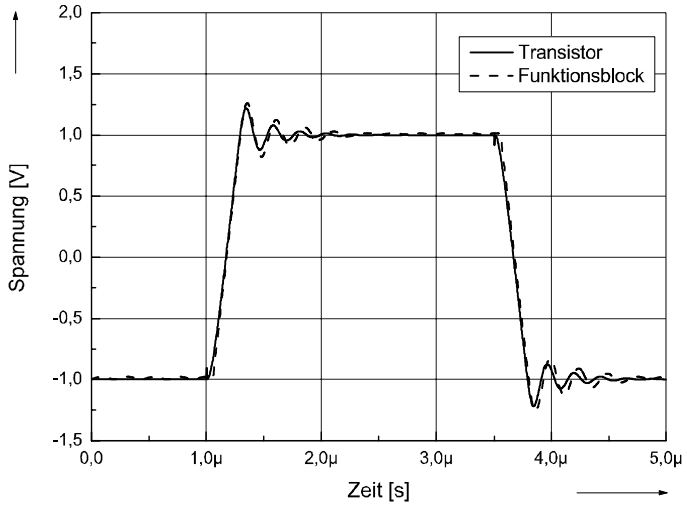

Abbildung 56: Das Einschwingen nach einem Spannungssprung am Eingang.

Grenzfälle (Kurzschluß und open loop) nicht hinreichend, um die Größe des Innenwiderstandes zu errechnen. Der nicht rückgekoppelte MOPA1 in Schaltung in 57 wird deshalb mit folgenden Lastwiderständen beschaltet: $R_L = \{0.0001\Omega, 100\Omega, 500\Omega, 1\text{K}\Omega, 5\text{K}\Omega, 10\text{K}\Omega, 50\text{K}\Omega, 100\text{K}\Omega, 500\text{K}\Omega, 10M\Omega\}$. Der erste Wert entspricht dem Kurzschluß, der letzte der open loop Beschaltung. An die Testbench wurde bei der DC-Analyse ein von $-0.05V$ auf $0.05V$ ansteigendes Eingangssignal angelegt, der MOPA1 wird hier in der Sättigung betrieben.

Generell gilt, daß die Last des Modells bekannt sein muß bzw. vom Simulator bei der Berechnung der Schaltung bestimmt werden muß. Die Last kann als Parameter beim Aufruf in das Modell übergeben werden. Das Funktionsblockmodell wird um eine weitere Methode `RI(`R_L`)` erweitert, das heißt in Abhängigkeit der anliegenden Last wird der Innenwiderstand des MOPA1-Modelles berechnet. Die Ausgangsspannung wird gemäß (3.31) unter Berücksichtigung von Innen- und Lastwiderstand berechnet.

Anhand der Ausgangsspannung kann jetzt der durch R_L fließende Strom und mittels der Maschenregel der Wert des Innenwiderstandes für die jeweilige Last bestimmt werden. Dieser Zusammenhang ist in Bild 58 illustriert.

Jetzt kann, wie schon erwähnt wurde, eine Methode `RI` für den Innenwiderstand in Abhängigkeit der Last erzeugt werden und in das Funktionsblockmodell eingefügt werden. Somit wird das Modell erweitert um die Ausgangsstrombegrenzung, und zwar um eine dynamische Begrenzung in Abhängigkeit von der Last am Ausgang. Die Validierung dieses Ansatzes erfolgt bei Beschaltung mit einer Last von $1K\Omega$. Das Ergebnis der entsprechenden DC-Analyse ist in Abbildung 57 dargestellt. Im Bild wird deutlich, daß der Übergang in die Sättigung zwar eine geringe Verschiebung aufweist, das Ergebnis im Gegensatz zum

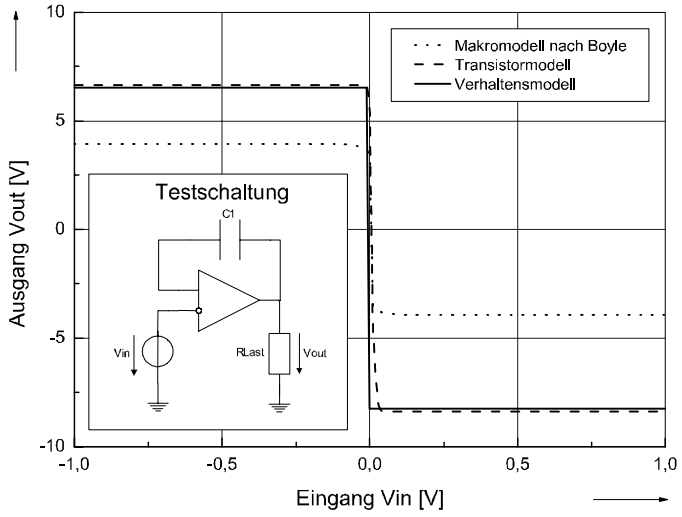

Abbildung 57: DC-Simulationsergebnisse unter Berücksichtigung einer Last.

Makromodell nach Boyle jedoch die Ausgangsspannung weitgehend korrekt liefert. Die Qualität der Modellierung ließe sich durch eine aufwendigere Charakterisierung sicherlich verbessern.

5.2.3 Biquadratischer Filter

In diesem Beispiel soll die Wiederverwendbarkeit des MOPA1 Funktionsblockmodells in einer komplexen Schaltung gezeigt werden. Als Beispielschaltung wurde ein biquadratischer Filter ausgewählt [CY94]. Seine Schaltung ist in Abbildung 59 dargestellt.

Um speziell den nichtlinearen Betrieb zu untersuchen, wurde ein Spannungssprung der Größe 5V an den Eingang der Filterschaltung angelegt. Wie im vorigen Beispiel wurden die drei Modelle für den MOPA1 in die Schaltung eingesetzt und die Ergebnisse der Simulationen miteinander verglichen. Abbildung 60 zeigt die Ergebnisse. Der besonders interessante Bereich, an dem das Einschwingen auf den Sollwert erfolgt, ist vergrößert dargestellt. Es wird deutlich, daß die mit der neuen Methodik erzeugten Funktionsblockmodelle die Nichtlinearitäten korrekt wiedergeben. Beim Einschwingen liegt das mit der neuen Methodik erstellte Verhaltensmodell wesentlich näher an den Originalergebnissen als das Makromodell. Das Makromodell zeigt eine Abweichung in der Flankensteilheit, die sich in einer deutlichen Verschiebung entlang der Abszisse bemerkbar macht.

Das Beispiel zeigt offensichtlich, daß ein derart erstelltes Modell in unterschiedlichen Beschaltungen wiederverwandt werden kann. So tritt MOPA1 in Abbildung 59 höchst unterschiedlich beschaltet auf. In Tabelle 13 sind die für die Simulationen benötigten

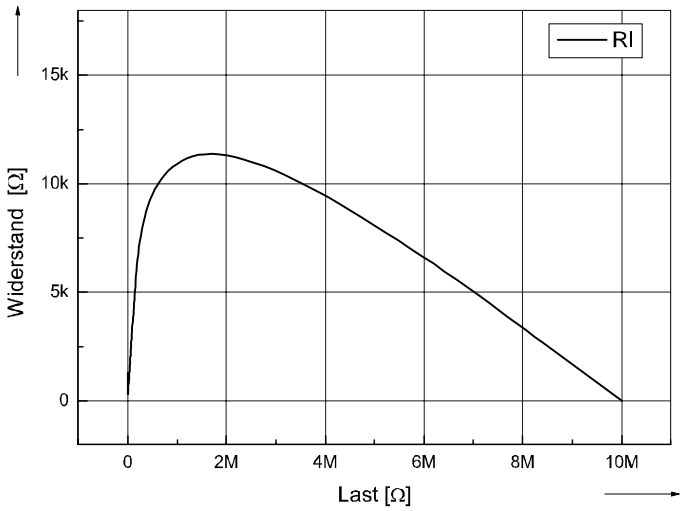

Abbildung 58: Zusammenhang Last-Innenwiderstand beim MOPA1.

Ressourcen dargestellt.

5.2.4 Vergleich der Ergebnisse

Die Simulationen mit unterschiedlichen Betriebsumgebungen erlauben einen guten Vergleich der Modelle - nicht nur rein qualitativ zum Nachweis der Korrektheit, sondern auch im Hinblick auf die Ressourcen Speicherplatz und Simulationszeit. Zum Vergleich wurden die Simulationen des Funktionsblocks und der Schaltung biquadratischer Filter verwandt. Tabelle 13 zeigt die Anzahl der Knoten in der simulierten Schaltung auf der Netzlisten-Ebene, den benötigten Speicherplatz und die benötigte Prozessorzeit auf einer mit 300MHz getakteten SUN UltraSparc mit 2 Prozessoren.

Modell	**Schaltung**	**Knoten [Anzahl]**	**Speicher [KB]**	**Zeit [s]**	**Speedup [Faktor]**
Transistor-ebene	MOPA1	83	930	55	1
	Filter	327	1324	123	1
Makro-ebene	MOPA1	15	877	18	3
	Filter	55	1098	27	4
Verhaltens-ebene	MOPA1	3	596	6	9
	Filter	9	915	12	10

Tabelle 13: Vergleich der Simulationsressourcen.

Die Ergebnisse zeigen, daß das Verhaltensmodell bei korrekter Beibehaltung der Cha-

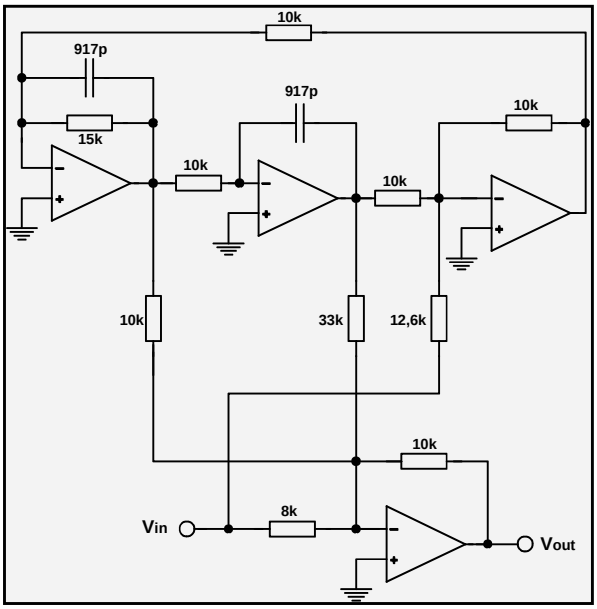

Abbildung 59: Biquadatischer Filter.

rakteristika der Kurvenform und hinreichender Genauigkeit einen Speedup der Größenordnung 10 erreicht - bereits bei diesem relativ kleinen Funktionsblock. Die Simulationszeit läßt sich also vom Stunden- in den Minutenbereich reduzieren, ohne Einschränkungen im Hinblick auf Genauigkeit oder bestimmte Einsatzbedingungen in Kauf zu nehmen. Dazu zeigt der Vergleich mit dem Makromodell nach Boyle, daß das Funktionsblockmodell universell, das heißt unabhängig von der Beschaltung mit einer bestimmten Last, einsetzbar ist.

