Simulation

Bernhard Hoppe

Fachbereich Elektrotechnik und Informationstechnik

Hochschule Darmstadt
University of Applied Sciences
Fachbereich EIT

15.9.08

Einleitung

Die Nachfrage für hoch integrierte technische Systeme, die Zustände erfassen, Informationen verarbeiten und komplexe Geräte steuern und regeln können und dabei Aufgaben wie Bild- und Signalverarbeitung übernehmen, steigt ständig. In diesen Systemen spielt die Elektronik eine zentrale Rolle, muss aber über Sensorik und Aktorik mit einer Umwelt in Kontakt treten und diese beeinflussen. Diese Umgebung ist aber weder primär elektronisch noch funktioniert sie nach digitalen Prinzipien, wie die eingesetzten Prozessoren. Die Entwicklungsprozesse für solche komplexen Systeme sind selbst hoch kompliziert, kostenintensiv, zeitaufwendig und risikobehaftet. Deshalb werden hier strukturierte softwarebasierte Methoden eingesetzt, die den Entwickler durch den gesamten Designprozess von der Systemdefinition, dem Austesten von Entwurfsalternativen, bis hin zur Validierung und Verifikation führen und begleiten.
Die zentrale Funktion dieser Designsysteme ist die Simulation der verschieden weit ausgearbeiteten Designvarianten. Mit Simulationen wird immer wieder die Übereinstimmung mit der Systemspezifikation überprüft und verschiedene Designalternativen werden ausgetestet. Simulation ist die Vorausberechnung des erwarteten tatsächlichen Systemverhaltens unter Verwendung von mathematisch physikalischen „Modellen“, d. h. Gleichungssysteme, die die Systemvariablen miteinander und den äußeren Vorgaben, den „Eingangsgrößen“, in Beziehung setzen. Diese Modellgleichungen sind in der Regel nicht mehr in geschlossener Form also formelmäßig aufzulösen. Sondern nur noch für bestimmte konkrete Sätze von Zahlenwerten numerisch auszuwerten. Die eingesetzten Lösungsalgorithmen bilden zusammen mit je nach Tool unterschiedlich weit ausformulierten Modelltemplates ein Simulationssystem. In diesem Lehrbrief werden die Verfahren, Konzepte und Methoden für verschiedene in der Technik weit verbreitete Simulationssysteme behandelt, SPICE, VHDL-AMS und MATLAB.
Diese drei Programme sind prototypisch zu sehen. SPICE ist ein Simulationsprogramm mit klar definierter und fest umrissener Anwendungsdomäne, dem Elektronikentwurf, welches alle Modelle, die benötigt werden, bereitstellt, während MATLAB ein allgemeines Werkzeug ist, das auf einer eigenen Sprache für das Programmieren technisch-naturwissenschaftlicher Probleme beruht und über eine spezielle Umgebung innerhalb derer die Modelle ausgewertet und die Lösungen visualisiert werden können. VHDL-AMS nimmt eine Zwischenstellung ein. Die Sprache ist eine Übermenge des Sprachumfangs der digitalen Hardwarebeschreibungssprache VHDL, die die Funktionalitäten, die mit SPICE nachgebildet werden können, in den Sprachumfang einbezieht und darüber hinaus, wie MATLAB, Systeme aus Komponenten aus verschiedenen physikalischen Teilgebieten einbeziehen kann (Optik, Mechanik, Thermodyna-

mik, Hydrodynamik, usw.). So können heterogene Systeme im Zusammenspiel mit analoger und digitaler elektronischer Hardware analysiert werden.
Nur über Simulation und Modellbildung unter Verwendung der genannten Werkzeuge oder ähnlicher Tools lassen sich die Entwicklungsprozesse für komplexe technische Systeme, die heute am Markt nachgefragt werden, beherrschen und in den sich immer weiter verkürzenden Zeitphasen für die Entwicklung neuer Systeme erfolgreich durchführen.
Die Simulation ist ein Standardwerkzeug im Rahmen der Systementwicklung und heute wird kein System mehr ohne vorherige Simulation zur Produktion freigegeben. Simulation wird zur Verifikation des Designs vor der physikalischen Implementierung eingesetzt und sie kann auch zum Vergleich verschiedener Designalternativen herangezogen werden. Simulation ist also ein Hilfsmittel zum Systemdesign. Die wesentliche Einschränkung der Simulation liegt darin, dass sich die Wirklichkeit nur näherungsweise in einem Computermodell nachbilden lässt. Aber auch so ist es für den Designer möglich, das Verhalten des Designs vor der Implementierung zu studieren, um Fehler zu korrigieren und den Entwurf in Bezug auf Herstellkosten und Systemeigenschaften zu optimieren.
Bei der Simulation wird die Realität aber nur nachgebildet. Wenn komplexe Systeme zu simulieren sind, dann werden bestimmte physikalische Eigenschaften, die man zunächst für unwichtig hält, nur rudimentär oder auch gar nicht berücksichtigt. Es kommt also entscheidend für die Aussagekraft von Simulationen darauf an, dass die verwendeten Modelle die relevanten Eigenschaften nachbilden. Es besteht aber immer die Gefahr, dass qualitative Abweichungen im Verhalten des realen Systems vom simulierten auftreten. Man kann also nicht prinzipiell von einer fehlerfreien Simulation auf ein fehlerfreies Systemverhalten in der Realität schließen.

Historie

Die ersten kommerziellen Simulationswerkzeuge wurden in der Mikroelektronik eingesetzt. Das bekannte Programm SPICE wird seit Anfang der 1970er Jahre zur analogen Schaltkreissimulation beim Schaltungsentwurf eingesetzt, weil nur durch Simulation die Funktion des Chipdesigns soweit überprüft werden kann, dass nur wenige kostspielige und zeitintensive Redesigns mit neuer Masken- und Prototypenproduktion nötig waren. SPICE spielt bis heute eine zentrale Rolle im technologienahen Designprozess auf Transistorebene. Auf Basis dieses Simulators haben sich zunächst proprietäre Simulations-Tools von verschiedenen CAE-Anbietern etabliert (Digitalsimulatoren wie QUICKSIM, Analogsimulatoren wie ELDO), bevor 1986 die ersten Hardwarebeschrei-

bungssprachen (VHDL, Verilog) als von der amerikanischen Ingenieursvereinigung IEEE genormtes Designeingabemedium für den digitalen Systementwurf auf den Markt kamen. Mit diesen Sprachen konnten Teile des Systems über eine Verhaltensbeschreibung and andere Teile als Netzliste digitaler Komponenten (Gatter, Flip-Flops) in einheitlichen Modellen beschrieben und mit Simulatoren (Verilog-XL, ModelSim) gleichzeitig simuliert werden.
In den frühen 1990er Jahren waren die Halbleiterprozesse mittlerweile soweit entwickelt, dass ganze System auf einem Chip realisiert werden konnten. In diesen Systemen sind typischerweise nicht nur digitale Komponenten konzentriert, sondern es werden gerade zur Anpassung und Interaktion mit dem umgebenden elektronischen System analoge Blöcke gebraucht, die technologisch problemlos mit integriert werden können. Um auch diese Systeme als Ganzes simulieren zu können, wurde von der IEEE eine Arbeitsgruppe eingesetzt, die 1999 die Erweiterung 1076.1 *Definition of Analog and Mixed Signal Extensions to IEEE Standard VHDL* herausgab. Mit dieser Sprache und den zugehörigen IEEE-Bibliotheken können analoge, digitale, gemischt analog-digitale (*mixed signal*) Systeme beschrieben und darüber hinaus nicht-elektrische Systemeigenschaften (Mechanik, Optik, Fluidik) mit und ohne Bezug zur Elektronik modelliert werden.
War VHDL-AMS eine Entwicklung, die von der elektrotechnischen Anwendung getrieben wurde, so hat sich aus der numerischen Mathematik heraus ein umfangreiches Softwarepaket etabliert, das die Programmiersprache MATLAB® zur Modelldefinition benutzt. Wie der Name MATrix LABoratory schon zum Ausdruck bringt, liegt die Stärke hier in der Matrizen- und Vektorrechnung. Auf Grundlage eines Basismoduls (Ein-Ausgabefunktionen, Programmablaufsteuerung und insbesondere umfangreiche zwei- und dreidimensionale Visualisierungsmöglichkeiten) sind zahlreiche Erweiterungspakete, so genannte Toolboxen verfügbar, mit deren Hilfe MATLAB® in den verschiedensten Anwendungsgebieten (Regelungstechnik, Signalverabeitung, Optimierung, usw.) als effizientes Werkzeug zur simulativen Untersuchung technischer Systeme eingesetzt werden kann.

Überblick

In diesem Lehrbrief werden Simulationsverfahren anhand von verschiedenen Beispielen vorgestellt und eingeübt: Analog-Simulation mit SPICE, Systemsimulation mit VHDL-AMS und Verhaltenssimulation mit MATLAB.
Die verwendeten Simulationsprogramme unterscheiden sich im Zuschnitt für die jeweiligen primären Anwendungsbereiche, beruhen aber auf denselben Prinzipien. Die Begriffe Modell, Theorie und Simulation werden in Kapitel 1

eingeführt und die verschiedenen Modellklassen und Simulationsdomänen für analoge, digitale und Mixed-Signal-Anwendungen gegeneinander abgegrenzt. Die Genauigkeit von Simulationen, ihre Verwendung während des Designprozesses werden erläutert und die heute üblichen Designstrategien und die Rolle der Simulation in diesen werden verglichen. An einfachen Systemen werden Simulationen mit den drei Simulationsverfahren beispielhaft dargestellt. Abschließend werden in diesem Kapitel Leitlinien diskutiert, wie Simulatoren im Designablauf am sinnvollsten eingesetzt werden können.

