

**Siegfried Puga-Reichle**

# Ansätze einer High-Level-Synthese in der Electronic Design Automation

**Diplomarbeit**

# BEI GRIN MACHT SICH IHR WISSEN BEZAHLT



- Wir veröffentlichen Ihre Hausarbeit, Bachelor- und Masterarbeit
- Ihr eigenes eBook und Buch - weltweit in allen wichtigen Shops
- Verdienen Sie an jedem Verkauf

Jetzt bei [www.GRIN.com](http://www.GRIN.com) hochladen  
und kostenlos publizieren



## **Bibliografische Information der Deutschen Nationalbibliothek:**

Bibliografische Information der Deutschen Nationalbibliothek: Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de/> abrufbar.

Dieses Werk sowie alle darin enthaltenen einzelnen Beiträge und Abbildungen sind urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsschutz zugelassen ist, bedarf der vorherigen Zustimmung des Verlanges. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen, Auswertungen durch Datenbanken und für die Einspeicherung und Verarbeitung in elektronische Systeme. Alle Rechte, auch die des auszugsweisen Nachdrucks, der fotomechanischen Wiedergabe (einschließlich Mikrokopie) sowie der Auswertung durch Datenbanken oder ähnliche Einrichtungen, vorbehalten.

Copyright © 2005 GRIN Verlag GmbH  
ISBN: 9783656998938

**Siegfried Puga-Reichle**

**Ansätze einer High-Level-Synthese in der Electronic  
Design Automation**

Examicus - Verlag für akademische Texte

Der Examicus Verlag mit Sitz in München hat sich auf die Veröffentlichung akademischer Texte spezialisiert.

Die Verlagswebseite [www.examicus.de](http://www.examicus.de) ist für Studenten, Hochschullehrer und andere Akademiker die ideale Plattform, ihre Fachtexte, Studienarbeiten, Abschlussarbeiten oder Dissertationen einem breiten Publikum zu präsentieren.

Ansätze einer  
High-Level-Synthese  
in der  
Electronic Design Automation

Diplomarbeit

Zur Erlangung des akademischen Grades

Dipl.-Ing. (FH)

Vorgelegt an der  
Fachhochschule Konstanz  
Fachbereich Elektrotechnik und Informationstechnik

---

**Diplomand:**  
**Siegfried Puga-Reichle**

**Fachhochschule Konstanz**  
**FB: Elektrotechnik und**  
**Informationstechnik**

**Erstellt in der Zeit**  
**Von 01.03.05 bis 24.10.2005**

Printed in Germany

Copyright © 2005 Siegfried Puga-Reichle

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), without the prior written permission of the author.

## Inhaltsverzeichnis

---

<b>1</b>	<b>Einleitung / Überblick .....</b>	<b>6</b>
<b>2</b>	<b>Hintergrund der Hardwarebeschreibungssprachen.....</b>	<b>7</b>
2.1	Motivation: IC-Entwurfsverfahren.....	7
2.2	Bestehende Probleme des heutigen Systementwurfs .....	8
2.3	Entwurfssichten .....	11
2.4	Entwurfsebenen.....	12
2.4.1	Systemebene.....	12
2.4.2	Algorithmische Ebene .....	12
2.4.3	Register-Transfer-Ebene .....	12
2.4.4	Logikebene .....	12
2.4.5	Schaltkreisebene.....	13
2.5	Hardware und Software Lösungen.....	14
2.6	Bedeutung der HW/SW-Architektur auf den einzelnen Abstraktionsebenen .....	16
2.6.1	Abstraktion .....	16
2.6.2	Beschreibungssprachen / Werkzeuge.....	16
2.7	Standardtechnologie ASIC.....	17
2.8	Besondere Bedeutung/Stellenwert des FPGA.....	19
2.9	Herstellungsprozess von digitalen integrierten Schaltkreisen.....	21
2.10	Probleme der Entwurfskomplexität hoch integrierter Systeme.....	23
2.11	Folgerungen zur marktgerechten Verkürzung der Entwurfsdauer.....	24
2.12	Effizienzsteigerung im Entwurf mittels grafischer HDL – Programmierung.....	26
2.13	Moderner Design Flow mittels grafischer HDL .....	27
<b>3</b>	<b>HDL – Design - Sprachen .....</b>	<b>28</b>
3.1	VHDL.....	28
3.1.1	Aufbau.....	28
3.1.2	Anwendung der VHDL .....	29
3.2	Verilog.....	30
3.2.1	Unterschiede von VHDL und Verilog .....	30
3.3	Sonstige HDL-Sprachen.....	31
<b>4</b>	<b>Systembeschreibungssprachen (System-Level-Entwurf.....</b>	<b>32</b>
4.1	SystemC .....	32
4.1.1	SystemC im Überblick .....	34
4.1.2	Vergleich VHDL und SystemC.....	34
4.2	SpecC .....	38
4.3	Superlog .....	38
4.4	Handel-C .....	39
4.5	PSL - Property Specification Language .....	39
4.6	SystemVerilog.....	39
4.7	VHDL-200x .....	40
4.8	Verilog-AMS.....	40
4.9	VHDL-AMS.....	41
4.10	Matlab/Simulink.....	42