5.3 A/D-Wandler

Das Verfahren wird für einen A/D-Wandler erläutert. Das Beispiel ist ein komplexes gemischt analog/digitales Bauteil mit einem Analogeingang und einem digitalen Ausgang pro Bit. Die Simulation der Schaltung benötigt hohe Ressourcen: Ein Simulationslauf für einen 6-Bit A/D-Wandler dauert auf einer UltraSparc mit 2 Prozessoren (300MHz) ca. 11 min. Als Referenz steht ein analoges Modell zur Verfügung, welches die in den Abbildungen 61 und 62 dargestellte Schaltungsstruktur aufweist. Als Modelle für den Operationsverstärker bzw. den Komparator wurden vom jeweiligen Hersteller gelieferten Makromodelle verwandt, siehe [Nat99] und [Max95]. Aus diesem Grund wird in diesem Kapitel das Referenzmodell immer als Makromodell bezeichnet.

Der nicht getaktete A/D-Wandler besteht aus einer Konverterstufe, welche für einen n-Bit Wandler n-1 mal in Serie geschaltet wird und mit einem Komparator ergänzt wird. Abbildung 61 zeigt das Schaltbild des 6-Bit A/D-Wandler. Es ist insofern vereinfacht,

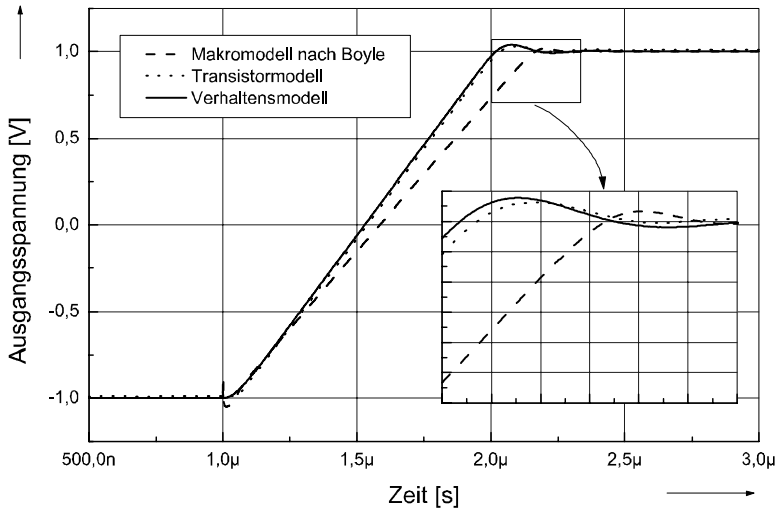

Abbildung 60: Simulationsergebnisse biquadratischer Filter.

als die in allen Konverterstufen benötigten Spannungen (minimale und maximale Referenzspannung und die Hälfte zwischen den Referenzspannungen) nicht dargestellt sind. Die halbe Referenzspannung ist als `Vthresh` bezeichnet und an den einen Eingang des Komparators angelegt.

Die Konverterstufe ist in Bild 62 dargestellt. Sie besteht aus einem Komparator, einem Schalter und einem Addierer/Verstärker. Die enthaltenen Komponenten sind Standardprodukte, für die Makromodelle vorliegen. Das Funktionsprinzip ist die sukzessive Approximation. Die Größe des Eingangssignals wird auf einen bestimmten Wert hin (die Hälfte zwischen Minimal- und Maximalwert) getestet, der Komparator setzt daraufhin das entsprechende Ausgangsbit. Je nach Wert des Bits wird eine bestimmte Spannung vom Eingangssignal subtrahiert und das Ergebnis mit dem Faktor 2 multipliziert. Im folgenden wird die Funktion der Konverterstufe in C-ähnlicher Notation dargestellt:

```
V50 = (Vref- + Vref+) / 2;
if (Vin >= V50) {
   Va = V50 - Vref- / 2;
   BIT = 1;
} else {
   Va = Vref- / 2;
   BIT = 0;
}
Vout = (Vin - Va) * 2;
```

Ein 6-Bit A/D-Wandler besteht aus fünf Konverterstufen und einem Komparator. Der Spannungsbereich sei für dieses Beispiel zu 1 bis 2 V gewählt. Jetzt wird Vin=1.25 V am

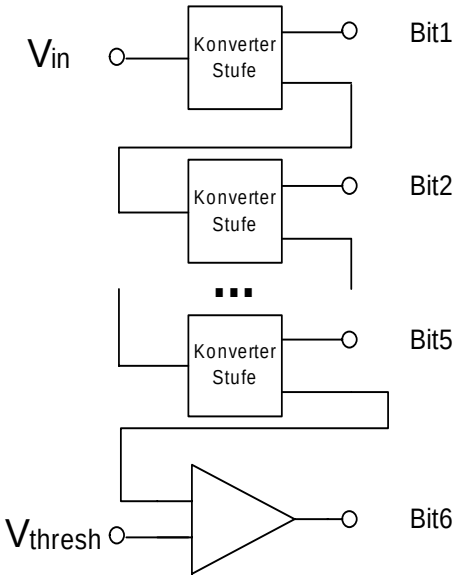

Abbildung 61: Vereinfachtes Schaltbild des 6-Bit A/D-Wandler.

analogen Eingang des A/D-Wandlers angelegt. Mit Vref-=1V, Vref+=2V und V50=1,5V ergeben sich nach dem oben skizzierten Verfahren die in Tabelle 14 dargestellten Werte für die einzelnen Stufen. *Dies ergibt das digitale Bitmuster 010000, das dem am Eingang anliegenden analogen Spannungswert von 1.25 V entspricht.*

	Stufe 1	**Stufe 2**	**Stufe 3**	**Stufe 4**	**Stufe 5**
Vin	1,25	1,5	1	1	1
Va	0,5	1	0,5	0,5	0,5
BIT	0	1	0	0	0
Vout	1,5	1	1	1	1

Tabelle 14: Ergebnisse der Konverterstufen.

5.3.1 Partitionierung der Schaltung

Die erste Aufgabe besteht nun darin, die Gesamtschaltung 6-Bit A/D Wandler zu partitionieren.

Bei der Partitionierung gilt es allgemein, Funktions-Einheiten zu identifizieren, welche während der Simulation hohe Anforderungen hinsichtlich Zeit- und Speicherressourcen haben. Darüberhinaus muß die Teilschaltung die Kriterien zur Anwendung der vorgeschlagenen Methodik erfüllen. Ein weiterer wichtiger Aspekt ist die Wiederverwendbarkeit: Wird ein Teil einer Schaltung modelliert, der - auch unterschiedlich beschaltet - in mehreren Teilschaltungen auftritt, so kann das Modell unverändert eingesetzt werden - wie

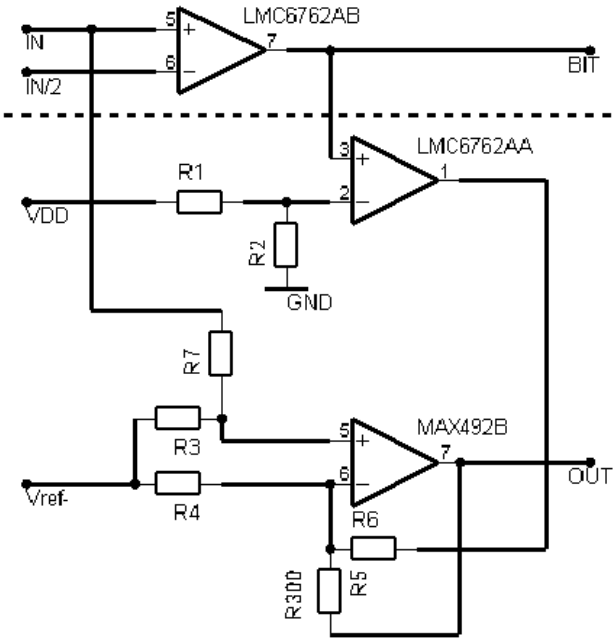

Abbildung 62: Konverterstufe des A/D-Wandlers.

bereits anhand des Beispiels biquadratischer Filter gezeigt wurde. Auf der anderen Seite gilt: Je größer die modellierte Teilschaltung ist, desto größer ist die Simulationsgeschwindigkeit, das heißt der Speedup des Verhaltensmodells gegenüber der Originalschaltung.

Die Zusammenschaltung von n-1 Konverterstufen zu einem n-Bit A/D-Wandler legt die Idee nahe, die gesamte Konverterstufe in ein Verhaltensmodell umzuwandeln, und das Element-Modell dann entsprechend zu unterschiedlichen A/D-Wandlern zu verschalten.

Ergebnis dieses ersten Charakterisierungsversuchs ist die Erkenntnis, daß die Zusammenfassung der gesamten Konverterstufe zu einem Modell nicht direkt durchführbar ist. Ursache hierfür ist der Konverter, der das Eingangssignal mit einem Referenzwert vergleicht: Je nachdem, ob das Eingangssignal größer oder kleiner als der Referenzwert ist, kommen unterschiedliche Signalformen am Ausgang zustande. Abbildung 63 zeigt das Ergebnis einer DC-Analyse der gesamten Konverterstufe: Das Ausgangssignal ist nicht stetig, es findet ein Sprung statt, wenn sich der Bit-Ausgang verändert. Die vorgeschlagene Methodik kann diese Unstetigkeit nicht korrekt nachbilden, das Verhaltensmodell müßte um eine Bedingung erweitert werden, die den Eingang gegen den Wert testet, bei dem der Komparator schaltet. Diese Bedingung kann zwar mittels eines if-Konstrukts eingebaut werden, vernachlässigt jedoch alle elektronischen Eigenschaften des Komparators - das Modell hätte keine garantierte Genauigkeit mehr. Deshalb führt die Lösung über die Partitionierung der Konverterstufe in Funktionsblöcke.