Das zweite Kapitel befasst sich mit der Simulation aus elektrotechnischer Sicht. Aus der Elektronik stammen die ersten Computerwerkzeuge, mit denen überhaupt Systeme modelliert und simuliert werden konnten. Der Prototyp eines Simulators ist das Programm SPICE (*Simulation Program with Integrated Circuit Emphasis*). SPICE unterstützt verschiedene Analyseformen (Kleinsignal, Großsignal, Arbeitspunktberechnung) und verfügt über eine Bibliothek generischer Bauelementmodelle, die über Parameter an bestimmte Bauteiltypen oder an bestimmte Herstellprozesse angepasst werden können (Abschnitt 2.1). Die Schaltungseingabe erfolgt über SPICE-Netzlisten mit einer speziellen Syntax, heute aber auch mit grafischen Oberflächen über die Schaltplaneingabe. Bereits in diesem doch recht betagten Programm sind bereits alle Möglichkeiten moderner Designsysteme enthalten: Hierarchisierung über die Definition von *Subcircuits* und die Wiederverwendung (*Design Reuse*) vordefinierten Funktionsblöcke (Abschnitte 2.2 und 2.3).

SPICE ist ein *Schaltkreissimulator*, der für Netzwerke von elektrischen Bauelementen die Spannungen in den Zweigen und die Ströme in den Knoten unter Einhaltung der physikalischen Prinzipien (hier der Kirchhoffschen Gesetze) numerisch berechnet. Bei digitalen Systemen wird dieser Detaillierungsgrad nicht benötigt (Abschnitt 2.4). Hier reicht es auch, die Systeme als Flussdiagramm mit logischen Operatoren aufzufassen, in dem die Systemvariablen binäre Zustände mit den Werten „0“, „1“, „X“ und „Z“ sind. Die zeitliche Entwicklung des Systems wird durch Signallaufzeiten in den einzelnen logischen Verknüpfungsstufen beeinflusst. Die Systemvariablen ändern sich nur bei Signalwechseln an den Eingangspins oder auf inneren Knoten durch verzögerte Signale. Zwischen diesen *Events* bleibt das System unverändert und deshalb genügt es, zur Simulation nur bei Ereignissen die Systemzustände neu zu berechnen (Ereignissteuerung). Diese einfache Modellierung reicht aus, um das dynamische Verhalten von Digitalschaltungen mit Tausenden von Gattern und Registern hinreichend genau zu beschreiben und zu simulieren und auf diese Weise Designfehler vor der Fabrikation der Schaltungen fest zu stellen.

Leider sind aktuelle technische Systeme weder rein analog noch rein digital aufgebaut und beinhalten häufig nicht nur elektrische oder elektronische Komponenten. Deshalb gibt es schon länger Versuche, die digitale und die analoge Simulationsdomäne zusammenzuführen und die Simulationsmöglichkeiten auf gemischt elektrische und nicht-elektrische Systeme auszudehnen (Abschnitt

2.4). Dies ist schwierig, weil hier die Zeit und die Systemvariablen in unterschiedlicher Form behandelt werden. Digital bedeutet diskrete Zustände, die sich ereignisgesteuert ändern. Analog heißt zeit- und wertekontinuierliche Systeme, die über DGLs beschrieben werden und für die physikalische Erhaltungssätze gelten. Erst mit der Einführung der Mixed-Signal-HDLs (VHDL-AMS als Standard IEEE1076.1) sind diese Schwierigkeiten gelöst, denn für genormte Sprachen können universelle Simulationsprogramme entwickelt und von EDA-Herstellern zur Verfügung gestellt werden.
Im dritten Kapitel werden die mathematischen und algorithmischen Methoden vorgestellt, die zur Simulation von technischen Systemen notwendig sind. Wir beginnen mit der analogen Simulation und hier mit der Berechnung von Zweipol-Netzwerken (Abschnitt 3.1). Bei linearen Netzwerkelementen können wir die Gleichstromanalyse und die Wechselanalyse im Kleinsignalbereich im Prinzip per Hand durchführen. Praktischer ist es numerische Methoden einzusetzen. Wir verwenden dafür das Programm MATLAB als universelles Numerik-Werkzeug. Nach linearen Systemen beschäftigen wir uns mit nicht linearen Schaltungen und berechnen zunächst Arbeitspunkte in solchen Systemen, führen dann Wechselstrom-Kleinsignalanalysen durch und schließlich transiente Großsignalbetrachtungen. Da exakte Lösungen meist nicht gefunden werden können, kommen zur Lösung nur numerische Verfahren in Frage. Die Algorithmen, die SPICE verwendet, erläutern wir anhand von MATLAB-Programmen. Die numerische Komplexität von analogen Simulationen beschränkt die Zahl der Bauelemente, die in einer Schaltung simuliert werden können, auf wenige Hundert. Moderne digitale Systeme bestehen aber aus Millionen von Transistoren. Deshalb werden hier weniger genaue und abstraktere Modelle als Designprimitive verwendet, die als logische Gatter aus mehreren elektrischen Bauelementen bestehen und deren Verhalten nur mit diskreten logischen Zuständen charakterisiert werden, die sich ereignisgesteuert nur bei diskreten Zeitpunkten ändern können. Für die Berechnung der Dynamik in solchen Systemen werden bestimmte Verfahren (Transaktionslisten) benutzt, die in Abschnitt 3.2 vorgestellt werden. Die Integration von digitalen und analogen Modellteilen stellt Simulatoren wieder vor spezifische neue Probleme, mit denen sich Abschnitt 3.3 und 3.4 beschäftigen.
In den Kapiteln 4, 5 und 6 werden Anwendungsbeispiele für die Simulatoren SPICE, VHDL-AMS und MATLAB vorgestellt, die typische Problemfelder behandeln, in denen das jeweilige Simulationstool benutzt wird. In Kapitel 4 behandeln wir die Simulation eines CMOS-Differenzverstärkers mit SPICE, um die Schaltungsauslegung zu testen und mit den analytisch berechneten Kennwerten zu vergleichen. Hier kommen alle Analyseformen zum Einsatz und in den Übungen werden einfache Schaltungsbeispiele vorgegeben, sie selbständig im SPICE-Netzlistenformat eingegeben werden und dann mit SystemVision® simuliert werden können.
In Kapitel 5 wird die Mixed-Signal-Analyse eines Systems aus der Übertra-

gungstechnik (FSK-Modulator) mit Hilfe der Sprache VHDL-AMS vorgestellt. Dieses Beispiel zeigt, wie einfach und systematisch gemischt analog-digitale Systeme mit VHDL-AMS oder einer anderen Mixed-Signal-HDL modelliert werden können. Die Modellierung erfolgt auf hohem Abstraktionsniveau und ist verhaltensorientiert. Das Modell besteht aus Modulator, Demodulator und einem digitalen Signalgenerator, die in einer Testbench eingebettet sind.
In Kapitel 6 betrachten wir als MATLAB-Anwendung die Lösung der Bewegungsgleichung des mechanischen Pendels, die zuerst mit Reibung und periodischer Antriebskraft formuliert wird. Im zweiten Teil lassen wir die Antriebskraft weg und fügen eine Wand durch den Ursprung unseres Koordinatensystems ein, an der das Pendel elastisch reflektiert wird. Dies führt zu Unstetigkeiten in der Bewegung des Pendels, die bei der Lösung der Differentialgleichung in geeigneter Weise behandelt werden müssen. In MATLAB werden die Unstetigkeiten als *Events* bezeichnet, während in VHDL-AMS der Begriff BREAK verwendet wird. Solche Unstetigkeiten sind typisch für technische Systeme und geeignete Simulatoren müssen über geeignete Verfahren verfügen, mit denen über Unstetigkeiten bei der Lösung von DGLs hinweg integriert werden kann. Das MATLAB-Beispiel zeigt deutlich, dass die allgemeine Ausrichtung des Tools auf Kosten der Effizienz in der Anwendung geht, denn der Nutzer muss die komplette Problemdefinition eingeben und kann nicht auf Bibliotheken mit vordefinierten Grundmodellen zurückgreifen, wie das bei SPICE oder VHDL-AMS möglich ist.
Den Abschluss des Lehrbriefs bildet eine zusammenfassende Schlussbetrachtung, in der noch mal die grundlegenden Prinzipien der Simulationstechnik herausgestellt werden.

Inhaltsverzeichnis

1. Theorie, Modell und Simulation

In der Technik werden die zu entwickelnden Systeme immer komplexer und höher integriert, wobei die zur Verfügung stehende Entwicklungszeit immer kürzer wird. Deshalb ist es fast zwingend auf Anhieb voll funktionsfähige Systeme zur Verfügung zu haben, um nicht durch Nacharbeiten wertvolle Zeit zur verlieren. Gleichzeitig werden Entwicklungsaufgaben auf verschiedene Teams im Unternehmen verteilt, die im Zuge der Globalisierung sogar in verschiedenen Kontinenten arbeiten können. Die Entwickler werden also mit den grundsätzlichen Problemen eines interdisziplinären technischen Systems konfrontiert und müssen dieses unter verschärften kommerziellen Randbedingungen entwickeln. Dies ist nur möglich, wenn moderne Entwicklungsmethoden eingesetzt werden, die ganz wesentlich auf Computersimulationen basieren.
Das primäre Ziel der Simulation ist das Auffinden aller Design-Fehler während des Entwurfsablaufs durch Vorhersage des dynamischen Verhaltens unter allen möglichen Betriebsbedingungen, die für das spätere technische System zu erwarten sind. Weiterhin können mit Simulationen verschiedenen Design-Alternativen miteinander verglichen werden.
Bevor simuliert werden kann, muss das zu entwickelnde System zuerst in Form eines Modells definiert und in simulierbarer Form also computergängig bereitgestellt werden. In der Regel werden Simulationsprogramme heute nicht mehr selbst von Entwicklern geschrieben, sondern es werden Standard-Simulations-Werkzeuge eingesetzt, für die bei der Modelleingabe bestimmte Syntax-Vorschriften einzuhalten sind. Modelle können nicht nur per ASCII-Code als Text eingegeben werden. Die meisten Simulatoren unterstützen auch die grafische Eingabe in Form von Schaltplänen, Blockschaltbildern oder Zustandsgraphen. In diesem Lehrbrief werden wir Modelle mit MATLAB® und SystemVision® simulieren, wobei wir uns aber im wesentlichen auf die Texteingabe konzentrieren, damit die generischen Eigenschaften des Modellierungs- und Simulationsvorgangs deutlich sichtbar werden..
Für jede Simulation wird ein mit dem Computer auswertbares Modell für das zu untersuchende System benötigt. Um so ein Modell aufzustellen, brauchen wir eine theoretische Vorstellung, aus der sich ein Satz von mathematischen Gleichungen entwickeln lässt, mit denen sich das dynamische Verhalten der **Systemvariablen** berechnen lässt. Zur Modelleingabe in mathematischer Form wird in der Regel ein bestimmtes Eingabeformat verwendet, das grafisch oder textbasiert sein kann. Theorien sind allgemeingültig, folglich gelten die mathematischen Gleichungen eines Modells nicht nur für bestimmtes System, sondern für eine Mannigfaltigkeit von technisch äquivalenten Systemen. Die Anpassung an eine ganz bestimmte Systemausprägung, an die für ein gegebenes System gewählte Technologie, sowie die Betriebsbedingungen erfolgt in der Regel über **Parameter**.

