5 Was ist Technische Informatik?

Die Technische Informatik ist die "Lehre von den apparativen Grundlagen der Informatik", oder einfacher ausgedrückt, die für Informatiker angepaßte Lehre von der *Hardware*. In der Mikroelektronik verbindet die Technische Informatik Stoffinhalte der Elektrotechnik mit solchen aus der Informatik und ein Vertreter der "Technischen Informatik" hat hier eigentlich zwei Seelen in seiner Brust: die eines Elektrotechnikers und die eines Informatikers. Die "Technische Informatik" ist also eigentlich interdisziplinär, obgleich erstere Bezeichnungsweise fast nur im Rahmen von Informatik-Studienplänen zu finden ist. Die Technische Informatik ist im Prinzip Technologie-unabhängig, da sie auch auf andere Techniken anwendbar ist (vgl. Kasten "Technologie-unabhängig." auf Seite 101). Als wichtige Orientierungshilfe der Technischen Informatik in der Mikroelektronik ist die Ordnung der Abstraktions-Ebenen (vgl. Kasten "Abstraktions-Ebenen" auf Seite 99) Hauptgegenstand dieses Kapitels.

Was ist ein Informatiker? Wenn in diesem Buch vom "Informatiker" die Rede ist, oder gar von einem "Technischen Informatiker", so muß deshalb eine solche Person nicht notwendigerweise ein Informatik-Studium absolviert haben und somit ein Informatik-Diplom besitzen. Dies hängt mit der historischen Entwicklung der noch sehr jungen Wissenschaft der Informatik zusammen. Viele Elektrotechnik-Studiengänge, nicht nur in Deutschland, enthalten in erheblichem Umfang Informatik-Stoffinhalte, beispielsweise im Rahmen von Stoffen der Digitaltechnik, der digitalen Signalverarbeitung, der Bildverarbeitung und anderen. Auch wenn Sie Elektrotechniker sein sollten, so fühlen Sie sich doch bitte mit der Bezeichnung "Informatiker" durchaus angesprochen, zumindest als "Technischer Informatiker". Mehr darüber wird im folgenden Abschnitt behandelt.

Abstraktions-Ebenen

Die Ordnung der Abstraktions-Ebenen ist für die Technische Informatik etwa so wichtig, wie das Periodische System der Elemente für die Chemie. Die Ordnung der Abstraktions-Ebenen ermöglicht eine klare Einordnung der Vielfalt von Entwurfs-Werkzeugen (vgl. Kap. 14) und macht den Entwurfsprozeß transparent und überschaubar.

5.1 Technische Informatik zwischen den Fachverbänden

Die Technische Informatik stand bis in jüngere Zeit mit im Spannungsfeld von Auseinandersetzungen um den Begriff der Informationstechnologie bzw. Informationstechnik. Nicht nur zwischen den Fachverbänden der Bundesrepublik (zwischen der GI und der ITG), sondern auch in einigen anderen Industrieländern gab, bzw. gibt es solche Spannungen. Durch ein "Friedenspapier" einer gemeinsamen Kommission zwischen den Fakultätentagen für Elektrotechnik und Informatik wurde der Streit beigelegt[33]. Das Papier definiert Zuständigkeiten in Forschung und Lehre. Zur weiteren Klarstellung sei deshalb folgende Abgrenzung zwischen Informatik und Elektrotechnik versucht. Wir können zwei Arten von Abgrenzungen unterscheiden:

- institutionelle Abgrenzung (Faktenlage an unseren Hochschulen)

5.1 Technische Informatik zwischen den Fachverbänden 99
5.2 Methodologische Ebenen der Technischen Informatik 105
5.2.1 Nicht-prozedurale Beschreibungen 105
5.2.2 Prozedurale Beschreibungen 109
5.3 Zusammenfassung 110
5.4 Literatur 111

