Teilprojekt A2: Systementwurf

Prof. Dr.-Ing. habil. Dietmar Müller
Professur Schaltungs- und Systementwurf (LSSE)

---

1. Kenntnisstand bei der letzten Antragstellung und Ausgangsfragestellung

Das Forschungsprogramm im Teilprojekt (TP) A2 umfaßte seit 1995 Arbeiten zur Modellierung, Simulation und Applikation von Systemen, in deren Mittelpunkt mikromechanische Arrays stehen. Im Antragszeitraum 2001-2003 lag der Schwerpunkt der Arbeiten

- im Entwurf neuer und der Verfeinerung bestehender methodischer Ansätze für den Mikrosystementwurf,
- in der Übertragung von Methoden der Mikroelektronik auf die Mikrosystemtechnik,
- in der Modellierung und Systemsimulation eines Demonstrator zur Verschleißanalyse mittels frequenzselektivem Vibrationssensor-Array,
- in der Verifizierung der geschaffenen Methoden anhand des Demonstrators,
- in der Weiterentwicklung eines Datenmanagementsystems zur Verwaltung der Entwurfsdokumente und -resultate.

Das gesamte Arbeitsprogramm wurde in zwei Bearbeitungspakete geteilt:

1. Methoden für den Mikrosystementwurf,
2. Datenmanagement.

1.1 Methoden für den Mikrosystementwurf

Die vom IEEE standardisierte Hardwarebeschreibungssprache VHDL-AMS [19], [20] hat sich bereits im letzten Antragszeitraum als Modellierungs- und Simulationssprache für heterogene Systeme/Mikrosysteme bewährt. Allerdings haben sich im Laufe der Arbeiten einige Probleme sowohl im Entwurf selbst als auch in der Zusammenarbeit zwischen den Entwerfern herauskristallisiert.

Der Entwurf elektronischer Schaltungen ist heute ein unverzichtbarer Bestandteil im Entwurfsprozeß moderner Mikrosysteme. Dafür kommt zunehmend die Hardwarebeschreibungssprache VHDL-AMS zum Einsatz, da sich mit ihr die elektrischen, mechanischen und z. B. thermischen

oder optischen Komponenten eines Mikrosystems gemeinsam beschreiben lassen. VHDL-AMS bietet aber von sich aus keine vordefinierten elektronischen Bauteile oder Elemente an, wie es der Entwerfer z. B. aus SPICE-basierten Simulatoren gewohnt ist. Das hat zur Folge, daß, angefangen bei elementaren Bauteilen wie Widerstand, Kondensator und Spule bis hin zu Leitungen, Transistoren, OPV oder digitalen Schaltkreisen, alle benötigten Bauteile selbst entworfen werden müssen. Um diesen zeitaufwendigen und fehleranfälligen Prozeß zu vermeiden, entstanden im Arbeitspaket 1.1c „Schaffung von VHDL-AMS Packages“ umfangreiche VHDL-AMS Bibliotheken mit SPICE-Modellen und MATLAB-Funktionen.

Der Entwurf von Modellen von Mikrosystemkomponenten erfordert mit VHDL-AMS sehr viel Know-How beim Entwerfer, da im Gegensatz z. B. zur FEM-Simulation die Funktion der Komponente analytisch beschrieben werden muß. Aufgrund der zum Teil immensen Komplexität von Komponenten ist dies oft nicht möglich, so daß die Modelle sehr stark vereinfacht werden müssen, worunter die Simulationsgenauigkeit erheblich leiden kann. Deshalb gibt es Bestrebungen (z. B. im SFB 379 TP A1 oder im Rahmen des SFB 358 an der FhG Institut für integrierte Schaltungen EAS Dresden [21]), genaue FEM-Modelle durch Ordnungsreduktion auf eine begrenzte Anzahl von Differentialgleichungen zu reduzieren. Die dabei entstehenden Makromodelle besitzen ähnliche Genauigkeit wie FEM-Modelle, sind aber erheblich schneller zu simulieren. Teil des Arbeitspaketes 1.1 „Makromodellierung/Multi-Architecture-Modellierung“ ist deshalb die Unterstützung des TPs A1 bei der Umsetzung der Makromodelle in VHDL-AMS.

Makromodelle stellen einen effizienten Ansatz dar, das Verhalten von Komponenten zu beschreiben und zu simulieren. Es besteht dabei aber die Möglichkeit, daß für eine Systemsimulation Makromodelle noch zu detailliert sind oder zu Beginn des Entwurfes noch nicht zur Verfügung stehen. VHDL-AMS besitzt nun die Eigenschaft, mehrere Architekturen (Modelle) für eine einheitliche Schnittstelle zu beschreiben. Diese Eigenschaft von VHDL bietet sich an, Komponentenmodelle auf niedrigem Abstraktionsniveau, Makromodelle und Modelle auf hohem Abstraktionsniveau, wie sie während eines Top-Down-basierten Systementwurfs entstehen können, zu verwalten und für die Simulation zur Verfügung zu stellen. Dabei ergibt sich aber das Problem, daß Modelle auf unterschiedlichen Abstraktionsniveaus oft unterschiedliche Schnittstellen besitzen.

Bild 9 (Abschnitt 7) verdeutlicht diese Problematik anhand eines Beispiels, bei dem zunächst Modelle auf der Basis funktioneller Blöcke beschrieben werden. Als Interface kommen hier nichtkonservative Knoten (`QUANTITY`) zum Einsatz. Während des Entwurfsprozesses wird eine Komponente zu einer Netzwerkbeschreibung weiterentwickelt oder durch ein Makromodell ersetzt, die andere dagegen nicht. Die weiterentwickelte (verfeinerte) Komponente benutzt nun als Interface konservative Knoten (`TERMINAL`), während die unveränderte Komponente nach wie vor `QUANTITIES` verwendet. Beide Modelle können dadurch nicht mehr miteinander verbunden werden (weitere Details siehe Abschnitt 2.1.2.2). Ziel des Arbeitspaketes 1.1 „Makromodellierung/Multi-Architecture-Modellierung“ ist es daher, einen methodischen Ansatz zu finden, durch den Modelle auf unterschiedlichen Abstraktionsebenen mit einheitlichen Schnittstellen verbunden werden können.

Mit der fortschreitenden Entwicklung von Entwurfswerkzeugen und Bibliotheken für den Mikrosystementwurf entsteht immer mehr die Möglichkeit/Notwendigkeit, Mikrosysteme nicht nur hinsichtlich funktioneller Parameter, sondern auch hinsichtlich von Kostenfaktoren[1] zu optimieren.

1. Der Begriff *Kosten* wird hier im Sinne der Terminologie der mathematischen Optimierung verwendet und repräsentiert das Ergebnis einer zu optimierenden Zielfunktion, mit der Größen wie Leistungsverbrauch, Chipfläche, Fehlertoleranz, Testbarkeit, Speicherverbrauch, Herstellungskosten u. a. gewichtet bewertet werden [24].

Erste Ansätze dazu sind z. B. in [23] dokumentiert. Im Rahmen des Arbeitspaketes 1.2 „Kosten-Modellierung“ entsteht ein methodischer Ansatz, der es in der Sprache VHDL-AMS erlaubt, Kostenfaktoren und Zielfunktionen in das Modell einzubringen, so daß diese Faktoren wie funktionelle Parameter behandelt werden können. Damit steht dem Nutzer die Möglichkeit zur Verfügung, die Modelle hinsichtlich der Kosten interaktiv oder automatisiert mit vorhandenen Werkzeugen zur Optimierung der Funktionalität [22] zu optimieren.

Die Wiederverwendung (Reuse) bereits existierender Modelle spielt in der Mikroelektronik, aber zunehmend auch in der Mikrosystemtechnik, eine große Rolle. Um flexibel einsetzbar zu sein, ist die Parametrisierbarkeit der Modelle notwendig. Durch die Parametrisierung besteht aber insbesondere bei Modellen der Mikrosystemtechnik die Gefahr, daß mögliche Grenzen technologischer oder physikalischer Parameter überschritten werden. Finden die Modelle im Rahmen der Multi-Architecture-Modellierung Anwendung, so entsteht die Notwendigkeit, die Grenzen (Constraints) der Modelle auch auf unterschiedlichen Abstraktionsebenen zu überwachen. Im Arbeitspaket 1.3 „Constraint-Modellierung“ wird daher, unter Anlehnung an das aus der Informatik bekannte Constraint-Programming, nach einer Möglichkeit gesucht, die Grenzen von Modellen hierarchieübergreifend zu beschreiben und die Zusammenhänge zwischen den Grenzen zu erfassen.

Um die geschaffenen Methoden zu verifizieren, erfolgt im Arbeitspaket 1.4 „Systementwurf für den Demonstrator Vibrationssensor-Array“ in enger Zusammenarbeit mit dem TP A4 der Systementwurf und die Systemsimulation für den im TP A4 entwickelten Demonstrator zur Verschleißanalyse mittels frequenzselektivem Vibrationssensor-Array.

1.2 Datenmanagement

Auf Basis der gewonnenen Erfahrungen in Bezug auf den Entwurfsablauf und dessen Organisation im SFB 379 in den Antragszeiträumen 1995-97 sowie 1998-2000 entstand der Prototyp für ein Datenmanagementsystem (DMS), das als Repository für verschiedenartige Dokumente und auch für die Abbildung von Entwurfsprozessen eingesetzt werden kann. Das DMS besteht aus zwei Komponenten, einem relationalen Datenbanksystem, das den gesamten Datenbestand verwaltet (Zugriffsmechanismen, Nutzerverwaltung, Konsistenzüberwachung, Kontrolle von Datenbereichsrestriktionen, ...) und der Applikation, die die Schnittstelle zur Datenbank bereit stellt und ein grafisches Benutzerinterface (GUI) generiert.

Nach erneuter Analyse der Ausgangssituation bezüglich des Datenbanksystems (Verfügbarkeit, Kostensituation) wurde festgelegt, an Stelle von ADABAS D das relationale Datenbanksystem PostgreSQL einzusetzen, da dies kostenlos zur Verfügung steht und volle Funktionalität besitzt. Es mußte folglich das Datenschema in PostgreSQL neu abgebildet und eine Datenmigration bereits vorhandener Daten aus ADABAS D nach PostgreSQL durchgeführt werden. Des weiteren wurde die Anpassung der Applikation an das neue Datenbanksystem in Bezug auf die Datenbankzugriffsschnittstelle (Zugriffsfunktionen auf der Basis der Skriptsprache Tcl) notwendig.

Ein weiterer Schwerpunkt (Arbeitspaket 2a) ist die Verbesserung des GUI in Bezug auf Handling, die grafische Darstellung von Datenstrukturen im Entwurf von Fertigungseinheiten und gefertigten Einheiten [16] und die Datenabfrage mit QBE (Query by Example), d. h., die Datenabfrage über die Ein-/Ausgabeformulare und die Datenabfrage über eine SQL-Schnittstelle (SQL = Structure Query Language). Die Applikation soll weiterhin die Möglichkeit einer flexiblen und dynamischen Erweiterbarkeit des DMS nach neuen Anforderungen bieten sowie die Volltextrecherche mit taxonomiebasierten Wiederverwendungsschemata bereitstellen. Geplant sind weiterhin die

Integration des an der Professur Schaltungs- und Systementwurf (LSSE) entwickelten Werkzeugs zur Spezifikationserfassung und Entwurfsdokumentation (ASPECTOR) und die WWW-Anbindung des DMS (Arbeitspaket 2b).

2. Angewandte Methoden (Forschungsarbeiten)

2.1 Methoden für den Mikrosystementwurf

2.1.1 Bibliotheksentwicklung für VHDL-AMS

Der Entwurf von Bibliotheken in VHDL-AMS wurde im Finanzierungsantrag als Teilaufgabe des Arbeitspaketes 1.1 Makromodellierung/Multi-Architecture-Modellierung aufgeführt. Aufgrund der umfangreichen Arbeiten erhält dieses Aufgabenpaket im Ergebnisbericht einen eigenen Abschnitt.

2.1.1.1 Bibliothek mit SPICE-Elementen

Der Entwurf elektronischer Schaltungen als Bestandteil von heterogenen Systemen oder Mikrosystemen setzt umfangreiche Bibliotheken mit elektronischen Bauteilen voraus. In VHDL-AMS existieren aber keine vordefinierten Bauteile, wie sie der Entwerfer z. B. aus SPICE gewohnt ist. Daher entstand im TP A2 „Systementwurf“ eine Bibliothek mit elektronischen Komponenten, die mit PSPICE-Elementen vergleichbar sind. Diese Bibliothek wird über das Datenmanagementsystem den anderen Teilprojekten des SFB 379 zur Verfügung gestellt. Damit können Modelle heterogener Systeme, für die bislang unterschiedliche Simulationssysteme notwendig waren, mit einem einheitlichen Werkzeug modelliert und simuliert werden. Der Vorteil der Beschreibung heterogener Systeme in VHDL-AMS besteht u. a. darin, daß Modelle auf unterschiedlichen Abstraktionsniveaus entworfen (siehe auch 2.1.2) und leicht in andere physikalische Domänen (mechanische, thermische, fluidische Systeme) konvertiert werden können.

Die Umsetzung der meisten Elemente der PSPICE-Bibliotheken „ABM“ (analog behavioral model), „analog“ und ausgewählter Elemente der Bibliotheken „port“ und „source“ nach VHDL-AMS gestaltet sich sehr einfach (z. B. Widerstand, gesteuerte Quellen etc.). Andere Elemente, wie z. B. Filter, erfordern in ihrer Modellierung etwas mehr Aufwand, sind aber dennoch einfach zu realisieren. Komplizierter gestaltet sich dagegen z. B. die Beschreibung einer verlustbehafteten, homogenen Leitung. Neben den Bibliotheken mit passiven und abstrakten Modellen wurden auch ausgewählte aktive Bauteile aus der PSPICE-Bibliothek „eval“ in VHDL-AMS realisiert. Dazu zählen parametrisierbare Modelle für Dioden, Bipolar- und Feldeffekttransistoren uvm. Für die Modelle der verlustbehafteten Leitung und der Halbleiterdiode ist der Entwurfsablauf exemplarisch dargestellt.

Übersicht über die in VHDL-AMS realisierten SPICE(PSPICE)-Bauelemente:

passiv:

- Widerstand mit Temperaturkoeffizient und Rauschen, Spule, Kondensator und Transformator
- Masse: gnd, agnd, egnd, gnd_analog, gnd_earth

- Spannungs- und Stromquellen: dc, src, sin, pulse, pwl, exp

abstrakt:

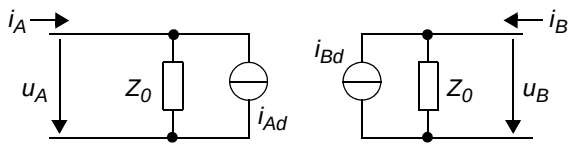
- gesteuerte Quellen: E, F, G, H, evalue, gvalue, const
- Vorlagen für analoge Verhaltensmodelle: ABM, ABM_I … ABM3, ABM_3I
- mathematische Funktionen: absolut, sin, cos, tan, asin, acos, atan, sqroot, exp, log, log10, pwr, pwrs, sum, diff, mult, emult, gmult, esum, gsum
- Verstärker und Begrenzer: gain, limiter, glimiter, softlim, hilo
- Ableitung, Integral: differ, integ
- Filter: lopass, hipass, bandpass, bandrej
- Laplace-Transferfunktion: laplace, elaplace, glaplace
- Leitungen: t (verlustfrei), tlossy (verlustbehaftet)
- Tabellen mit Interpolation: table, etable, gtable

aktiv:

- Halbleiterdioden: allgemeines Modell nach [34],
  Modelle der 1N4148, ZD47 (Z-Diode) und S10 (Schottky-Diode)
- Bipolartransistoren: allgemeines Modell nach Gummel-Poon [35],
  Modelle des BC547B und BC557B
- Sperrschicht-Feldeffekttransistoren: allgemeines Modell nach [35],
  Modelle des 2N3819 und 2N4221
- MOS-Transistoren: allgemeines Modell nach Berkeley Level 1 und 3 [36], [37],
  Modell des IRF150,
- OPV: Modell des μA741
- Thyristor, DIAC, TRIAC, IGBT: allgemeine Modelle
  Modelle 2N1595, DB3TG, 2N5444, IRG4RC10K

Modellierung der verlustbehafteten Leitung

Ausgangspunkt für das Modell einer verlustbehafteten Leitung ist zunächst die verlustfreie Leitung mit folgenden Beziehungen:



$$Z_0 = \sqrt{\frac{L'}{C'}} \tag{1}$$

$$v = \frac{1}{\sqrt{L' \cdot C'}}, \tau = \frac{l}{v} \tag{2}$$

Bild 1 Ersatzschaltbild der verlustbehafteten Leitung

$$i_{Bd}(t) = \frac{u_A(t-\tau)}{Z_0} + i_A(t-\tau) \tag{3}$$

$$i_{Ad}(t) = \frac{u_B(t-\tau)}{Z_0} + i_B(t-\tau) \tag{4}$$

und die Telegraphengleichung [25]:

$$\frac{\partial^2 u}{\partial x^2} = R'G' \cdot u + (R'C' + L'G') \cdot \frac{\partial u}{\partial t} + L'C' \cdot \frac{\partial^2 u}{\partial t^2} \tag{5}$$

Diese partielle Differentialgleichung repräsentiert die Änderung der Spannung in einem infinitesimal kleinen Abschnitt der Leitung. Sie ist in dieser Form nicht lösbar, es existieren allerdings verschiedene Näherungsansätze zur Lösung.

Eine Möglichkeit besteht darin, die Leitung in endlich viele Teile zu zerlegen und für jedes dieser Teilstücke ein Netzwerk aus diskreten R, G, C und L aufzubauen. Dieser Ansatz ist leicht realisierbar, hat aber den Nachteil, daß ein großes Gleichungssystem mit vielen Netzwerkelementen berechnet werden muß, was sehr zeitintensiv ist. Deshalb wurde eine Lösung gesucht, die die Leitung als nur ein Netzwerkelement erscheinen läßt. Einen interessanten Ansatz zur Lösung der Telegraphengleichung im Zeitbereich bietet das in [26] beschriebene Verfahren. Daraus resultieren die folgenden, zu simulierenden Gleichungen:

$$i_{Bd}(t) = \left(\frac{u_A(t-\tau)}{Z_0} + i_A(t-\tau)\right) \cdot e^{-\alpha l} + \frac{br_B(t)}{Z_0} + \frac{brr_B(t)}{Z_0} \tag{6}$$

$$i_{Ad}(t) = \left(\frac{u_B(t-\tau)}{Z_0} + i_B(t-\tau)\right) \cdot e^{-\alpha l} + \frac{br_A(t)}{Z_0} + \frac{brr_A(t)}{Z_0} \tag{7}$$

mit:

$$\alpha = \frac{R'}{2Z_0} + \frac{Z_0 G'}{2} \tag{8}$$

$$br_{A,B}(t) = \vartheta^2 \cdot br_{A,B}(t-\Delta t) + br2_{A,B} - br3_{A,B} \tag{9}$$

$$br2_{A,B} = \rho \cdot b_{A,B}(t-\Delta t) \tag{10}$$

$$br3_{A,B} = \rho\vartheta^{2n} \cdot b_{A,B}(t-2\tau) \tag{11}$$

$$brr_{A,B} = b_{A,B}(t) \cdot \frac{\rho^2 \cdot \vartheta^n}{1-\vartheta^2} \cdot \left(n - \frac{1-\vartheta^{2n}}{1-\vartheta^2}\right) \tag{12}$$

$$\rho = \frac{R' \cdot \Delta l - Z_0^{\,2} \cdot G' \cdot \Delta l}{2Z_0 + R' \cdot \Delta l + Z_0^{\,2} \cdot G' \cdot \Delta l} \tag{13}$$

$$\vartheta = \frac{2Z_0}{2Z_0 + R' \cdot \Delta l + Z_0^{\,2} \cdot G' \cdot \Delta l} \tag{14}$$

$$b_{A,B}(t-\Delta) = u_{A,B}(t-\Delta) + Z_0 \cdot i_{A,B}(t-\Delta) \tag{15}$$

$$\Delta\tau = \frac{\tau}{n},\ \Delta t = \frac{2\tau}{n} \tag{16}$$

Der erste Term der Gleichungen (6) und (7) entspricht dem aus der verlustfreien Leitung bekannten Ansatz (3) und (4), erweitert um einen Faktor e^x, der ein Maß für die Dämpfung ist. Der Term $br_{A,B}$ beschreibt einfache Reflexionen an den Verlustelementen innerhalb der Leitung und Term $brr_{A,B}$ beschreibt doppelte Reflexionen. Mehrfachreflexionen höherer Ordnung werden vernachlässigt.