1.1 Theorie

Eine Theorie ist eine formalisierte Beschreibung von natürlichen Phänomenen, Abläufen in Wirtschaft und Gesellschaft oder in technischen Systemen. In den Naturwissenschaften insbesondere in der Physik werden Modelle aus allgemein gültigen Prinzipien abgeleitet. Physikalische Theorien beschreiben deshalb nicht nur bekannte Vorgänge, sondern besitzen auch Vorhersagekraft. So hat bspw. der Physiker W. Pauli aufgrund von theoretischen Überlegungen die Existenz des Neutrinos vorhersagen können, dessen experimenteller Nachweis erst vierzig Jahre später gelungen ist. In der uns hier stärker interessierenden Technik geht es nicht so sehr um neue Erkenntnisse, sondern um praktikable möglichst einfache aber gleichzeitig genaue, modellhafte Beschreibungen technischer Systeme. Einfachheit und Genauigkeit sind in der Regel konkurrierende Kriterien, zwischen denen man bei der Modellentwicklung abwägen muss. Die Beschreibungen müssen nicht notwendigerweise aus grundlegenden Prinzipien abgeleitet werden. Das Modell eines Transistors, dessen Funktion unmittelbar mit den Eigenschaften dotierter idealer Halbleiterkristalle, den quantenmechanischen Eigenschaften der Elektronen als Fermionen (Bändertheorie) und der Wirkung elektrodynamischer Felder auf die Bandstruktur zusammenhängen, kann zwar unter Berücksichtigung all dieser allgemeinen Prinzipien aufgestellt werden, das Modell ist dann aber nicht mehr auswertbar, d.h. das implizierte Verhalten kann nicht berechnet werden. Es ist viel praktischer, den Transistor als spannungsgesteuerte Stromquelle aufzufassen und deren Eigenschaften durch Anpassung an gemessene Kennlinien realer Transistoren festzulegen. Wir erkennen, dass die Trennung zwischen Modell und Theorie unscharf ist.

1.2 Modelle

Modelle werden in der Regel auf Basis einer allgemeinen Theorie abgeleitet und müssen dann für jedes System, das modelliert werden soll, konkretisiert werden. Wenn wir also den Transistor als spannungsgesteuerte Stromquelle modellieren, gehen wir davon aus, dass die Gesetze der klassischen Elektrodynamik gelten, wie etwa das Ohmsche Gesetz, die Kirchhoff-Gesetze etc. Das Modell von Cob und Sah, das den MOS-Transistor in eine Stromquelle abbildet, vereinfacht die Problematik noch weiter:

- Der leitfähige Kanal zwischen Source und Drain des Transistors bildet sich erst oberhalb einer Einsatzspannung aus, ist aber nur infinitesimal dick.

- Der Transistor liefert im Sättigungsbereich einen konstanten Strom.

Das Modell ist dann recht einfach, je nach Betriebszustand wird der Strom von zwei Gleichungen geliefert, die in quadratischer Form von der Drain-Source- und der Gate-Source-Spannung abhängen. Wenn man aber versucht, auf Basis dieses einfachen Modells den zeitlichen Verlauf des Ladestroms für einen Kondensator am Ausgang (Drain) des Transistors in Abhängigkeit von der Gate-Source-Spannung zu berechnen, dann führt dies auf eine nichtlineare Differentialgleichung, die nur in Spezialfällen (Eingangsrampe oder Spannungssprung am Gate des Transistors) gelöst werden kann. In der Regel haben die meisten nicht linearen Differentialgleichungen und fast alle partiellen Differentialgleichung keine analytische Lösung, die die zeitliche Entwicklung der Systemgrößen als Formel angibt. Für die Ladekurve gilt dies, wenn der Ausgang eines anderen Transistors das Gate des Ladetransistors ansteuert. Ist eine mathematische Lösung nicht möglich, dann muss man sich mit Computersimulationen begnügen. Hier werden die (Differential-)Gleichungen des Modells mit numerischen Verfahren ausgewertet und die Ergebnisse in Listenform oder direkt als Kurvenzüge grafisch ausgegeben.
Damit ist das Problem aber nicht völlig gelöst, denn anstelle einer vollständigen Lösung liegen nur Ergebnisse für die Fälle vor, die bei der Simulation Berücksichtigung fanden. Es kann durchaus sein, dass die wichtigsten Fälle bei der Simulation nicht erfasst worden sind. Die Funktion des entworfenen Systems kann mit Simulationen also nur mit Einschränkungen sichergestellt werden, genau wie in der Mathematik die Aufzählung von Beispielen nicht den mathematischen Beweis einer Vermutung ersetzen kann.

1.2.1 Klassifikation von Modellen

Generell werden Modelle mit mathematischen Formeln beschrieben, mit denen das Verhalten des Systems berechnet werden kann. Es wird zwischen **statischen** und **dynamischen** Modellen unterschieden. Bei statischen Modellen geht es um die Arbeitspunkte des Systems. Dies sind stationäre also zeitunabhängige Zustände, die ein System annehmen kann. Ein Beispiel sind die Arbeitspunkte einer elektronischen Schaltung, die über Strom- und Spannungswerte in den Knoten und längs der Zweige des elektrischen Netzwerkgrafen abgegeben werden können. Dynamische Modelle beschreiben die zeitliche Entwicklung eines Systems von einem Zustand in den nächsten. Auch die Systemsimulationen in der **Frequenzdomäne** gehören in den Bereich der dynamischen Analyse.
Werden die Systemzustände mit **kontinuierlichen** Größen gekennzeichnet (also mit reellen Zahlen), dann handelt es sich um ein **analoges** Modell, in dem

sich die Zustände kontinuierlich in Zeit und Amplitude ändern. Sind die Systemzustände **diskret**, lassen sich also mit einem endlichen Satz von Werten angeben, dann liegt ein digitales System vor. Beispiele sind digitale Schaltungen, in denen die einzelnen Knoten nur die beiden logischen Zustände (0 und 1 für wahr und falsch) annehmen können. Die zeitliche Entwicklung bei digitalen Systemen kann ebenfalls diskretisiert werden. Für die Dynamik zählen nur einzelne Zeitschritte. Der Systemzustand wird immer dann neu berechnet, wenn ein Ereignis an einer Stelle im System stattgefunden hat, also z.B. sich eine Eingangsgröße geändert hat. Anders als in analogen Systemen, ist das Systemverhalten bei dieser **Ereignissteuerung** zwischen Ereignissen nicht relevant, weil die Systemgrößen dann ihre diskreten Werte beibehalten.
Ein **digitales** System ist nach dieser Definition also ein System, das dadurch gekennzeichnet ist, dass es werte-quantisierte Informationen in quantisierten Zeitschritten verarbeitet oder speichert. Im Gegensatz dazu verarbeitet ein **analoges** System kontinuierliche Größen, die auch in kontinuierlicher Form von der Zeit abhängen. Die typische Beschreibungsweise für solche Systeme sind gewöhnliche Differentialgleichungen oder algebraische Gleichungen, wie das Ohmsche Gesetz. Mit diesen beiden Definitionen kann ein gemischt digital-analoges System (*Mixed Signal-System*) als ein System aufgefasst werden, in dem die Information zum Teil digital und teilweise analog verarbeitet wird. An den Schnittstellen werden die Signale digital-analog bzw. analog-digital gewandelt (Bild 1.1).
Ein typisches Beispiel für solche gemischten Systeme sind Prozesskontrollsysteme, die den Zustand des Prozesses mit Sensoren analog erfassen, die Sensordaten in digitale Signale wandeln, dann mittels eines digitalen Rechners die Sensordaten auswerten, auf den Zustand des Prozesses schließen und über Aktoren bei Bedarf in den Prozess eingreifen. Die Aktoren werden wieder mit analogen Größen angesteuert, was eine weitere Wandlung zwischen digitalen und analogen Signalen erforderlich macht.
Elektronische und elektrische Systeme interagieren stets mit Systemen aus anderen physikalischen Bereichen, mit mechanischen Systemen, optischen Systemen, chemischen Systemen usw. Auch in diesen nichtelektrischen Domänen werden die Systemzustände wie in der Analogelektronik mit zeitlich variierenden kontinuierlichen Größen beschrieben, deshalb können wir auch diese Systeme als analoge Systeme betrachten und die Modellierungsmethodik aus der Analogelektronik auf diese anderen Domänen erweitern, oder genauso auch die Simulationsmethoden aus der Mechanik, Chemie usw. auf die Elektrotechnik übertragen.
Ein weiteres wichtiges Unterscheidungsmerkmal für Modelle ist die Modellierungsstrategie, die verhaltensorientiert oder physikalisch sein kann. Bei **Verhaltensmodellen** interessiert lediglich eine möglichst einfache aber auch möglichst genaue formelhafte Nachbildung des Systemverhaltens. Bei der **physikalischen Modellierung** werden bestimmte grundlegende physikalische Ge-

setzmäßigkeiten, wie die Erhaltung von Energie, Ladung und Masse in die Modelle eingearbeitet. Ein Netzwerksimulator für elektronische Schaltungen verwendet physikalische Bauelementmodelle. Deshalb sind die Kirchhoffschen Gesetze hier während der Auswertung des Schaltungsmodells stets erfüllt. Ein digitaler Logiksimulator, in dem nur logische Zustände an den einzelnen Knoten betrachten werden und keine physikalischen Größen, berücksichtigt keinerlei Erhaltungssätze und gibt nur das Verhalten des Systems im Hinblick auf die zeitliche Entwicklung der logischen Zustände wieder. Dies vereinfacht die Simulation zwar erheblich, erlaubt es aber nicht, physikalische Effekte im Zeitverhalten, wie etwa das Übersprechen zwischen Signalleitungen (EMV) zu erfassen.