<b>5</b>	<b>High-Level-Synthese-Tools.....</b>	<b>43</b>
5.1	Speedchart .....	44
5.2	Visual HDL von Summit.....	44
5.3	Visual Elite / FastC von Summit.....	45
5.4	Statemate von I-Logix .....	46
5.5	HDL Designer Series von Mentor Graphics .....	47
5.6	DSP-Builder von Altera .....	48
5.7	DK Design Suite von Celoxica .....	49
5.8	System Generator for DSP von Xilinx .....	50
5.9	AccelChip DSP Synthesis von AccelChip.....	51
5.10	Synplify DSP von Synplicity .....	52
5.11	Discovery Verification Platform von Synopsys.....	53
5.12	Catapult C Synthesis von Mentor Graphics .....	54
5.13	Agility Compiler von Celoxica .....	55
5.14	CoDeveloper von Impulse Accelerated Technologies .....	56
5.15	Filter Design HDL Coder 1.2.....	57
<b>6</b>	<b>Simulations-Programme .....</b>	<b>58</b>
<b>7</b>	<b>Design-Entry-Tools .....</b>	<b>62</b>
<b>8</b>	<b>Weiterführende EDA-Organisationen und LINKs.....</b>	<b>65</b>
8.1	EDA-Organisationen .....	65
8.2	Konferenzen .....	67
8.3	Online-Elektronik-Magazine.....	68
<b>9</b>	<b>Automatische Synthese von VHDL mit dem DSP-Builder.....</b>	<b>69</b>
9.1	DSP-Builder .....	69
9.1.1	System Voraussetzungen.....	69
9.1.2	Installationsprozess unter Windows.....	70
9.1.3	Arbeiten mit dem DSP-Builder .....	71
9.1.4	Einführendes Beispiel .....	72
9.2	Quartus II.....	74
9.3	Verifikation des DSP-Builder mit händischem VHDL- Codes .....	75
9.3.1	Beispiel 1: Addier- und Multiplizier-Werk.....	75
9.3.2	Beispiel 2: Zähler .....	78
9.3.3	Beispiel 3: 8 Point Radix 8 DIT FFT .....	81
9.4	Ergebnis der High-Level Synthese.....	83
<b>10</b>	<b>Zusammenfassung.....</b>	<b>85</b>
<b>11</b>	<b>Abbildungsverzeichnis .....</b>	<b>86</b>
<b>12</b>	<b>Tabellenverzeichnis .....</b>	<b>89</b>
<b>13</b>	<b>Stichwortverzeichnis .....</b>	<b>90</b>
<b>14</b>	<b>Quellenverzeichnis .....</b>	<b>96</b>