Modellierung der Halbleiterdiode

Ein einheitliches Modell, entsprechend dem beispielsweise für Bipolartransistoren, existiert für Dioden nicht [35]. Das in PSPICE verwendete Modell [34] ist in großen Teilen mit dem in [35] beschriebenen vollständigen Modell für Dioden identisch. Die entsprechende Ersatzschaltung ist in Bild 2 dargestellt.

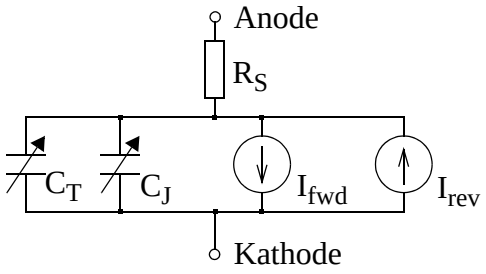


Bild 2 Ersatzschaltbild einer Diode [34], [35]

Das Modell besteht aus einer idealen Diode, die durch die Stromquellen mit den Strömen I_{fwd} und I_{rev} dargestellt wird und den dazu parallel geschalteten Kapazitäten C_T und C_J. Dieser Anordnung wird ein Serienwiderstand R_S vorgeschaltet. Die Beschreibung des statischen Strom-Spannungs-Verhaltens der Diode beruht im wesentlichen auf dem allgemeinen Zusammenhang für Strom und Spannung an pn-Übergängen nach Shockley [35]. Um das reale Verhalten näher beschreiben zu können, wird dieses ideale Verhalten um die Modellierung zusätzliche Effekte erweitert, wie z. B. um den Vorwiderstand R_S, der den Spannungsabfall über den Bahngebieten charakterisiert, sowie um den Hochstromeffekt (Hochinjektion). Die Generation und Rekombination von Ladungsträgern in der Sperrschicht findet ebenso Beachtung, wie die arbeitspunktabhängige Ausdehnung der Raumladungszone.

Ein weiterer wichtiger Effekt ist das Sperrverhalten der Diode mit dem daraus resultierenden Rückwärtsstrom I_{rev}. Die Modellierung dieses Verhaltens ist nicht physikalisch exakt, sondern

beruht auf der Nachbildung von Meßkurven [38]. Dazu werden ein low-level- und ein high-level-Durchbruch definiert.

Viele wichtige Parameter der Diode, so z. B. der Sättigungssperrstrom oder der Serienwiderstand, sind außerdem temperaturabhängig.

Neben den statischen Eigenschaften wird auch das dynamische Verhalten durch entsprechende Modellgleichungen für die Sperrschichtkapazität C_J sowie die Diffusionskapazität C_T modelliert. Die Beschreibung des Rauschens der Diode erfolgt durch das thermische Rauschen des Serienwiderstandes R_S sowie das Schrot- und 1/f-Rauschen der inneren Diode. Dazu enthält das Modell an den entsprechenden Stellen Rauschquellen. Der Übersichtlichkeit halber wurden diese Rauschquellen in Bild 2 nicht mit dargestellt.

2.1.1.2 Bibliothek mit MATLAB-Funktionen

Für die Verhaltenssimulation auf algorithmischer Ebene sind oft komplexe mathematische Funktionen notwendig. Als Simulationswerkzeug dafür kommt häufig MATLAB zur Anwendung. Möchte man diese Simulationen in VHDL-AMS durchführen, so müssen komplexe mathematische Funktionen ebenfalls in Form einer Bibliothek zur Verfügung stehen. Dazu wurden mathematische Grundoperationen und ein Teil der LINPACK und LAPACK Fortran-Algorithmen [27], [28], auf denen die MATLAB-Funktionen beruhen, in VHDL implementiert.

Auf eine ausführliche Herleitung einzelner Modelle wird hier verzichtet, da dies sehr umfangreich ist und es sich um bekannte Algorithmen handelt. Die Bibliothek umfaßt dabei einfache Operatoren und Funktionen wie Addition, Subtraktion, Winkelfunktionen, Logarithmen etc., die auf Elemente von Matritzen angewand werden können. Darüber hinaus enthält die Bibliothek verschiedene Funktionen zur Matrixtransformationen, zur Berechnung von Determinanten und Standardabweichungen, zur Dreieckszerlegung bis hin zur Berechnung der Eigenwerte einer Matrix. Mit diesen Funktionen ist es z. B. möglich, alle Nullstellen oder alle Wurzeln eines Polynoms zu bestimmen. Eine Auflistung der in VHDL realisierten Funktionen befindet sich im Anhang.

Für die automatisierte Umsetzung von MATLAB-Scripten nach VHDL wurde auf der Basis von Lex&Yacc ein Übersetzungs-Tool programmiert. Dieses Tool setzt die Steuerstrukturen (`IF`-Anweisungen, Schleifen etc.) in VHDL-Äquivalente um, und ersetzt die mathematischen Funktionen der MATLAB-Scripte durch die entsprechenden in VHDL realisierten Funktionen und Operatoren.

2.1.2 Makromodellierung/Multi-Architecture-Modellierung

2.1.2.1 Makromodellierung

Die Erstellung von Makromodellen durch Ordnungsreduktion von FEM-Modellen ist ein wesentlicher Bestandteil der Arbeiten im TP A1. Um die entstehenden Makromodelle sofort im TP A2 im Systementwurf nutzen zu können, war es notwendig, die Modelle in der Hardwarebeschreibungssprache VHDL-AMS zu erzeugen. Dabei erfolgte eine intensive Zusammenarbeit zwischen den Teilprojekten A1 und A2. In vielen Projektbereichsberatungen wurden so Synergieeffekte zwischen den Teilprojekten geschaffen.

Nach einer Einarbeitungsphase entstand zunächst zu Versuchszwecken ein Makromodell der ersten Präparation des Vibrationssensor-Arrays mit Stress-Stiffening Tuning in VHDL-AMS. Anhand der mit diesem Testmodell gewonnenen Erfahrungen konnte TP A1 die Methode der Makromodellierung entsprechend optimieren und anpassen. Für den Einsatz in der Systemsimulation wurde anschließend ein Makromodell des Vibrationssensor-Arrays mit direkter elektrostatischer Abstimmung generiert, das das vom TP A2 erstellte abstrakte analytische Modell ersetzt (s. Abschnitt 2.1.5).

2.1.2.2 Multi-Architecture-Modellierung (MAM)

Im Entwurfsprozeß von Mikrosystemen können digitale, analoge elektrische und nichtelektrische Modelle auf unterschiedlichen Abstraktionsebenen entstehen. Der Abstraktionsgrad eines Modells bestimmt in der Regel den Abstraktionsgrad der Schnittstellen. Dadurch kann es während des Systementwurfs dazu kommen, daß die Schnittstellen der Modelle mit jedem Entwicklungsschritt aufgrund der damit verbundenen Verringerung des Abstraktionsgrades angepaßt werden müssen. Das Ziel dieses Arbeitspaketes ist es, einen methodischen Ansatz zur effizienten Schnittstellengestaltung von Modellen zu finden. Dieser Ansatz soll es ermöglichen, die Schnittstellen eines Modells während des Entwurfsprozesses unverändert zu lassen, womit die zeitaufwendige und fehleranfällige Anpassung der Schnittstellen entfallen kann.

Beim Entwurf digitaler Systeme erlaubt VHDL die Beschreibung mehrerer Architekturen (`architecture`) für eine Schnittstellenbeschreibung (`entity`). Mit der Erweiterung von VHDL zu VHDL-AMS *(VHDL Analog and Mixed Signal)* steht diese Eigenschaft von VHDL auch für die Beschreibung analoger und heterogener Systeme zur Verfügung. In einem Top-Down-basierten Systementwurf entstehen zu Beginn Modelle auf einem hohen Abstraktionsgrad. Der Abstraktionsgrad der Modelle verringert sich mit Fortschreiten des Entwurfsprozesses. Durch die Eigenschaft von VHDL-AMS, für eine Entity mehrere Architekturen beschreiben zu können, ist es nun möglich, Architekturen von Komponenten heterogener Systeme auf unterschiedlichen Abstraktionsebenen für ein Modell zu hinterlegen. Dabei kann bisher das Problem auftreten, daß die Architekturen der Modelle auf den unterschiedlichen Abstraktionsebenen unterschiedliche Schnittstellen benutzen.

Bild 9 verdeutlicht diese Problematik anhand eines Beispiels, bei dem zunächst Modelle auf der Basis funktioneller Blöcke beschrieben werden. Als Interface kommen hier nichtkonservative Knoten (`QUANTITY`) zum Einsatz. Während des Entwurfsprozesses wird die rechte Komponente zu einer Netzwerkbeschreibung weiterentwickelt, die linke dagegen nicht. Die weiterentwickelte (verfeinerte) Komponente benutzt nun als Interface konservative Knoten (`TERMINAL`) während die unveränderte Komponente nach wie vor `QUANTITIES` benutzt. Beide Modelle können dadurch nicht mehr miteinander verbunden werden.

In den folgenden Abschnitten wird daher ein neuer methodischer Ansatz – das „Multi-Architecture-Modeling“ (MAM) – vorgestellt. Wird dieser Ansatz bereits beim Entwurf der ersten abstrakten Modelle im Rahmen eines Top-Down Designs berücksichtigt, so können die später im Entwurfsprozeß entstehenden verfeinerten Modelle problemlos an Stelle der abstrakten Modelle eingesetzt werden. Dabei soll dieser Ansatz leicht anwendbar sein und nur einen geringen Mehraufwand bei der Modellierung hervorrufen.

Digitale Schnittstellen

Für digitale Schnittstellen finden normalerweise `SIGNALS` Verwendung. Der Abstraktionsgrad bestimmt in der Regel den verwendeten Datentyp. So können z. B. auf funktioneller oder algorithmischer Ebene Datentypen wie `integer` oder `real` verwendet werden, im Gegensatz dazu sind auf RTL-Ebene Vektoren der Datentypen `bit` oder `std_logic` üblich.

Für die MAM sind nun zwei Ansätze denkbar: Entweder man verwendet die Schnittstelle, die das Modell auf hohem Abstraktionsniveau besitzt auch für die weiteren Modelle auf niedrigem Abstraktionsniveau, oder man benutzt die Schnittstelle des detaillierteren Modells bereits auf hoher Abstraktionsebene. Dabei hat sich der erste Ansatz als nicht praktikabel erwiesen. Er hätte zur Folge, daß innerhalb eines Modells auf RTL-Ebene Werte aus einer mehrwertigen Logik (z. B. 'X' oder 'Z') entstehen, die über ein Interface vom Typ `integer` oder `real` nicht zum nächsten Modell transportiert werden können.

Dagegen ist der zweite Ansatz anwendbar und bedeutet für digitale Schnittstellen die Verwendung der Datentypen `bit_vector` oder eines Vektors einer mehrwertigen Logik bereits im ersten abstrakten Modell. Gegenüber der Verwendung der Datentypen `integer` oder `real` erzeugt dies nur einen sehr geringen Mehraufwand. Eine evtl. notwendige Konvertierung der Datentypen zu `integer` oder `real` kann u. U. zusätzliche Delta-Zyklen im Simulator verursachen. Zustände wie 'X', 'Z' usw. werden außerhalb des Modells korrekt propagiert, im Modell kann die Handhabung entsprechend der Anforderungen an das Modell frei implementiert werden.

Analoge elektrische und nichtelektrische Schnittstellen

Abstrakte analoge Komponenten werden oft als funktionelle Blöcke beschrieben. Als Interface-Objekt kommen dabei in VHDL-AMS sog. `QUANTITIES` zum Einsatz. Diese repräsentieren einen zeit- und wertkontinuierlichen Informationsverlauf. Im Gegensatz dazu benutzt ein Modell auf niedrigerem Abstraktionsgrad `TERMINALS` als Interface. `TERMINALS` enthalten eine `ACROSS` und eine `THROUGH QUANTITY` (z. B. Spannung und Strom) und erfüllen die kirchhoffschen Gesetze. Da man `TERMINALS` nicht direkt mit `QUANTITIES` verbinden kann, müssen bislang während des Entwurfsprozesses die Schnittstellen der Modelle geändert werden, wenn z. B. funktionelle Blöcke mit Netzwerken verbunden werden sollen.

Wie schon bei den digitalen Schnittstellen hat es sich auch hier als ungünstig erwiesen, die Schnittstelle, die auf hohem Abstraktionsniveau verwendet wird, auch in dem Modell auf niedrigem Abstraktionsniveau zu benutzen. Der Einsatz von `QUANTITIES` auf niedrigem Abstraktionsniveau hätte zur Folge, daß `ACROSS` und `THROUGH QUANTITY` separat im Interface angelegt und die kirchhoffschen Knoten- und Maschenregeln explizit in jedem Modell beschrieben werden müßten. Die Verwendung von `TERMINALS` für analoge Schnittstellen auf allen Abstraktionsebenen dagegen ist möglich.

Bild 10 zeigt anhand eines Beispiels die Anwendung der MAM beim Entwurf analoger Schnittstellen. Die z. B. als funktioneller Block beschriebenen Modelle erhalten als Interface im Gegensatz zu Bild 9 keine `QUANTITY` sondern gleich ein `TERMINAL`. Die Anpassung innerhalb des Modells an dieses Interface besteht lediglich aus der Definition von Branch-`QUANTITIES` (`ACROSS` und `THROUGH QUANTITY`). Verfeinert man nun (wie in Bild 9) eine Komponente, die andere aber nicht, so lassen sich beide Modelle immer noch problemlos miteinander verbinden. Eine zu übertragende `QUANTITY` wird dabei als ideale Spannungs- oder Stromquelle zwischen dem `TERMINAL` und dem Masseknoten `electrical_ground` beschrieben. Die Empfängerkomponente ermittelt die übertragene `QUANTITY` aus der Spannung bzw. dem Strom zwischen

dem TERMINAL und electrical_ground. Dieser Ansatz verursacht nur einen geringen Mehraufwand in der Modellierung und ist leicht anzuwenden.

Digital/analoge Schnittstellen

Im Rahmen einer Variantendiskussion (z. B. Ausführung eines Filters analog, digital oder mechanisch) oder in zeitkritischen digitalen Systemen kann es erforderlich sein, daß Schnittstellen auf hohem Abstraktionsniveau digital, auf niedrigem Niveau aber analog beschrieben werden müssen. Da digitale Schnittstellen SIGNALS, analoge Schnittstellen jedoch TERMINALS benutzen, muß ein gemeinsames Interface-Objekt gefunden werden, das die Eigenschaften beider Schnittstellenobjekte nachbilden kann. Da es nicht möglich ist, die Eigenschaften von TERMINALS auf SIGNALS abzubilden, mußte auch hier ein TERMINAL als gemeinsames Objekt zum Einsatz kommen.

Für die Verbindung zwischen SIGNALS und TERMINALS finden spezielle D/A und A/D-Wandler Verwendung. Diese ermöglichen das Abbilden der Zustände einer mehrwertigen Logik als Spannung auf ein TERMINAL. Da der analoge Wert auf dem TERMINAL den digitale Zustand genau eines SIGNALS repräsentiert, treten keine Fehler in der Wandlung auf (z. B. Quantisierungsfehler). Eine geeignete Beschreibung der D/A und A/D-Wandler erlaubt außerdem die Rückgewinnung der events auf den digitalen Signalen. Die Parameter R_S, R_W, R_I, U_X, U_1, U_0, U_Z, U_{i1} und U_{i0} (s. Bild 11) der D/A und A/D-Wandler kann man dabei auf technologiebezogene oder auf abstrakte Werte setzten, so daß sich entweder ein realitätsnahes Verhalten ergibt oder die *VHDL resolution function* z. B. für den Datentyp std_logic nachgebildet werden kann. Für die Parameter der D/A und A/D-Wandler gelten folgende Beziehungen:
n = maximale Anzahl von Treibern für ein TERMINAL

Parameter	technologieabhängig	technologieunabhängig
U_{i1}: minimale Eingangsspannung, so daß der Zustand '1' erkannt wird, wenn kein Treiber den Zustand 'X' oder '0' hat. U_{i0}: maximale Eingangsspannung, so daß der Zustand '0' erkannt wird, wenn kein Treiber den Zustand 'X' oder '1' hat	U_{i1} and U_{i0} sind durch die Technologie vorgegeben	$\frac{U_x + U_1 \cdot (n-1)}{n} < U_{i1} < U_1$ $\frac{U_x + U_0 \cdot (n-1)}{n} > U_{i0} > U_0$
R_s: Ausgangswiderstand für starke Treiber, R_w: Ausgangswiderstand für schwache Treiber, R_l: Eingangswiderstand	R_s, R_w und R_l sind durch die Technologie vorgegeben $0 < R_s < R_w \ll R_l$	$0 < R_s < R_w \ll R_l$ $R_w \geq max\left[(n-1) \cdot R_s \cdot \frac{U_0 - U_{i1}}{U_{i1} - U_1}, (n-1) \cdot R_s \cdot \frac{U_1 - U_{i0}}{U_{i0} - U_0}\right]$
Maximales Verhältnis von Treibern mit den Zuständen 'W', 'L' oder 'H' zu Treibern mit den Zuständen '1' oder '0', bei dem am Eingang die Zustände '1' und '0' wiedererkannt werden.	$\frac{N_{w,l,h}}{N_{1,0}} = \frac{R_w}{R_s} \cdot \frac{U_{i1,0} - U_{1,0}}{U_{x,0,1} - U_{i_{1,0}}}$	durch eine *resolution function* vorgegeben

TABELLE 1. Konfigurationsmöglichkeiten der A/D- und D/A Wandler für die MAM

2.1.3 Kosten-Modellierung

Das Ziel dieses Arbeitspaketes ist die Erstellung eines methodischen Ansatzes zur Modellierung und Auswertung von Kostenparametern in VHDL/VHDL-AMS Modellen. Unter Kosten sind hier nicht nur fiskalische Kosten wie Anschaffungs- und Entwicklungskosten, sondern auch Kenngrößen wie Platzbedarf, Leistungsaufnahme, Testbarkeit oder Fehlersicherheit zu verstehen. Solche Kostenparameter stellen neben funktionalen Daten wichtige Kenngrößen eines Systems dar, die im Rahmen eines Systementwurfes zu optimieren sind. Bisher werden dazu vorrangig Tools verwendet, die die Kostenparameter parallel zum VHDL-Modell erfassen und vom VHDL-Modell getrennt auswerten und optimieren. Ziel des hier vorgestellten Ansatzes ist es, diese Kostenparameter zu einem Bestandteil des Verhaltens des Modelles werden zu lassen und gleichzeitig mit der Simulation auszuwerten und zu optimieren.

Um diese Kostenparameter zu einem Bestandteil des Modells zu machen, müssen zuerst geeignete Datentypen geschaffen werden, die die Informationen aufnehmen können.

```
type criterion is (develop_cost, product_cost, space, power,
                   testability, error_safety);

type criterion_val is record
                        val:real;
                        limit:real;
                   end record;

type criterion_vect is array (criterion) of criterion_val;
```

Der Typ `criterion` ist eine Aufzählung von Kostenparametern, die optimiert werden sollen. Der Inhalt dieses Typs kann dabei auf die im jeweiligen Entwurf erforderlichen Parameter angepaßt werden. Der Typ `criterion_val` ermöglicht die Angabe eines Wertes für einen Kostenparameter und die Definition einer Grenze für diesen Wert. Wird für die Grenze der Wert –1 eingesetzt, so wird bei späteren Berechnungen diese Kostengrenze ignoriert. Der Typ `criterion_vect` ist ein Vektor, der alle Kostenparameter und Kostengrenzen einer Komponente zusammenfaßt. Dieser Datentyp kann nun auf Konstanten (`CONSTANT`) oder Signale (`SIGNAL`) Anwendung finden. Das Verwenden von Konstanten hat den Vorteil, daß die Simulations-Berechnung zur Kostenmodellierung nur zu Beginn der Simulation einmal ausgeführt und die Simulationsgeschwindigkeit dadurch nicht beeinträchtigt wird. Dafür sind die Kostenfaktoren zur Laufzeit nicht änderbar. Ist dies erforderlich, so kann der Datentyp `criterion_vect` auf ein Signal angewandt werden, dessen Wert sich simulationsabhängig ändert. Ergibt sich der Kostenparameter aus einer `QUANTITY` als Ergebnis der Berechnung eines *simultaneous statements* (Differentialgleichung), so ist dafür Sorge zu tragen, daß diese `QUANTITY` bei der Zuweisung auf das `SIGNAL` auch ein *event* auslöst (z. B. mittels des Attributs `'above`). Durch die Verwendung eines Signals kann sich allerdings, je nachdem wie oft sich die Werte darauf ändern, die Simulationsgeschwindigkeit verringern.