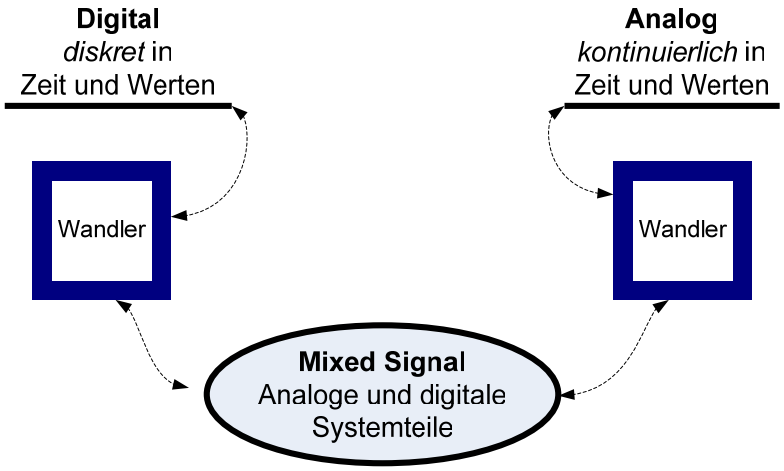

Bild 1.1 *Analoge und digitale Teilsysteme bilden ein Mixed-Signal-System*

1.2.2 Abstraktionsebenen und Sichtweisen

Je nachdem wie ein Modell das zu analysierende System beschreibt, unterscheidet man verschiedene *Sichtweisen*:

- **Funktionale Sicht**: Hier werden nur die Funktionen des Systems als Operatoren, die auf die Systemzustandsvariablen wirken, berücksichtigt (Verhaltensbeschreibung in physikalischer oder reiner Verhaltensorientierung). Dies ist im Kontext eines technischen Entwicklungsprojektes die abstrakteste Form der Beschreibung, da das Modell nicht auf die technische Umsetzung der Systemfunktionen eingeht.
- **Strukturelle Sicht**: Hier wird bereits erfasst, wie das System aus inter-

agierenden Untersystemen aufgebaut werden kann oder worden ist. Die Subsysteme können wieder strukturiert sein, aber auf der untersten Ebene müssen die *primitiven* Modellelemente auf funktionalem Niveau beschrieben sein.

- **Geometrische Sicht**: Auf dieser Ebene wird die physikalische Dimensionierung der einzelnen Bauelemente auf der primitiven Modellebene mit eingegeben, um das Funktionieren des Systems in allen Details so technologienah wie möglich zu überprüfen.

Jede der drei Sichtweisen unterstützt die Designentscheidungen während der Systementwicklung auf jeweils verschiedenem Abstraktionsniveau (Bild 1.2). Man beginnt in der Regel mit der funktionalen Sicht und arbeitet sich bis zur Geometriesicht in jedem Strang immer weiter an die endgültige technische Systemauslegung heran, wobei die Simulationsergebnisse in den verschiedenen Sichten auf jeder Ebene miteinander verglichen werden. Jeder Modellstand wird simuliert und die abstraktere bereits verifizierte Version der nächst konkreteren Modellvariante gegenübergestellt. Diese Verzahnung im Vorgehen soll eine fehlerfreie Umsetzung der Designvorgaben sicherstellen.

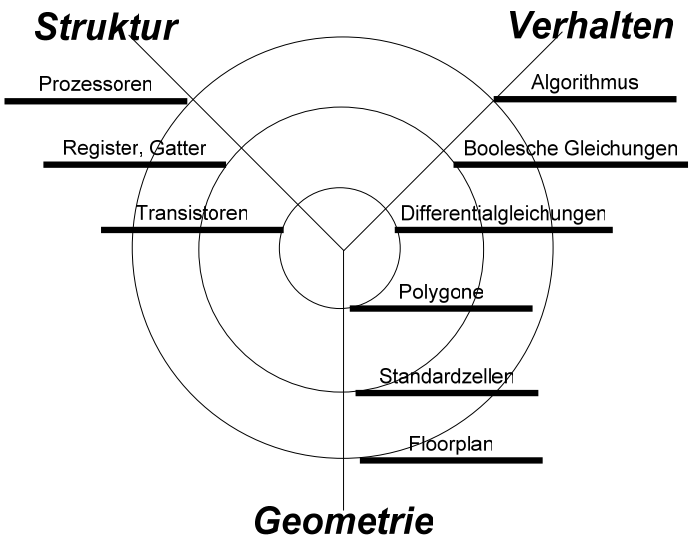

Bild 1.2 *Sichtweisen und Abstraktionsebenen am Beispiel eines digitalen Systems. Die konzentrischen Ringe kennzeichnen die verschiedenen Abstraktionsebenen. Mit dem Radius nimmt das Abstraktionsniveau zu. Die drei radialen Achsen in Y-Form entsprechen den verschiedenen Sichtweisen. Diese Auftragung heißt deshalb auch Gajskysches Y-Diagramm.*

1.2.3 Designstrategien

Je stärker die Modellbildung von der physikalischen Implementierung des zu entwickelnden Systems abstrahiert, desto einfacher wird die Modellbeschrei-

bung und desto schneller liegen die Simulationsergebnisse vor. Auf der hohen Abstraktionsebene eines rein funktionalen Modells werden aber noch viele Details des Systemverhaltens nicht erfasst, die für die konkrete Funktion wichtig sein können. Deshalb müssen die Modelle im Zuge des Entwicklungsablaufs schrittweise verfeinert werden, damit sie sich immer mehr dem erwarteten physikalischen Verhalten des Systems annähern.
Bei dieser zergliedernden Entwurfsstrategie (**Top Down**) wird ein Design also auf hohem Abstraktionsniveau begonnen und mit einer Verhaltensbeschreibung spezifiziert. Dann wird der Entwurf schrittweise verfeinert und die Struktur des Systems mit einbezogen, um zu einer physikalischen Realisierung aus verbundenen miteinander wechselwirkenden Funktionsblöcken zu kommen. Bei diesen Blöcken kann es sich um diskrete Schaltkreise oder integrierte Schaltkreise, ASICs, Prozessoren, Softwarekomponenten, Einzelbauelemente, elektrische Maschinen, Sensoren oder Aktoren etc. handeln. Dieser Zugang erlaubt es, die Technologieauswahl und damit zusammenhängende geometrische Auslegung der einzelnen physikalischen Komponenten erst ganz am Ende des Entwurfszyklus zu treffen.
Deshalb betrifft ein eventueller späterer Technologiewechsel nicht mehr die ersten Entwurfsetappen, wie dies bei der umgekehrten Designstrategie, beim „**Bottom-up**“-Verfahren (aufbauender Entwurfsstil), der Fall ist. Dort werden zuerst alle Grundbausteine technologiebezogen entwickelt und vollständig geometrisch ausgelegt, bevor man mit dem Zusammensetzen beginnt (mit Fehlerrisiken beim Verbinden der Bausteine). Diesen Stil wendet man immer dann an, wenn man sehr schwierig zu erfüllende technische Anforderungen umzusetzen hat, die an der Grenze der Leistungsfähigkeit der eingesetzten Technologie liegen. Hier ist erst zu überprüfen, ob überhaupt ausreichend leistungsfähige primitive Komponenten verfügbar sind, aus denen dann das komplexe Gesamtsystem zusammengesetzt werden kann. Ein Beispiel wäre ein Mixed-Signal-Schaltkreis bei dem in einem Chip aufwendige Analogfunktionen (z.B. hoch auflösende Wandler) mit einer komplexen digitalen Umgebung zusammenarbeiten sollen.
In der Praxis benutzt man häufig die zergliedernde und die aufbauende Technik nebeneinander (Meet-In-The-Middle-Strategie). Beim Design eines ASICs (anwendungsspezifischer integrierter Schaltkreis) arbeitet man häufig z.B. mit vorcharakterisierten Zellen, die mit einem Top-Down-Entwurfskonzept zusammengesetzt werden, wobei die Zellen selbst mit einem Bottom-up-Verfahren aus Einzeltransistoren erstellt wurden.
In komplexeren Modellen können auch Objekte verwendet werden, die selbst auf verschiedenen Abstraktionsebenen beschrieben sind. So können Systeme simuliert werden, bei denen bestimmte Teile auf der Verhaltensebene definiert sind (zum Beispiel als mathematische Gleichung), während andere Teile durch strukturelle Verknüpfungen von Komponenten in der gewählten Technologie

modelliert werden, um den dort relevanten physikalischen Phänomenen Rechnung zu tragen. Man spricht in diesem Zusammenhang von **Multi-Abstraktion**.
Mit Hilfe der Multiabstraktion kann man die Rechenzeit für Simulationen von aus mehreren Teilen zusammengesetzten Systemen verkürzen. Je mehr Objekte in einem Design auf einem hohen Abstraktionsniveau beschrieben sind, umso schneller läuft die Systemsimulation ab. Durch Verwendung von möglichst hohen Abstraktionsebenen für alle die Teilmodelle, die nicht auf dem niedrigsten Beschreibungsniveau verifiziert werden sollen, kann man ein System komplett Teilmodell für Teilmodell validieren, ohne dass man es in seiner Gesamtheit auf der niedrigsten, geometrischen Abstraktionsebene simulieren müsste.