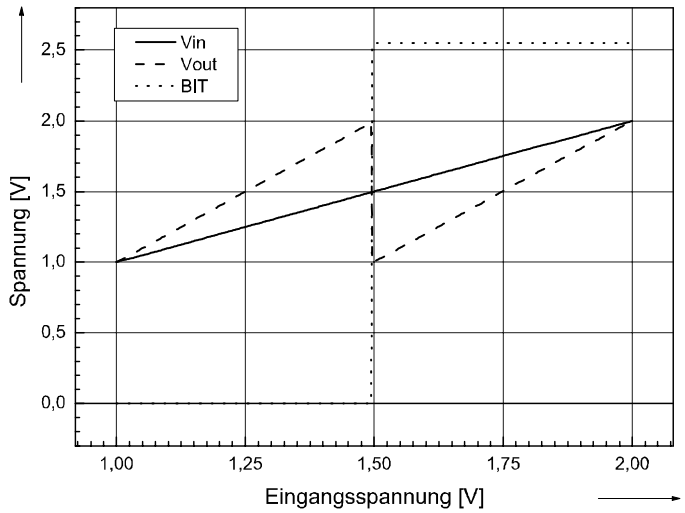

Abbildung 63: DC-Analyse der Konverterstufe.

5.3.2 Charakterisierung der Teilschaltung

Der Komparator, der dafür verantwortlich ist, daß die Charakterisierung nicht ohne weiteres durchführbar war, wird aus der Teilschaltung herausgelöst. Er kann separat modelliert werden oder als Transistormodell eingebaut werden. Die Teilschaltung ohne den Komparator am Eingang ist in Bild 62 durch eine gestrichelte Linie nach oben abgetrennt gezeichnet, sie wird als Funktionsblock charakterisiert. Der Block hat zwei Eingänge: einen analogen (Eingangssignal) und einen digitalen (Bit-Ausgang der Stufe). Der Ausgang ist ein analoges Signal, welches an den Eingang der nächsten Konverterstufe geschaltet wird. Entsprechend gliedert sich die Charakterisierung in zwei Teile. Zunächst erfolgt die Charakterisierung für das nicht gesetzte Bit, in einem zweiten Schritt wird die Charakterisierung wiederholt mit gesetztem Bit. Teil 1 der Charakterisierung ist die DC-Analyse. Bild 64 zeigt die Ergebnisse für die unterschiedliche Beschaltung des Bit Eingangs.

Das DC-Verhalten in Bild 64 beschreibt eine Hystheresekurve. Die DC-Kurven bilden die Übergangsfunktion für interne Ereignisse und werden als Methoden in dem Modell eingesetzt. Sie werden dazu verwendet, um zu Beginn der Simulation den Arbeitspunkt zu bestimmen, und um dann während der Simulation die Berechnung der effektiven Eingangsspannungsdifferenz zu erlauben.

Die Simulationen im Zeitbereich erfordern, im Gegensatz zu der DC-Simulation, nicht nur einen einzigen Lauf des Simulators, sondern einen Lauf für jeden diskreten Wert, den der am Eingang wirkende Spannungssprung annimmt. Für das Beispiel wurde ein entsprechender Charakterisierungsplan erstellt. Das Ergebnis der Charakterisierung ist eine Tabelle mit der direkten Zuordnung Größe Eingangssprung zu der Slewrate. Bild 65

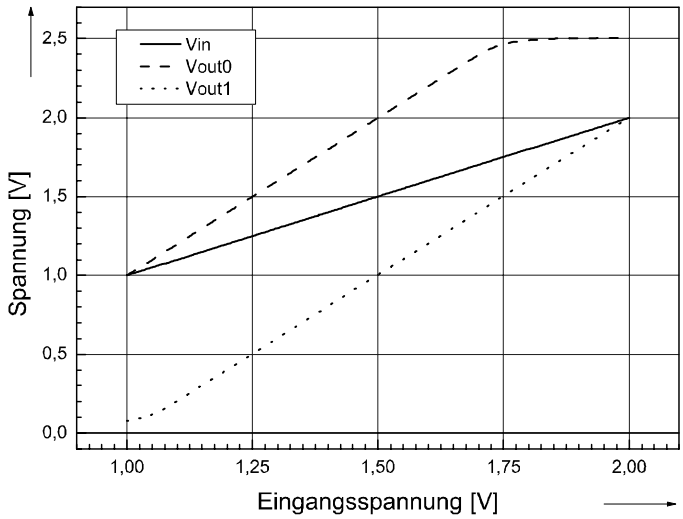

Abbildung 64: Ergebnis der DC-Analyse für Bit=0 und Bit=1.

illustriert den Zusammenhang.

Dieser tabellarische Zusammenhang wird mittels folgendem Mathematica-Aufruf in eine Methode `slewrate` umgewandelt, der maximal erlaubte Fehler (prozentualer Abstand zwischen mathematischer Modellgleichung und Datensatz) wird für das Modell auf 2.5 % festgelegt.

```
slewrate = BuildMethod[ daten, MaxError -> 2.5]
```

Die einzelnen Abschnitte der resultierenden Methode sind in Bild 65 durch vertikale Trennlinien visualisiert. Das Ergebnis kann in Mathematica weiter verarbeitet werden oder mittels eines Code-Generators in VHDL-AMS Code umgewandelt und direkt in das ausführbare Modell eingebunden werden. Wie im Beispiel Operationsverstärker sind die Werte der Verzögerungszeit in dem vorgegebenen Fehlerband konstant und fließen dementsprechend als Konstante `TD` in das Modell ein. Hier ein Auszug aus dem Simulatorcode, der die Methode `slewrate` im Modell repräsentiert:

```
-- Begin *******  Method slewrate0  ********
function slewrate0(Vdiff: real) return real is
variable sign, V_diff, SR: real;
begin
if Vdiff < 0.0 then
        sign := -1.0;
        V_diff := -1.0 * Vdiff;
else
        V_diff := Vdiff;
        sign := 1.0;
end if;
...  return sign * SR;
end;
-- End *******  Method slewrate0  ********
```

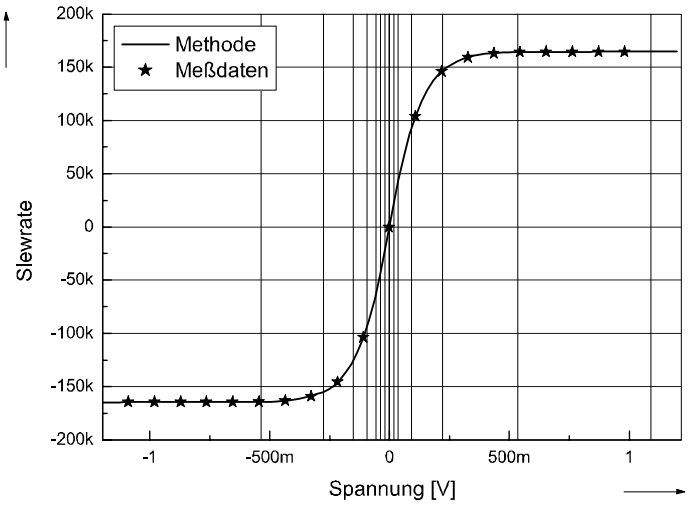

Abbildung 65: Meßdaten und Methode `slewrate` in Abhängigkeit der Größe der Differenz der Eingangsspannung.

Der gesamte VHDL-AMS Code für das Funktionsblockmodell ist im letzten Abschnitt dieses Kapitels aufgelistet.

5.3.3 Funktionsblockmodell der Teilschaltung

Das Funktionsblockmodell der Teilschaltung kann mit den Originalergebnissen auf der Makroebene verglichen werden. Dazu wird ein Sprung auf den Eingang gelegt. Dies ist eine für den A/D-Wandler relevante Testschaltung, denn die Beschaltung im Gesamtmodell erfolgt in Folge auf den Ausgang der vorhergehenden Konverterstufe, an welcher Span-

nungssprünge stattfinden. Abbildung 66 zeigt die Ergebnisse graphisch. Tabelle 15 zeigt den Geschwindigkeitsvergleich mit dem Makromodell.

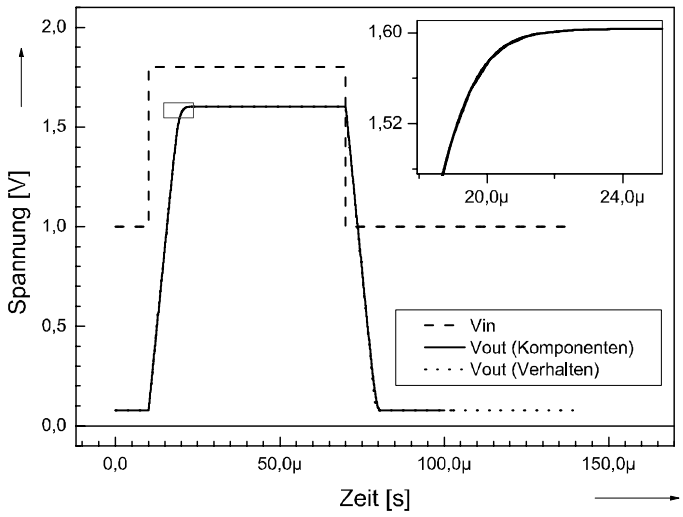

Abbildung 66: Vergleich Makromodell mit dem Funktionsblockmodell.

Abbildung 66 zeigt ein interessantes Verhalten, denn die Schaltung liefert nicht das beim Operationsverstärker typische Einschwingen, siehe Abbildung 56 im vorigen Beispiel. In diesem Beispiel erfolgt ein asymptotisches Angleichen an den stabilen Endwert, welches vom Funktionsblockmodell, wie in der eingesetzten Graphik deutlich wird, absolut korrekt vollzogen wird. Die Modellierungsmethodik gibt diesen Spezialfall wieder, ohne daß Änderungen bei der Erstellung des Modelles notwendig sind.