Im nächsten Schritt müssen die Kostenvektoren der einzelnen Komponenten im übergeordneten Strukturmodell zu einem neuen Kostenvektor, der die Kosten z. B. einer Baugruppe repräsentiert, zusammengefaßt werden.

$$K_{j_{val,\,limit}} = \sum_{i=1}^{n} K_{i,\,j_{val,\,limit}} \qquad \forall j(criterion(j)) \tag{17}$$

Dabei muß die Anzahl der Komponenten *n* dieser Struktur variabel sein und darf nicht durch die Funktion, die die Kostenvektoren zusammenfassen soll, eingeschränkt werden. Normale Funktionen haben aber eine feste Anzahl von Parametern und sind somit nicht zur Zusammenfassung einer variablen Anzahl von Kostenvektoren geeignet. Als Lösung bieten sich hier die VHDL *resolution functions* an. Diese Funktionen dienen dazu, die Konflikte, die entstehen wenn mehrere Treiber auf ein *resolved signal* schreiben, aufzulösen.

Die Kostenvektoren der einzelnen Komponenten werden über Signal-Ports mit dem Richtungsattribut `out` in die nächsthöhere Hierarchieebene übertragen und dort mit einem *resolved* Signal vom Typ `criterion_vect` verbunden. Dadurch existieren mehrere Treiber auf dieses Signal, die einen Konflikt auslösen, der dann durch eine geeignete *resolution function* aufgelöst werden kann. Diese spezielle *resolution function* für die Zusammenfassung der einzelnen Kostenvektoren addiert im einfachsten Fall Kostenwert und Kostengrenze für alle Kostenfaktoren. Ist die Kostengrenze eines Kostenfaktors in einer Komponenten auf –1 gesetzt, so wird diese Grenze auch im resultierenden Kostenvektor auf –1 gesetzt. Die *resolution function* ermittelt ebenfalls, ob ein Kostenparameter einer Komponente seine Kostengrenze übersteigt und gibt ggfs. eine Warnung aus.

Darüber hinaus wurden weitere Funktionen geschaffen, mit deren Hilfe der von der *resolution function* erzeuge Kostenvektor bearbeitet werden kann. Mit diesen Funktionen ist es möglich, die Kostenwerte zu ändern bzw. die Kostengrenzen neu festzulegen.

Die Optimierung der Kostenparameter erfordert es im allgemeinen, aus den Kostenvektoren ein skalares Gütemaß *S* zu bestimmen.

$$S = \sum_{\forall j(criterion(j))} \frac{K_{j_{val}}}{K_{j_{limit}}} \cdot W_j \tag{18}$$

Dazu existiert im Rahmen der Methode der Kosten-Modellierung ebenfalls eine Funktion, die auf der Basis der Kostenparameter K_{val}, der Kostengrenzen K_{limit} und eines Wichtungsvektors W einen skalaren Wert ermittelt, der ein Gütemaß für die Einhaltung der Kostengrenzen darstellt.

Der hier aufgeführte methodischen Ansatz ermöglicht die interaktive Optimierung der Kostenparameter während der Simulation des Systems. Ein automatisierts Arbeiten mit Werkzeugen zur Verhaltensoptimierung wird ebenfalls möglich.

2.1.4 Constraint-Modellierung

Für die Wiederverwendung (Reuse) von parametrisierbaren Komponentenmodellen ist es notwendig, physikalische Grenzen (Constraints) zu überwachen. Dafür bietet VHDL *assert statements* an. Diese Statements sind allerdings nur wenig flexibel einsetzbar, da sie nur einen Parameter auf einen bestimmten Wert überprüfen. Mit Hilfe von Ansätzen des aus der Informatik bekannten *constraint programming*, soll eine Transformation von Constraints über Hierarchie- und Abstraktionsebenen hinweg möglich werden. Da die Constraints des *constraint programming* ähnliche Eigenschaften (Multidirektionalität, Kontrollstrukturenfreiheit, Eventorientierung) wie die *concurrent* und *simultaneous statements* von VHDL/VHDL-AMS aufweisen, lag es nahe, diese Ansätze nach VHDL-AMS zu übertragen.

Der erste Schritt bei der Erstellung der Methode der Constraint-Modellierung ist die Aufstellung der einzuhaltenden Grenzen im Komponentenmodell. Für das in Abschnitt 2.1.5 betrachtete

Vibrationssensor-Array kommen dabei z. B. minimale und maximale Strukturgrößen für Federn und Massen, minimale und maximale Chipflächen oder maximale Tuningspannungen in Frage. Als Anforderungen (Requirements) an diese Komponente können von einer höheren Hierarchie- oder Abstraktionsebene Größen wie minimale und maximale Frequenzen, Empfindlichkeit und maximale Beschleunigung etc. stehen. Das Modell muß nun über geeignete Transformationen verfügen, die die Constraints auf die Requirements abbilden. Dabei hat sich gezeigt, daß die digitalen *concurrent statements* dazu nicht in der Lage sind, da diese Transformationsvorschriften nur selten als Zuweisung formuliert werden können. Die Transformationsvorschriften ergeben einen Satz zu lösender Gleichungen, wofür analoge *simultaneous statements* erforderlich sind. Daraus ergibt sich jedoch einen entscheidenden Nachteil: *Simultaneous statements* (Differentialgleichungen) werden Bestandteil des Gleichungssystems, das der Simulator zu jedem Rechenschritt löst. Dies verringert die Simulationsgeschwindigkeit des Modells und kann das Konvergenzverhalten des Simulators negativ beeinflussen. Den Einfluß auf die Simulationsgeschwindigkeit kann man dadurch umgehen, indem man die Transformationsgleichungen mittels eines *simultaneous-if-statements* nur zum Zeitpunkt der Berechnung des Arbeitspunktes (`quiescent_domain`) durchführen läßt. Der Einfluß auf die Simulationsstabilität bleibt allerdings erhalten, was dazu führt, daß der Simulator unter Umständen keinen Arbeitspunkt bestimmen kann und die Simulation schon zu Beginn abbricht.

Ein weiteres Problem tritt dann auf, wenn die Transformationsvorschriften nicht als Gleichung sondern als Ungleichung zu beschreiben sind. Mit VHDL/VHDL-AMS existiert keine Möglichkeit Ungleichungen abzubilden, oder Gleichungssysteme die Ungleichungen enthalten zu lösen. Dies schränkt die Anwendungsmöglichkeiten des *constraint programming* in der Modellbildung mit VHDL weiter stark ein.

Zu diesen simulationstechnischen Problemen kommt noch ein entwurfstechnisches hinzu. Die Transformationsvorschriften können ein Teil des Verhaltens des Modells selbst sein. Ist dies nicht der Fall, müssen sie vom Entwerfer manuell hinzugefügt werden. Eine Automatisierung oder eine Wiederverwendung aus Bibliotheken erscheint aufgrund der Mannigfaltigkeit der möglichen Kombinationen aus Constraints und Requirements schwierig. Das hat für den Entwerfer einen erheblichen Mehraufwand zur Folge, so daß die Notwendigkeit dieses methodischen Ansatzes im Verhältnis zum Arbeitsaufwand nur schwer vermittelbar sein wird.

Die in diesem Arbeitspaket aufgezeigten Probleme lassen eine weitere Bearbeitung der Constraint Modellierung deshalb als wenig erfolgsversprechend erscheinen.

2.1.5 Systementwurf für den Demonstrator „Vibrationssensor-Array“

2.1.5.1 Systemspezifikation

Für die Aufstellung einer Anforderungsspezifikation für den Demonstrator wurden neben einer Literaturrecherche [39] auch selbst Messungen zum Vibrationsverhalten einer Drehmaschine vorgenommen. Dafür kamen handelsübliche, piezokeramische Beschleunigungssensoren zum Einsatz. Diese wurden so an einem Drehmeißel angebracht, daß Schwingungen in horizontaler und vertikaler Richtung aufgenommen werden konnten. Die gemessenen zeitlichen Signalverläufe wurden anschließend mittels FFT in ein Frequenzspektrum überführt. Dieses Spektrum ergab eine deutliche Verschiebung der Resonanzfrequenzen bei einem sich abstumpfenden Meißel. Aus diesen Messungen konnte die Anforderung abgeleitet werden, daß der Sensor einen Frequenzbereich von 2 kHz bis 5 kHz zu überwachen hat. Die sich anschließende Signalverarbeitung muß die

Maxima der Schwingungsamplituden in diesem Frequenzbereich bestimmen, wobei die Größe des Maximums kein signifikantes Merkmal ist. Der Klassifikator muß anschließend die Lage des Maximums zusammen mit Informationen zu Drehzahl, Vorschub, Abhub und Materialparametern beachten, um eine Entscheidung treffen zu können. Bei den Messungen wurde ebenfalls festgestellt, daß sich die auftretenden Schwingungen aus den Eigenfrequenzen des Meißels bestimmen, so daß der Klassifikator für den jeweils verwendeten Meißel angelernt werden muß.

2.1.5.2 Systempartitionierung

Die Systempartitionierung dient der Unterteilung des Gesamtsystems in kleinere Teilsysteme. Die Modelle der Teilsysteme sind kleiner und übersichtlicher und daher mit weniger Rechenaufwand simulier- und validierbar. Der erste Schritt der Partitionierung trennt die einzelnen Baugruppen des Demonstrators (siehe Bild 13) in ihre – für die Simulation relevanten – Bestandteile. Der zweite Partitionierungsschritt trennt die im Rahmen der MAM zur Verfügung stehenden Architekturalternativen.

Ausgehend vom im Finanzierungsantrag vorgestellten Systemkonzept und der Systemspezifikation wurde das System „Demonstrator“ wie folgt partitioniert:

- Umgebungsmodell „Virtuelle Werkzeugmaschine“
  - Transient-Simulation aus Waveform-Daten
  - Transient-Simulation aus Spektral-Daten
  - Wechselstrom(AC)-Simulation aus Spektral-Daten
- Sensor
  - abstraktes Modell mit linearem Ansatz
  - vom TP A1 entwickeltes Makromodell
- Analog-Signalverarbeitung
  - Modell mit funktionellen Blöcken
  - analoge Schaltung
- Mikrocontroller
  - algorithmisches Modell
- Fuzzy-Pattern-Klassifikation
  - algorithmisches Modell
  - taktgenaues, synthetisierbares Modell

2.1.5.3 Systemmodellierung und -simulation

Umgebungsmodell „Virtuelle Werkzeugmaschine“

Die „virtuelle Werkzeugmaschine“ ist ein im TP A2 geschaffenes VHDL-AMS Modell, das die Umgebung des Sensors simuliert. Das Modell stellt als Stimuli für den Sensor Daten der Vibration einer Maschine und der Temperatur des Werkzeuges bereit. Das Verhalten der „virtuellen Werkzeugmaschine“ ist hochgradig parametrisierbar. Um das Modell dennoch übersichtlich zu halten, werden die Parameter nicht als `GENERIC` übergeben, sondern in einem `PACKAGE` und `PACKAGE BODY` beschrieben. Ändern sich die Parametersätze, so kann man entweder den `PACKAGE BODY` modifizieren, oder ein anderes `PACKAGE` einbinden. Dieses Verfahren hat sich schon im Antragszeitraum 1997-2000 bei der Simulation des Demonstrators „Rundumprojektion“ bewährt.

Das Umgebungsmodell ist in der Lage, sowohl spektrale Vibrationsdaten als auch Vibrationsdaten als Waveform zu verarbeiten und für eine Transient- oder AC-Simulation als Stimuli zur Verfügung zu stellen. Das Einlesen der Daten erfolgt derzeit noch über die `PACKAGES` mit den Maschinendaten, da in den zur Verfügung stehenden VHDL-AMS Simulatoren das Einlesen von Dateien noch nicht implementiert ist. Wenn ein Simulator zur Verfügung steht, der diese Funktion beherrscht, kann hier das Modell noch flexibler und weniger speicherintensiv gestaltet werden.

Für den Einsatz der virtuellen Werkzeugmaschine ergeben sich 4 Einsatzfälle:

a) Erzeugung von Stimuli für Transient-Simulation aus Waveform-Daten
   Hierbei werden aus einem mit `t_sample` abgetasteten Amplitudenverlauf im Intervall von `t_sample` Daten entnommen und einer internen, mechanischen Quelle als Beschleunigungs- oder Weg-Information zugeordnet.

b) Erzeugung von Stimuli für Transient-Simulation aus Spektral-Daten
   In diesem Einsatzfall werden eine variable Anzahl *n* relevanter Amplituden- und Phasenwerte aus einem Spektrum entnommen und damit *n* mechanische Sinus-Quellen gesteuert. Die Ausgangssignale der Quellen werden anschließend addiert.

c) Erzeugung von Stimuli für Wechselspannungs(AC)-Simulation aus Spektral-Daten
   Die Erzeugung der Stimuli für diesen Einsatzfall erfolgt ähnlich Fall b). Das Hauptproblem hierbei lag allerdings in der im VHDL-AMS Simulator nur mangelhaft implementierten Möglichkeiten zur Beschreibung von Wechselspannungs-Quellen. Die Lösung besteht darin, für eine Anzahl *n* relevanter Amplitudenwerte zunächst *n* breitbandige mechanische Wechselspannungs-Quellen mit den entsprechenden Amplituden zu erzeugen, deren Ausgangssignal von einem idealen Bandpaß auf die gewünschte Frequenz beschnitten wird.

d) Die Erzeugung von Stimuli für eine AC-Simulation aus Waveform-Daten erfordert eine Umwandlung von Daten im Zeitbereich in den Frequenzbereich mittels FFT. Da eine Implementation in VHDL die Systemsimulation unnötigerweise verlangsamen würde, werden in diesem Fall Daten im Zeitbereich mittels eines kleinen MATLAB-Modells und der dort eingebauten FFT in den Frequenzbereich abgebildet. Anschließend können die Stimuli wie in Fall c) generiert werden.

Die Einsatzfälle a) bis c) wurden ursprünglich in einem einheitlichen, universellen Modell beschrieben. Später wurde das Modell unter Anwendung der MAM in drei unterschiedliche

Architekturen bei einheitlicher Schnittstelle getrennt, wodurch sich für das Umgebungsmodell die Rechenzeit etwa halbierte.

Die Simulation einer Erwärmung der Maschine wird über einen `GENERIC` zu- oder abgeschaltet. Die Erwärmung erfolgt linear oder exponentiell von einer Start-Temperatur auf eine End-Temperatur mit einer bestimmten Zeitkonstante. Bei Bedarf kann die Simulation auch um die Erzeugung eines bestimmten Temperaturprofils aus gemessenen Daten erweitert werden.

Ein weiteres grundlegendes Problem bei der Modellierung der virtuellen Werkzeugmaschine betrifft die Wahl der charakteristischen Zustands (ACROSS)- und Fluß (THROUGH)-Größe für die mechanische Schwingung. Betrachtet man eine Analogiebeziehung unter SPICE, so muß als Zustandsgröße die Geschwindigkeit auf eine Spannung und als Flußgröße die Kraft auf einen Strom abgebildet werden. Im Gegensatz dazu kann man unter VHDL-AMS das mechanische Verhalten direkt, ohne Analogiebeziehung beschreiben. Dabei ist man auch nicht auf Kraft und Geschwindigkeit zur Beschreibung eines mechanischen Systems angewiesen. Prinzipiell sind die drei folgenden Fälle sinnvoll:

ACROSS-Größe	THROUGH-Größe
Weg s	Kraft F
Geschwindigkeit v	Kraft F
Beschleunigung a	Kraft F

TABELLE 2. Möglichkeiten zur Beschreibung translatorischer Bewegung

Selbst für eine Beschreibung von Weg, Geschwindigkeit oder Beschleunigung als THROUGH-Größe und Kraft als ACROSS-Größe lassen sich gültige und sinnvolle Beziehungen aufstellen.

Eine Messung von Vibrationen erfolgt üblicherweise in der Form von gemessenen Beschleunigungen, so daß es nahe lag, den mechanischen Teil des Umgebungsmodells auf der Basis von Beschleunigung und Kraft zu modellieren. Dies ist auch problemlos möglich, solange man nicht die aus der Beschleunigung resultierende Geschwindigkeit oder den Weg benötigt. Bekanntlich gilt:

$$v = \int a\,dt + C_0 \quad \text{und} \quad s = \int v\,dt + C_1 \tag{19}$$

Das Problem sind nun die Integrationskonstanten C_0 und C_1. Diese können bei der Anwendung des `'integ` Attributes von VHDL-AMS nicht explizit gesetzt werden. Dies hat z. B. zur Folge:

$$a = A \cdot \sin(\omega t), \quad v = A \cdot \int \sin(\omega t) + C_0 = -\frac{A}{\omega} \cdot \cos(\omega t) + C_0 \tag{20}$$

Der Simulator setzt nun:

$$v(0) = -\frac{A}{\omega} \cdot \cos(0) + C_0, \quad C_0 = \frac{A}{\omega} \tag{21}$$

Damit kommt es in der Simulation zu einem in der Realität nicht vorhandenen Geschwindigkeitsoffset, der bei der Berechnung des Ortes mit integriert wird. Bildlich gesprochen würde der Sensor eine gerichtete Bewegung ausführen und dabei die Werkzeugmaschine hinter sich herziehen.

Die Lösung dieses Problems besteht darin, als mechanische Größen Weg und Kraft zu verwenden. Benötigt man die Geschwindigkeit oder die Beschleunigung, so lassen sich diese Größen jederzeit durch Ableitungen nach der Zeit eindeutig aus dem Weg bestimmen. Die als Beschleunigungswerte vorliegenden Vibrationsmeßwerte müssen jetzt vor Beginn der Simulation in Positionswerte umgerechnet werden. Bei Vorlage der Meßwerte als Spektraldaten kann dies analytisch erfolgen, bei vorliegenden Wave-Daten muß man im Bedarfsfall die Integrationskonstanten so wählen, daß kein Geschwindigkeitsoffset entsteht.

Neben diesen vom TP A2 entwickelten VHDL-AMS Modellen entstand für den Drehmeißel im TP A1 ein FEM-Modell und im TP A6 ein Modell auf der Basis der Mehrkörpersimulation. Anhand der aus diesen Modellen gewonnenen Informationen konnten einerseits die Funktion der VHDL-AMS Modelle, zum anderen die im Rahmen der Systemspezifikation gewonnenen Erkenntnisse verifiziert werden.

Sensor

Für das Sensor-Array ist der Einsatz eines Makromodells vorgesehen. Um erste Simulationen durchführen zu können, wird jedoch zunächst ein einfacheres, auf analytisch erfaßbaren Zusammenhängen beruhendes Modell benötigt. Der geplante Aufbau des Sensor-Arrays eliminiert weitgehend Nichtlinearitäten im Verhalten, wodurch für die Modellierung ein linearer Ansatz gewählt werden konnte.

Jedes Sensorelement stellt einen gedämpften Feder-Masse-Schwinger dar. Zur Beschreibung wurde eine Bibliothek mit mechanischen Grundelementen für die drei möglichen Beschreibungsformen der translatorischen Bewegung (s. TABELLE 2.) erstellt. Dafür gelten die folgenden Gleichungen und Analogien:

Domäne	Feder	Dämpfer	Masse
a, F	$F = k \cdot \iint a\,dt\,dt$	$F = c \cdot \int a\,dt$ Spule	$F = m \cdot a$ Widerstand
v, F	$F = k \cdot \int v\,dt$ Spule	$F = c \cdot v$ Widerstand	$F = m \cdot \frac{dv}{dt}$ Kondensator
s, F	$F = k \cdot s$ Widerstand	$F = c \cdot \frac{ds}{dt}$ Kondensator	$F = m \cdot \frac{d^2 s}{dt^2}$

TABELLE 3. Analogien zwischen mechanischen und elektrischen Elementen

Aus dieser Tabelle ist ersichtlich, daß mit einer SPICE-Analogie nur eine Beschreibung mittels Kraft und Geschwindigkeit möglich ist, wogegen mit VHDL-AMS das mechanische Verhalten mit Beschleunigung, Geschwindigkeit oder Verschiebung beschreibbar ist. Aufgrund der im vorigen Abschnitt zur „virtuellen Werkzeugmaschine dargelegten Probleme besteht das Modell des

Sensors aus einer Kopplung mechanischer Federn, Massen und Dämpfern mit den charakteristischen Größen Weg und Kraft.

Um die Zellen des Sensor-Arrays auf eine bestimmte Frequenz trimmen zu können, muß die Federkonstante der Resonatoren änderbar sein; dafür kommen zwei Wirkprinzipien zum Einsatz:

Tuning mittels Stress-Stiffening-Effekt

Die erste Variante für die Beeinflussung der Federkonstante ist der Stress-Stiffening-Effekt. Bild 3 zeigt das Funktionsprinzip eines solchen Resonators.