1.3 Modelleingabe und Spezifikation

Es gibt verschiedene Methoden, Modelle in den Computer einzugeben. Am einfachsten ist es für den Anwender, grafische Stromlaufpläne aus vernetzten elektrischen Komponenten über Schaltplaneditoren oder entsprechend auf andere physikalische Domänen angepasste Tools einzugeben. Diese grafischen Beschreibungen werden dann in der so genannten Elaboration des Modells automatisch in Text, also eine bestimmte Computersprache übersetzt. Um diesen Zwischenschritt zu vermeiden und um genormte Eingabeformate benutzen zu können, die nicht der Kontrolle des CAE-Werkzeugherstellers unterliegen, der den Schaltplaneditor geliefert hat, haben sich in letzter Zeit standardisierte Beschreibungssprachen bei der Modelleingabe durchgesetzt, die bei Bedarf auch grafisch unterstützt verwendet werden können. Die häufig in Form von Netzlisten vorliegenden textuellen Beschreibungen werden dann vor der eigentlichen Simulation weiter ausgearbeitet („elaboriert“) und in Gleichungssysteme übersetzt, die den Lösungsalgorithmen der jeweils verwendeten Simulationswerkzeuge zugeführt werden. Dann werden die Modelle ausgewertet und die Ergebnisse entweder als Text, oder aber in der Regel als grafische Wellenzüge ausgegeben (Bild 1.3). Die grafischen Daten können dann weiter prozessiert werden (Fourier-Transformation, statistische Analyse, etc.).
Sinn und Zweck der Simulation ist es, das Modellverhalten auf jeder Abstraktionsebene und in jeder Sichtweise mit den Designvorgaben aus der Spezifikation abzugleichen.

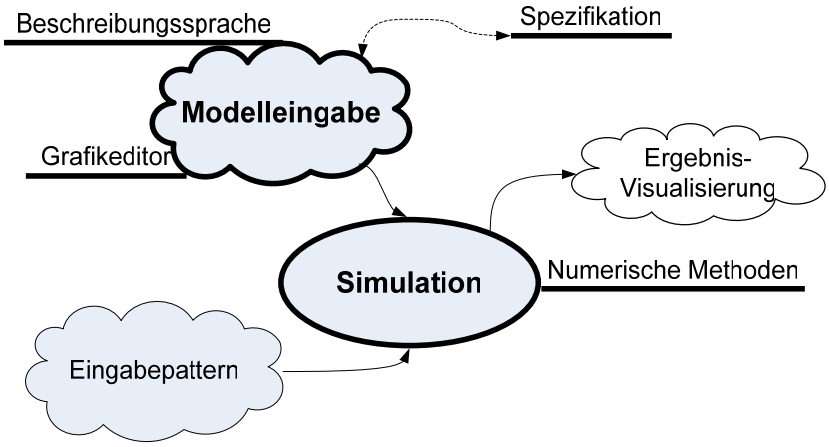

Bild 1.3 *Modelleingabe mit den Medien Grafik oder Sprache und simulative Auswertung mit geeigneten Ansteuersignalen (Eingabepattern). Die Simulationsergebnisse werden mit numerischen Methoden gewonnen und grafisch als Kurven ausgegeben.*

Das Erstellen einer vollständigen und genauen Spezifikation eines komplexen Systems, in der die Vorgaben enthalten sind, die bei der Simulation abgeprüft werden sollen, ist ebenfalls eine sehr schwierige Aufgabe. Die Funktionen und die zugehörigen Randbedingungen werden bisher meist in natürlicher Sprache in einem Dokument beschrieben, das sehr umfangreich sein kann. Diese Vorgehensweise ist anfällig für Ungenauigkeiten, Mehrdeutigkeiten, Auslassungen und sogar Widersprüche. Es ist vorab keine Voraussage über die Leistungsfähigkeit oder die Kosten des neuen Systems möglich, lediglich aufgrund der Erfahrung des Projektleiters und des Systemarchitekten sind Abschätzungen möglich.
Die aktuelle Richtung ist daher, formale Beschreibungssprachen zur Spezifikation einzusetzen, die eine teilweise oder sogar vollständige Verhaltenssimulation ermöglichen. Man kann so die Funktion unabhängig von der physikalischen Implementierung ausdrücken. Diese simulierbare Spezifikation wird dann immer als Referenz verwendet, wenn man von einem Entwurfsschritt zum nächsten geht. So können sich alle Zwischenergebnisse beim Design bis einschließlich der Fertigungsvorlagen zu Vergleichs- und Prüfzwecken auf die Ergebnisse der funktionalen Simulation der Spezifikation beziehen. Diese simulierbaren Modelle, die die Systemvorgaben abbilden, heißen **Testbenches**.

1.4 Simulation von Modellen

Simulation ist die computergestützte Auswertung der Modelle die bei der Systemspezifikation in eine Netzliste eingebettet wurden. Bild 1.4 zeigt das Zusammenspiel zwischen den verschiedenen Ebenen. Wie schon erwähnt, wird das eingegebene Modell elaboriert, um es für die Computerauswertung aufzubereiten. Vorher prüft ein Syntax-Checker, ob das Modell sprachlich oder grafisch ohne Fehler eingegeben wurde. Wenn standardisierte Modelle, wie die Bauelementmodelle von SPICE verwendet werden, dann müssen Technologieparameter als SPICE-Modellparameter bereitgestellt werden, die diese Modelle an die zu verwendende Technologie anpassen. Die numerische Genauigkeit kann bei den Simulatoren ebenfalls über bestimmte Parameter gesteuert werden. Unter Umständen sind bestimmte Anfangswerte zu setzen, von denen ab das Modellverhalten zeitlich berechnet wird.

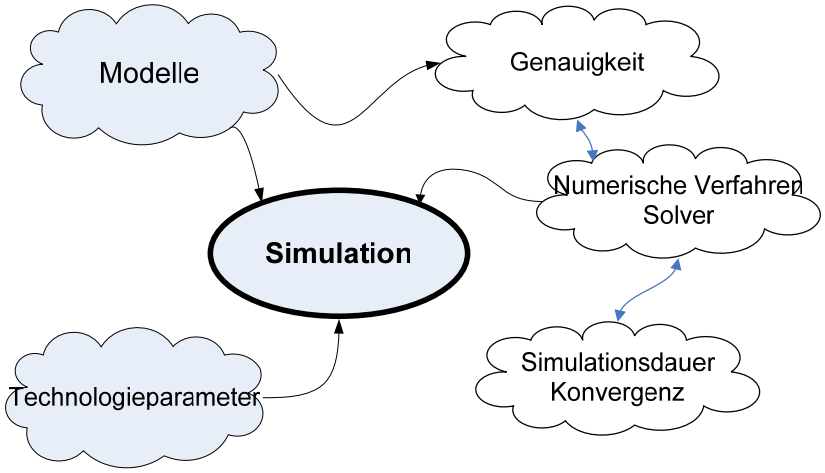

Bild 1.4 *Aspekte, die bei der Simulation eine Rolle spielen*

Eine besondere Rolle spielen bei den heute zu entwickelnden komplexen Systemen die oben erwähnten Testbenches. In diese Testbenches wird das zu simulierende Systemmodell eingesetzt und dann zusammen mit der zugehörigen Testumgebung simuliert (Bild 1.5). Man kann die Testbench-Modelle so gestalten, dass sie nicht nur die relevanten Stimuli bereit stellen, anhand derer die Systemeigenschaften abgeprüft werden, sondern kann auch während der Testbenchsimulation die Simulationsergebnisse automatisch überprüfen lassen. Dann ist es nicht nötig, komplexe Kurvenzüge optisch zu inspizieren. Die Testbenchsimulation gibt die Pass/Fail-Information direkt aus.

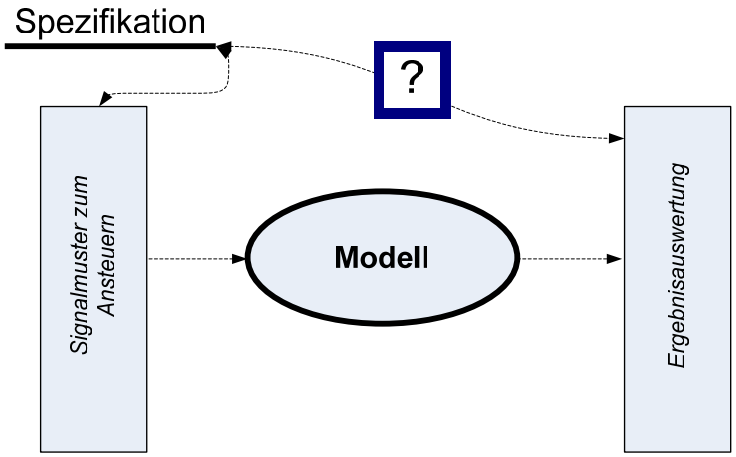

Bild 1.5 *Testbenchsimulation. Die Spezifikation wird in eine simulierbare Version übersetzt, die zusammen mit dem zu untersuchenden Modell als strukturelles Modell verschaltet wird. Es gibt Testbenches in verschieden tiefer Detaillierung. Die höchste Ausbaustufe ist die gezeigte, in der nicht nur Eingangsmuster von der Testbench generiert werden, was die Programmierung von Signalquellen im Modell überflüssig macht, sondern auch die Simulationsresultate mit den Spezifikationsvorgaben abgleicht.*

1.4.1 Simulationsmodelle des Simulators SPICE

Ein SPICE-Modell ist eine Textbeschreibung einer elektronischen Komponente, die in einer bestimmten Syntax geschrieben ist und mit SPICE-Simulatoren im Rahmen der Schaltkreissimulation ausgewertet werden kann, um das Verhalten des Bauteils unter verschiedenen Bedingungen zu untersuchen. SPICE-Modelle können aus einfachen einzeiligen Anweisungen für passive Bauelemente (Widerstand) bestehen, oder aber aufwendige Beschreibungen komplexerer Blöcke sein, wie z.B. Operationsverstärker, die hunderte von Programmzeilen umfassen. Fast alle Halbleiterhersteller stellen SPICE-Modelle für ihre Produkte auf ihrer Website zur Verfügung, um die Entwicklung von elektronischen Schaltungen mit ihren Bauteilen zu unterstützen. Im Folgenden ist auszugsweise das SPICE-Modell eines MAXIM-Analogschalters als *SPICE-Netzliste* wiedergegeben, um die Syntax und die Form von SPICE-Modellen zu veranschaulichen:

```
* MAX4601 MACROMODEL
* ------------------------------
* Revision 0, 01/2005
* ------------------------------
* MAX4601 quad analog switch with low-on resistance and has 4
* normally closed(NC) switches. Switches operate from a single
* supply of 4.5V to 36V or from dual supplies of +-4.5V to
* +- 20V.
```

```
* ------------------------------
* Connections
*       1  = IN1
*       2  = COM1
……
……
*      15  = COM2
*      16  = IN2
************************************
.SUBCKT MAX4601 1 2 3 4 5 6 7 8 9 10
+               11 12 13 14 15 16
X1 3 2 1 14 15 16 13 4 12 5 MAX4601DUAL
X2 6 7 8 11 10 9 13 4 12 5 MAX4601DUAL
.ENDS
************************************

*****************
.SUBCKT MAX4601DUAL 1 2 3 4 5 6 7 8 9 10
S1 1 2 3 10 SMOD2
S2 4 5 6 10 SMOD2
*ANALOG SIGNAL RANGES
DNO11 1 7 DX
……
……

DCOM22 8 5 DX
*SUPPLY CURRENTS
IVDD 7 10 0.5N
*SWITCH INPUT LEAKAGE CURRENT
ILENC1 1 10 0.01N
FNC1 1 10 POLY(1) VSCALED2 0 0.19N 0 0 0
ILENC2 4 10 0.01N
FNC2 4 10 POLY(1) VSCALED2 0 0.19N 0 0 0
*INPUT LEAKAGE CURRENT
ILIN1 3 10 1N
……
……
*****************
*MODELS USED
.MODEL SMOD1 VSWITCH(RON=1.7 ROFF=0.5E6 VON=2.4 VOFF=0.8)
.MODEL SMOD2 VSWITCH(RON=1.7 ROFF=0.5E6 VON=0.8 VOFF=2.4)
.MODEL DX D(IS=1E-18 N=0.001)
.ENDS
****************
```