5.3.4 Simulationsergebnisse für einen 6-Bit ADC

Der 6-Bit A/D Wandler besteht aus 5 Konverterstufen, die in Serie geschaltet sind. Der analoge Ausgang der ersten Stufe ist an den Eingang der zweiten Stufe geschaltet usw. Die Serienschaltung bewirkt, daß sich Fehler in der Phase einer vorderen Stufen bis zur letzten Stufe fortpflanzen und additiv überlagern. Dazu kommt, daß die Flankensteilheit des analogen Ausgangs richtig sein muß, um die richtigen Schaltzeiten der folgenden Komparatoren zu gewährleisten.

An dieser Stelle wird der Unterschied zu vereinfachten Verhaltensmodellen deutlich. Durch die Berücksichtigung der Flankensteilheit durch die Methode Slewrate und durch die Berücksichtigung der Phasenverschiebung durch die Methode Verzögerungszeit ist ein korrektes Schaltverhalten der folgenden Stufen gewährleistet.

Zur Validierung des korrekten Verhaltens der Teilschaltung wird das so erzeugte Ver-

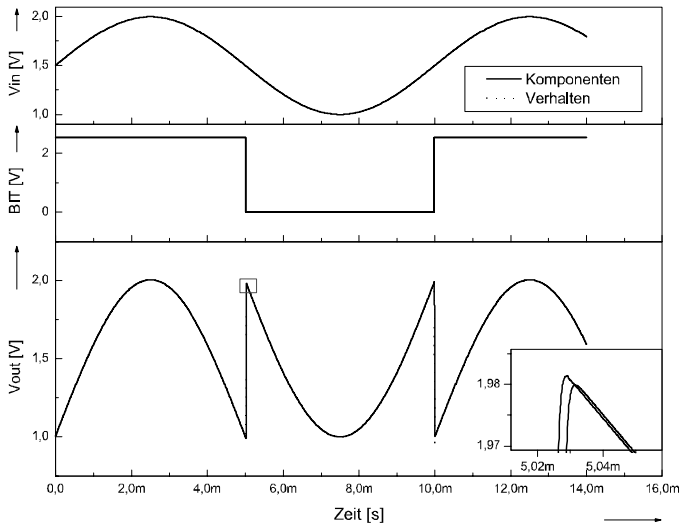

Abbildung 67: Vergleich Original mit Verhaltensmodell, gesamte Konverterstufe.

haltensmodell der Konverterstufe mit der Originalschaltung verglichen. Die Testbench wird mit einer Sinusschwingung angeregt, deren Amplitude den Arbeitsbereich der Konverterstufe abdeckt. Die Simulation erfolgt im Zeitbereich. Der Vergleich der Ergebnisse ist in Bild 68 dargestellt. Der digitale Ausgang vollzieht die Schaltvorgänge korrekt, der analoge Ausgang weist eine Phasenverschiebung der steigenden Flanke mit einem relativen Fehler kleiner als 1% auf. Die Genauigkeit läßt sich durch entsprechend höheren Aufwand bei der Charakterisierung weiter verbessern.

Tabelle 15 zeigt einen Vergleich der für die Simulation für verschiedene Schaltungskomplexitäten benötigten Rechenzeiten. Der Funktionsblock ist das Black-Box Modell der Teilschaltung, wie es zur Charakterisierung verwendet wurde, also unbeschaltet. Die Ergebnisse für die Konverterstufe beziehen sich auf die in Abbildung 67 dargestellte Simulation. Der 6-Bit A/D-Wandler wurde, wie oben beschrieben, mit einem Sinussignal angeregt. Die Ergebnisse aus Tabelle 15 zeigen, daß der erzielbare Geschwindigkeitsgewinn der Modelle mit zunehmender Schaltungsgröße ansteigt. Dies ist insbesondere bei der Simulation von komplexen Systemen von Interesse.

	Spice-Level	**Funktionsblockmodell**	**Speedup**
Funktionsblock	0:03	0:02	1,5
Konverterstufe	2:18	0:38	3,6
6-Bit A/D-Wandler	10:50	0:54	12

Tabelle 15: Vergleich der Simulationszeiten.

Tabelle 15 zeigt eine Verringerung der Simulationszeit um eine Größenordnung. Da-

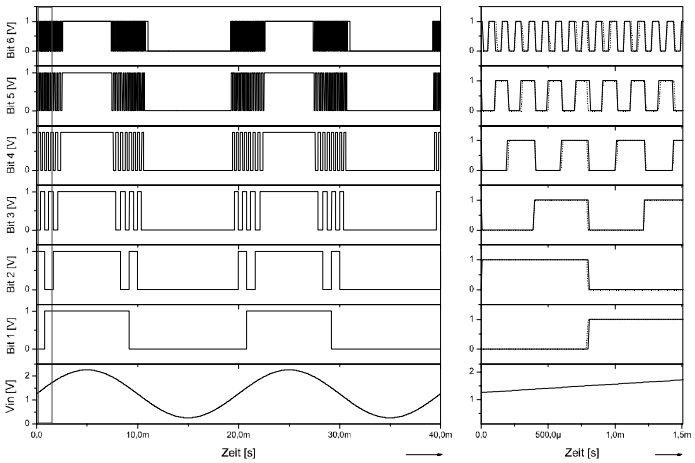

Abbildung 68: Ergebnisse der Simulation mit dem 6-Bit A/D-Wandler.

mit wird eine Reduktion der Zeit für eine typische Simulation aus dem Stunden- in den Minutenbereich erreicht. Der folgende Abschnitt zeigt den großen Vorteil der Funktionsblockmodelle im Vergleich zu Modellen auf der funktionalen Ebene, nämlich die durch die korrekte Wiedergabe der internen analogen Signalen erreichte Genauigkeit, speziell bei der im A/D-Wandler wichtigen Schaltzeitpunkten.

5.3.5 Vergleich mit einem funktionalen Modell

Ein typisches VHDL-AMS Verhaltensmodell für einen A/D-Wandler ist in Kapitel 2.4.2 dargestellt. Das Modell enthält eine mathematische Beschreibung des Ausgangsverhaltens in Abhängigkeit der Eingangsgrößen. Streng genommen steht das Modell gemäß der Abstraktionsebenen für analoge Schaltungen auf der funktionalen Ebene, denn es enthält eine rein algorithmische Beschreibung ohne Bezug auf die Erhaltungssätze. Da diese aber für den digitalen Teil (die Ausgänge sind digital) nicht relevant sind, kann das Modell sicher als eine Mischform betrachtet werden. Auf jeden Fall ist dieses Verhaltensmodell gut geeignet zum Vergleich mit dem mittels der neuen Methodik gewonnenen Verhaltensmodell, das im folgenden zur Unterscheidung von dem VHDL-AMS Verhaltensmodell als Funktionsblockmodell bezeichnet wird. Beide Modelle können so parametrisiert werden, daß sie einen A/D-Wandler mit identischen Arbeitsbereich darstellen und können mit identischen Signalen beschaltet werden. Als drittes Modell steht das Makromodell zur Verfügung, welches für die Generierung des Funktionsblockmodells verwandt wurde.

Im in DC-Analysen untersuchten Großsignalverhalten zeigen sich keine Unterschiede zwischen den Modellen - nach hinreichend langer Zeit befinden sich die Modelle im identischen stabilen Zustand. Hier sind die Modelle also beliebig untereinander austauschbar. Die weiteren Tests erfolgen im Transientenbereich.

Abbildung 69 zeigt einen Auszug der Simulationsergebnisse des ADC, nämlich das 6. Bit des Ausgangs. Die Schaltzeitpunkte im letzten Bit sind unterschiedlich, im Verhaltensmodell liegen sie deutlich daneben. Der Grund hierfür ist die additive Fortpflanzung der Fehler von Stufe zu Stufe. Ein Ausgang der Konverterstufe ist ein analoges Signal, dessen Flankensteilheit begrenzt ist. Während diese Flankensteilheit im Funktionsblockmodell mit der Methode `slewrate` modelliert und berücksichtigt wird, tritt im Verhaltensmodell ein idealer Sprung auf den neuen Wert auf. Dieser Sprung sorgt dafür, daß der nächste Schwellwert zu früh überschritten wird, der Komparator zu früh schaltet und somit ein Fehler in der Phase der nächsten Konverterstufe auftritt. Dieser Fehler ist zwar minimal, pflanzt sich aber bis zur fünften Stufe fort und nimmt in diesem Fall jeweils um eine Größenordnung zu.

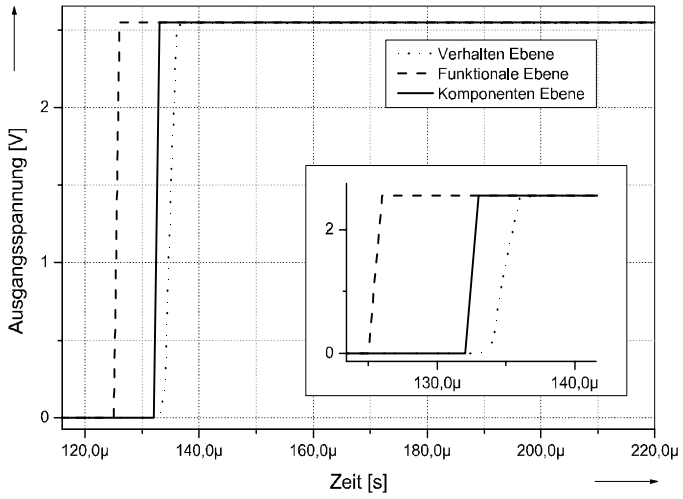

Abbildung 69: Vergleich der Simulationsergebnisse für Bit 6 (LSB).

Als nächstes Beispiel wird ein sinusförmiges Eingangssignal mit 10 kHz betrachtet. Diese Frequenz des Eingangssignals ist für die Originalschaltung zu groß: aufgrund der internen Verzögerungen der Bauteile kann der Ausgang nicht dem Eingang folgen. Der Wandler funktioniert nicht so, wie er mathematisch betrachtet sollte.