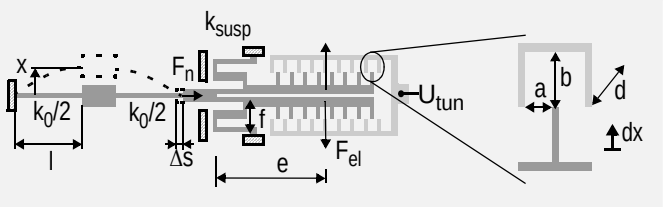


Bild 3 extrahiertes Detail eines Resonators mit Stress-Stiffening-Tuning

Unter der Annahme eines homogenen Feldes in der Kammstruktur gilt:

$$F_{el} = \frac{dW}{dx}, \quad dW = \frac{1}{2} \cdot U_{tun}^{\ 2} \cdot dC \tag{22}$$

$$C = 2n \cdot \varepsilon \cdot \frac{x_{ss} \cdot d}{a}, \quad \frac{dC}{dx} = 2n \cdot \varepsilon \cdot \frac{d}{a} \tag{23}$$

$$F_{el} = U_{tun}^{\ 2} \cdot \frac{n \cdot \varepsilon \cdot d}{a} \tag{24}$$

wobei n die Anzahl der Finger eines Kammes ist. Die Kraft F_{el} erzeugt über einen Hebelmechanismus (e, f) die Normalkraft F_n, die die Federkonstante durch den Stress-Stiffening-Effekt beeinflußt.

$$F_n = 2 \cdot \frac{e}{f} \cdot F_{el} + \frac{6 \cdot k_{susp}}{5 \cdot l} \cdot x^2, \qquad k = k_0 + \frac{12 \cdot F_n}{5 \cdot l} \tag{25}$$

Tuning mittels einer elektrostatischen Feder

Das Tuning mittels Stress-Stiffening-Effekt hat zwei Nachteile. Die Herstellung dieser Struktur gestaltet sich wegen Schichtspannungen schwierig, und es treten funktionsbedingt Nichtlinearitäten auf (Gleichung (25)). Deshalb wurde ein Ansatz für das Resonanzfrequenz-Tuning gewählt, bei dem eine Kurvenkammstruktur eine Anziehungskraft erzeugt, die linear zur Auslenkung ist (für Details s. TP A4).

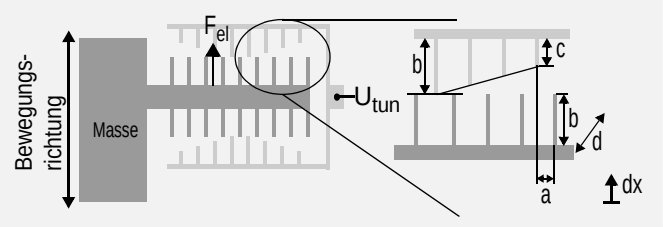


Bild 4 extrahiertes Detail eines Resonators mit Tuning mittels elektrostatischer Feder

Unter der Annahme eines homogenen Feldes in der Kammstruktur gilt:

$$F_{el} = \frac{1}{2} \cdot U_{tun}^{\ 2} \cdot \frac{dC}{dx} \tag{26}$$

Im Bereich $-(b-c) \le x \le (b-c)$ gilt:

$$C = 2 \cdot \varepsilon \cdot \frac{d \cdot x}{a} \cdot \frac{x}{b-c} \cdot n \cdot \frac{1}{2} \tag{27}$$

$$C = \varepsilon \cdot \frac{d \cdot x^2}{a} \cdot \frac{n}{b-c}, \quad \frac{dC}{dx} = 2 \cdot \varepsilon \cdot \frac{d \cdot x}{a} \cdot \frac{n}{b-c} \tag{28}$$

$$F_{el} = U_{tun}{}^2 \cdot \varepsilon \cdot \frac{d \cdot x}{a} \cdot \frac{n}{b-c} \tag{29}$$

Da F_{el} entgegengesetzt zur Kraft einer normalen Feder wirkt, kann man schreiben:

$$F = k \cdot x, \quad k = -\left(U_{tun}{}^2 \cdot \varepsilon \cdot \frac{d}{a} \cdot \frac{n}{b-c} \right) \tag{30}$$

Für den Bereich $|x| > (b-c)$ gilt:

$$C = 2n \cdot \varepsilon \cdot \frac{(x-(b-c)) \cdot d}{a} + C_0, \quad \frac{dC}{dx} = 2n \cdot \varepsilon \cdot \frac{d}{a} \tag{31}$$

$$F_{el} = U_{tun}{}^2 \cdot n \cdot \varepsilon \cdot \frac{d}{a} \quad \text{für } x > (b-c) \tag{32}$$

$$F_{el} = -U_{tun}{}^2 \cdot n \cdot \varepsilon \cdot \frac{d}{a} \quad \text{für } x < -(b-c) \tag{33}$$

Auslenkungs-Strom-Transducer

Der Auslenkungs-Strom-Transducer für den Sensor mit Stress-Stiffening und für den Sensor mit elektrostatischer Feder besteht ebenfalls aus einer gebogenen Kammstruktur. Für die Systemsimulation wurde dieser bogenförmige Kammverlauf aufgrund der geringen Krümmung vernachlässigt.

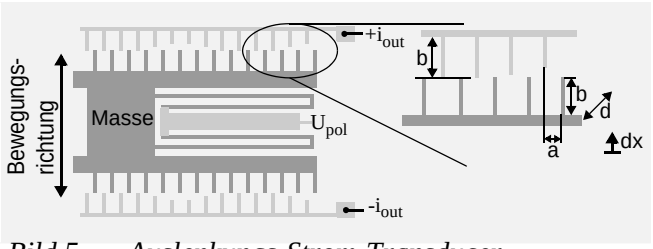


Bild 5 Auslenkungs-Strom-Transducer

Unter der Annahme eines homogenen Feldes in der Kammstruktur gilt:

$$i(t) = U_{pol} \cdot \frac{dC}{dt} \tag{34}$$

$$C = 2n \cdot \varepsilon \cdot \frac{x \cdot d}{a}, \quad \frac{dC}{dt} = 2n \cdot \varepsilon \cdot \frac{d}{a} \cdot \frac{dx}{dt} \tag{35}$$

Für die Umwandlung der mechanischen Bewegung in einen Strom gilt demzufolge:

$$i(t) = U_{pol} \cdot 2n \cdot \varepsilon \cdot \frac{d}{a} \cdot \frac{dx}{dt} \tag{36}$$

Dabei sind: U_{pol} die angelegte Polarisationsspannung; *n, d, a* die geometrischen Abmessungen der stromerzeugenden Kammstruktur; *s* die Auslenkung der Kammstruktur.

Makromodelle

Wie schon im Abschnitt 2.1.2.1 beschrieben, erstellte das TP A1 Makromodelle für das Sensor-Array in VHDL-AMS. Das erste Makromodell entstand aus der FEM-Simulation der ersten Präparation des Sensor-Arrays mit Stress-Stiffening-Tuning und diente zur Evaluierung des Ansatzes

der Makromodellierung. Das zweite Makromodell ist ein Modell des Sensors mit elektrostatischer Feder zum Frequenztuning, wie es im Demonstrator eingesetzt wird. Dieses Modell umfaßt die Feder-Masse-Resonatoren, die Kammstruktur zum Frequenztuning und den Auslenkungs-Strom-Transducer. Das Makromodell weist dabei eine höhere Genauigkeit auf als die auf den vereinfachten analytischen Beziehungen beruhenden Modelle, womit die Genauigkeit der Systemsimulation erhöht werden konnte. Die Anwendung der Multi-Architecture-Modellierung auf den Entwurf des analytischen Modells und des Makromodells ermöglichte das problemlose Ersetzen des analytischen Modells durch das Makromodell.

Analog-Signalverarbeitung

Die analoge Signalverarbeitung besteht aus mehreren Verstärkerstufen, einer Stufe zur Trennung von Real- und Imaginärteil für einen bestimmten Frequenzanteil im Meßsignal (Lock-In-Verstärker) sowie aus einer Stufe zur Betragsbildung. Bei die Modellierung wurde die vorhandene elektrische Schaltung in VHDL-AMS sowohl mit funktionalen Blöcken als auch als elektrische Schaltung unter Anwendung der Multi-Architecture-Modellierung beschrieben. In der zur elektrischen Schaltung verfeinerten Modellvariante erfolgte teilweise eine direkte Übernahme der Schaltung in das Modell oder aus Rechenzeitgründen eine Beschreibung als Verhalten (z. B. bei den OPV). Für ein Filter im Lock-In-Verstärker kam sowohl das Modell der elektrische Schaltung als auch eine Übertragungsfunktion zum Einsatz, wobei die MAM ebenfalls Anwendung fand. Da diese Filterschaltung eine relativ lange Einschwingzeit besitzt, die die notwendige Rechenzeit des Gesamtsystems erheblich verlängert, wurde noch eine dritte Architekturvariante auf der Basis eines Tschebyscheff-Filters 11. Ordnung [32] aus Abschnitt 2.1.1.1 realisiert, der eine kürzere Einschwingzeit besitzt, womit sich die Simulationsdauer halbiert. Wird die Genauigkeit des Original-Filters benötigt, läßt sich dieser jederzeit wieder einsetzen.

Mikrocontroller

Das Modell des Mikrocontrollers besteht aus einem einfachen algorithmischen Ablauf, der die zur Messung eines Amplitudenwertes eines Frequenzanteils notwendigen Signale steuert. Auf die Modellierung weitergehender Funktionen wie z. B. Selbstkalibrierung oder Kommunikation zum PC wurde verzichtet. Die Kalibrierdaten werden aus einer Tabelle entnommen und entsprechen dem idealen Verhalten. Auch der A/D Wandler, der die Ausgangsdaten der analogen Signalverarbeitung der digitalen Signalverarbeitung zur Verfügung stellt, wurde stark abstrahiert. Am Ausgang stellt der Mikrocontroller die gemessenen Daten dem Fuzzy-Pattern-Klassifikator zur Verfügung. Das Modell befindet sich noch in Entwicklung und konnte noch nicht in die Gesamtsystemsimulation integriert werden. (Arbeitspaket 1.4n liegt im Zeitplan)

Fuzzy-Pattern-Klassifikation

Der Fuzzy-Pattern-Klassifikator hat die Aufgabe, zu entscheiden, ob die gemessenen Vibrationen von einem scharfen oder einem verschlissenen Werkzeug verursacht werden. Für den Klassifikator wurde sowohl ein Modell im Rahmen der Systemsimulation erstellt als auch die Hardware als Teil des Demonstrators aufgebaut. Der Aufbau des Demonstrators ist laut Finanzierungsantrag Bestandteil der Arbeiten des TPs A4. Da das Expertenwissen zur Fuzzy-Pattern-Klassifikation aber im TP A2 konzentriert ist, erfolgte der Aufbau des Klassifikators durch A2.

Der Klassifikator beruht auf dem Fuzzy-Pattern-Verfahren nach [40], [18] und bestimmt mit Hilfe folgender Zugehörigkeitsfunktion die Sympathie eines Satzes von Merkmalen zu einer bestimmten angelernten Klasse:

$$y(i) = \frac{1}{1 + \sum_{l=1}^{m} \left(\frac{1}{B_{i,l}} - 1\right) \cdot \left(\frac{\left|x_l - S_{i,l}\right|}{C_{i,l}}\right)^{D_{i,l}}} \tag{37}$$

Dabei ist $S_{i,l}$ der Schwerpunkt eines Merkmals l in der Klasse i. x_l ist das zu bewertende Merkmal, $B_{i,l}$ und $C_{i,l}$ beschreiben die Zugehörigkeit des Merkmales zum Rand der Klasse und $D_{i,l}$ ist der Formfaktor für den Rand der Klasse. Das Resultat ist der Sympathievektor y, der die Sympathie eines Satzes von m Merkmalen zur Klasse i beschreibt. Der Klassifikator arbeitet dabei mit unsymmetrischen Zugehörigkeitsfunktionen, d. h., B, C und D können unterschiedliche Werte annehmen, je nachdem ob x größer oder kleiner als der Klassenschwerpunktes ist. Weiterhin enthält das Verfahren eine Koordinatentransformation der Merkmale bezüglich der Achsen des Merkmalsraumes. Dies ist dann erforderlich, wenn die Lage der Klasse im Merkmalsraum nicht parallel zu einer Merkmalsachse liegt. Der Vorteil der Fuzzy-Pattern-Klassifikation besteht darin, daß die Parameter B, C, D und S sowohl angelernt als auch aus Expertenwissen bestimmt werden können.

Für die Simulation des Klassifikators auf hoher Abstraktionsebene wurde der Fuzzy-Pattern-Algorithmus in VHDL-AMS abgebildet. Für die Hardwareimplementierung des Klassifikators ist diese abstrakte Beschreibung allerdings nicht geeignet, da ein solches Modell nicht synthetiserbar ist. Für die Implementierung des Fuzzy-Klassifikators benötigt man ein Modell auf Register-Transfer-Ebene. Dazu wurde zuerst versucht, den Klassifikator fest in Hardware zu implementieren. Da aber insbesondere die Koordinatentransformation sequentiellen Charakter trägt, erfolgte die Entwicklung eines speziellen Prozessors, dessen Befehlssatz, Ein- und Ausgänge auf die Klassifizierung von Meßdaten nach dem Algorithmus der Fuzzy-Pattern-Klassifikation abgestimmt sind. Einen Überblick über die Systemstruktur gibt Bild 16. Für die Realisierung des Prozessor kam ein VIRTEX E 400 FPGA zum Einsatz. Für den Klassifikator mußte zudem eine eigene Leiterplatte entwickelt werden, die neben dem FPGA diverse RAM, FLASH und I/O-Bausteine trägt. Die Hardware der Komponente „Fuzzy-Pattern- Klassifikator“ funktioniert, und wird entsprechend Zeitplan mit dem restlichen System verbunden.

2.2 Datenmanagement

2.2.1 Erweiterung des Datenschemas

Aufgrund neuer Anforderungen an das Datenmanagementsystem (DMS) kommt es zu Erweiterungen des Datenschemas in der Datenbank. Dazu zählen:

- die Adreß- und Namensverwaltung (Entität Adresse) beispielsweise für das Zuordnen von Angaben zum Verfasser von Dokumenten,
- das Einfügen des Attributes „owner“ in jede Tabelle der Datenbank, damit eine genaue Zuordnung jedes Datensatzes möglich ist. Es wird somit ein erhöhter Zugriffsschutz auf die Daten erreicht (nur der Dateneigentümer darf seine Daten entsprechend der für die Tabelle festgeleg-

ten Zugriffsrechte manipulieren). Eine Tabelle in der Datenbank kann eine Entität oder ein Relationship sein [33].

- das Einfügen des Attributes „privaccess_rwd“ in jede Tabelle, um für den Datensatzeigentümer eine eigene Rechtevergabefunktion zu implementieren,
- die Möglichkeit der Erfassung von Layoutdaten, Wafern und zugeordneten Layoutebenen.

2.2.2 Neue Funktionalitäten in der DMS-Applikation

- verbesserte Gestaltung und Bedienbarkeit der Formulare,
- Aufruf der Formulare über Baumstruktur,
- Grafische Übersichtsanzeige aller aktiver Formulare und deren Verknüpfungen (falls solche existieren),
- E/A-Formulare dienen gleichzeitig zur Eingabe von Suchbegriffen. Eine Kombination mehrerer Suchbegriffe und die Suche nach Teilzeichenketten in einer Tabelle sind möglich.
- Datensortierung durch Anklicken der gewünschten Datenfeldbezeichnung,
- Durch die Verknüpfung von Formularen sind automatisch auch die darin enthaltenen Daten verknüpft (1…n - Beziehung).
- Aktivierte Verknüpfungen von Formularen werden hervorgehoben und geben dadurch einen schnellen Überblick zu den Relationen des aktuellen Datensatzes.
- Bei der Erzeugung von Strukturen (Fertigungseinheiten, Konstruktionselemente, gefertigte Einheiten) wird über eine Baumstrukturansicht immer die aktualisierte Verknüpfung aller Einzelkomponenten angezeigt.
- selbsterklärende Schaltflächen,
- datensensitive Steuerung der Schaltflächen (aktiv/deaktiviert),
- Erweiterung der DMS-Applikation um Rechtevergabe für anwendereigene Datensätze,
- Adressenverwaltung,
- Zugriff auf die Datenbank mittels Web-Browser durch eine Retrievalkomponente mit Volltextrecherchemöglichkeit,
- Eine Erweiterung des Zugriffs über einen Web-Browser unter Nutzung der vollen Funktionalität der Tcl/Tk-Applikation wird getestet.

2.2.3 Abbildung des Entwurfs und der Herstellung mikromechanischer Komponenten im Datenmanagementsystem (DMS)

2.2.3.1 Zielsetzung

Zu realisieren war eine plattformübergreifende, intranetbasierte Verwaltung von Entwurfsdokumenten, so z. B. von Simulationsmodellen, Stimuli und zugehörigen Simulationsergebnissen, Meß- und Charakterisierungsdaten, Dokumentationen usw. sowie deren Metadaten. Das DMS ist eine Client-Server-Lösung auf der Basis eines relationalen Datenbanksystems, wobei sich der Datenbankserver auf einem Linux-PC befindet und der Zugriff der Klienten von Windows 9x/NT oder UNIX über Intranet ermöglicht wird. Mittels einer Retrievalkomponente wird das DMS zu einem intuitiv bedienbaren Informationssystem, mit dem man in der Lage ist, sowohl umfangreiche Suchvorgänge im (Meta-) Datenbestand als auch Volltextrecherchen mit unscharfen Suchbegriffen in den Dokumenten durchzuführen. Weiterhin soll die Übertragung der aus dem Entwurf digitaler Systeme stammenden methodischen Ansätze zur Wiederverwendung von Komponenten auf die Mikrosystemtechnik unterstützt werden. Das DMS wird deshalb zu einem projektübergreifenden Modell-Repository für Komponenten und Systeme der Mikrosystemtechnik mit Komponentenretrieval weiterentwickelt. Für die einfache Suche im Datenbestand dienen die I/O-Formularfenster der Applikation, d. h., durch die Eingabe beliebiger Suchbegriffe in die Datenfelder können passende Teildatenmengen selektiert werden. Unter Kenntnis der Datenbankabfragesprache SQL (Structure Query Language) und des Datenschemas des DMS sind über eine vorhandene SQL-Schnittstelle ebenfalls Daten aus der Datenbank abrufbar. Ein weiterer Schwerpunkt ist die Integration des experimentellen Werkzeuges zur Spezifikationserfassung und Entwurfsdokumentation ASPECTOR [17] in das DMS.

Das der Datenbank zugrunde liegende Datenschema ist unter Anwendung einer relationalen Datenstruktur problemlos erweiterbar, so daß neue Anforderungen entsprechend der Entwurfsprozeßmethodik nach dem Phasenmodell [33] eingearbeitet werden können. Auf Grundlage einer fein differenzierten Vergabe von Nutzerrechten bis auf Datensatzebene ist man in der Lage, Projekte mit allen Verknüpfungen in die Datenbank aufzunehmen und erst zu einem gegebenen Zeitpunkt (z. B. Fertigstellung eines Projektes) für andere DMS-Nutzer freizugeben. Dabei kann der Zugriff lesend, schreibend und/oder löschend erfolgen.

2.2.3.2 Allgemeiner Aufbau des DMS

Die Basis des DMS ist eine Datenbank, in der alle Daten strukturiert nach dem Datenschema physisch abgelegt sind. Ein Anwenderprogramm, das über ein graphisches Interface alle Daten für den Nutzer in aufbereiteter Form anzeigt und Funktionen zur Dateneingabe und -manipulation zur Verfügung stellt, ist über spezielle Zugriffsroutinen mit der Datenbank verbunden (Bild 19).

2.2.3.3 Kategorien im DMS

Den Mittelpunkt des Datenschemas bilden die Verhaltensbeschreibungen sowie die Anwendungsbereiche mikromechanischer Komponenten und Systeme. Die dazu anfallenden Entwurfs- und Charakterisierungsdaten werden grob den Kategorien Projekt, Entwurf, Technologie, Fertigung/Qualitätssicherung, Modellierung, Simulation zugeordnet. Bild 20 zeigt den Zusammen-

hang zwischen Entwurfs- und Fertigungsprozeß mikromechanischer Komponenten innerhalb eines Mikrosystems mit rückgekoppeltem Optimierungsmechanismus.

Kategorie Projekt

Der Projektname dient als Oberbegriff für die Zuordnung von mikromechanischen Komponenten zu einem Projekt. Informationen, Zielstellungen, gegenwärtige Stände, Statistiken, Ergebnisse usw. werden dem Projekt in Form von Dokumenten zugeordnet.