Auf die einzelnen Einträge in diesem und anderen SPICE-Modellen wird genauer in Abschnitt 2.3 eingegangen. SPICE-Modelle von Schaltungen setzen sich aus den Modellen der einzelnen Bauelemente zusammen, die über ihre Eingangs- und Ausgangsknotenbezeichnungen miteinander verbunden werden. So wird die Struktur der Schaltung vorgegeben. Statt einzelne Textmodelle zusammenzufügen, ist es in der Technik üblicher, Schaltpläne zu zeichnen, in denen die einzelnen Schaltelemente durch Symbole dargestellt werden, denen die textliche Beschreibung über Symboleigenschaften zugeordnet sind. Die

einzelnen grafischen Symbole werden dann mit Linien verbunden, die elektrische Leitungen darstellen. Vor der Simulation bei der Elaboration werden die grafisch eingegebenen Schaltpläne wieder in Netzlisten übersetzt und dann in mathematische Gleichungssysteme überführt, die dann mit numerischen Methoden auf dem Computer gelöst werden können.

1.4.2 Simulationsmodelle in VHDL-AMS

Diese Analog-Digitale-Beschreibungssprache wurde bereits im Lehrbrief *Systemspezifikation* vorgestellt. Hier können anders als in SPICE nicht nur strukturelle Modelle sondern auch Verhaltensbeschreibungen auf verschiedenen Abstraktionsebenen (rein abstraktes Verhalten bis hinunter zur Bauelementebene) vorgegeben werden. Diese Sprache wurde entwickelt, damit gleichzeitig analoges und digitales Verhalten innerhalb eines Modells beschrieben werden kann. SPICE unterstützt nur analoges Verhalten. Ältere Hardwarebeschreibungssprachen, wie das ebenfalls im Lehrbrief *Systemspezifikation* behandelte VHDL, können nur digitale Systeme modellieren. VHDL-AMS schafft den Übergang zwischen diesen grundsätzlich verschiedenen Simulationsdomänen.
Ein VHDL-AMS-Modell besteht aus zwei grundlegenden Teilen: der Spezifikation der **Entität** (kurz *die Entity*; Schlüsselwort `ENTITY`), die der äußeren Sicht des Models entspricht und der Spezifikation der **Architektur** der Entity (Schlüsselwort `ARCHITECTURE`), die das Verhalten als innere Sicht des Models enthält.
In der Entity können Bibliotheken (Schlüsselwort: `LIBRARY`) aufgerufen werden und Ports (`PORT`s) definiert werden. Über die Ports kann die Entity Informationen mit ihrer Umgebung austauschen. Es gibt zwei verschiedene Informationsklassen, die über die Ports transportiert werden, die Klasse `SIGNAL` für digitale und die Klasse `TERMINAL` für analoge Informationen.
Die Architektur beginnt mit einem Bereich für Deklarationen und legt dann die Funktion des Modells auf folgende Arten fest:

- über verbundene instanziierten **Komponenten** (d.h. durch Definition einer Netzliste von Objekten),
- durch **zeitlich nebenläufige** Anweisungen, die digitale Information verarbeiten und
- durch **simultane** Anweisungen, die analoge Werte ins Modell einsetzen.

Die Architektur kann insbesondere auch die Informationen verarbeiten, die über die Ports der Entität zur Verfügung gestellt werden.
Eine Entität ist nur dann simulierbar, wenn sie mit einer zugehörigen Architektur verknüpft ist. Man bezeichnet ein Modell, das sich kompilieren lässt, als

Entwurfseinheit (UC = *Design Unit*). Die Architektur einer Entität ist nur dann kompilierbar, wenn die Entität, auf die sie sich bezieht, schon kompiliert ist.
Der schematische Aufbau eines VHDL-AMS-Modells ist wie folgt:

- Spezifikation der Entität (`ENTITY`), bestehend aus
  - der Definition der generischen Parameter (`GENERIC`),
  - der Definition der Ports für mögliche Verbindungsstellen (`PORT`),
  - der Deklaration der Signale (`SIGNAL` in/out), die zeitlich diskrete Ereignisse transportieren,
  - der Deklaration von *Quantities* (`QUANTITY` in/out), als Größen, die sich kontinuierlich ändern können,
  - der Deklaration von Terminals (`TERMINAL`) mit denen Verbindungen geschaffen werden können, die physikalische Größen, unter Berücksichtigung von Erhaltungssätzen (z.B. Kirchhoff´sche Gesetze) beschreiben.
- Architektur der Entität, die den Bereich für Deklarationen enthält, und den Körper (*Body*) der Architektur, in dem
  - Komponenten eingesetzt werden können, um **hierarchische** Modelle zu definieren,
  - zeitlich **nebenläufige** Anweisungen, wie z.B. die PROCESS-Anweisung, eingesetzt werden können, die bei zeitlich **diskreten** Ereignissen (*Events*) verarbeitet werden und
  - **simultane** Anweisungen auftreten können, die zeitlich **kontinuierlich** bearbeitet werden.

Der folgende VHDL-AMS-Code veranschaulicht beispielhaft die generelle Struktur eines gemischt analog/digitalen Modells. Es ist hier noch nicht wichtig, diese Struktur vollständig zu verstehen.

```
LIBRARY Disciplines, IEEE; -- Bibliotheken
USE  Disciplines.electric_systems.ALL,  IEEE.MATH_REAL.ALL;  --
Packages
ENTITY example IS
            GENERIC (tplh : time := 4 ns);
            PORT (SIGNAL sig_ext : OUT real; TERMINAL vp, vm :
            electrical);
END ENTITY example;

ARCHITECTURE archi1 OF example IS
            TYPE list_ex IS (el1, el2, el2);
            CONSTANT cst1, k : REAL := 3.0;
            SIGNAL sig1, sig2, sig3 : INTEGER;
            QUANTITY vbias ACROSS ibias THROUGH vp TO vm;
            QUANTITY free_quant : REAL;
BEGIN
      u1 :  ENTITY model_externe(archi)
-- Instanziierung einer Komponente
            GENERIC MAP (100.0e3, 5.0);
            PORT MAP (vp, vm, sig1);
```

```
-- Simultane Anweisung
            free_quant == 3.0 * sinus(k * NOW);
            ibias == free_quant'DOT;
-- Nebenläufige Anweisung
            sig_ext <= '0' after tplh;
-- Prozess (nebenläufig)
            p1 : PROCESS
                  VARIABLE x : real := 5.5;
            BEGIN
                  WAIT ON sig3 UNTIL sig2 > 3 FOR 25 ms;
                  x := 2 * x;
                  sig_ext <= sig3 AFTER 1 ms;
            END PROCESS p1;
END ARCHITECTURE archi1;
```

1.4.3 Modelle in der universellen Simulationsumgebung MATLAB

MATLAB, ein Produkt der Firma *The MathWorks*, ist ein universelles Softwarepaket zur Lösung mathematischer Probleme und zur grafischen Darstellung der Ergebnisse. Im Gegensatz zu Computeralgebrasystemen, wie etwa MAPLE oder MATHEMATICA werden mit MATLAB mathematische Modelle nicht symbolisch, sondern primär numerisch, also zahlenmäßig gelöst. MATLAB ist für Berechnungen mit Matrizen ausgelegt, woher sich auch der Name ableitet: *MATrix LABoratory*.
Programmiert werden die zu lösende Probleme unter MATLAB in einer plattformunabhängigen eigenen MATLAB-Programmiersprache, die auf dem verwendeten Rechner interpretiert wird. Kleinere Programme können als so genannte Skripts oder Funktionen abgelegt werden, was das Erstellen von anwendungsorientierten Funktionspaketen (**Toolboxes**) erlaubt. Viele solcher Pakete sind auch kommerziell erhältlich. Durch die vereinfachte, mathematisch orientierte Syntax der MATLAB-Skriptsprache und die umfangreichen Funktionsbibliotheken für Statistik, Signalverarbeitung, Bildverarbeitung u.v.m. ist die Erstellung entsprechender Programme wesentlich einfacher möglich als in den üblichen Programmiersprachen wie z. B. in C.
MATLAB ist auch die Basis für die grafischen Erweiterungen *Simulink*, das zur zeitgesteuerten Simulation dient, und für *Stateflow*, das für die ereignisorientierte Simulation benutzt wird, sowie für zahlreiche anwendungs- und domänenspezifische Erweiterungen, wie die Control-Toolbox, die Signal Processing-Toolbox usw. In diesem Lehrbrief werden wir uns auf die grundlegenden Funktionen von MATLAB konzentrieren. Mit MATLAB können wir lineare Gleichungssysteme lösen, Matrizen verarbeiten (invertierten, diagonalisieren etc.) und vor allem Modellgleichungen für das zeitliche Verhalten von Systemen aufstellen, auswerten und sehr komfortabel grafisch darstellen.