Die Ergebnisse für den Verlauf im 1. Bit (MSB), die von den beiden Modellen geliefert werden, sind sehr unterschiedlich, wie in Abbildung 70 deutlich wird. Das VHDL-AMS Verhaltensmodell liefert ein ideales Signal, welches zwar mathematisch korrekt ist, jedoch nicht der Praxis entspricht. Dem Anwender würde an dieser Stelle nicht bewußt werden, daß die Schaltung, an deren Entwicklung er arbeitet, in der Praxis nicht funktionieren wird.

Die Ergebnisse des mit der vorgeschlagenen Methodik erstellten Funktionsblockmodells stimmen zwar nicht mit der Originalschaltung überein, erlauben jedoch eine klare qualita-

tive Aussage: die Schaltung funktioniert in dieser Beschaltung mit diesem Eingangssignal nicht.

In Abbildung 70 wird dieses Verhalten deutlich: ganz unten das ideale Ausgangssignal, welches vom Verhaltensmodell geliefert wird. Darüber wird in den Makro- und Funktionsblockmodellen deutlich, daß der A/D-Wandler so nicht korrekt arbeitet.

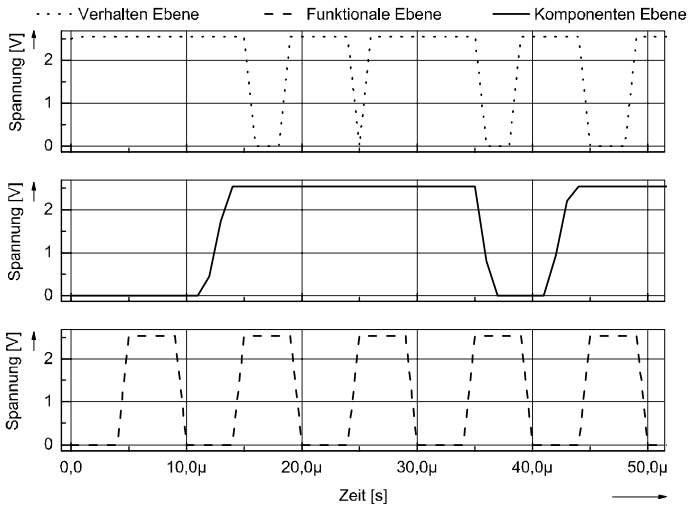

Abbildung 70: Vergleich des 1. Bit (MSB) der Modelle bei einem hochfrequenten Eingangssignal.

Dieser Vergleich macht gleichzeitig gewisse Grenzen des Funktionsblockmodells deutlich, denn die physikalischen Effekte, die für die Fehlfunktion des Makromodells verantwortlich sind, wurden bei der Charakterisierung nicht erfaßt und spiegeln sich dementsprechend nicht im Funktionsblockmodell wider.

5.3.6 Der VHDL-AMS Code des Funktionsblockmodells

Das Verhaltensmodell für die Teilschaltung ist in der Entity `generic_CONVSTAGE` definiert. Dies ist die Umsetzung des Funktionsblockmodells in VHDL-AMS. In diesem Modell ist die Verzögerungszeit auf einen konstanten Wert `TD` gesetzt, die Methoden `VoutDC` und `Slewrate` sind jeweils doppelt enthalten: Einmal für die Situation `Bit=0` und einmal für `Bit=1`. Beim Zusammensetzen des Ausgangs werden je nach Zustand des Bits die entsprechenden Methoden zur Berechnung verwandt.

```
library ieee;
use ieee.math_real.all;
   library disciplines;
```

```
use disciplines.electromagnetic_system.all;

entity generic_CONVSTAGE is

generic (
        V_out_min : emf := -9.6;
        V_out_max : emf :=  9.6;
        TD        : real := 5.8e-8);

port(
        terminal  Uin, Uout   : Electrical;
        signal Bit   : BOOLEAN);

end generic_CONVSTAGE;


architecture behavioral of generic_CONVSTAGE is

-- Begin *******  Method slewrate0  ********
-- Number of sections:  5, Maximum Error:  2.5%, Maximum Degree:  4
function slewrate0(Vdiff: real) return real is
variable sign, V_diff, SR: real;
begin
if Vdiff < 0.0 then
        sign := -1.0;
        V_diff := -1.0 * Vdiff;
else
        V_diff := Vdiff;
        sign := 1.0;
end if;

if V_diff <= 2.96e-6 then
        SR := 0.0;
elsif V_diff <= 1.419e-1 then
        SR := -4.4832957e3 - 1.1722595e6 * V_diff
+ 1.7897751e9 * V_diff ** 4.0e0 - 5.8280225e8 * V_diff ** 3.0e0
+ 6.0415514e7 * V_diff ** 2.0e0;
elsif V_diff <= 3.9859e-1 then
        SR := 3.1599691e4 + 6.3580799e5 * V_diff
- 7.7975484e5 * V_diff ** 2.0e0;
elsif V_diff <= 1.61786e0 then
        SR := 1.5973821e5 + 3.5260018e3 * V_diff;
else    SR := 165442.786;
end if;
return sign * SR;
end;
-- End *******  Method slewrate0  ********


-- Begin *******  Method slewrate1  ********
-- Number of sections:  9, Maximum Error:  2.5%, Maximum Degree:  3
```

```
function slewrate1(Vdiff: real) return real is
variable SR, V_diff, sign: real;
begin
if Vdiff < 0.0 then
        sign := -1.0;
        V_diff := -1.0 * Vdiff;
else
        V_diff := Vdiff;
        sign := 1.0;
end if;

if V_diff <= 2.96e-6 then
        SR := 0.0;
elsif V_diff <= 1.36e-3 then
        SR := -1.1791821e2 + 1.2488487e6 * V_diff;
elsif V_diff <= 1.772e-2 then
        SR := -1.1376556e2 + 1.2457953e6 * V_diff;
elsif V_diff <= 3.68e-2 then
        SR := 1.7868687e3 + 1.138536e6 * V_diff;
elsif V_diff <= 1.1176e-1 then
        SR := -3.5795385e1 + 1.3051697e6 * V_diff
-3.1821959e6 * V_diff ** 2.0e0;
elsif V_diff <= 2.5896e-1 then
        SR := 2.5414163e4 -1.5785964e6 * V_diff ** 2.0e0
+ 9.008659e5 * V_diff;
elsif V_diff <= 5.9152e-1 then
        SR := 8.5254322e4 + 4.443516e5 * V_diff
   + 5.3807284e5 * V_diff ** 3.0e0 - 8.4308599e5 * V_diff ** 2.0e0;
elsif V_diff <= 1.089e0 then
        SR := 1.6416098e5 + 4.863889e2 * V_diff;
else    SR := 165442.786;
end if;
return sign * SR;
end;
-- End *******  Method slewrate1  ********


-- Begin *******  Method VOUTDC0  ********
-- Number of sections:  3, Maximum Error:  2%, Maximum Degree:  3
function VOUTDC0(Vin: real) return real is
variable DC0: real;
begin
if Vin <= 1.646 then
        DC0 := 2.0 * Vin - 1.0004;
elsif (1.646 < Vin and Vin <= 1.79 ) then
        DC0 := -25.0895 * Vin ** 3.0e0 + 122.245 * Vin ** 2.0e0
- 196.394 * Vin + 106.242;
else
        DC0 := -0.628326 * Vin ** 2.0e0 + 2.45802 * Vin + 0.0998593;
end if;
```

```
return DC0;
end;
-- End *******  Method VOUTDC0    ********


-- Begin *******  Method VOUTDC1  ********
-- Number of sections:  3, Maximum Error:  2%, Maximum Degree:  3
function VOUTDC1(Vin: real) return real is
variable DC1: real;
begin
if Vin <= 1.052  then
        DC1 := 52.7017 * Vin ** 3.0e0 - 154.027 * Vin ** 2.0e0
   + 150.414 * Vin - 49.0104;
elsif Vin <= 1.116 then
        DC1 := -76.3004 * Vin ** 3.0e0 + 253.087 * Vin ** 2.0e0
- 277.849 * Vin + 101.159;
else
        DC1 := 1.99983 * Vin - 1.99626;
end if;

return DC1;
end;
-- End *******  Method VOUTDC1  ********

-- input
quantity U_in across
        Uin;

-- output
quantity
        U_out across
        U_out_I through
        Uout;

quantity VI : emf;



begin
        -- calculation of effective Output Difference
        if Bit use
                VI == VOUTDC1( U_in ) - U_out;
        else
                VI == VOUTDC0( U_in ) - U_out;
        end use;


        -- calculation of the output voltage
        -- limit voltage to maximum / minimum values
        -- through current is not limited, set by simulator
        if (now < TD and Bit) use
```

```
        -- if (now < TD) use
                U_out == VOUTDC1( U_in );
        elsif (now < TD) use
                U_out == VOUTDC0( U_in );
        elsif (U_out >= V_out_max) and
                (slewrate0( U_in'Delayed( TD )) > 0.0) use
                U_out == V_out_max;
        elsif (U_out <= V_out_min) and
                (slewrate0( U_in'Delayed( TD )) < 0.0) use
                U_out == V_out_min;
        else
                U_out'dot == slewrate1( VI'Delayed( TD ) );
        end use;


     -- telling the simulator engine about discontinouities
     break when now=TD;
     break when U_out'above(V_out_max);
     break when not(U_out'above(V_out_min));
end behavioral;
```

Die Konverterstufe ist ein Strukturmodell, in welchem das Funktionsblockmodell für die Teilschaltung mit einem Konverter verschaltet wird. Es entspricht der in Abbildung 62 dargestellten Schaltung, ergänzt um das Element `dtoa`, welches das digitale Signal `bit` in ein (analoges) Terminal umwandelt. Die Referenzspannungen, mit welchen die Originalschaltung verschaltet sind, fließen wie oben beschrieben als "generics" in das Modell ein.

```
-------------------------------------
-- Converter stage, consists  of comparator, switch,
-- and adder/amplifier
-------------------------------------
library disciplines;
use disciplines.electromagnetic_system.all;
entity Converterstage is
generic (
        vthresh: REAL := 1.5);

port(
        terminal  p1, pout, pgnd, pbit  : Electrical);

end entity Converterstage;


library disciplines;
use disciplines.electromagnetic_system.all;
use WORK.all;
architecture struct of Converterstage  is
        signal Bit   : BOOLEAN;
```

```
begin

        conv1   : entity dtoa(Behavioral) generic map(0.0, 2.55)
                        port map(bit, pbit);
        comp1   : entity LMC6762AB(Behavioral) generic map(vthresh)
                        port map(p1, pgnd, bit);
        stage1  : entity generic_CONVSTAGE(Behavioral)
                        port map(p1, pout, bit);
end architecture struct;
```