Kategorie Entwurf

Hierzu zählen Daten, die die mikromechanischen Komponenten beschreiben und charakterisieren. Neben dem Namen und dem Entwurfsdatum der Komponente sind Angaben über ihre Struktur und Eigenschaften, wie geometrische Abmessungen, materialspezifische Kenngrößen oder Systemkenngrößen relevant. Diese Charakteristika bilden die Grundlage für einen späteren Modellierungsprozeß. Außerdem sind die Eigenschaften stets objektabhängig und unterscheiden sich u. U. von einer Komponente zur anderen erheblich. Deshalb bietet das DMS die Möglichkeit des freien Konfigurierens von charakterisierenden Eigenschaften. Aus einer Anzahl von zuvor definierten Größen, Einheiten und Eigenschaften erfolgt die Auswahl und Zuordnung dieser zu Konstruktionselementen und Fertigungseinheiten, die Bestandteile des Entwurfs darstellen. Konstruktionselemente sind die kleinsten Bausteine einer Fertigungseinheit, die wiederum aus anderen Fertigungseinheiten bestehen kann und somit die qualitative Ebene einer mikromechanischen Komponente darstellt. Angaben zur Struktur und Teilehierarchie der mikromechanischen Komponenten dienen dem automatischen Generieren von Stücklisten oder Informationen über Baugruppen sowie deren Verwendung.

Ein weiterer Bestandteil des Entwurfs ist die Verwaltung von Layoutstrukturen mit verwendeten Wafern und der Zuordnung der technologischen Ebenen zu den Wafern innerhalb der Datenbank.

Kategorie Technologie

Sie beschreibt Technologien, nach denen eine mikromechanische Komponente gefertigt werden soll. Dokumente, die z. B. eine Technologie allgemein beschreiben oder eine Technologie zu einer konkreten mikromechanischen Komponente darstellen, gehören ebenso zu dieser Kategorie. Ausgewählte technologische Parameter und Verfahren wie z. B. die Lithographie, Dick- oder Dünnschichttechnologie, Vereinzeln, Bonden und Verkappen werden einer Komponente zugeordnet und bestimmen in Form einer technologischen Sequenz den Fertigungsprozeß.

Kategorie Fertigung/Qualitätssicherung

Diese Kategorie enthält alle Daten, die zur Produktion und Qualitätssicherung der Komponente dienen. Das Endprodukt ist durch seinen Namen, die Seriennummer und das Herstellungsdatum referenziert. Mittels eines Verknüpfungsbausteins werden der mikromechanischen Komponente Technologien zugeordnet. Ein wichtiger Aspekt ist die Qualitätsbewertung/-sicherung, die u. a. auf der Grundlage von Charakterisierungsdaten näher spezifiziert ist. Diese Daten können in Form von Meßreihen als charakterisierende Eigenschaften gespeichert werden. Darüber hinaus ist die Verknüpfung zu Dokumenten, die den Meßaufbau, Meßbedingungen, Testpattern und Meßergebnisse der Charakterisierung sowie Zertifikate enthalten, für eine detaillierte Beschreibung eines Mikrosystems möglich.

Kategorie Modellierung

Die dieser Kategorie zugehörigen Simulationsmodelle sind durch Gültigkeitsbereich, Qualität, Modellierungsverfahren und Simulationsplattform (Simulator) näher beschrieben. Das Modellierungsverfahren gibt an, wie das Modell gebildet wurde, d. h., wie es auf Grundlage physikalischer Gesetze aus der Theorie abgeleitet (Energieerhaltungssatz, Prozesse mit Differentialgleichungen) oder durch Bestimmung des Systemverhaltens infolge umfangreicher Messungen (Parameteradaption) ermittelt wurde. Die im Simulationsmodell verwendeten Parameter, die als Charakterisierungsdaten im DMS gespeichert werden, sind z. B. Systemkennwerte wie Trägheitsmoment, Federkonstante, Resonanzfrequenz, Dämpfungsfaktor o. ä.

Kategorie Simulation

Die Kategorie Simulation enthält die Ergebnisse durchgeführter Simulationen sowie Angaben zum Simulationsablauf, zum verwendeten Simulationsmodell und zur Art der Simulation. Der Simulationsablauf liefert Informationen zu den Stimuli und zu Simulationsparametern (Berechnungsmethode, Schrittgrößen der Berechnung, Simulationsstartzeit und -dauer). Die Simulationsart beschreibt die Untersuchungsmethode näher [16]. Dokumente, die mit einem Datensatz in der Kategorie Simulation verknüpft sind, liefern weiterführende Angaben zur Simulation selbst. Dazu zählen u. a. Informationen zur Generierung der Stimuli bzw. der Anzeige und Auswertung von Simulationsergebnissen.

2.2.3.4 Dokumentverwaltung

Da in allen Kategorien Dokumente verschiedenster Art und unterschiedlichster Dateitypen entstehen bzw. vorhanden sind, wurde ein DMS-Formular entwickelt, das beliebige Dateien aufnehmen kann. Diese werden in der Datenbank als Large Object Files binär abgelegt. Bei Aufruf eines Dokumentes ordnet das DMS in Abhängigkeit des Dateityps das zugehörige Betrachter- oder Textverarbeitungsprogramm zu. Über nutzergenerierte Verknüpfungen werden jeder Kategorie die entsprechenden Dokumente zugewiesen. Dabei besteht einerseits die Möglichkeit, dem Datensatz einer Kategorie mehrere Dokumente zuzuordnen, andererseits kann ein Dokument mit Datensätzen unterschiedlicher Kategorien verknüpft sein (Bild 21). Der Dokumentname bildet dabei den eineindeutigen Schlüssel zur Identifikation. Eine Exportfunktion dient dazu, ein Dokument aus der Datenbank in ein beliebiges Zielverzeichnis zu kopieren.

2.2.3.5 Komponenten im Entwurfsprozeß

Die Wiederverwendung (Reuse) vorhandener Komponenten im digitalen Entwurf gewinnt stetig mehr an Bedeutung. Grund dafür ist die Forderung nach immer kürzeren Entwurfszeiten. Der interdisziplinäre Einsatz des DMS als Modell-Repository mit Komponenten-Retrieval gibt den kooperierenden Komponenten- und Systementwerfern ein wichtiges informationstechnisches Werkzeug in die Hand. Das aus dem Entwurf abgeleitete Mikrosystem wird in einem baumstrukturähnlichen Datenmodell im DMS nachgebildet. Diese Struktur gibt Aufschluß über den Aufbau und die Verwendung eines Mikrosystems. Außerdem erlaubt sie die Generierung von Stücklisten. Ein heterogenes Mikrosystem besteht aus Komponenten, die selbst Bestandteil anderer sein können. Die Komponente wiederum besteht aus sogenannten Formelementen, die einfach strukturiert sind und als kleinste Bausteine innerhalb der Mikrostruktur gelten. Das sind beispielsweise Spiegelplatten, Federbänder in einem mikromechanischen Spiegelarray oder spezielle Lackschichten

zum Schutz von Elektroden. In Bild 23 ist die Struktur eines Spiegelarrays mit Komponenten und Formelementen dargestellt. Die Abkürzung „K“ steht für Komponente, „FE“ bedeutet Formelement. Über die Kategorie Modellierung kann ein entsprechendes Simulationsmodell, das Beschreibungs-, Erstellungs- und Bewertungsdaten enthält, zugeordnet werden. Entstandene Daten von Simulationsergebnissen, Stimuli und weitere Informationen werden in der Kategorie Simulation gespeichert. Weitere Eigenschaften, die zur Charakterisierung der Formelemente und Komponenten sowie des Simulationsmodells und der Simulationsergebnisse dienen, sind über die Verknüpfung zur Eigenschaften-Verwaltung frei definierbar. Technische Größen und Einheiten oder andere elementare Eigenschaften lassen sich parallel dazu in einer eigenen Tabelle vordefinieren.

2.2.3.6 Technologien zur Herstellung mikromechanischer Komponenten im DMS

Die Fertigung einer mikromechanischen Komponente setzt sich aus einer Vielzahl technologischer Verfahren zusammen, die in einer bestimmten Abfolge ineinander greifen. Eine Technologie in ihrer Gesamtheit abzubilden, ist rechentechnisch außerordentlich anspruchsvoll, da sich Parameter sehr oft ändern. Für jeden Wafer müßten alle Bearbeitungsschritte gespeichert werden. Dies würde einen erheblichen Zeitaufwand zur Pflege der Daten verursachen, so daß lediglich ein bestimmter Ist-Zustand einer Herstellungssequenz ohne Vor- und Fehlversuche im DMS abgebildet wird. Der Entwerfer einer Komponente benötigt üblicherweise nur für sich relevante Technologien und keine technologischen Zusammenhänge. Wird dennoch im Entwurfsprozeß eine neue Technologie erforderlich, z. B. durch die Veränderung von Herstellungsanlagen eines Technologiezentrums, so besteht im DMS die Möglichkeit, einer mikromechanischen Komponente verschiedene Technologien zuzuordnen, wobei die aktuelle in Form eines Vermerkes gekennzeichnet werden sollte. Das Ablegen von Layouts bzw. Teillayouts, REM-Bildern, geometrischen Daten, Meßergebnissen o. ä. erfolgt über die Dokumentverwaltung oder über Eigenschaften-Links, die zur näheren Charakterisierung einer Technologie verwendet werden.

2.2.4 Wiederverwendbarkeit (Reuse) von Entwurfsmodellen

Um die komplette Struktur eines Mikrosystems im DMS abzubilden, werden Komponenten mit zugehörigen Formelementen aus der Kategorie Entwurf benötigt. Zum Aufbau einer Komponenten-Struktur sind diese dann dem zu entwerfenden Mikrosystem zuzuordnen. Die verschiedenen mikromechanischen Komponenten eines Systems können Elemente des Entwurfs (Entwurfsmodelle) gemeinsam nutzen. Das spart Speicherplatz und Eingabeaufwand. Eine Änderung im Entwurf einer Komponente wird, wenn es der Entwerfer wünscht, sofort in allen Mikrosystemen, die einen Link zu dieser besitzen, wirksam (Bild 22). Sollen die Änderungen keinen Einfluß auf andere Mikrosysteme aufweisen, so entsteht für diese geänderte Komponente ein neuer Eintrag in der Datenbank. Die Abbildung von Layouts und zugehörigen Layout-Ebenen im DMS ist in ähnlicher Weise strukturiert wie für mikromechanische Komponenten. Dem Layout als Oberbegriff mit charakterisierenden Eigenschaften und Dokument-Links werden die entsprechenden Layout-Ebenen untergeordnet. Dabei besteht die Möglichkeit, einmal definierte Ebenen mit verschiedenen Layouts zu verbinden. Eine Layoutstruktur läßt sich wiederum mikromechanischen Komponenten zuordnen und mit Technologien verknüpfen. Das Mikrosystem ist das Endprodukt, das aus unterschiedlichen Komponenten (Entwurfsmodellen und Technologien) besteht. Simulationen (Stimuli und Ergebnisse) sind für Entwurfsmodelle und Mikrosysteme über zugehörige Simula-

tionsmodelle getrennt zu speichern. Die mikromechanische Komponente kann aus anderen Komponenten bestehen oder selbst Bestandteil einer anderen sein.

2.2.5 Systemeinführung

In Vorbereitung zur Einführung des Datenmanagementsystems (DMS) in den SFB 379 wurde mit 3 - 4 Mitarbeitern aus verschiedenen Projektbereichen des SFB eine Evaluierungsphase durchlaufen mit dem Ziel, erste Erfahrungen im Umgang mit dem DMS zu sammeln. Dabei ging es um die Bedienbarkeit und das generelle Verständnis dieses Systems aus der Sicht des Anwenders. Wünsche und Hinweise (Adreß- bzw. Namensverzeichnis, das Sperren/Freigeben von eigenen in der Datenbank abgelegten Dokumenten zur Einsicht für andere Nutzer) wurden und werden sukzessive in das DMS eingearbeitet.

3. Ergebnisse und ihre Bedeutung

3.1 Methoden für den Mikrosystementwurf

3.1.1 Bibliotheksentwicklung für VHDL-AMS

Im Antragszeitraum 2001-2003 entstanden zwei umfangreiche Bibliotheken mit SPICE-Modellen und MATLAB-Funktionen, die über die Datenbank den anderen Teilprojekten des SFB 379 zur Verfügung stehen.

3.1.1.1 Bibliothek mit SPICE-Elementen

Die entstandenen Modelle wurden getestet und bezüglich Funktionalität und Geschwindigkeit mit ELDO bzw. PSPICE verglichen. Die Differenzen der Simulationsergebnisse zwischen den Modellen liegen in der Regel unter 1 %. Ausnahmen bilden lediglich die Modelle für verlustbehaftete Leitung, MOSFET-Level 3 und Triac, bei denen der Fehler, je nach gewählten Modellparametern bis zu 10 % betragen kann. Die Ursache hierfür liegt darin begründet, daß in diesen Fällen die in SPICE verwendeten Gleichungen nicht exakt bekannt waren und so Gleichungssätze aus anderen Simulationssystemen bzw. Näherungslösungen verwendet werden mußten. Im Rahmen einer **System**simulation sind Fehler bis zu 10 % im allgemeinen aber unbedenklich. Die Simulationsgeschwindigkeit der VHDL-AMS Modelle war mit der Simulationsgeschwindigkeit der SPICE-Modelle vergleichbar. Für die Initialisierung der VHDL-AMS Modelle ist gegenüber SPICE mit zum Teil größeren Zeiten zu rechnen, was bei einer Transient-Simulationen über einen längeren Zeitraum allerdings nur einen kleinen Teil der Gesamtsimulationszeit ausmacht. Die längeren Initialisierungszeiten lassen sich damit erklären, daß in SPICE-kompatiblen Simulatoren die Modelle simulatorimmanent und optimiert vorliegen. Bei der Simulation des OPV, des Thyristors und des Triac traten Konvergenzprobleme in der Simulation auf, die bei OPV und Thyristor durch Wahl geeigneter Simualtionsparameter behoben werden konnten. Stellvertretend für die passiven Elemente der Bibliothek zeigen Bild 6 in Abschnitt 7. eine Simulation der verlustbehafteten Leitung und Bild 7 die Simulation der in der Bibliothek realisierten Diode 1N4148.

Die hier geschaffene Bibliothek ermöglicht es dem Mikrosystementwerfer, elektrische Komponenten nicht nur abstrakt, sondern auch auf Schaltungslevel zu simulieren. Die Verwendung von

VHDL-AMS gestattet es, die passiven und abstrakten Modellen leichte auf andere physikalische Domänen (z. B. mechanische oder fluidische Systeme) abzubilden. Außerdem hat der Anwender jederzeit die volle Kontrolle über die Modellgleichungen und kann diese bei Bedarf – im Gegensatz zu SPICE – auf spezielle Anforderungen anpassen.

3.1.1.2 Bibliothek mit MATLAB-Funktionen

Die komplexen mathematischen Funktionen entsprechend der LINPACK und LAPACK Algorithmen wurden in VHDL umgesetzt. Da es sich um rein numerische Algorithmen handelt, die sich mit digitalen Sprachkonstrukten von VHDL beschreiben lassen, gibt es bis auf numerische Ungenauigkeiten keine Abweichungen in den Resultaten. Eine Ausnahme bildet die *LU*-Funktion zur *LU*-Faktorisierung (Zerlegung einer quadratischen Matrix *X* in eine obere Dreiecksmatrix *U* und in eine Permutation der unteren Dreiecksmatrix *L,* so daß gilt *L*U=X*). Hier liefert der LINPACK Algorithmus ein anderes Resultat als MATLAB, was wahrscheinlich auf eine Modifikation des Algorithmus' seitens MATLAB zurückzuführen ist. Die in VHDL realisierten Funktionen erreichen eine deutlich höhere Simulationsgeschwindigkeit als in MATLAB. So benötigt eine 100000 fache Wiederholung der *LU*-Funktion auf einer SUN ULTRA60 Workstation mit UltraSPARC-II 296 MHz CPU in VHDL (Modelsim) etwa 2 Sekunden, in MATLAB ca. 17 Sekunden. Bei der Bewertung der höheren Simulationsgeschwindigkeit muß man aber beachten, daß der VHDL-Code compiliert, ein MATLAB-Script dagegen interpretierend abgearbeitet wird.

Die in dieser Bibliothek entstandenen Modelle ermöglichen es dem Entwerfer, auch komplexe mathematische Zusammenhänge mit VHDL bzw. VHDL-AMS zu beschreiben.

3.1.2 Makromodellierung/Multi-Architecture-Modellierung

3.1.2.1 Makromodellierung

Durch den Wissenstransfer bezüglich VHDL-AMS vom TP A2 nach TP A1, wurde es möglich, die im TP A1 entwickelte Methode der Makromodellgenerierung so zu erweitern, daß auch Modelle in VHDL-AMS erzeugt werden können. Mit dieser Methode konnten aus den im TP A4 entstandenen FEM-Modellen des Vibrationssensor-Arrays entsprechende VHDL-AMS-Makromodelle abgeleitet werden. Details zu den Ergebnissen des Einsatzes der Makromodelle im Entwurf des Demonstrators sind im Abschnitt 3.1.5 zu finden.

3.1.2.2 Multi-Architecture-Modellierung (MAM)

Die im Abschnitt 2.1.2.2 beschriebene Methode wurde beim Systementwurf des Demonstrators angewandt und hat sich dabei bewährt (Bild 14). Durch Nutzung der MAM mußte bei der Modellierung dieses Systems kein Interface nachträglich geändert werden. Auch das Bestimmen des notwendigen Interfaces der detaillierteren Komponente auf hohem Abstraktionsniveau hat sich als praktikabel erwiesen. Werden einige Modellierungsregeln [6] beachtet, ist nur mit einer geringfügigen Verringerung der Simulationsgeschwindigkeit zu rechnen. Dafür erspart man sich beim Entwurf die zeitaufwendige und fehleranfällige Konvertierung der Schnittstellen beim Einfügen von Modellen oder Architekturen auf unterschiedlichen Abstraktionsebenen. Dies ist insbesondere dann von Vorteil, wenn z. B. für eine oder mehrere Komponenten Verhaltens- und

Makromodelle zur Verfügung stehen. Während der Simulation kann so für jede Komponente, für die Architekturen auf unterschiedlichen Abstraktionsebenen existieren, je nach Erfordernissen bezüglich Simulationsgeschwindigkeit und Genauigkeit, entweder eine abstrakte oder eine detaillierte Version eingesetzt werden.

Im Vergleich zu einem herkömmlichen Top-Down-Entwurf ergibt sich mit MAM der in Bild 12 abgebildete Entwurfsablauf. Bei rein digitalen Systemen oder Systemen, bei denen Modelle oder Schnittstellen automatisch generiert werden (Synthese), sollte man die MAM nicht anwenden, da bei solchen Systemen keine Steigerung der Entwurfseffizienz durch die MAM zu erwarten ist. Bei heterogenen Systemen dagegen, bei denen eine Synthese der Modelle nur selten zum Einsatz kommen kann, hat sich der Einsatz der MAM bewährt.

3.1.3 Kosten-Modellierung

Die Kostenmodellierung wurde implementiert und anhand eines Beispiels getestet. Bereits in diesem ersten Beispiel zeigte sich die sehr gute Funktionalität des Ansatzes. Negative Auswirkungen auf Rechenzeit oder Simulationsstabilität traten nicht auf. Auf die Modellierung des Demonstrators konnte diese Methode noch nicht angewandt werden, da der zur Verfügung stehende VHDL-AMS Simulator Teile der für diese Methode notwendigen Typdefinitionen noch nicht beherrscht. Um unnötigen Arbeitsaufwand zu vermeiden, soll deshalb das nächste Simulator-Update abgewartet werden. Wenn dann dieses Problem immer noch besteht, so muß eine Umstellung der Datenstruktur erfolgen, was aber die Übersichtlichkeit der Darstellung der Kostenwerte und Kostengrenzen verringert.

3.1.4 Constraint-Modellierung

Wie im Abschnitt 2.1.4 dargelegt, traten bei der Entwicklung des methodischen Ansatzes zur Constraint-Modellierung Probleme auf. Da diese nur zu einem geringen Teil lösbar waren, wird eine Anwendung dieses Ansatzes in der geplanten Art und Weise nicht für erfolgversprechend erachtet.

Das Ziel, die Grenzen der Modelle bei der Parametrisierung zu überwachen, kann, wie in 2.1.4 beschrieben, auch direkt mit den VHDL-AMS *assert statements* erfolgen. Diese Vorgehensweise ist weniger flexibel aber dafür weniger aufwendig, und fand anstelle der Constraint-Modellierung im Entwurf des Demonstrators Anwendung.

Einen etwas anderen Ansatz zur Überwachung von Parametern digitaler Komponenten wird derzeit an der Professur Schaltungs- und Systementwurf im Rahmen des Projekts IPQ (Intellectual Property Qualification) entwickelt, bei dem die Abbildung von Requirements auf Constraints auf der Basis von unidirektionalen Zuweisungen erfolgt. Im Rahmen der im Bearbeitungszeitraum 2001 - 2003 verbleibenden Zeit soll daher untersucht werden, inwiefern sich dieser Ansatz auch für heterogene System eignet. Dieser Ansatz erreicht zwar nicht die Leistungsfähigkeit der ursprünglich geplanten Methode, würde aber die derzeit zur Verfügung stehenden Möglichkeiten zur Constraint-Überwachung deutlich erweitern.