<b>ANHANG A: SystemVerilog Support .....</b>	<b>98</b>
<b>ANHANG B: DSP-Builder Design-Code .....</b>	<b>101</b>
ANHANG B I: VHDL-Code des Beispiel 1 (AddMultWerk) aus Kapitel 9.3.1 .....	101
ANHANG B I-I: Händischer VHDL-Code des Addier-/Multiplizier- Werk .....	101
ANHANG B I-II: Fitter-Ergebnis-File des AddMultWerk (händisch generiertes VHDL) .....	101
ANHANG B I-III: Timing-Ergebnis-File des AddMultWerk (händische generiertes VHDL) .....	102
ANHANG B I-IV: Aus dem DSP-Builder generierter VHDL-Code des Addier- /Multiplizier- Werk .....	103
ANHANG B I-V: Fitter-Ergebnis-File des AddMultWerk (DSP-Builder VHDL) .....	105
ANHANG B I-VI: Timing-Ergebnis-File des AddMultWerk (DSP-Builder generiertes VHDL) .....	105
ANHANG B II: VHDL-Code des Beispiel 2 (Zähler) aus Kapitel 9.3.2 .....	107
ANHANG B II-I: Händischer VHDL-Code des Zählers .....	107
ANHANG B II-II: Fitter-Ergebnis-File des Zählers (händische generiertes VHDL)....	108
ANHANG B II-III: Timing-Ergebnis-File des Zählers (händische generiertes VHDL)	108
ANHANG B II-IV: Aus dem DSP-Builder generierter VHDL-Code des Zählers .....	110
ANHANG B II-V: Fitter-Ergebnis-File des Zählers (DSP_Builder generiertes VHDL) .....	112
ANHANG B II-VI: Timing-Ergebnis-File des Zählers (DSP_Builder generiertes VHDL) .....	112
ANHANG B III: VHDL-Code des Beispiel 3 (8er FFT) aus Kapitel 9.3.3 .....	114
ANHANG B III-I: Aus dem DSP-Builder generierter VHDL-Code der 8er-FFT .....	114
ANHANG B III-II: Fitter-Ergebnis-File der 8er-FFT (DSP_Builder generiertes VHDL) .....	131
ANHANG B III-III: Timing-Ergebnis-File der 8er-FFT (DSP_Builder generiertes VHDL) .....	131
<b>ANHANG C: Visual HDL: Beispielprogramme, VHDL-Code automatisch synthetisiert aus grafischen Spezifikationen. [27] .....</b>	<b>133</b>
ANHANG C I: Multiplizierwerk .....	133
ANHANG C II: Steuerpfad synchron .....	136
ANHANG C III: Steuerpfad asynchron .....	139
ANHANG C IV: Ampel.....	142

## 1 Einleitung / Überblick

Die Gesellschaft wandelt sich immer mehr zu einer Informations- und Kommunikationsgesellschaft. Die Schlüsseltechnologie in dieser Entwicklung stellt die Mikroelektronik dar. Die Mikroelektronik ist heute allgegenwärtig und aus unserer Gesellschaft nicht mehr weg zu denken und sie gewinnt immer noch mehr an Bedeutung in allen Lebenslagen.

Im Jahre 2010 werden über 5 Milliarden Transistoren auf einem einzigen Chip integrierbar sein und die Entwicklungszyklen werden aus Wettbewerbsgründen immer kürzer. Das Entwurfsteam muss trotz der Komplexitätsexplosion dem Kosten- und Zeitdruck entgegenwirken. Aufgrund dessen muss sich die Entwurfsproduktivität in jedem Jahr mehr als verdoppeln, will sie der Chipentwicklung folgen.

Der ungebrochene Technologiefortschritt hat dazu geführt, dass heute ganze Systeme aus mehreren Prozessoren und komplexen Verbindungsstrukturen auf einem einzelnen Chip gefertigt werden können (*SoC*). Um die Komplexität dieser Systeme und mögliche Anwendungen kontrollieren zu können, bedarf es einer Automatisierung des Entwurfs auch auf höheren Entwurfsebenen (High-Level-Synthese). Die Automatisierung des Entwurfs (Electronic Design Automation, EDA) stellt deshalb den Schlüssel zur Mikroelektronik und damit zu den Systemen der Zukunft dar.

Heutiger Standard des Schaltungsentwurfs ist die Hardwarebeschreibung durch Hardwarebeschreibungssprachen (*HDL*), die durch *CAE*-Werkzeuge (Computer Aided Engineering) zur Schaltungssimulation und –synthese benutzt werden. Dabei dient die *Simulation* der Überprüfung der Funktion des Entwurfs und die *Synthese* der Umsetzung der Beschreibung in eine Netzliste für die Implementierung der Schaltung auf die gewählte Zieltechnologie wie ASICs oder FPGAs. Als Hardwarebeschreibungssprachen haben sich weltweit die beiden Sprachen Verilog und VHDL etabliert.

Die *Simulation* und *Verifikation* gewinnt zunehmend immer mehr an Bedeutung, je komplexer die Schaltungen werden. Es ist nicht mehr möglich Signale Takt für Takt auf ihre Richtigkeit zu überprüfen, sondern es müssen neue Verifikationsstrategien gefunden werden. Einer davon ist die Entwicklung von *HDVL*- Sprachen (Hardware Description and Verification Language).