Die Struktur des 6-Bit A/D-Wandler, wie sie in Abbildung 61 dargestellt ist, wird im folgenden VHDL-AMS Code abgebildet. Der Wandler hat einen analogen Eingang und 6 digitale Ausgänge, welche als `port terminals` realisiert sind, um die Integration des Modells in die Eldo Simulatorumgebung und den direkten Vergleich mit dem (analogen) Makromodell zu ermöglichen, siehe die weiter unten dargestellte Testbench. Aus diesem Grund wird der Ausgang des Komparators LMC6762AB (das LSB), welcher als Modell auf der Ebene Verhalten eingesetzt ist, mittels dem Element `dtoa` in ein analoges Signal umgewandelt.

```
-------------------------------------
-- 6Bit ADC
-- Combining 5 Converter-Stages and one
-- Comparator results in a 6 Bit ADC
-------------------------------------
library disciplines;
use disciplines.electromagnetic_system.all;
entity ADC6 is
generic(
        vlow : REAL := 1.0;
        vhigh : REAL := 2.0);
port(
        terminal  p1, pgnd, pb1, pb2, pb3, pb4, pb5, pb6  : Electrical);

end entity ADC6;


library disciplines;
use disciplines.electromagnetic_system.all;
use WORK.all;
architecture struct of ADC6  is
        signal Bit6   : BOOLEAN;
        terminal pout1, pout2, pout3, pout4, pout5 : Electrical;
        constant vthresh : REAL := (vhigh + vlow) / 2.0;
begin

        conv1   : entity dtoa(behavioral) generic map(0.0, 2.55)
                        port map(Bit6, pb6);
        cs1     : entity ConverterStage(struct) generic map(vthresh)
```

```
                    port map(p1, pout1, pgnd, pb1);
        cs2     : entity ConverterStage(struct) generic map(vthresh)
                    port map(pout1, pout2, pgnd, pb2);
        cs3     : entity ConverterStage(struct) generic map(vthresh)
                    port map(pout2, pout3, pgnd, pb3);
        cs4     : entity ConverterStage(struct) generic map(vthresh)
                    port map(pout3, pout4, pgnd, pb4);
        cs5     : entity ConverterStage(struct) generic map(vthresh)
                    port map(pout4, pout5, pgnd, pb5);
        comp1   : entity LMC6762AB(behavioral) generic map(vthresh)
                    port map(pout5, pgnd, Bit6);

end architecture struct;
```

Die Testbench für den 6-Bit A/D-Wandler ist im folgenden dargestellt. Auf den Eingang wird eine Sinusschwingung gelegt, die Simulation erfolgt im Zeitbereich. Die Testbench ist in ELDO-FAS realisiert, das oben dargestellte VHDL-AMS Modell für den 6-Bit A/D-Wandler wird als externes Modell eingebunden. Die Ergebnisse der Simulation sind in Abbildung 68 dargestellt.

```
*ADC Simulation mit vhdlams-Modell

.model ADC6 MACRO LANG=VHDLAMS LIB=ADC

* Anweisungen:
.OP
.TRAN   1us     0.1ms
.PLOT  tran V(1)  V(400) V(401) V(402) V(403) V(404) V(405)


* Eingangssignal:
VIN     1       0       SIN(1.25 1.0 50)

* Dies paßt zur .model Einbindung
YADC   ADC6  1      0      400    401    402    403    404    405

.OPTIONS ASCII NOMOD LIST NODE NOPAGE

.option smooth
.option reltol=1N
.option vntol=1N
.option EPS=1N

.end
```

Das Einbinden des VHDL-AMS Modells in die Eldo Simulatorumgebung hat den Vorteil, daß die Vergleiche mit dem Modell auf Makroebene innerhalb einer Umgebung erfolgen kann. Die Simulationszeiten beziehen sich also auf den selben Simulator auf dem selben Rechner und sind so direkt miteinander vergleichbar.

5.4 Modellgenerierung

In diesem Abschnitt werden die drei präsentierten Anwendungsbeispiele zusammengefaßt. Zunächst werden die einzelnen Schritte dargestellt, die beim Erstellen eines Funktionsblockmodells durchgeführt werden müssen. Abschließend erfolgt die Auflistung der in den Beispielen benutzten Modelle, speziell die Übersicht über die Bestandteile der Funktionsblockmodelle.

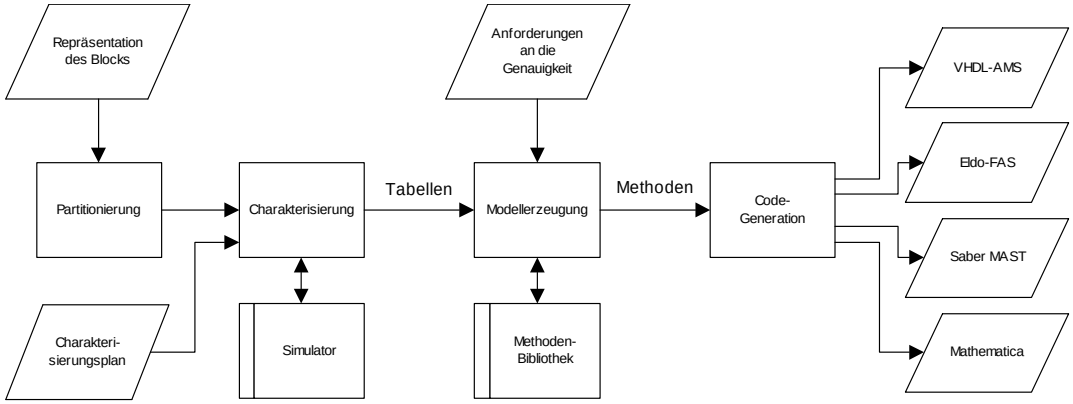

Abbildung 71: Zusammenfassende Darstellung des Ablaufs.

Die Schritte, die der Schaltungsentwickler durchführen muß, um von der Repräsentation der zu modellierenden Schaltung auf einer niederen Abstraktionsebene zum ausführbaren Funktionsblockmodell zu gelangen, sind in Abbildung 71 dargestellt. Die Eingaben sind die Repräsentation der Schaltung in der Sprache des benutzten Simulators, der Charakterisierungsplan und die Anforderungen an die Genauigkeit. Als Ergebnis wird das Funktionsblockmodell in einer HDL bzw. einer beliebigen Sprache - wie Mathematica - geliefert.

Partitionierung Der Partitionierung kommt bei großen Schaltungen eine wichtige Bedeutung zu. Die für die Charakterisierung benötigten Ressourcen (die Größe des Funktionsblocks korreliert mit der für die Charakterisierung benötigten Zeit) und die für die Ausführung des Gesamtmodells notwendigen Ressourcen hängen von der Partitionierung ab. Auch der Grad der Wiederverwendbarkeit läßt sich durch die Partitionierung steuern, so kann das Funktionsblockmodell MOPA1, das Teil der Schaltung des biquadratischen Filters ist, in jeder beliebigen Operationsverstärker-Schaltung eingesetzt werden, unabhängig davon, aus welcher Schaltung es herausgelöst wurde. Durch eine geeignete Partitionierung kann auch der Anwendungsbereich der Methodik erweitert werden: Die Konverterstufe der A/D Wandlerschaltung kann als Ganzes wegen der Unstetigkeit des Ausgangs nicht modelliert werden, nach der Aufteilung in zwei Funktionsblöcke - welche die Anforderungen erfüllen - wird die Modellierung erst möglich.

Charakterisierung Die Charakterisierung ist der mit dem meisten Aufwand verbundene Teil des Ablaufes. Der Aufwand kann reduziert werden, wenn im Charakterisierungsplan nur die Bereiche, in denen die Ergebnisse nichtlinear sind, mit sehr großer Genauigkeit simuliert werden, lineare Bereiche aber mit einer entsprechend reduzierten Anzahl Stützstellen. Auch kann, wenn der Anwendungsbezug von Anfang an klar ist, auf das Erzeugen von Daten verzichtet werden, die u.U. im Modell nicht benötigt werden, wie die Methode `RI` in einem Modell für die funktionale Ebene. Das Ergebnis der Charakterisierung sind, wie in Abbildung 71 gezeigt, Tabellen mit den Ergebnissen. Der Netzwerksimulator wird "ferngesteuert", d.h. die für die Durchführung des Charakterisierungsplanes notwendigen Simulationen werden mit der Repräsentation des Funktionsblocks automatisch durchgeführt und die relevanten Ergebnisse extrahiert.

Modellerzeugung Während der Modellerzeugung werden die Tabellen in Methoden umgewandelt. Hierzu wird direkt auf die Ergebnisse der Charakterisierung zurückgegriffen und mittels der Methodenbibliothek und den als Eingabe spezifizierten Anforderungen an die Genauigkeit die für das Funktionsblockmodell notwendigen Methoden erzeugt. Für unterschiedliche Anwendungsfälle können Methoden für dieselbe Funktion mit unterschiedlichen Anforderungen erzeugt werden, das Funktionsblockmodell kann dann bei der Ausführung entsprechend skaliert werden. So kann beispielsweise für eine Simulation auf der funktionalen Ebene eine Methode mit einem größeren maximalen Fehler benutzt werden, um ein effizientes Gesamtmodell zu erzeugen, für die Verifikation auf der Verhaltensebene können Methoden mit höherer Genauigkeit benutzt werden.

Code-Generation Der letzte Schritt, bevor das Modell ausgeführt werden kann, ist die Umwandlung der Methode in die vom Zielsimulator unterstützte HDL. Hierzu dient der Code-Generator, der leicht an neue Sprachen angepasst werden kann.

Das Ergebnis dieses Vorgehens ist das nach der vorgeschlagenen Methodik erstellte Funktionsblockmodell. Der Charakterisierungsplan und die Anforderungen an die Genauigkeit sind im Modell berücksichtigt, seine Funktion kann jetzt durch Vergleich mit der ursprünglichen Repräsentation verifiziert werden.