Als Beispiel für eine elememtare MATLAB-Anwendung betrachten wir ein Modell aus der Biologie, das die Populationsdynamik zweier Fischsorten in einem Teich beschreibt, von denen die eine die Beute ist, die die andere Art jagt (Lotka-Volterra-Modell). Diese Autoren nahmen für die Population der Beutetiere x_2 ein lineares Wachstum mit der Geschwindigkeit c an. Dieses Wachstum wird durch die Anwesenheit der Räuber x_1, gewichtet mit einer Beutefangrate d, vermindert. Je mehr Raubfische es gibt, desto schwieriger wird es für den einzelnen, Beutefische zu fangen. Das führt für die Raubfische zu einer negativen Wachstumsrate $-a$, die durch eine entsprechend hohe Reproduktionsrate b der Beutefische ausgeglichen werden kann. Dies führt auf die folgenden Gleichungen:

$$\frac{dx_1}{dt} = x_1\left(-a + b \cdot x_2\right)$$
$$\frac{dx_2}{dt} = x_2\left(c - d \cdot x_1\right)$$

Dieses Gleichungssystem ist ein System aus gewöhnlichen Differentialgleichungen erster Ordnung, für die in MATLAB das Runge-Kutta-Verfahren als Lösungsalgorithmus („solver“) ODE45 implementiert ist. Das folgende MATLAB-Skript-File löst das System für die Anfangswerte (bei $t = 0$) 300 Räuber und 7000 Beutetiere für 30 Zeitschritte mit den Replikationsraten der Räuber von 0,08 und der Beute von 1,0 bei einer Fangquote von 0,002 und einer Räuberreproduktionsrate von 0,00001:

```
% Jäger Beute Modell
t0 = 0;
tende = 30;
x01 = 300;
x02 = 7000;
[t, x] = ode45('DGLSystem',[t0, tende],[x01,x02]);
plot (x(:,1),x(:,2))
xlabel ('Raubtiere')
ylabel ('Beutetiere')
```

t0 und *tende* geben den Zeitbereich vor, über den integriert werden soll. Der Aufruf `[t, x] = ode45('DGLSystem',[t0, tende],[x01,x02]);` startet den Lösungsalgorithmus. Dabei werden der interessierende Zeitbereich und die beiden Anfangswerte mit übergeben. Die mathematische Funktion `DGLSystem` ist das Gleichungssystem, das über eine separate MATLAB-Funktion bereitgestellt und das ebenfalls an den Lösungsalgorithmus übergeben wird:

```
%Funktion zur Definition des DGL-Systems
%Argumente t,x als Eingabe
%Funktionswerte als Ausgabe
function xPunkt = DglSystem(t,x)
xPunkt = zeros(2,1);
```

```
xPunkt = [x(1)*(-0.08 + 0.00001*x(2)); x(2)*(1.0-0.002*x(1))];
```

`zeros` erzeugt eine 2x1-Null-Matrix (Spaltenvektor mit 2 Komponenten). Die beiden Differentialgleichungen für x_1 und x_2 sind als Komponenten des Spaltenvektors definiert.
Wenn das MATLAB-Skript ausgeführt wird, dann erhalten wir die folgende Phasendarstellung, die die Zahl der Beutefische als Funktion der Räuber darstellt (Bild 1.6).

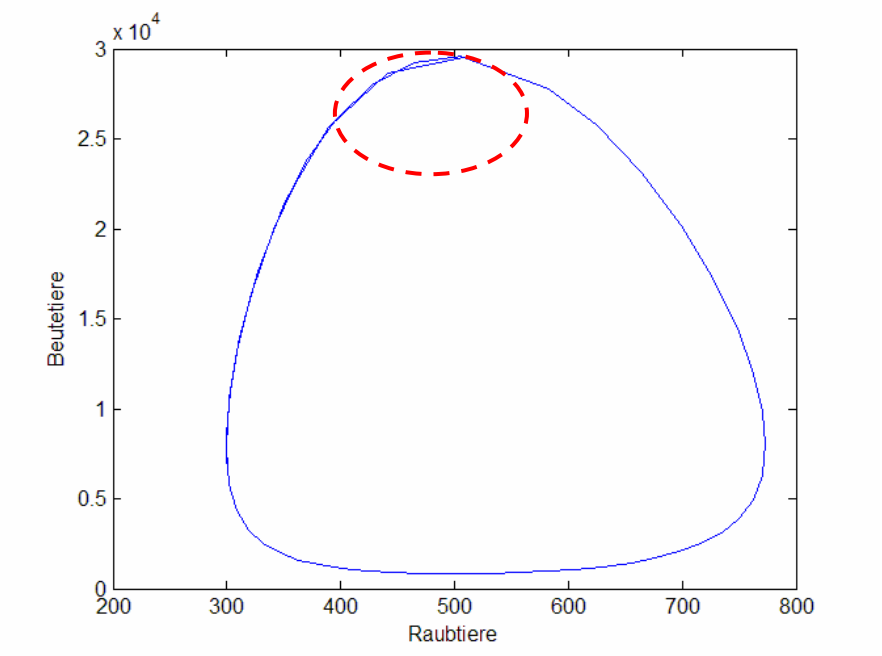

***Bild 1.6**: Phasendiagramm des Räuber Beute Modells. Bei dieser Auftragung sollte jeder Zyklus der Populationen deckungsgleich auf dem vorherigen liegen. Wie Bild 1.7 zeigt, simulieren wir 1,5 Zyklen. Im oberen Bereich ist eine Abweichung der ersten und des zweiten Zyklus eingekreist. Diese Ungenauigkeit, kann durch Verkleinerung des zulässigen Restfehlers bei der Iterationslösung der DGLs beseitigt werden.*

Die eingekreiste Ungenauigkeit in der Abbildung, kann durch Verkleinerung des zulässigen Restfehlers bei der Iterationslösung der DGLs beseitigt werden. Der Befehl lautet:

```
[t, x] = ode45('DGLSystem',[t0, tende],[x01,x02], ode-
set('RelTol',1e-6));
```

Der Parameter `RelTol` bestimmt die Zahl der korrekten Stellen im Ergebnis. Hier wird sichergestellt, dass die Ziffern bis zur sechsten Stelle nach dem Komma korrekt sind. Wird das `plot`-Kommando abgeändert, können wir das zyklische Anwachsen und Abnehmen der Tierpopulationsstärken als Zeitfunktionen auftragen lassen (Bild 1.7).

```
plot (t, x)
xlabel ('Zeit')
ylabel ('Raubtiere, Beutetiere')
```

Wir erkennen die Voltera-Lotka-Regeln: Im zeitlichen Mittel bleibt die Räuber- und Beutepopulation konstant, oszilliert aber um diesen Mittelwert.

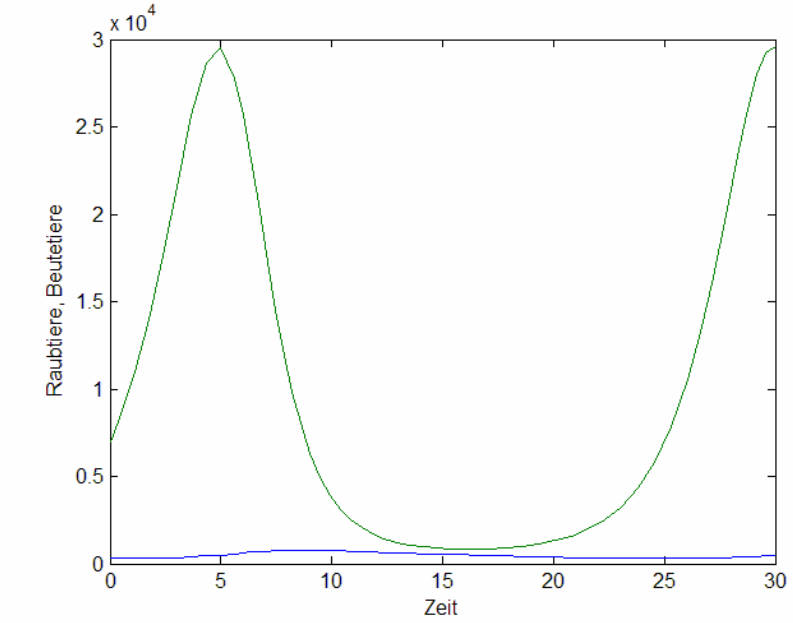

Bild 1.7 *Beutetiere (große Amplitude) und Räuber (kleine Amplitude). Wächst das Beuteangebot nehmen die Räuber zeitversetzt zu. Dies führt zu einer Überjagung, das Beuteangebot sinkt, die Jägerpopulation in Folge ebenfalls, die Beutetiere vermehren sich wieder usw.*

1.5 Simulationsgenauigkeit

Eine wichtige Frage, die sich nach einer Simulation stellt, ist wie genau die Simulation die Realität widerspiegelt. Wie gut sagen die Simulationsresultate die späteren Messungen an Prototypen vorher? Bild 1.8 zeigt, welche Einflussgrößen für die Vorhersagekraft und die Genauigkeit des Simulationsergebnisses eine Rolle spielen.