Diese Arbeit beschäftigt sich daher mit den bestehenden Problemen im Systementwurf und behandelt neue Sprachen und Werkzeuge die eine High-Level-Synthese ermöglichen.

Im ersten Teil sollen die Problematik näher dargestellt werden und einen kleinen Background geschaffen werden. Nachfolgend sollen neue *HDL*-Sprachen vorgestellt werden, bevor ein Überblick über High-Level-Synthese-Tools gegeben werden soll.

Eine Sammlung derzeitiger Simulatoren und Back-End-Werkzeuge sollen die Tools abschließend ergänzen. Da die Herausforderung der Entwurfskomplexität nicht von einem einzelnen Unternehmen beherrschbar ist, wird auf weiterführende EDA-Organisationen und Quellen eingegangen.

Abschließend soll anhand des DSP-Builders von Altera untersucht werden wie eine High-Level-Synthese aus Matlab/Simulink nach VHDL funktioniert und wie Effizient sie arbeitet.

## 2 Hintergrund der Hardwarebeschreibungssprachen

Seit vielen Jahren besteht eine große Dynamik im Entwurf von Digitalschaltungen und die Entwurfsverfahren wandeln sich ständig. Früher wurden integrierte Schaltungen mit graphischen Rechner-Werkzeugen in Form von Schaltplänen manuell erstellt. Dazu mussten die Grundelemente der Schaltung - logische Gatter, die in einer Bibliothek zur Verfügung stehen - bzw. deren Symbole ausgewählt, auf einem Schaltplan platziert und anschließend miteinander verbunden werden. Auf diese Weise konnten zunächst einfache Module generiert und mit diesen wiederum komplexere Schaltungen zusammengesetzt werden, was einem sog. Bottom-Up Entwurf entspricht. Dieser Prozess konnte bei umfangreichen Schaltungen sehr viel Zeit in Anspruch nehmen. Darüber hinaus war ein derartiger Schaltungsentwurf wenig flexibel gegenüber Änderungen, die wiederum ein mühsames Umzeichnen der Pläne erforderlich machten.

### 2.1 Motivation: IC-Entwurfsverfahren

Die heutige Situation beim Entwurf elektronischer Systeme erfordert jedoch den Umgang mit immer komplexeren Aufgabenstellungen, deren Integration durch die gestiegenen Packungsdichten ermöglicht wird. Um dem wachsenden Konkurrenzdruck und den Anforderungen der Kunden genügen zu können, ist zusätzlich eine kurze Entwicklungszeit ein entscheidender Faktor. Auch wird versucht, einmal generierte Funktionsblöcke bzw. Module in neuen Systemen wieder zu verwenden, was eine technologieunabhängige Beschreibung erforderlich macht.

Durch diese Randbedingungen wird heute die Entwicklung von Digitalschaltungen streng methodisch als Top - Down Entwurf vorgenommen. Zum Einsatz kommen hierbei Hardware-Beschreibungssprachen (*HDL* - Sprachen), die eine Modellierung des Systems auf vielen Abstraktionsebenen ermöglichen. Bedingt durch die beim Top - Down Entwurf ständig durchgeführte, schrittweise Verfeinerung müssen, angefangen von der Systemspezifikation über eine Beschreibung der Algorithmen und deren Aufteilung in Funktionsblöcke bis hin zu einer Gatternetzliste, alle Ebenen mit einer solchen Sprache realisiert werden können. Wichtiger Punkt im heutigen Entwurfsprozess ist die Unterstützung durch Synthesewerkzeuge, die auf der Grundlage einer Verhaltensbeschreibung eine Gatternetzliste automatisch erstellen. Dies setzt jedoch eine normierte Beschreibungssprache voraus. Des Weiteren soll ein damit modelliertes System in verschiedenen Abstraktionsebenen simulierbar sein um schon in einem frühen Stadium der Entwicklung Systemfehler zu entdecken.