Modus	**Ebene**	**RC Filter**	**OpAmp**	**A/D Wandler**
DC	Verhalten	`VoutDC, RI`	`VoutDC, RI`	`VoutDC0, VoutDC1, RI`
DC	Funktional	`VoutDC`	`VoutDC`	`VoutDC0, VoutDC1`
TRAN	Verhalten	`VoutDC, RI, slewate, TD`	`VoutDC, RI, slewate, TD`	`VoutDC0, VoutDC1, RI, slewate0, slewrate1, TD`
TRAN	Funktional	`VoutDC,TD, slewate`	`VoutDC, TD, slewrate`	`VoutDC0, VoutDC1, slewate0, slewrate1, TD`

Tabelle 16: Bestandteile der Funktionsblockmodelle.

Tabelle 16 stellt die in den Beispielen dieses Kapitels erarbeiteten Funktionsblockmodelle vor und listet, welche Methoden für welchen Anwendungsfall notwendig sind. Die

Anwendungsfälle unterscheiden zwischen Simulationen im Großsignal- und Transientenbereich und Modelle für die Funktionale bzw. Verhaltensebene. Die Methoden sind beliebig austauschbar, d.h. dieselbe Methode `VoutDC`, die für eine Großsignalsimulation in einem funktionalen Modell benutzt wird, kann unverändert in ein Verhaltensmodell zur Transientensimulation eingebaut werden. Der A/D Wandler enthält die Methoden `VoutDC` und `slewrate` jeweils doppelt, da zwischen gesetztem und nicht gesetztem Biteingang unterschieden werden muß, damit die Hysteresekurve korrekt wiedergegeben wird.

Bei der Darstellung des Beispiels RC Filter wurde besonderer Wert auf die Methodik gelegt, die zur Charakterisierung und Modellerzeugung notwendigen Schritte wurden ausführlich gezeigt. Es wurde bewußt eine einfache Schaltung gewählt, damit der Vergleich mit herkömmlichen Verfahren möglich war.

Das Beispiel Operationsverstärker listet die in der Signatur des Funktionsblockmodells enthaltenen Elemente. Schwerpunkt in diesem Beispiel ist die Wiederverwendbarkeit: ein Modell wird in ganz unterschiedlichen Beschaltungen und mit unterschiedlichen Anregungen betrieben. Dazu kommt die Skalierbarkeit: Durch Hinzufügen der Methode Innenwiderstand `RI` wird das Modell zu einem echten Verhaltensmodell, die Kirchhoff'schen Gesetze werden dadurch erfüllt. Das Modell wurde mit den vom Hersteller ausgelieferten Modellen auf der Transistor- und Makroebene verglichen. Es wurde gezeigt, daß neben dem reduzierten Ressourcenbedarf der Anwendungsbereich breiter ist.

Das Beispiel A/D Wandler schließlich wurde gewählt, um die Anwendbarkeit der vorgeschlagenen Methodik anhand einer größeren gemischt analog/digitalen Schaltung zu demonstrieren. Der Schwerpunkt in diesem Beispiel liegt auf der Partitionierung, d.h. der Auswahl der modellierten Teilschaltung. Darüberhinaus ermöglichte dieses Beispiel einen Vergleich mit einem herkömmlichen VHDL-AMS Verhaltensmodell, die Vorteile der vorgeschlagenen Modellierungsmethodik traten dabei deutlich auf.

6 Zusammenfassung und Ausblick

Die vorliegende Arbeit stellt eine Methodik zur Modellierung gemischt analog / digitaler Schaltungen auf den Abstraktionsebenen Verhalten und Funktion vor. In diesem Zusammenhang erfolgt die Einordnung in die Abstraktionsebenen des analogen und digitalen Schaltungsentwurfs und der Vergleich mit dem Stand der Technik. Als Vorteile der vorgestellten Methodik gegenüber etablierten Modellierungsansätzen wurden folgende Punkte herausgearbeitet:

- Der Entwickler im Bottom-Up Entwurfszyklus kann selbst Modelle vollautomatisch generieren.
- Der Entwickler muß über kein detailliertes Wissen hinsichtlich des zu modellierenden Funktionsblock verfügen: Eine Repräsentation auf einer unteren Abstraktionsebene und die Spezifikation der Anforderungen an die Genauigkeit sind hinreichend.
- Die generierten Modelle sind unabhängig von einem bestimmten Simulator. Sie lassen sich durch die ausschließliche Verwendung von Standard-Sprachelementen mittels eines Codegenerators in unterschiedliche HDL umsetzen.
- Gemischt analog / digitale Modelle können erzeugt werden. Dabei ist wichtig, daß die Methodik nicht nur eine Schnittstelle zur digitalen Welt enthält, sondern daß die Ein- / Ausgänge der Modelle digitale Signale enthalten können. Die digitalen Signale werden im Modell genau wie die analogen gehandhabt.
- Die generierten Modelle lassen sich einfach skalieren. Ein und dasselbe Modell kann sowohl auf der Funktionalen wie auch auf der Verhaltensebene eingesetzt werden. Auch unterschiedliche Anforderungen an die Genauigkeit können in einem Modell realisiert werden.

Nichtlinearitäten werden unterstützt und die relevanten charakteristischen Eigenschaften des Funktionsblocks sind im Modell implizit berücksichtigt. Modelle für eine lineare Filterschaltung, einen Operationsverstärker und einen A/D-Wandler wurden mit der neuen Methodik entwickelt und im Kapitel Beispiele dargestellt. Die Modelle wurden in den für die jeweilige Schaltung relevanten Standard-Testschaltungen (z.B. Sprungantwort und Großsignalverhalten beim Operationsverstärker) eingesetzt. Dabei erfolgte nicht nur der Vergleich zu Modellen auf der Komponentenebene, sondern speziell mit herkömmlichen Modellen auf höheren Abstraktionsebenen wie den von den Herstellern bereitgestellten Makromodellen. Dabei konnten die oben gelisteten Vorteile nachgewiesen werden. Die Einhaltung der gewünschten Anforderungen an die Genauigkeit wurde durch Vergleich mit den Modellen auf der Netzlisten-Ebene verifiziert. Auch eine Steigerung der Effizienz (Speicherplatz und Simulationszeit) wurde nachgewiesen, sowohl im Vergleich mit Transistormodellen als auch mit Makromodellen.

Ein Schwerpunkt dieser Arbeit liegt auf der Definition der systemtheoretischen Basis für die Methodik. In der Systemtheorie steht dazu ein erweiterter Formalismus für kombinierte Modelle als Instrumentarium zur Verfügung, mit dem die formale Spezifikation des Konzepts erfolgte. Auf diese theoretische Spezifikation setzt die Implementierung der Verfahren zur Generierung der Modelle auf. Der Schwerpunkt bei der Implementierung liegt auf der mathematischen Modellbildung und den Fehlerfortpflanzungsmodellen, die die Aussagen zur Genauigkeit des Modells erlauben.

Für den längerfristigen Einsatz in der Praxis muß die Charakterisierung weiter automatisiert werden. Das Generieren der Modelle soll durch die Integration in eine visuelle Benutzeroberfläche erleichtert werden. Dieser Schritt wird das Aufstellen der Modelle weiter erleichtern.

Die Abbildung der Methodik auf den neuen VHDL-AMS Industriestandard ermöglicht nicht nur einen wesentlichen Fortschritt im Entwicklungsablauf gemischt analog/digitaler Schaltungen, sondern insbesondere den Transfer der Methodik auf interdisziplinäre heterogene Systeme. Die systemtheoretische Spezifikation liefert hierbei die Basis für die Simulation und Modellierung allgemeiner dynamischer Systeme. Die auf dem Konzept zur Modellerstellung aufbauende Methodik muß in Bezug auf ihre Anwendbarkeit auf praktische Aufgabenstellungen außerhalb des Bereichs der elektronischen Schaltungen bewertet werden: Im Zentrum des Interesses stehen dabei heterogene Systeme, die Komponenten aus unterschiedlichen Ingenieursdisziplinen (z. B. Maschinenbau, Elektrotechnik) enthalten.

Literatur

[Ana96] Analogy, Inc.: *MAST Language Reference Manual*, 1996.

[Ant01] Antrim: *Mixed-Signal Characterization and Library Generation*, http://www.antrim.com, 2001.

[Arm89] J. Armstrong: *Chip-Level Modeling with VHDL*, Prentice-Hall, 1989.

[BCP74] R. Boyle, M. Cohn und O. Pederson: *Macromodeling of Integrated Circuit Operational Amplifiers*, in *IEEE J. Solid-State Circ.*, Seiten 353–364, IEEE Press, 1974.

[BD87] G.E.P. Box und N.R. Draper.: *Empirical Model-Building and Response Surfaces*, John Wiley and Sons, 1987.

[Bev69] P. R. Bevington: *Data Reduction and Error Analysis for the Physical Science*, McGraw-Hill Book Company, New York, 1969.

[BGHW96] A. Bleck, M. Goedecke, S. Huss und K. Waldschmidt: *Praktikum des modernen VLSI-Entwurfes*, B.G. Teubner Stuttgart, 1996.

[Bli94] J. Bliesener: *Approximation von Ausgangsfunktionen analoger Komponenten in Mathematica*, Diplomarbeit, TH Darmstadt, FB Informatik, Fachgebiet Integrierte Schaltungen und Systeme, 1994.

[Bor97] C. Borchers: *Automatische Generierung von Verhaltensmodellen für nichtlineare Analogschaltungen*, Volume 254 *Rechnerunterstützte Verfahren*, VDI Verlag GmbH, 1997.

[Bos87] H. Bossel: *Systemdynamik*, Vieweg & Sohn, Braunschweig, 1987.

[Box78] G.E.P. Box: *Statistics for Experimenters*, John Wiley and Sons Inc., 1978.

[Cel91] F. Cellier: *Continuous System Modeling*, Springer-Verlag, New York, 1991.

[CL75] L.O. Chua und P.M. Lin: *Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques*, Englewood Cliffs, NJ, Prentice Hall, 1975.

[CY94] G. Casinovi und I.-M. Yang: *Multi-Level Simulation of Large Analog Systems Containing Behavioral Models*, IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems, 13(11):1391–1399, 1994.