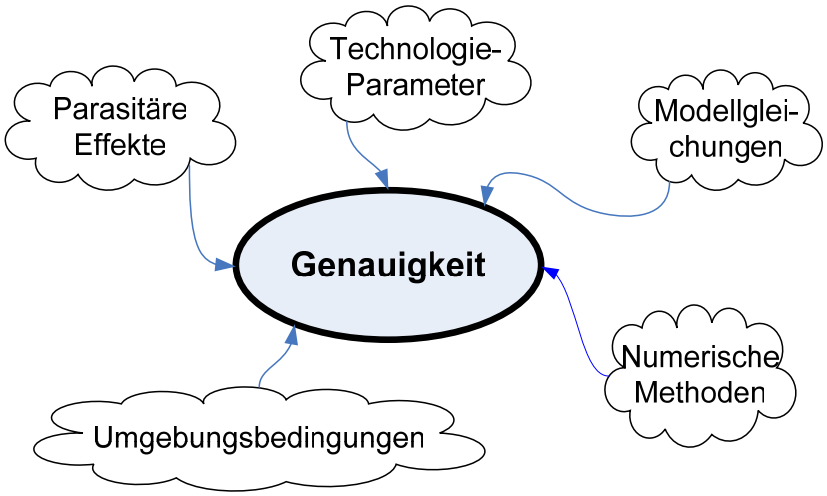

Bild 1.8 *Einflussgrößen für die Genauigkeit einer Simulation*

Jedes numerische Lösungsverfahren ist nicht beliebig genau und führt zu Restfehlern. Deshalb müssen die Abbruchbedingungen bei der iterativen Lösung des Modellgleichungssystems ausreichend scharf vorgegeben werden. In Zusammenhang mit dem Lotka-Voltera-Modell (Bild 1.6) ist ein entsprechender Fall bereits diskutiert worden.
Auch der beste numerische Algorithmus kann die Schaltung nur so gut simulieren, wie dies die Modelle und Parameter zulassen. Die Modelle in SPICE sind physikalisch orientiert, die Modelle in MATLAB und VHDL-AMS können aber vollkommen abstrakt formuliert sein.
SPICE verwendet für die nichtlinearen Bauelemente, wie Transistoren, ausgetestete physikalische Modelle, in denen bestimmte Größen als Parameter aufgefasst werden, die aus Messungen an Testbauelementen gewonnen werden. Bei einem MOS-Transistor ist bspw. die Einsatzspannung ohne Substratvorspannung VT0 ein solcher Anpassparameter. Die Einsatzspannung lässt sich zwar auch theoretisch berechnen, aber durch die Einbeziehung von Messresultaten können komplexere Effekte, die in die gesuchte Größe eingehen, implizit mit erfasst werden. So bleiben die Modelle mathematisch relativ einfach, geben aber doch die Bauelementeigenschaften genau wieder. Wenn die Modelle akkurat die Eigenschaften der Bauelemente im Netzwerk beschreiben und die Modellparameter die verwendete Herstelltechnologie gut abbilden, dann kann man bei SPICE mit weniger als 10%-Abweichung gegenüber den bei der Parameterextraktion herangezogenen Messdaten rechnen.
Bei Verhaltensmodellen, die mit MATLAB oder in VHDL-AMS geschrieben worden sind, bestimmt letztlich der Ingenieur-Aufwand, der in die Modelle investiert wird, wie zuverlässig und genau die beschriebenen Systeme auf dem Computer nachgebildet werden.

Jedes Modell sei es verhaltensorientiert oder physikalisch enthält in der Regel bestimmte **Parameter**, die Diode bspw. die Kniesspannung, den Bahnwiderstand und die Sperrschichtkapazität. Parameter werden in der Elektronik nicht nur für typische Herstellprozesse sondern auch für verschiedene grenzwertige Extremlagen mit angegeben (min-, max- Parameter, Eckparameter). Diese Parameter stehen dann für den kleinstmöglichen oder größtmöglichen Parameterwert (Verstärkung, Widerstand, etc.), der im Rahmen der Technologietoleranzen auftreten können. Solche Extremfälle können für eine Schaltung besonders kritisch sein. Deshalb ist nicht nur für typische Parameter, sondern auch für die kritische Parameterwahl zu simulieren. Häufig beeinflussen auch die Betriebsbedingungen, wie Temperatur der Schaltung oder die Versorgungsspannung, die Systemeigenschaften nachhaltig. Zwischen dem bestmöglichen und dem schlechtmöglichsten Parameterfall, über den gesamten beispielsweise für den Automobilbau interessanten Temperaturbereich (-40° C bis 120° C) unterscheiden sich z.B. die Verstärkungsfaktoren von MOS-Transistoren um den Faktor 2! Deshalb ist es wichtig, die für die spätere Anwendung relevanten Betriebsbedingungen bei der Simulation abzuprüfen und auch die Eckparametersätze einzubeziehen.
Jede Komponente eines Systems ist mit bestimmten **parasitären Effekten** behaftet. Dies sind Effekte, die für die eigentliche Funktion nicht benötigt werden, aber bei der technischen Implementierung zwangsläufig mit entstehen und die Systemeigenschaften beeinflussen. Bei einer elektrischen Leitung, die ein Signal von einer Komponente zu nächsten weiterleiten soll (im Idealfall ohne Widerstand), kommt es aufgrund des unvermeidbaren elektrischen Widerstands zu einer Signalverschlechterung und aufgrund von kapazitiven Kopplungen zu benachbarten Leitungen ggf. auch zum Übersprechen.
Beim Systemdesign denkt man normalerweise nicht an solche parasitären Effekte, sie können auch evtl. in den ersten Designstadien noch gar nicht erfasst werden, weil man den physikalischen Verlauf der Leitungen im System noch gar nicht kennt. Im Zuge des Designfortschritts sind aber die parasitären Effekte einzubeziehen. Dies wird meist erst für das ausoptimierte Systeme durchgeführt, kann aber grundlegende Änderungen des Entwurfs nach sich ziehen und weitere Designiterationen notwendig machen.

1.6 Leitlinien für die Verwendung von Simulationen

Um die Simulation beim Design effizient und zielführend einzusetzen, sollte man sich an folgende Regeln halten:

- Verwende keinen Simulator, wenn der Bereich der erwartenden Ergebnisse nicht von vorneherein bekannt ist.

- Beschränke Dich bei der Simulation auf die wesentlichen Systemkomponenten und simuliere nie mehr vom System als nötig!
- Ändere nie mehr als eine Designvariable zwischen zwei Simulationen, sonst sind die Auswirkungen der Designverbesserung nicht zuzuordnen.
- Lerne die wesentlichen Funktionsprinzipien des verwendeten Simulators kennen, sonst kann man dieses Werkzeug nur eingeschränkt benutzen!
- Simulationen ersetzen nicht das Nachdenken!

Übungsaufgaben

1. Was sind die Grundbestandteile eines Modells?
2. Was unterscheidet ein Modell von einer Theorie oder von einem realen System?
3. Stellen Sie die Bottom-Up- der Top-Down-Design-Stategie gegenüber und benennen Sie die Vor- und Nachteile der beiden Vorgehensweisen!
4. Warum wird der Simulator SPICE als Schaltkreissimulator bezeichnet?
5. Wie wird die Zeitentwicklung eines digitalen Systems im Rahmen der Logiksimulation berechnet?
6. Was verstehen Sie unter „Gewöhnlichen Differentialgleichungen“?
7. Wie werden im Rahmen von VHDL-AMS die digitale Simulation und die analoge Simulation vereinheitlicht?
8. Wofür eignen allgemeine Numeriktools wie MATLAB prinzipiell?
9. Was verstehen Sie unter Eckparametern und was sind „Parasitäten“?
10. Skizieren und erläutern Sie das Y-Diagramm!
11. Was bezeichnet der Begriff „Event“?
12. Wo liegen die Genauigkeitsgrenzen bei der Simulation und wie kann man diese beeinflussen?
13. Welche Sichtweisen gibt es bei der Modellierung?
14. Wo liegen die Vorteile beim Einsatz genormter Designeingabesprachen? Ist die Sprache von MATLAB genormt?
15. Was sind SPICE-Parameter und wie werden SPICE-Modelle entwickelt?
16. Was ist grundlegender Zweck einer Simulation und warum sind heute Simulatoren beim Systemdesign nicht mehr weg zu denken?
17. Wie unterscheiden sich Computeralgebrasysteme und MATLAB?
18. Was verstehen Sie unter einem Mixed-Signal-Design und einem heterogenen System?
19. Wie ist eine Testbench aufgebaut und warum werden Testenches verwendet, wenn komplexe Designs verifiziert werden müssen?

Literaturverzeichnis

[ABRW05] A. Angermann, M. Beuschel, M. Rau und U. Wohlfarth : „MATLAB-Simulink-Stateflow", Oldenbourg Verlag München, 2005

[AH02] P. Allen und D. R. Holberg *CMOS Analog Circuit Design*, 2nd Edition, Oxford University Press, 2002

[APT03] P. J. Ashenden, G. D. Peterson, Darell A. Teegarden: „The System Designer's to VHDL-AMS", Morgan Kaufmann Publishers, Amsterdam, 2003

[CRB99] E. Christen und K. Bakalar: „VHDL-AMS – a hardware description language for analog and mixed signal applications", IEEE Transactions on Circuits and Systems, part II, S. 1263 -1272, 1999

[GG04] F. Grupp, F. Grupp: „MATLAB7 für Ingenieure", Oldenbourg Verlag München, 2004

[H2006] B. Hoppe, *Verilog*, Oldenbourg Verlag München, 2006

[IEEE Standard 1076] Standard 1076-1993: IEEE Standard VHDL Language Reference Manual; IEEE Standards Department; 1993

[IEEE Standard 1076.1] Standard 1076.1-1999: IEEE *Standard VHDL Analog and Mixed-Signal Extensions*, ISBN 0-7381-1640-8 (http://www.vhdl.org/analog/).

[WE94] N. H. E. Weste und K. Eshraghian Principles of CMOS VLSI Design, 2nd Edition, Addison Wesley, Reading Mass. U.S.A. 1994

Anhang: Simulation von SPICE Netzlisten mit System-Vision®

Statt ein komplett neues Projekt anzulegen und zu simulieren, kann man eine mit Signalquellen versehene und damit vollständig simulierbare SPICE-Netzliste einem bestehenden Projekt als *auxiliary file* (Hilfsdatei) zuordnen und dieses File dann unabhängig vom gerade aktiven Projekt simulieren.
Um SPICE Netzlisten für die Simulation vorzubereiten, verfährt man folgendermaßen:

1. Im Project Navigator klicken wir auf den **Simulation** Reiter.
2. Dann erweitern wir die Anzeige mit Betätigen des [+]-Symbols bei **Simulation, Analysis, and Results**.
3. Dann selektieren wir mit der rechten Maustaste auf **Testbenches**. Dann öffnet sich das Popup-Menü.
4. Wir wählen **New Testbench**.
5. Dann suchen wir die SPICE Netzliste, die simuliert werden soll und bestätigen mit **OK**.

Dann öffnet sich der **New Testbench** - Dialog.

6. In diesem Dialog bestätigen wir mit **OK**.

Die SPICE-Datei erscheint daraufhin als Testbench-File im Project Navigator.

8. Ein Klick mit der rechten Maustaste öffnet ein Menü.
9. Dort selektieren wir **Simulate [**Spice Testbench**]**.

Stichwortverzeichnis