3.1.5 Systementwurf für den Demonstrator „Vibrationssensor-Array“

Mit dem abstrakten Modell der virtuellen Werkzeugmaschine, dem auf vereinfachten analytischen Beziehungen beruhenden, abstrakten Modell der ersten Präparation des Sensors und dem Modell der Analog-Signalverarbeitung konnten erste Simulationen durchgeführt werden, die die Funktion der Modelle und das Systemkonzept bestätigen. Darauf aufbauend erfolgte die Optimierung dieser Modelle hinsichtlich ihrer Simulationszeit sowie erste Schritte zur Verifizierung der Methode der Multi-Architecture-Modellierung. Nach der Fertigstellung des Layouts des Sensors, der im Demonstrator zum Einsatz kommt, wurde das Sensormodell für diesen neuen Sensor angepaßt. Mit dem vom TP A1 generierten Makromodell stand anschließend eine zweite Architektur für das Sensormodell zur Verfügung

Die Simulation auf einer SUN ULTRA60 Workstation mit UltraSPARC-II 296 MHz CPU ergab die folgende Resultate:

	abstraktes, analytisches Modell vom TP A2	Makromodell vom TP A1	Messung am Sensor
Resonanzfrequenz Zelle 1	1,8 kHz	1,7 kHz	1,7 kHz
Ausgangsspannung	0,6 V	0,8 V	0,5 V
Simulationszeit	7 Sekunden	2 Sekunden	

TABELLE 4. Ergebnisse der AC-Simulation mit Tuningspannung 35 V, Polarisationsspannung 3 V, Anregungsamplitude 0,6 mm

Es zeigt sich, daß bei einer AC-Simulation das Makromodell schneller berechnet werden kann als das auf vereinfachten analytischen Beziehungen beruhende, abstrakte Modell. Das abstrakte Modell scheint ein genaueres Ergebnis als das Makromodell zu liefern, was aber darauf zurückzuführen ist, daß die Abstraktion des Modells und die Fertigungstoleranzen des realen Sensors zufälligerweise einen in die gleiche Richtung wirkenden Fehler hervorrufen. Der Fehler des Makromodells gegenüber dem ursprünglichen FEM-Modell beträgt weniger als 1%.

Das Modell der Systemsimulation besteht aus den Modellen der virtueller Werkzeugmaschine, des Sensor-Arrays und der Analog-Signalverarbeitung. Die Simulation läuft über einen Zeitraum von 50 ms, für die Stimuli wird ein Vibrationsspektrum mit mehreren Frequenzanteilen benutzt. Die Polarisationsspannung beträgt 3 V.

	abstraktes Modell vom TP A2	Makromodell vom TP A1
Tuningspannung für eine Resonanzfrequenz von Zelle 1 bei 1 kHz	36,65 V	36,0 V
Ausgangsspannung	1,2 V	1,1 V
Simulationszeit	15 Minuten	7 Minuten

TABELLE 5. Ergebnisse der Transient-Simulation des Systems

Ein Simulationsplot der Simulation mit Makromodell ist in Bild 15 zu sehen. Dieser Plot zeigt die von der virtuellen Werkzeugmaschine erzeugten Stimuli, die Reaktion der Resonatoren 1 und 2

aus dem Array, die vom Array gelieferte Ausgangsspannung und das Ausgangssignal der analogen Signalverarbeitung. Es ist deutlich zu erkennen, wie nur die auf Resonanz abgestimmte Zelle 1 schwingt.

Anhand der Simulation zeigt sich auch hier, daß das Makromodell gegenüber dem abstrakten Modell sowohl eine höhere Genauigkeit als auch eine höhere Simulationsgeschwindigkeit aufweist. Ebenso konnten bei dieser Simulation einige Nebeneffekte wie z. B. das „Schnappen“ der Resonatoren beim Überschreiten einer bestimmten kritischen Tuningspannung festgestellt werden, die im Entwurf des Demonstrators berücksichtigt werden müssen. Dank der Anwendung der Multi-Architecture-Modellierung dauert das Umkonfigurieren des Systems zwischen Makromodell und analytischem Modell nur etwa 5 Sekunden.

Im Modell der analogen Signalverarbeitung wurde unter Anwendung der MAM ein Filter gegen ein abstraktere Version getauscht. Mit dem Originalmodell des Filters würden sich die Simulationszeiten verdoppeln. Das Austauschen der abstrakten Blöcke der analogen Signalverarbeitung gegen ein Modell der elektrischen Schaltung brachte dagegen keine nennenswerten Änderungen in den Simulationszeiten.

Mit der Fertigstellung des Modells des Mikrocontrollers und der Einbindung des Modells der Fuzzy-Pattern-Klassifikation in das Systemmodell, lassen sich dann auch ganze Meßzyklen simulieren.

Anhand der Modellierung und Simulation des Gesamtsystems Demonstrator konnte zum einen das Systemkonzept des Simulators evaluiert und zum anderen die Methode der Makromodellierung und der Multi-Architecture-Modellierung erfolgreich verifiziert werden.

3.2 Datenmanagement

Die Entwicklung des DMS erfolgte mit dem Datenbanksystem PostgreSQL und der Skriptsprache Tcl/Tk für die graphische Oberfläche. Für die Programmiersprache Tcl/Tk stehen Interpreter sowohl für UNIX-/Linux- als auch Windows-Plattformen zur Verfügung, so daß eine hohe Flexibilität und Portierbarkeit des DMS besteht. Eine fein differenzierbare Nutzerrechte-Vergabe gewährleistet die notwendige Datensicherheit innerhalb der Datenbank. Das Datenmanagementsystem kommt im Rahmen des Sonderforschungsbereiches 379 zum Einsatz. Die Teilprojekte innerhalb des SFB 379 haben mit Hilfe des DMS die Möglichkeit, die verschiedenen Entwurfs- und Fertigungsebenen von mikromechanischen Komponenten und Systemen datentechnisch zusammenführen und bereits erworbenes Wissen zu sammeln. Der Mehraufwand an Datenpflege zahlt sich letztendlich durch eine gemeinsame und effizientere Nutzung der Wissensbasis innerhalb des SFB 379 aus.

Aufgrund neuer Anforderungen in Bezug auf Bedienbarkeit und Funktionalität des DMS sowie die Umstellung des Datenbanksystems von ADABAS D auf PostgreSQL, waren massive Änderungen an der Oberfläche des DMS und die Erweiterung des Datenschemas erforderlich. In Bild 24 ist das überarbeitete Hauptmenü des DMS mit dem erweiteren Datenschema sowie der übersichtlichen Darstellung aktiver Formulare und Verknüpfungen abgebildet. Die komplett neu gestalteten Formularlayouts (Bild 25), die Verkettung von Formularen (Bild 26) und damit die Möglichkeit der relationalen Datenanzeige, die vereinfachte Nutzung von Suchfunktionen und die grafischen Übersichten (Bild 27) über erzeugte Datenstrukturen gewährleisten ein verbessertes Programmhandling. Der Zugriff auf das DMS über einen Web-Browser ist in der aktuellen Ausbaustufe mit einer Retrievalkomponente auf Basis der Programmiersprache PHP3 bereits reali-

siert. Eine Erweiterung der Nutzung mit der vollen Funktionalität der graphischen Tcl/Tk-Oberfläche des DMS wird gegenwärtig getestet. Der jetzt erreichten Ausbau der Datenbank ermöglicht weiterhin die Integration des am Lehrstuhl entwickelten Werkzeugs zur Spezifikationserfassung ASPECTOR.

Für einen effizienteren und leichteren Umgang mit dem DMS wurde im Zuge der Einführung der Datenbank eine Benutzerreferenz erstellt. Für die Nutzung des Datenmanagementsystems im gesamten SFB 379 und der damit verbundenen erhöhten Nutzung der Datenbank sollte die derzeit lokale Installation durch eine Netzwerkinstallation abgelöst werden.

4. Vergleiche mit Arbeiten außerhalb des Sonderforschungsbereichs und Reaktionen der wissenschaftlichen Öffentlichkeit auf die eigenen Arbeiten

Die Hardwarebeschreibungssprache VHDL-AMS gewinnt zunehmend Einfluß in universitären und auch industriellen Bereichen. Dadurch ergibt sich eine steigendes Interesse an Forschungsaktivitäten im Bereich VHDL-AMS. Der größte Teil dieser Aktivitäten beschäftigt sich aber mehr mit der Modellbildung als mit Methoden des Systementwurfes.

Die Erprobung und Weiterentwicklung von Methoden, Algorithmen und Verfahren für die Modellierung, Beschreibung und Simulation von komplexen Systemen ist u. a. Gegenstand der Arbeiten im Rahmen des SFB 358 [21] an der TU Dresden und der FhG-IIS/EAS Dresden, wobei sich die Arbeiten hier nicht nur auf VHDL-AMS konzentrieren, sondern auch auf andere Beschreibungssprachen und Simulatorkopplungen.

Bezüglich der Arbeiten zur Kostenmodellierung und -Optimierung heterogener Systeme existieren unabhängig von VHDL-AMS Arbeiten an anderen Forschungseinrichtungen (z. B. Uni Frankfurt [23]). Diese unterscheiden sich im Wesentlichen darin, daß dort neue Tools zur Kostenoptimierung heterogener Systeme entstehen, während die Arbeiten am SFB 379 TP A2 darauf abzielen, die Kosten so in VHDL-AMS Modelle zu implementieren, daß eine Optimierung mit vorhandenen Optimierungswerkzeugen durchgeführt werden kann.

Die in Bearbeitungszeitraum 2000-2003 im TP A2 erzielten Ergebnisse fanden auf nationalen [3], [4], [6], [7], [9], [10] und internationalen [1], [2], [5], [8] Konferenzen ein breites Interesse. Für die im Abschnitt 2.1.1.1 beschriebenen Modelle, insbesondere das der verlustbehaftete Leitung, gab es seitens der FhG-IIS Dresden und Daimler Chrysler Anfragen.

Das Datenmanagementsystem ist im Rahmen des SFB 379 erfolgreich eingeführt worden. Daneben gibt es an der TU Chemnitz auch Anfragen von Mitarbeitern außerhalb des SFB zur evtl. Nutzung des DMS für Themen wie z. B. Dokumenteverwaltung für Bildverarbeitung.

5. Offene Fragen

Zum gegenwärtigen Zeitpunkt sind noch einige wenige Punkte aus dem Arbeitsprogramm abzuarbeiten:

- Abschnitt 2.1.3 Kostenmodellierung

  Der methodische Ansatz der Kostenmodellierung ist weitestgehend fertiggestellt und getestet. Wegen der in Abschnitt 2.1.3 beschriebenen Probleme bezüglich der Sprachunterstützung des VHDL-AMS Simulators, war die Methode auf den Systementwurf des Demonstrators bisher noch nicht anwendbar.

- Abschnitt 2.1.5 Systementwurf

  Die vom TP A2 zu entwickelnde Hardwarekomponente (Fuzzy-Pattern-Klassifikator) ist fertiggestellt und muß im nächsten Schritt an die vom TP A4 entwickelte Signalverarbeitung angebunden werden. Damit ist der Demonstrator einsatzbereit. Anschließend ist der Klassifikator auf Vibrationsmuster von scharfen und verschlissenen Drehmeißeln anzulernen, und es sind unter realen Bedingungen Messungen vorzunehmen.
  Für die Systemsimulation ist das Modell des Mikrocontrollers noch in die Systemsimulation einzufügen und damit dann das gesamte System, bestehend aus virtueller Werkzeugmaschine, Sensor-Array, analoger und digitaler Signalvorverarbeitung und Fuzzy-Pattern-Klassifikator, zu simulieren.

Auch in Zukunft wird die Schaffung neuer Methoden und die Übertragung von Methoden des Mikroelektronikentwurfes in den Mikrosystementwurf eine große Rolle bei den Forschungsaktivitäten auf dem Gebiet der Mikrosystemtechnik spielen müssen, um die immer noch bestehende Lücke im Entwurfsprozeß zwischen Elektronik- und Mikrosystementwurf verkleinern zu können. Im Mittelpunkt der geplanten Arbeiten im Antragszeitraum 2004-2006 des TPs A2 werden deshalb wieder Arbeiten zur Methodik des Systementwurfs für Mikrosysteme stehen. Der Schwerpunkt wird dabei auf der Entwicklung und Untersuchung von methodischen Zugängen zur Anwendung der neuen Systembeschreibungssprache SystemC-AMS bei der Modellierung und Simulation heterogener Systeme (Mechanik, analoge und digitale Elektronik, Software) liegen. SystemC-AMS erlaubt eine abstraktere Beschreibung von Systemen als VHDL-AMS und ermöglicht gleichzeitig die Einbeziehung von Software in die Modelle. Desweiteren sollen die im Bearbeitungszeitraum 2001-2003 entstandenen Methoden der Multi-Architecture-Modellierung und Kosten-Modellierung in SystemC-AMS übertragen, und damit weiterentwickelt werden. SystemC-AMS bietet außerdem erste Ansätze, die im Mikroelektronik-Entwurf bekannte Methode des Hardware-Software-Codesign auf heterogene Systeme zu übertragen, wozu ebenfalls Arbeiten vorgesehen sind.

6. Literatur

6.1 Begutachtete Veröffentlichungen

[1] Kuerschner, R.; Richter, A.; Herrmann, G.; Mueller, D.: *Data Management System for design and characterization data of micromechanical components in heterogeneous microsystems.* 2nd International Conference on Materials for Advanced Technologies - ICMAT, 2003, Singapore, 29. Juni - 04. Juli 2003 (accepted)

[2] Schlegel, M.; Herrmann, G.; Müller, D.: *Application of „Multi Architecture Modeling“ design method in system level MEMS simulation*. DTIP 2003, 5.-7. May 2003, Mandelieu-La Napoule, France (accepted)

[3] Schlegel, M.; Herrmann, G.; Müller, *D.: Ein System-Level Modell in VHDL-AMS eines mikromechanischen Vibrationssensor-Arrays*. Scientific Reports, Journal of the University of Applied Science Mittweida, 15th International Scientific Conference Mittweida, Nr. 10, 2002, Mittweida, Germany, 07.-11. November 2002, ISSN 1437-7624

[4] Richter, A.; Müller, D.: *Datenmanagementsystem für Entwurfs- und Charakterisierungsdaten von Komponenten in heterogenen Mikrosystemen.* Scientific Reports, Journal of the University of Applied Science Mittweida, 15th International Scientific Conference Mitt-

weida, Nr. 10, 2002, Mittweida, Germany, 07.-11. November 2002, ISSN 1437-7624

[5] Schlegel, M.; Herrmann, G.; Müller, D.: *A system level model in VHDL-AMS for a micromechanic vibration sensor array*. The First IEEE International Conference on Sensors, Orlando, Florida, USA, June, 11-14 2002, ISBN 0-7803-7454-1

[6] Schlegel, M.; Herrmann, G.; Müller, D.: *Multi-Architecture-Modeling: Entwurfsmethode für Mixed-Signal- und Multi-Domain-Systemsimulation.* GI/ITG/GMM-Workshop: Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen, Tübingen, 25.-27. Februar 2002, Berichte aus der Informatik, Shaker Verlag, Aachen 2002, ISBN 3-8265-9859-8, S. 18-25

[7] Schlegel, M.; Herrmann, G.; Müller, D.: *Entwicklung eines effizienten VHDL-AMS-Modells der verlustbehafteten Leitung*. 5. Chemnitzer Fachtagung Mikrosystemtechnik – Mikromechanik & Mikroelektronik, Chemnitz, 23./24. Oktober 2001, Tagungsband, S. 12-16; ISBN 3-00-008201-8

[8] Schlegel, M.; Herrmann, G.; Müller, D.: *Multi Architecture Modeling Desing Method for Mixed Signal and Multi Domain System Simulation - First Solutions*. International MEMS Workshop, Singapore, July 4-6, 2001, pp. 662, ISBN 981-04-4165-7

[9] Schlegel, M.; Müller, D.: *Systementwurf und Systemsimulation mit VHDL-AMS am Beispiel der Rundumprojektion mit 2D-Mikrospiegelarray*. 3. ITG/GI/GMM-Workshop „Multi-Nature Systems: Optoelektronische, mechatronische und andere gemischte Systeme“, Hamburg-Harburg, 22. Februar 2001, ISBN 3-930400-31-6

[10] Schlegel, M.; Müller, D.: *Gesamtsystemsimulation mit VHDL-AMS am Beispiel der Rundumprojektion mit 2D-Mikrospiegelarray*. Scientific Reports, Journal of the University of Applied Science Mittweida, 14th International Scientific Conference Mittweida, Nr. 10, 2000, Mittweida, Germany, 08.-11. November 2000, ISSN 1437-7624

6.2 Vorträge und Zeitschriftenartikel

[11] Schlegel, M.: *Neue Methoden zur Optimierung des Entwurfs heterogener Systeme mit VHDL-AMS.* Vortrag auf dem Doktorandenseminar des Zentrums für Mikrotechnologien der TU Chemnitz, November 2001

[12] Richter, A.: Poster und Programmpräsentation des Datenmanagementsystems zur 5. Chemnitzer Fachtagung Mikrosystemtechnik - Mikromechanik & Mikroelektronik; Chemnitz, 23./24. Oktober 2001

6.3 Studien- und Diplomarbeiten, Dissertationen

[13] Franke, M.: *Erstellung und Test von Modellen aktiver elektronischer Bauelemente in VHDL-AMS und Vergleich gegenüber SPICE-Modellen*. Diplomarbeit an der TU Chemnitz, Fakultät für Elektrotechnik und Informationstechnik, Professur Schaltungs- und Systementwurf, 2002

[14] Horn, T.: *Implementation der berechnung von Eigenwerten einer Matrix in VHDL*. Studienarbeit an der TU Chemnitz, Fakultät für Elektrotechnik und Informationstechnik, Professur Schaltungs- und Systementwurf, 2003

[15] Richter, D.: *Erarbeitung eines Konzeptes zur automatischen Konvertierung von MATLAB-Scripten nach VHDL*. Studienarbeit an der TU Chemnitz, Fakultät für Elektrotechnik und Informationstechnik, Professur Schaltungs- und Systementwurf, 2003

[16] Waldenburger, J.: *Rechnergestütztes Datenmanagementsystem für Entwurfs- und Charakterisierungsdaten mikromechanischer Komponenten*. Diplomarbeit, Chemnitz 1998

[17] Barthel, T.: *Eine Methode zur Spezifikationserfassung heterogener Systeme.* Chemnitz, Technische Universität, Fakultät für Elektrotechnik und Informationstechnik, Dissertationsschrift, Technische Universität Chemnitz, Fortschritt-Berichte VDI, Reihe 10, Nr. 599, 1999, ISBN 3-18-359910-4

[18] Eichhorn, K.: *Entwurf und Anwendung von ASICs für musterbasierte Fuzzy-Klassifikationsverfahren*. Dissertation, Technische Universität Chemnitz, 21. März 2000

6.4 Sonstige Literatur

[19] The Institute of Electrical and Electronics Engineers, Inc.: *IEEE Standard VHDL Language Reference Manual (Integrated with VHDL-AMS changes)*, IEEE Std. 1076.1, 1999, ISBN 0-7381-1640-8

[20] Christen, E.; Bakalar, K.; Dewey, A. M.; Moser, E.: *Analog and Mixed-Signal Modelling Using the VHDL-AMS Language*. 36th Design Automation Conference, http://www.eda.org/analog/ftp_files/documentation/tutdac99.pdf

[21] Haase, J.; Reitz, S.; Schwarz, P.: *FEM-basierte Generierung von Verhaltensmodellen.* 4. Chemnitzer Fachtagung Mikromechanik & Mikroelektronik, Chemnitz, 11., 12. Okt. 1999, ISBN 3-00-004902-9

[22] Schneider, A.; Schneider, P.; Bastian, J.: *MOSCITO-Ein modulares, internetbasiertes Programmsystem für die Optimierung von Mikrosystemen.* Statusseminar zum Verbundprojekt OMID Optimierung von Mikrosystemen für Diagnose- und Überwachungsanwendung, Bremen, 9., 10. Okt. 2001

[23] Heuschen, F.; Grimm, C.; Waldschmidt, K.: *Modellierung des Implementationsraumes im Analog/Digital Co-Design*. 3. ITG/GI/GMM-Workshop der Fachgruppen 3 und 4 der Kooperationsgemeinschaft „Rechnergesteuerter Schaltungs- und Systementwurf“, Forschungs-Report Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Systemen und Schaltungen, Frankfurt am Main, 28.–29. Feb., 1. März 2000. ISBN 3-8007-2524-X