Es gibt viele verschiedene Ansätze all diese Forderungen in einer Sprache zu vereinen. Eine davon ist die Hardwarebeschreibungssprache VHDL. VHDL (**V**HSIC **H**ardware **D**escription **L**anguage; VHSIC (**V**ery **H**igh **S**peed **I**ntegrated **C**ircuit)) erfüllt all diese Anforderungen. Es verfügt über Konstrukte zur Beschreibung von nebenläufigem oder sequentiellm Verhalten von Digitalschaltungen mit oder ohne Timing auf einer Vielzahl von Abstraktionsebenen. Auch besteht die Möglichkeit, durch Verschalten von Submodulen einen hierarchischen Entwurf aufzubauen. Durch die Normierung als *IEEE* Standard unterstützen alle wichtigen Entwurfswerkzeuge diese Sprache weshalb sie sich auch gut als Format zum Datenaustausch eignet. [1]

## 2.2 Bestehende Probleme des heutigen Systementwurfs

Elektronikkomponenten und die daraus aufgebauten informationsverarbeitenden Komponenten sehen sich ständig wachsenden Anforderungen entgegen. Gleichermäßen bezieht sich dies auch auf die Leistungsfähigkeit ihrer Prozessschnittstellen, die Komplexität und Verarbeitungsgeschwindigkeit der realisierten Steuer-, Regler- und Signalverarbeitungsfunktionen sowie die Flexibilität einer integrierten Netz- oder Systemschnittstelle. Ursachen liegen darin, dass bei vielen technisch hochwertigen Produkten eine Verdrängung mechanischer, elektrischer oder fluidischer Lösungen durch elektronische und informationsverarbeitende Systeme erfolgt. Umgekehrt bedeutet dies, dass viele mechanische, elektrische oder fluidische Funktionen, die von ihren physikalischen Phänomenen meistens als zeit- und amplitudenkontinuierliche Größen zu beschreiben sind, in informationsverarbeitende Komponenten möglichst wirklichkeitsgetreu abgebildet werden müssen.[4]

Rapide zunehmende Komplexität, Funktionsvielfalt und Leistungsanforderungen von elektronischen Systemen finden sich nahezu in jedem Bereich. Nachfolgend soll ein kleiner Überblick über die gestiegene Komplexität von Systemen unterschiedlicher Bereiche aufgezeigt werden.

### Bereich

### Anforderungs- spektrum

#### Automobilelektronik: (Abb. 2-1)

- Echtzeit,
- Ausfallsicherheit,
- Flexibilität,
- Nachrüstbarkeit,
- Hohe Performanz,
- Niedrige Kosten, ...

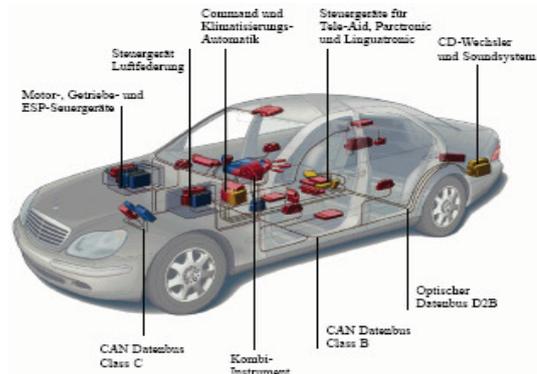


Abb. 2-1 30% der Wertschöpfung und 90% der Innovationen resultieren aus Elektronik und Software. Ca. 70 teilweise vernetzte Steuergeräte pro high-end Fahrzeug

#### Mobilkommunikation: (Abb. 2-2)

- Funktionsvielfalt
- (PC-) Rechenleistung
- GByte Speicherkapazität
- Hohe Integrationsdichte
- Geringe Verlustleistung



Abb. 2-2 Mobiltelefon, Megapixel-Kamera, www-Browser, MP3-Player, PDA, Diktiergerät, Computerkonsole, ...

**Unterhaltungselektronik:**  
(Abb. 2-3)

- (Supercomputer-) Rechenleistung
- Vernetzung,
- „Plug & Play“
- Miniaturisierung,
- geringe Verlustleistung und Kosten



Abb. 2-3 Unterhaltungselektronik wird immer kleiner, Leistungsfähiger und billiger

**Supercomputer on a Chip:**  
(Abb. 2-4)

- 2x64-bit *PPUs*
- 8x SPEs mit 32 Gops
- 4.6 GHz Takt, 234 M Transistoren, 90 nm SOI *CMOS*
- 76 GBytes/s I/O
- 25 GBytes/s Speicherbandbreite