[DP99] J. Dabrowski und A. Pulka: *Experiences with Modeling of Analog and Mixed A/D Systems Based on PWL Technique*, in DATE Conference, München, Deutschland, 1999.

[Eic92] K. Eickhoff: *Table Models for efficient MOS Circuit Simulation*, EDAC, 1992.

[FRHG98] F. Fernandez, A. Rodriguez-Vazques, J. Huertas und G. Gielen: *Symbolic Analysis Techniques*, in *Applications to Analog Design Automation*, Piscataway, NJ:IEEE Press, 1998.

[Fuc96] B. Fuchssteiner: *Computeralgebrasysteme: Stand der Technik und Perspektiven*, in *4. GMM/ITG Diskussionssitzung Entwicklung von Analogschaltungen mit CAE-Methoden*, Seiten 17–31, C-LAB / Analoge Systemtechnik, Paderborn, 1996.

[GDWL92] D. Gajski, N. Dutt, A. Wu und S. Lin: *High-Level Synthesis: Introduction to Chip and System Design*, Kluwer Academic Publishers, 1992.

[Ger90] M. Gerbershagen: *Ein Programm zur prozeduralen Simulation und Charakterisierung analoger Zellen*, AEG Aktiengesellschaft, 1990.

[GH94] M. Goedecke und S.A. Huss: *An interactive Environment for the Characterization of mixed-signal Circuits*, in *3. ITG/GME-Conf. "CAE-Methods for analog Design"*, 1994.

[GHH95] M. Goedecke, H. Hamad und S.A. Huss: *A Methodology for the Development of System-Level Simulation Models for Analog Functional Blocks*, Archiv für Elektronik und Übertragungstechnik, 49:72–80, 1995.

[Gie96] G. G. E. Gielen: *Progress and Potential of Symbolic Techniques in Analog Design Automation*, in *4. GMM/ITG Diskussionssitzung Entwicklung von Analogschaltungen mit CAE-Methoden*, Seiten 3–15, C-LAB / Analoge Systemtechnik, Paderborn, 1996.

[Goe01] M. Goedecke: *Verfahren zur Beschreibung von Simulations- und Charakterisierungsabläufen für gemischt digital/analoge Schaltungen*, Dissertation, Technische Universität Darmstadt, FB Informatik, Fachgebiet Integrierte Schaltungen und Systeme, 2001.

[GW96] C. Grimm und K. Waldschmidt: *KIR - A Graph-Based Model for Description of Mixed Analog/Digital Systems*, in *European Design Automation Conference*, Genf, Schweiz, 1996.

[GW98] C. Grimm und K. Waldschmidt: *Hybride Datenflussgraphen*, Interner Bericht, Johann Wolfgang Goethe-Universität Frankfurt, Fachbereich Informatik, 1998.

[Ham95] H. Hamad: *Konzepte zur Entwicklung von abstrakten Verhaltensmodellen analoger Funktionsblöcke für die Systemsimulation*, Dissertation, Technische Hochschule Darmstadt, FB Informatik, Fachgebiet Integrierte Schaltungen und Systeme, 1995.

[HGT91] S.A. Huss, M. Gerbershagen und G. Tränkle: *Automatic Performance Characterization of Analog Functional Blocks*, Analog Integrated Circuits and Signal Processing, 1:277–286, 1991.

[HH98] T. Halfmann und E. Henning: *Analog Insydes Tutorial*, ITWM, Kaiserslautern, 1998.

[HKR99] S.A. Huss, S. Klupsch und R. Rosenberger: *Modellierung gemischt analog/digitaler Schaltungen mit VHDL-AMS*, in W. Gross W. John und H. Luft (Editor), *8. GMM Workshop Methoden und Werkzeuge zum Entwurf von Mikrosystemen*, Nachtrag, FhG IZM Advanced System Engineering, Paderborn, 1999.

[IEEE01] IEEE Computer Society: http://www.vhdl.org/analog/, 2001.

[IMGB82] A. Ipri, L. Mewin, N. Goldsmith und F. Brehm: *Two-Dimensional dynamic Analysis of Short-Channel thin Film MOS Transistors using a Mini Computer*, IEEE Trans. Electron Devices, ED-29:618–625, 1982.

[Ise92] R. Isermann: *Identifikation Dynamischer Systeme*, Springer-Verlag, 1992.

[JRS91] Y.-C. Ju, V. Rao und R. Saleh: *Consistency Checking and Optimization of Macromodels*, IEEE Transactions on Computer-Aided Design, 10(8):957–967, 1991.

[Kas00] M. Kasper: *Mikrosystementwurf*, Springer Verlag, 2000.

[KC87] A.I. Khuri und J.A. Cornell: *Response Surfaces: Designs and Analyses*, Marcel Dekker Inc., 1987.

[MAP97] V. Moser, H. P. Amann und F. Pellandini: *Behavioral Modeling of analogue Systems with Absynth*, in O. Leria, A. Vachoux, J.-M. Berge und J. Rouillard (Editor), *Analog and Mixed-Signal Hardware Description Languages*, Kluwer Academic Press, 1997.

[Max95] Maxim Integrated Products: *MAXIM Single/Dual/Quad, Micropower, Single-Supply Rail-to-Rail Op Amps*, Maxim Integrated Products Datasheet, 1995.

[MM91] M. Maynard und A. Maynard: *ELDO-FAS Dynamic System Modeling*, Anacad Computer Systems, 1991.

[Mon91] D.C. Montgomery: *Design and Analysis of Experiments*, John Wiley and Sons, 1991.

[Nat99] National Semiconductor: *LMC6762 Dual MicroPower Rail-To-Rail Input CMOS Comparator with Push-Pull Output*, National Semiconductor Product Folder, http://www.national.com, 1999.

[Pra91] H. Praehofer: *System Theoretic Foundations of Combined Discrete-Continous System Simulation*, Phd. Thesis, Johannes Kepler Universitaet Linz, Department of Systems Theory, 1991.

[Rat83] D. Ratkowsky: *Nonlinear Regression Modeling*, Volume 48 of *Statistics, Textbooks and Monographs*, Marcel Dekker Inc., 1983.

[Ray90] Raytheon Company - Semiconductor Division: *RLA Linear Macrocell Array Breadboarding Kit*, 1990.

[RH98] R. Rosenberger und S. A. Huss: *A Systems Theoretic Approach to Behavioural Modeling and Simulation of Analog Functional Blocks*, in *DATE Conference*, Seiten 721–728, Paris, Frankreich, 1998.

[Ros94a] R. Rosenberger: *Entwurf und Implementierung einer Methodenbibliothek zur Verhaltensmodellierung analoger Schaltungen*, Diplomarbeit, Technische Hochschule Darmstadt, FB Informatik, Fachgebiet Integrierte Schaltungen und Systeme, 1994.

[Ros94b] R. Rosenberger: *Integration von Mathematica in die Charakterisierungssprache CLANG mittels MathLink*, Studienarbeit Technische Hochschule Darmstadt, FB Informatik, Fachgebiet Integrierte Schaltungen und Systeme, 1994.

[Sof94] M. Sofroniou: *An Efficient Symbolic-Numeric Environment By Extending Mathematicas Format Rules*, MathSource Item No. 0205-254, 1994.

[STH98] R. Sommer, M. Thole und E. Hennig: *A Generic Circuit Modeling Strategy Combining Symbolic and Numeric Aanalysis*, in *Proc. SMACD'98*, Kaiserslautern, 1998.

[Str97] B. Stroustrup: *The C++ Programming Language*, Addison Wesley Publishing Company, 1997.

[Tho90] J.U. Thoma: *Simulation by Bondgraphs*, Springer-Verlag Berlin Heidelberg New York, 1990.

[VHDL97] IEEE DASC 1076.1 Home Page: *Analog and Mixed-Signal Extensions to VHDL*, http://vhdl.org/vi/analog/index.html, 1997.

[VS83] J. Vlach und K. Singhal: *Computer Methods for Circuit Analysis and Design*, Van Nostrand Reinhold, New York, 1983.

[Weh92] N. Wehn: *Rechnergestützter Entwurf mikrolektronischer Schaltungen*, Unterlagen zur Vorlesung Wintersemester 1992/1993, 1992.

[Wei01] E. Weisstein: *Eric Weisstein's World of Mathematics*, http://www.mathworld.-wolfram.com/ErrorPropagation.html, 2001.

[Wol98] Wolfram Research: *Experimental Data Analyst Version 2*, 1998.

[Wol99] S. Wolfram: *The Mathematica Book*, Fourth Edition, Cambridge University Press, Cambridge, 1999.

[WPHH99] T. Wichmann, R. Popp, W. Hartong und L. Hedrich: *On the Simplification of Nonlinear DAE Systems in Analog Circuit Design*, in *Proc. CASC'99*, München, Juni 1999.

[YA91] K. Yoon und P. Allen: *An Adjustable Accuracy Model for VLSI Analog Circuits Using Lookup Tables*, Analog Integrated Circuits and Signal Processing, 1:45–63, 1991.

[Yoo90] K.S. Yoon: *A Precision Analog Small-Signal Model for Submicron MOSFET Devices*, PhD Thesis, Georgia Institute of Technology, 1990.

[Zei84] B.P. Zeigler: *Multifacetted Modelling and Discrete Event Simulation*, Academic Press, 1984.

[Zei89] B.P. Zeigler: *Models as System-Theoretic Specifications*, in M. Singh (Editor), *Encyclopedia of Systems and Control*, Pergamon Press New York, 1989.

Lebenslauf

Name:	Ralf Peter Rosenberger	
geboren:	22. Juli 1968 in Frankfurt am Main	
Familienstand:	verheiratet seit 2000	
Schulausbildung:	1975 – 1979	Lindenschule Kriftel
	1979 – 1985	Weingartenschule Kriftel
	1985 – 1988	Main-Taunus-Schule Hofheim Abschluß: Allgemeine Hochschulreife
Wehrdienst:	1988 – 1989	2. Raketenartilleriebataillon 52 Gießen
Studium:	1989 – 1994	Technische Hochschule Darmstadt Abschluß: Diplom-Informatiker
Berufliche Tätigkeit:	1995 -2001	Externe Promotionsstelle FG Integrierte Schaltungen und Systemen
	1995-heute	ADDITIVE GmbH, Friedrichsdorf Abt. Engineering & Science Technischer Softwarevertrieb