[24] Teich, J.: *Digitale Hardware/Software-Systeme: Synthese und Optimierung*. Springer Verlag, 1997, ISBN 3-540-62433-3

[25] Philippow, E.: *Grundlagen der Elektrotechnik.* Akademische Verlagsgesellschaft Geest & Portig K.G., Leipzig, 1976

[26] Schlagenhaufer, F.: *Berechnung transienter Vorgänge auf verlustbehafteten Leitungen mit Feldanregung*. Dissertation an der Technischen Universität Hamburg-Harburg, 1994

[27] Dongarra, J.; Bunch, J.; Moler, C.; Stewart, G.: *LINPACK Users Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1978

[28] Anderson, A.; et al.: *LAPACK Users Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999

[29] Rosenberger, R.; Huss, S. A.: *A Systems Theoretic Approach to Behavioral Modeling and Simulation of Analog Functional Blocks*. IEEE/ACM Design, Automation and Test Conference Europe, Paris, 1998

[30] Hanna, J. P.; Hillman, R. G.: *A Common Basis for Mixed-Technology Micro-System Modeling*. International Conference on Modeling and Simulation of Microsystems MSM 99, San Juan, Puerto Rico, U.S.A., Apr. 19-21, 1999, ISBN 0-9666135-4-6

[31] Bennini, F.; Mehner, J.; Dötzel, W.: *A modal decomposition technique for fast harmonic and transient simulation of MEMS*. The International MEMS Workshop 2001, Singapore, July 4-6, 2001, ISBN 981-04-4165-7

[32] Achenbach, J.-J.: *Analoge und digitale Filter und Systeme*. BI-Wissenschaftsverlag, 1992, ISBN 3-411-15341-5

[33] P. P. Chen; *The Entity-Relationship-Model Toward a Unified View of Data*. ACM/TODS 1976

[34] MicroSim Corporation: *PSPICE A/D Reference Manual*. 1997

[35] Tietze, U.; Schenk, Ch.: *Halbleiter-Schaltungstechnik*. Springer Verlag, 1999

[36] Antognetti, P.; Massobrio, G.: *Semiconductor Device Modelling with Spice*. McGraw-Hill Book Company, 1988

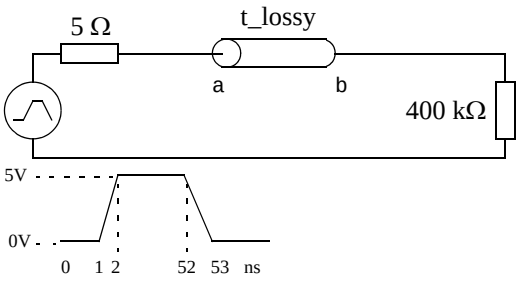
[37] Shichman, H. Hodges, D. A.: *Modeling and Simulation of Insulated-Gate Field-Effect Transistor Switching Circuits*. IEEE Journal of Solid-State Circuits, SC-3, 1968

[38] Reisch, M.: *Elektronische Bauelemente*. Springer Verlag 1997

[39] John, R.; Schaub, R.: *Werkzeugschwingungen im Ergebnis des Spanungsprozesses*. Dissertation an der Technischen Hochschule Karl-Marx-Stadt, 1975

[40] Bocklisch, S. F.: *Prozeßanalyse mit unscharfen Verfahren*. Verlag Technik, Berlin 1987

7. Bilder, Grafiken



mit:
len = 0.6 m,
R‘ = 24.4 Ω/m, L‘ = 616.3e-9 H/m,
G‘ = 3016e-6 S/m und C‘ = 31e-12 F/m

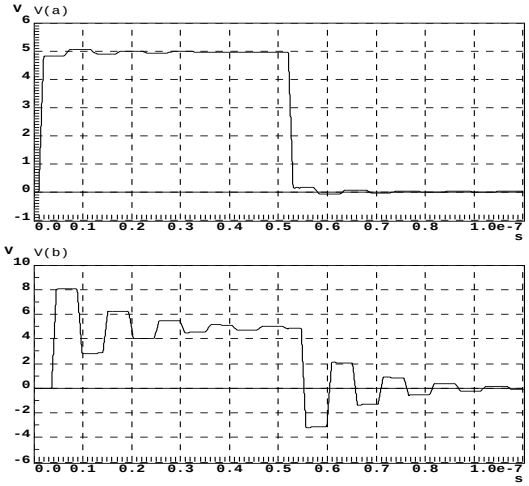
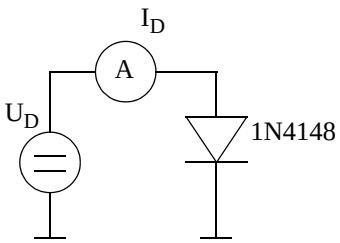


Bild 6 Testschaltung und Simulationsresultat der verlustbehafteten Leitung



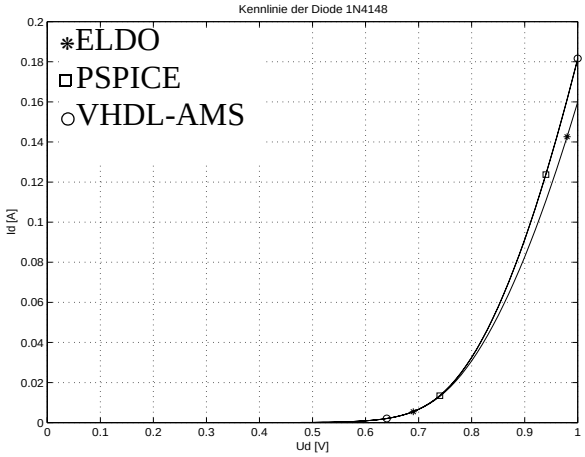


Bild 7 Testschaltung und Simulation des VHDL-AMS Modells der Halbleiterdiode

```
In VHDL realisierte Vektor- und Matritzenoperationen:
"+", "-", "*", "**", "/"
Funktionen auf Vektoren und Matritzen:
CROSS, SPAT -- Vektorprodukt, Spatprodukt
SIN, COS, TAN, COT, SINH, COSH, TANH, COTH, ARCSIN, ARCCOS, ARCTAN, ARCCOT
LOG, LOG10, LOG2, SQRT, CBRT, ABS, EXP, CEIL, FLOOR, ROUND, TRUNC, MOD
length    -- Länge eines Vektors
size      -- Ermittlung Anzahl Zeilen und Spalten einer Matrix
eye       -- Erstellen einer Matrix nach vorgegebener Ordnung
rot90     -- Rotieren einer Matrix um 90 Grad
xorm      -- xor Verknüpfung von zwei Matrizen, Ergebnis in einer dritten Matrix
ones      -- Erzeugen einer mit '1.0'en gefuellten Matrix
zeros     -- Erzeugen einer mit '0.0'en gefuellten Matrix
nonzeros  -- Liefert von 0.0 verschiedene Elemente einer Matrix
sum       -- Berechnung der Summe aller Elemente eines Vektors
trace     -- Summe der Diagonalelemente der Matrix
spones    -- Erzeugung einer neuen Matrix gleicher Struktur mit '1' an Stellen, die ungleich null sind
nnz       -- Ermittlung der Anzahl von Elemente einer Matrix ungleich null
max       -- spaltenweise Bestimmung des Maximumwertes einer Matrix
max       -- Vergleich der Elemente zweier Matrizen, Speichern des groesseren in einer neuen Matrix
mini      -- spaltenweise Bestimmung des Minimumwertes einer Matrix
mini      -- Vergleich der Elemente zweier Matrizen, Speichern des kleineren in einer neuen Matrix
mean      -- Erzeugung eines Vektors mit jeweils dem Mittelwert der jeweiligen Spalte
sort      -- Sortiert Matrix spaltenweise in aufsteigender Reihenfolge
median    -- Zerlegung der Matrix in Vektoren(Spalten), Sortieren eines Vektors,
reshape   -- Wiedergabe eine XxY-Matrix als MxN-Matrix
nextpow2  -- Finden der einem gegebenen Wert naechsthoehere 2^n-Potenz
cumprod   -- Produkt aller Elemente einer Matrix
cumsum    -- Summe aller Elemente einer Matrix
diff      -- Differenz aller Elemente einer Matrix
polyval   -- Polynomberechnung
fliplr    -- Spiegeln einer Matrix links/rechts(
flipud    -- Spiegeln einer Matrix oben/unten
tril      -- Alle Elemente auf und unterhalb der k-ten Diagonale sind '1', darueber '0'
triu      -- Alle Elemente auf und oberhalb der k-ten Diagonale sind '1', darunter '0'
makvec    -- Erzeugung eines Vektors bestimmter Laenge, in den die Indizes der Elemente
find      -- Rueckgabe eines Vektors, der alle Elemente ungleich 0.0 eines gegebenen Vektors beeinhaltet
pow2      -- Berechnung von 2^n mit gegebenem n fuer jedes Element einer Matrix
stad      -- Berechnung der Standardabweichung aller Elemente einer Matrix
conv      -- Verknuepfung der Vektoren zu einem neuen Vektor: c(k)=summe[a(j)b(k+1-j)]
diag      -- Matrixdiagonale normal in Vektor umwandeln
det       -- Berechnung der Determinante einer Matrix mit Hilfe des lu-Algorithmuses
alle, any -- Erzeugung eines Vektors mit Laenge der Spaltenanzahl der Matrix
compan    -- Erzeugung der Companion Matrix zu einem vorgegebenen Vektor
lu        -- LU - Faktorisierung einer Matrix
angle     -- Rueckgabe des Phasenwinkels fuer jedes Element einer komplexen Matrix
conj      -- Rueckgabe der Komplekonjugierten fuer jedes Matrixelement
imag      -- Rueckgabe des Imaginaerteils fuer jedes Matrixelement
realt     -- Rueckgabe des Realteils fuer jedes Matrixelement
hankel    -- Erzeugt eine Hankel-Matrix
hilb      -- Erzeugt eine Hilbert-Matrix
invhilb   -- Erzeugt eine inverse Hilbert-Matrix
toeplitz  -- Erzeugt eine Toeplitz-Matrix
vander    -- Erzeugt eine Vandermonde-Matrix
kron      -- Berechnet Kronecker Tensor Produkt
deconv    -- Polynomdivision
unwrap    -- Korrigiert Phasenwinkel eines Vektors
gamma     -- Gamma-Funktion
beta      -- Beta-Funktion
erf       -- Error-Funktion
erfinv    -- inverse der Error-Funktion
chol      -- Cholesky-Faktorisierung
corrcoef  -- Berechnung von Korrelationskoeffizientem
eigen     -- Berechnung der Eigenwerte einer Matrix
```

Bild 8 Übersicht über einen Teil der in VHDL realisierten mathematischen Funktionen aus MATLAB

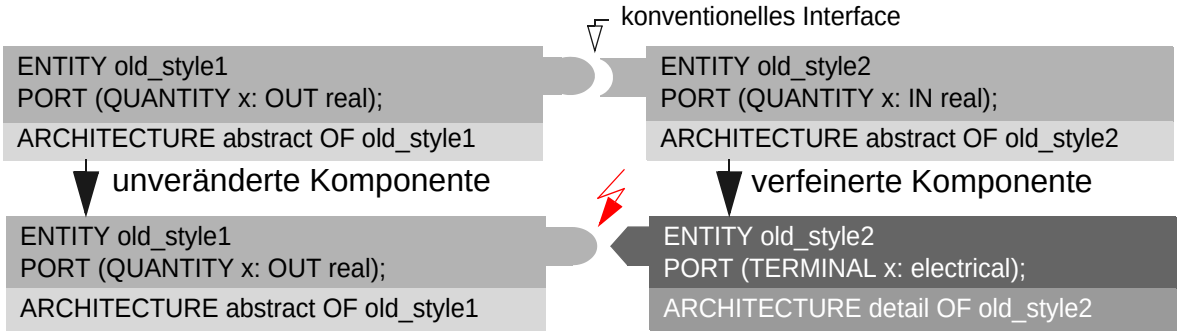


Bild 9 Problem beim Verbinden von Modellen mit unterschiedlichen Abstraktionsgraden