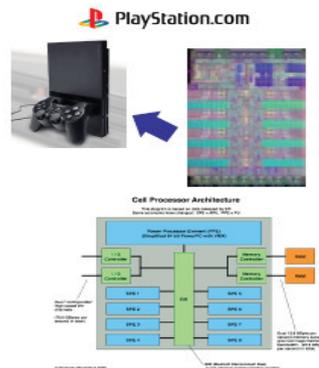


Abb. 2-4 Leistungsfähige Systeme werden immer kleiner

**Industriesteuerungen:**  
(Abb. 2-5)

- hohe Ausfallsicherheit
- Echtzeitfähigkeit
- extreme Umweltbedingungen und mechanische Belastungen



Abb. 2-5 Anforderungen an die Elektronik steigen ständig an

**Vernetzte Welten:**  
(Abb. 2-6)

- Breitband Netzzugangsdienste x-DSL, Cable Modem
- Internet: Terabit *IP* Router mit Dienstgüten und Datensicherheit
- „Jeder mit jedem“



Abb. 2-6 Die Informationstechnologie ist aus unserer Welt nicht mehr weg zu denken

Durch die zunehmende Komplexität und die Zahl der Aufgaben eines elektronischen Systems werden jedoch auch die Entwurfsaufgaben zunehmend anspruchsvoller. Zusätzlich wird eine sehr geringe Zeit vom Beginn der Entwicklung bis zur Marktreife (engl. *time-to-market*) gefordert, die ohne aufwendige nachträgliche Korrekturen am Endprodukt in ihrer Einsatzumgebung auskommen müssen. Die schnell wachsende Komplexität der Entwurfsaufgaben ist durch den hohen Anteil an Wechselwirkungen für den Menschen nur noch schwer nachzuvollziehen. Auch die Qualitätssicherung gestaltet sich als zunehmendes Problem. Um Entwurfsfehler in frühen Phasen zu vermeiden, werden verstärkt formale Hilfsmittel zur Spezifikation und zur abstrakten Modellierung verwendet. Mit den formalen Hilfsmitteln finden auch die *Simulation* und formale *Verifikation* der Modellierung Einzug in den Entwurf elektronischer Systeme. Diese Techniken sind in vielen Jahren gewachsen. Deswegen existieren mehrere parallele Beschreibungsformen, mit denen dieselbe Funktionalität modelliert werden kann. Zusätzlich sind die Beschreibungsformen sowie *Simulation* und formale *Verifikation* oftmals auf einzelne Werkzeuge festgelegt, die keine oder eine eingeschränkte Interaktion mit weiteren Werkzeugen erlauben.

Diese parallele *Simulation* und *Verifikation* im Systementwurf stellt ein Problem dar, da keine Durchgängigkeit der einzelnen Ebenen gegeben ist. In jeder Systemebene kommen so beispielsweise unterschiedliche Beschreibungssprachen vor, deren Interaktion mittels weiterer Werkzeuge hergestellt werden kann, vgl. Abb. 2-7. Diese Problematik verkompliziert den gesamten Systementwurf unnötig. Ein neuer Ansatz, unter vielen, versucht mittels der *HDL*-Sprache SystemC eine Durchgängigkeit des gesamten Systementwurfs zu ermöglichen. Die verschiedenen Ansätze der Durchgängigkeit werden in Kapitel 4 Systembeschreibungssprachen (System-Level-Entwurf näher erläutert.