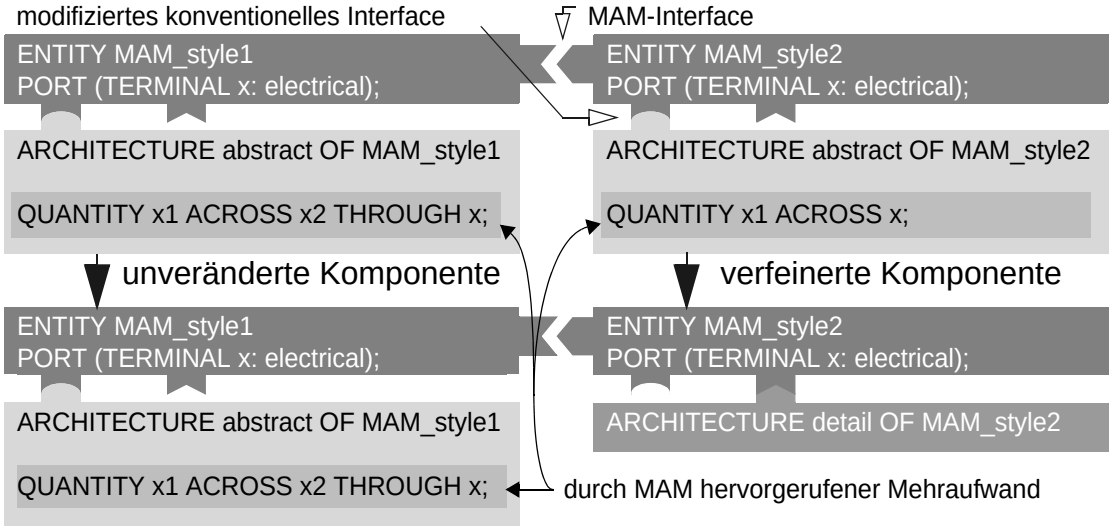


Bild 10 Verbindung analoger Modelle mittels MAM

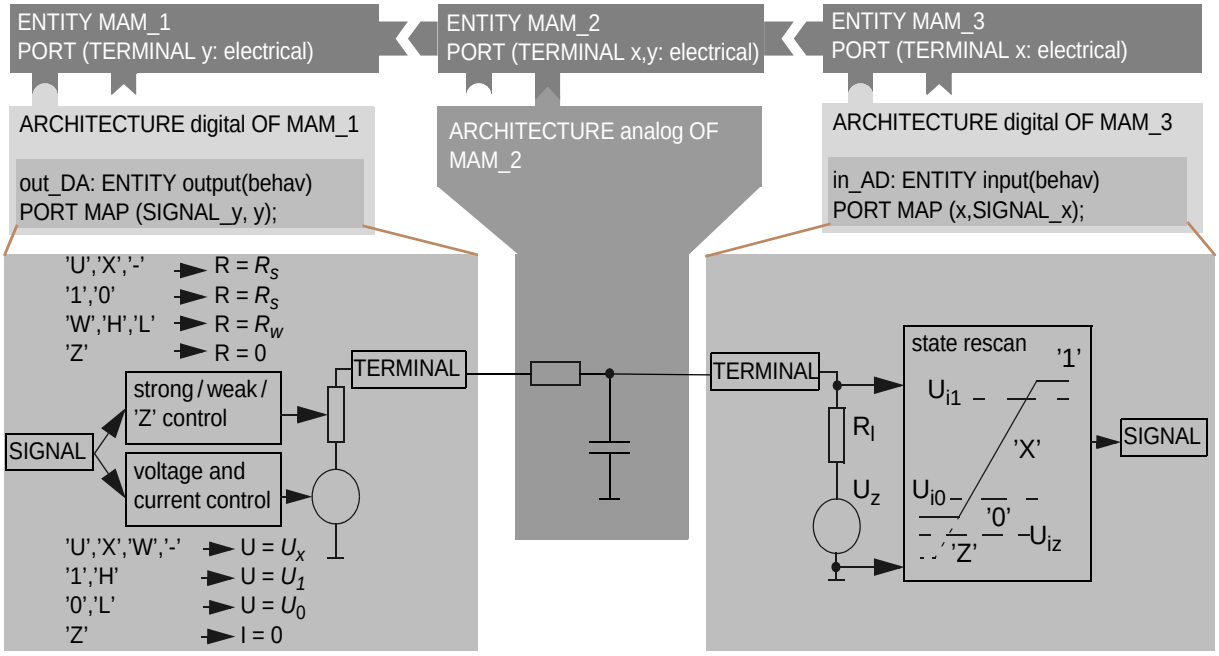


Bild 11 Verbindung analoger und digitaler Ports mittels MAM

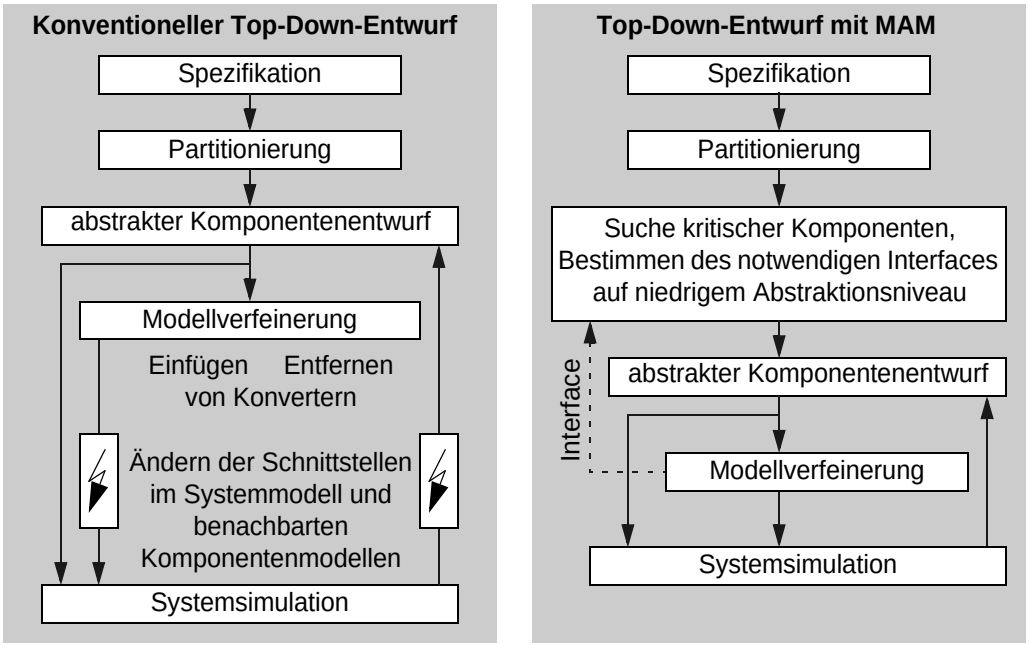


Bild 12 Top Down Entwurf ohne und mit MAM

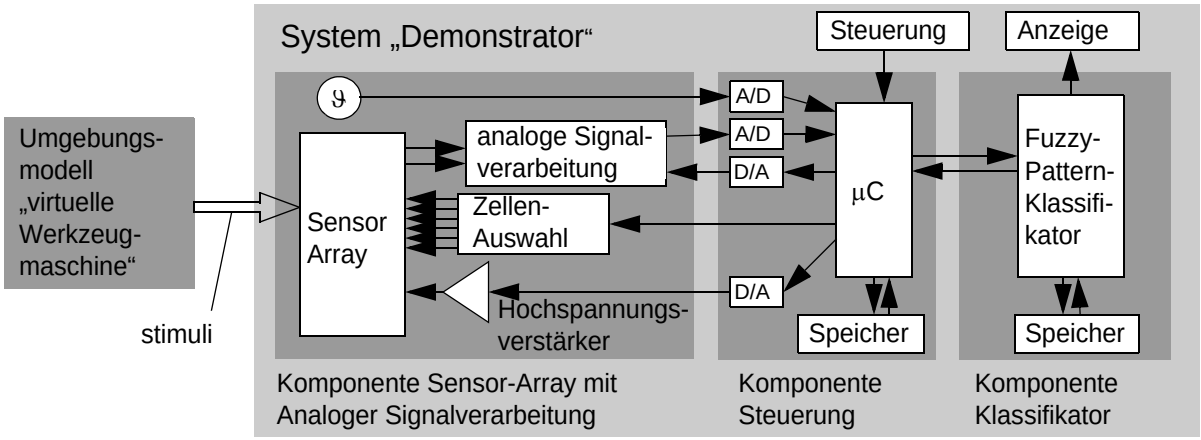


Bild 13 Systemkonzept und Partitionierung des Demonstrators

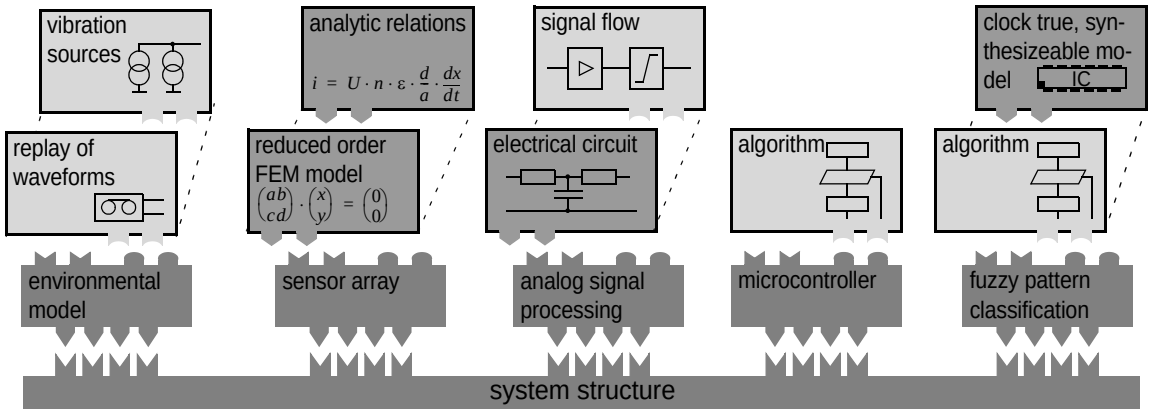


Bild 14 Anwendung der MAM im Systementwurf des Demonstrators

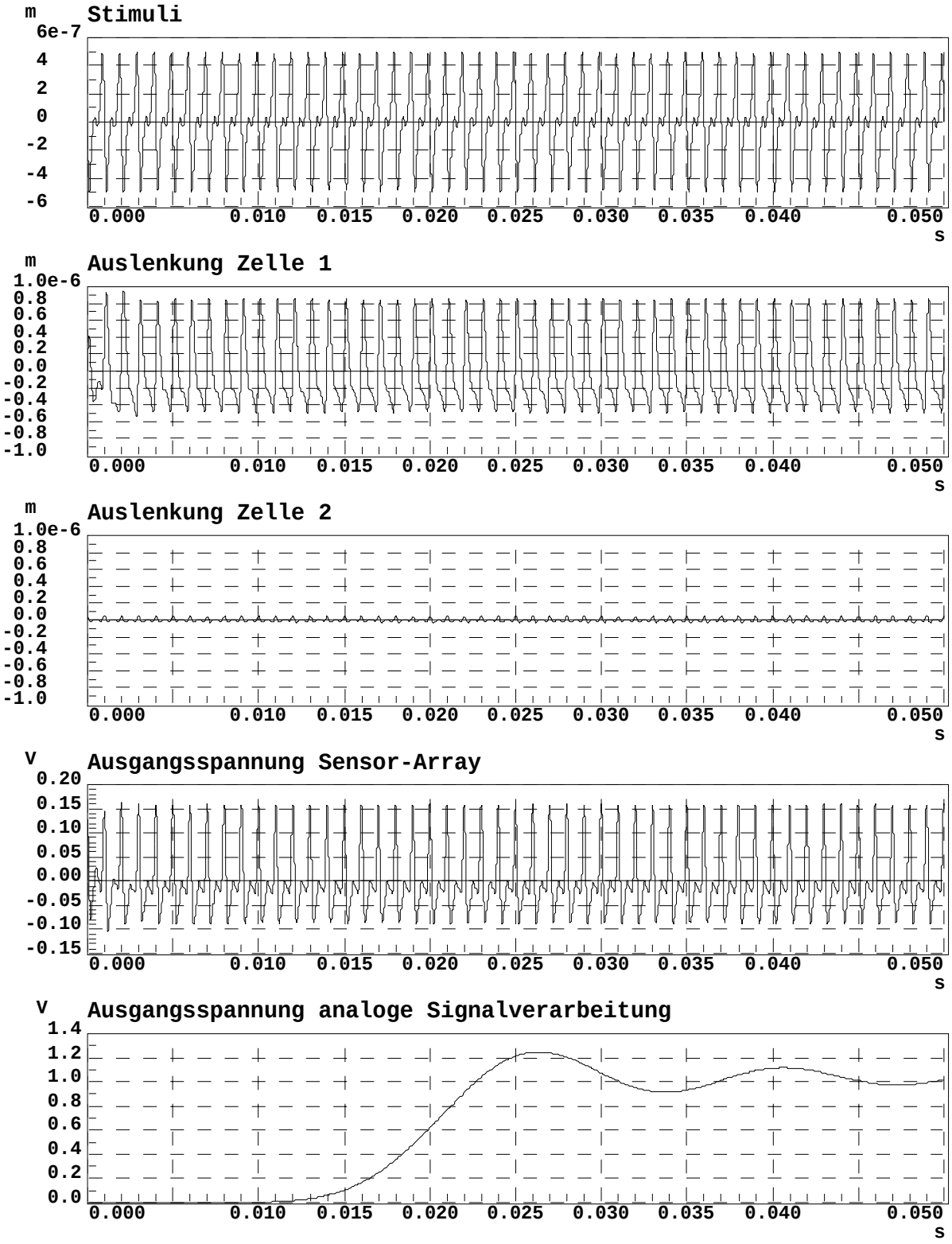


Bild 15 Systemsimulation des Demonstrators

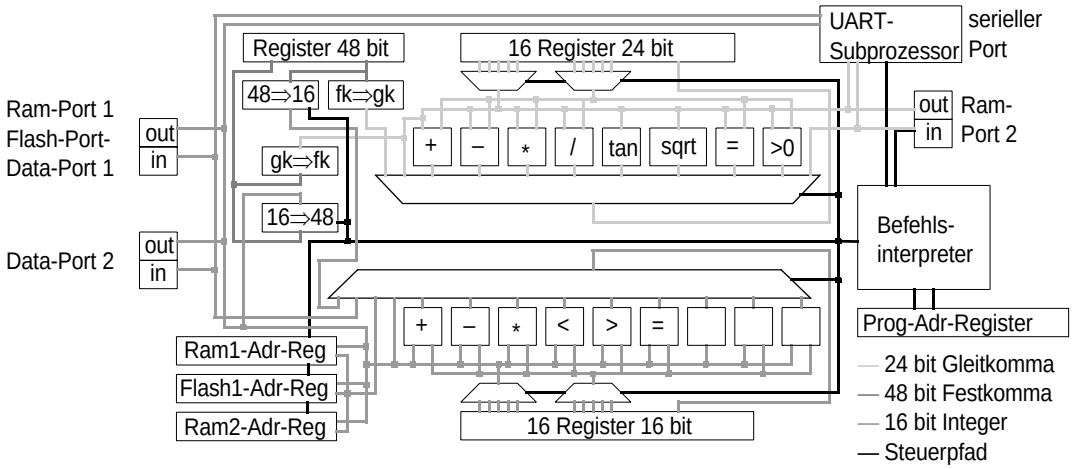


Bild 16 Systemstruktur des Fuzzy-Klassifikationsprozessors

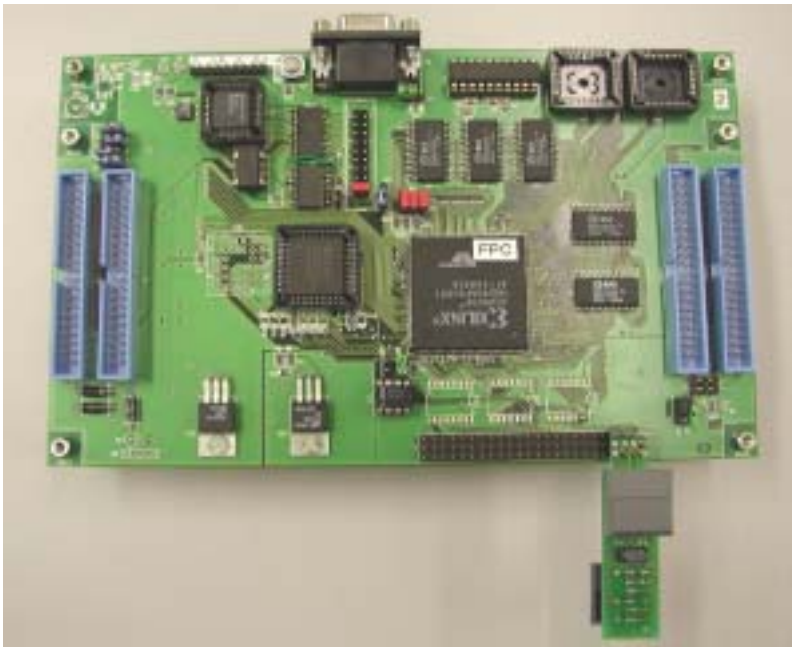


Bild 17 Foto des Fuzzy-Pattern-Klassifikators

Sensor-Array (A4)

Sensor-Steuerung und analoge Signalverarbeitung (A4)

Fuzzy-Pattern-Klassifikator (A2)

Bild 18 Foto des Gesamtsystems „Demonstrator“

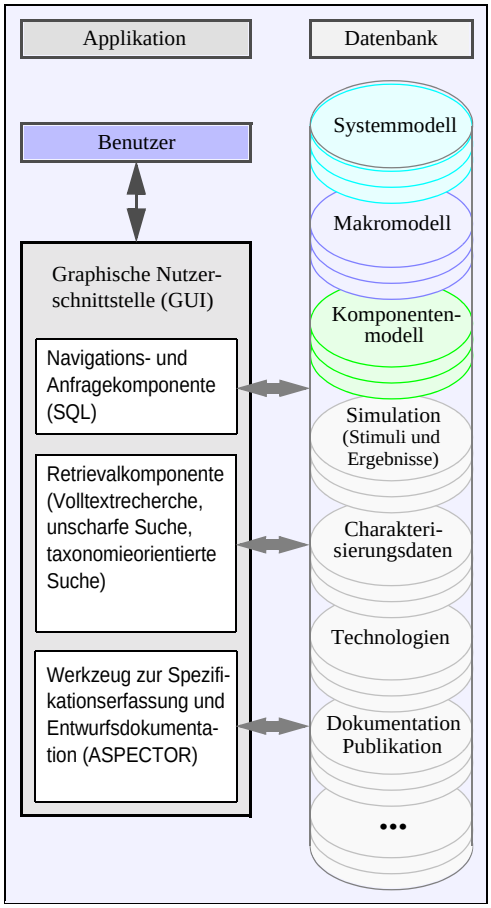


Bild 19 prinzipieller Aufbau des DMS

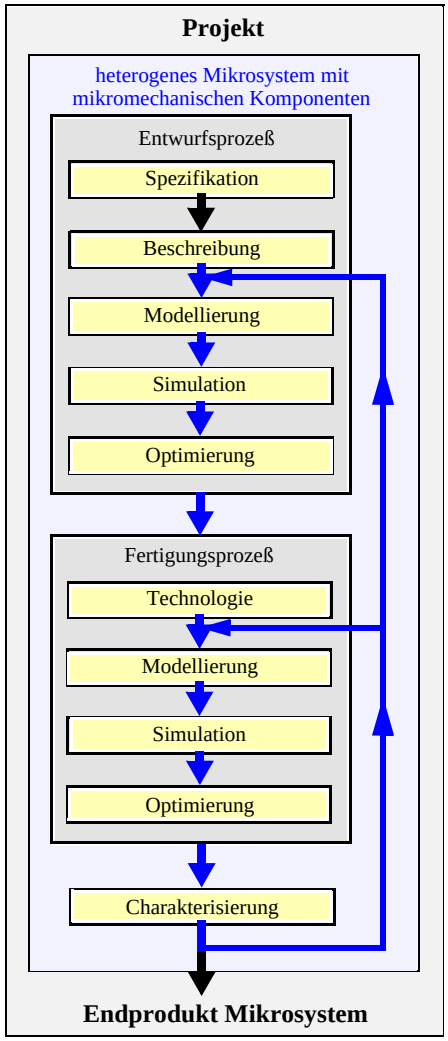


Bild 20 Abbildung des Entwurfs- und Fertigungsprozesses einer mikromechanischen Komponente im DMS

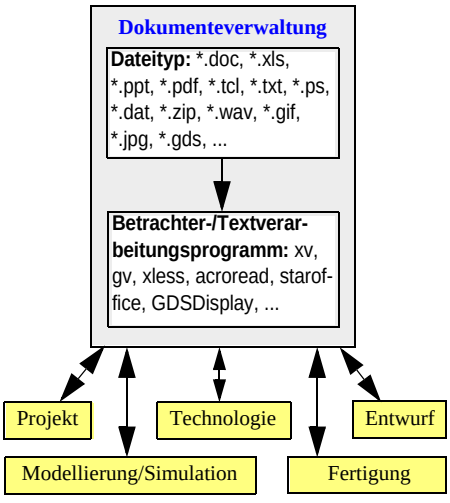


Bild 21 Dokumentzuordnungen zu den Kategorien des DMS

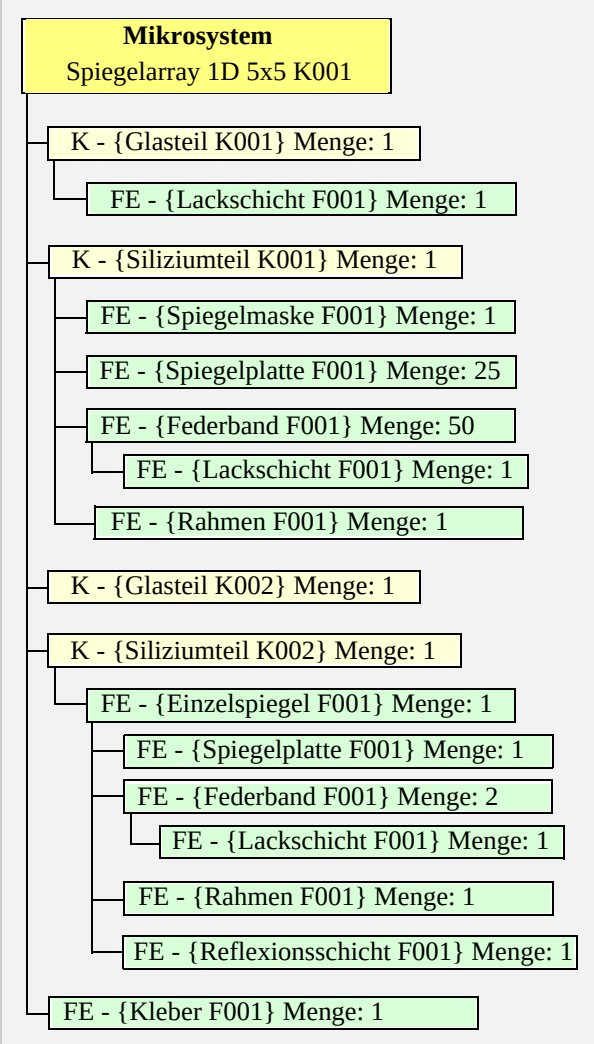


Bild 23 Struktur bzw. Stückliste eines Mikrosystems am Beispiel eines Spiegelarrays 1D 5x5

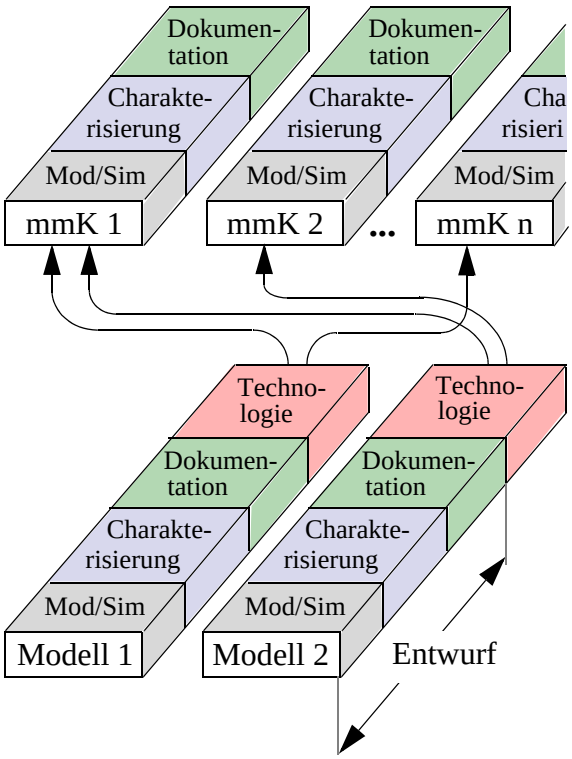


Bild 22 Zuordnung von Entwurfsmodellen zu verschiedenen mikromechanischen Komponenten (mmK) (Mod/Sim = Modellierung und Simulation)

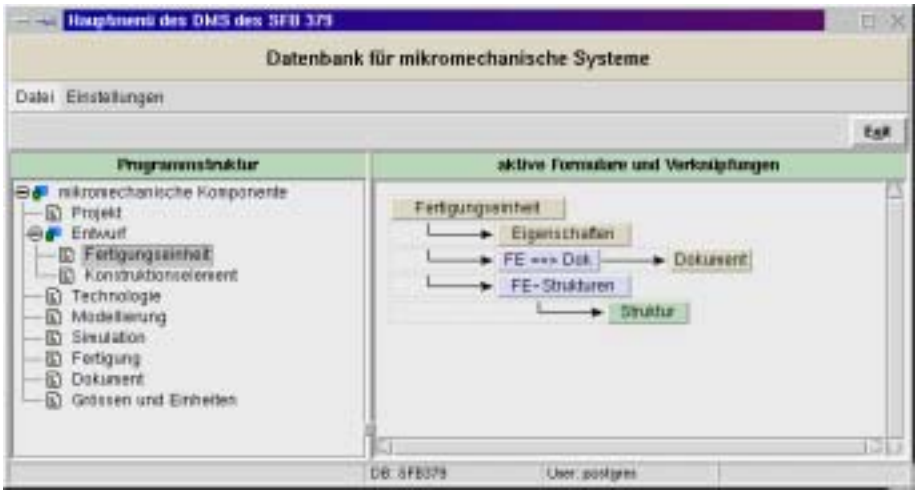


Bild 24 grafische Darstellung einer Formularverknüpfung im Hauptmenü des DMS

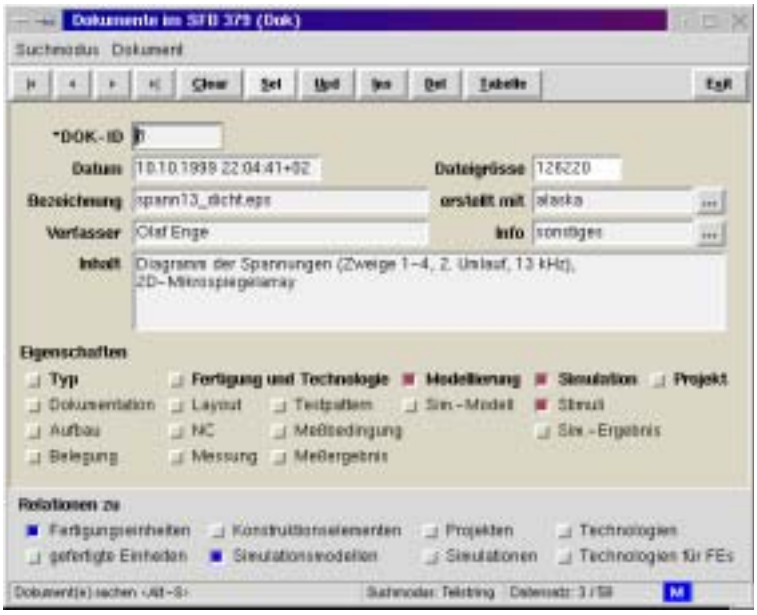


Bild 25 Entitäten-Formular zur Dokumentenverwaltung

Bild 26 Relationship-Fenster der Fertigungseinheit - Konstruktionselemente - Struktur. Das Attribut Menge dient der näheren Beschreibung der Verknüpfung, über den Schalter Baugruppe kann die gesamte Struktur der Fertigungseinheit und ihrer Konstruktionselemente graphisch dargestellt werden

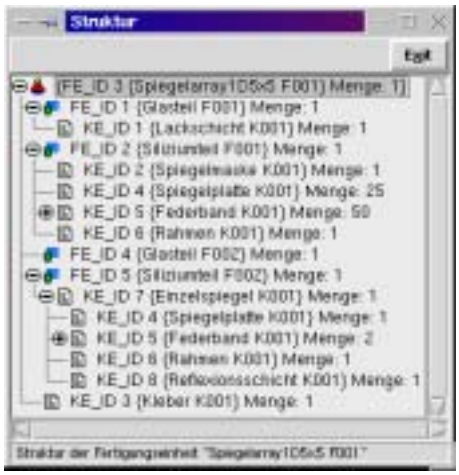


Bild 27 Baumstrukturdarstellung einer Fertigungseinheit im DMS