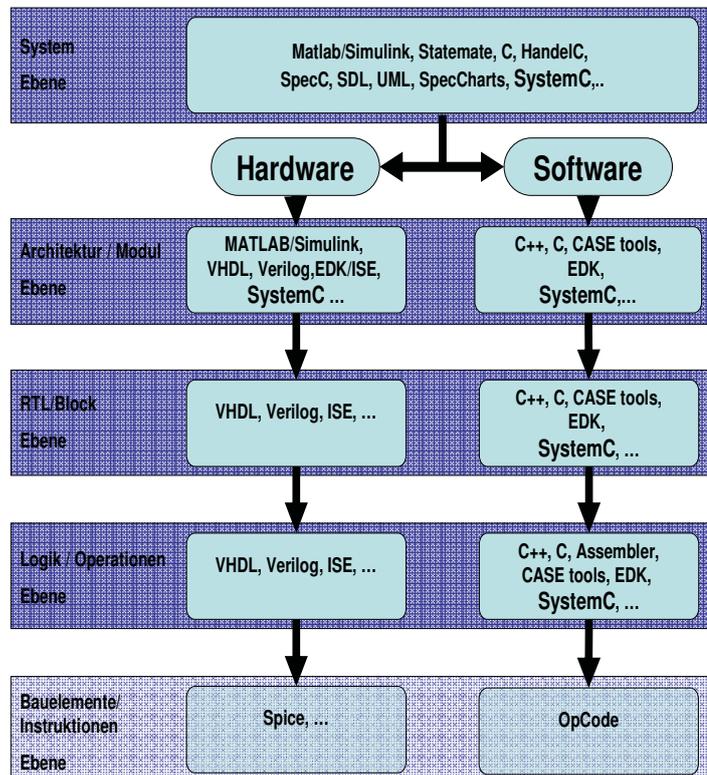
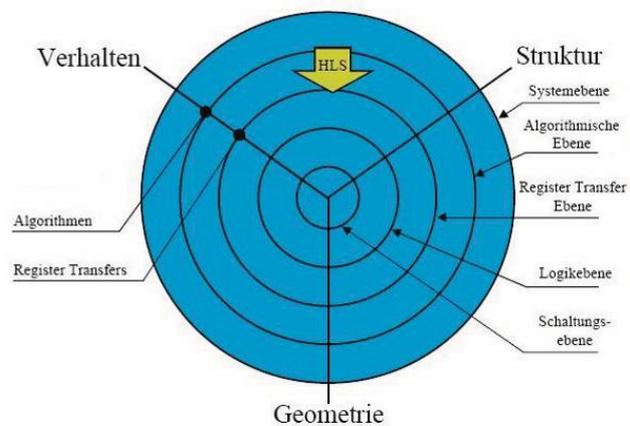


Abb. 2-7 Durchgängigkeitsproblem im (HW/SW) Systementwurf.

## 2.3 Entwurfssichten

Die Entwicklung elektronischer Systeme ist bei der heutigen Komplexität und den genannten Anforderungen nur durch eine strukturierte Vorgehensweise beherrschbar. Idealerweise wird, ausgehend von einer Spezifikation auf Systemebene, die Schaltungsfunktion partitionieren und die grundsätzlichen Funktionen den einzelnen Modulen zugeordnet. Schrittweise wird der Entwurf weiter strukturiert und zunehmend mit Details der Implementierung versehen, bis die für die Fertigung des elektronischen Systems notwendigen Daten vorliegen. Dies können Programmierdaten für Logikbausteine, Layouts für Leiterplatten oder Datensätze für die IC-Fertigung sein. Im Entwurf elektronischer Systeme wird üblicherweise in die drei Sichtweisen Verhalten, Struktur und Geometrie unterschieden. Diese Einteilung wird durch das sog. Y-Diagramm verdeutlicht (Abb. 2-8):

Gleichzeitig zu den drei Sichtweisen, die durch die Äste im Y-Diagramm repräsentiert werden, sind auch die verschiedenen Abstraktionsebenen durch Kreise mit unterschiedlichen Radien dargestellt. Ein großer Radius bedeutet hohe Abstraktion. Es kann nun vereinfacht der Entwurf elektronischer Systeme als eine Reihe von Transformationen (Wechsel der Sichtweise auf einem Abstraktionskreis) und Verfeinerungen (Wechsel der Abstraktionsebene innerhalb einer Sichtweise) im Y-Diagramm dargestellt werden. Beginnend auf dem Verhaltensast in der Systemebene wird der Entwurf durch Verfeinerungs- und Syntheseschritte bis hin zum Layout auf dem geometrischen Ast durchgeführt. Ein "Place- and Route" - Werkzeug für Standardzellenentwürfe überführt beispielsweise eine strukturelle Beschreibung in der Logikebene (Gatternetzliste) in eine geometrische Beschreibung in der Schaltungsebene (IC-Layout). Das reine Top-Down- Vorgehen (Entwicklung in Richtung Kreismittelpunkt) kann dabei nicht immer konsequent beibehalten werden. Verifikationsschritte zwischen den einzelnen Ebenen zeigen Fehler beim Entwurf auf. Gegebenenfalls muss das jeweilige Entwurfsergebnis modifiziert, der Entwurfsschritt wiederholt oder sogar auf höherer Abstraktionsebene neu eingesetzt werden. Dies wird auch als "Jojo-Design" bezeichnet. [10]



**Abb. 2-8 Y-Diagramm nach Gajski-Walker und die einzelnen Abstraktionsebenen**