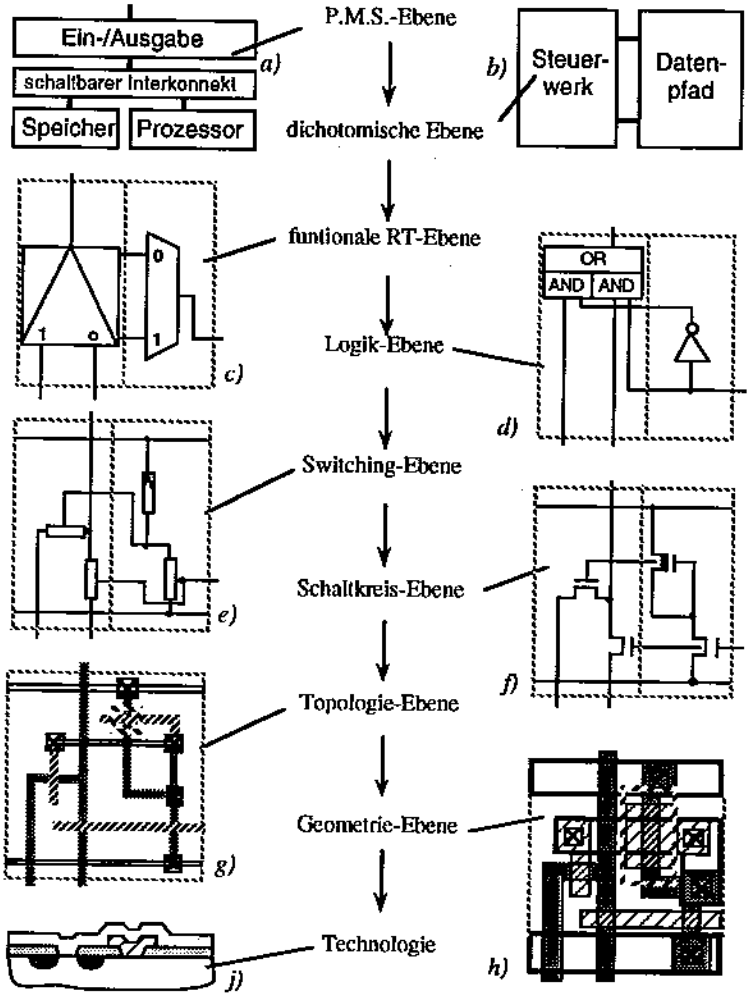

Bild 5.1: Ordnung der Abstraktions-Ebenen in der Technischen Informatik.

- systematische Abgrenzung (kurrikuläre Abgrenzung: vom Stoffinhalt her begründet)

Erstere ist eher quasi-politischer Natur, während letztere eine sachliche Begründung anstrebt.

Technologie-unabhängige Grundlagen. Systematisch gesehen behandelt die Informatik in der Mikroelektronik all diejenigen Grundlagen, Prinzipien und Methoden, die Technologie-unabhängig sind, zu deren Realisierung also vom Prinzip her nicht die Elektrizität benötigt wird [39]. Hierzu gehört beispielsweise der Logik-Entwurf, dessen Ergebnisse nicht nur in der

Mikroelektronik anwendbar sind, sondern z. B. auch für Realisierungen in mechanischer Logik (die Idee der logischen Formalisierung ist fast 100 Jahre alt: s. Bild 5.7), hydraulischer, fluidischer oder pneumatischer Logik etc. Bild 5.2 veranschaulicht die Funktion eines mechanischen And. Auch eine Reihe anderer Mikroelektronik-Entwurfs-Verfahren sind im Prinzip Technologie-unabhängig. Hierzu gehören u. a. auch sogenannte "routing and placement"-Verfahren (automatische Verdrahtungs-Generierung), deren Algorithmen beispielsweise auch auf die Generierung von Rohrleitungsnetzen anpaßbar sind (Bild 5.3). Auch die Informations-Speicherung in Genen beruht auf einem digitalen System (mit quaternären Ziffern, die gemäß Bild 5.4 durch Moleküle dargestellt werden).

Technologie-unabhängig.
Die Technische Informatik ist im Prinzip Technologie-unabhängig, wie das Beispiel des mechanischen And-Gatters in Bild 5.2 zeigt (s. auch Bild 5.3). Es gibt u. a. auch hydraulische, fluidische, oder pneumatische Logik. Die in der Datenverarbeitung derzeit wichtigste Anwendung ist die Mikroelektronik. Wird im nächsten Jahrhundert die Optronik aufholen?

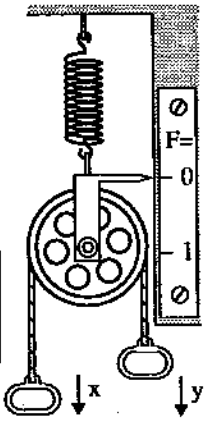

Bild 5.2: Mechanisches And, wobei F = x and y.

Auch "Extraktoren" ([17] [21], vgl. Kapitel 14) sind nicht nur in der Mikroelektronik bekannt (zur Schaltungsrückgewinnung aus dem Layout), sondern beispielsweise auch im Bauingenieurwesen (beispielsweise die Extraktion von Stabnetzwerken aus Baukonstruktionsdaten). Auch die heute übliche strenge Hierarchie der Zellen bei CAD-Werkzeugen in der Mikroelektronik hat ihre Parallelen in anderen CAD-Bereichen, wie beispielsweise im Maschinenbau, wo das "Chassis" die Wurzel einer Hierarchie zur Ordnung mechanischer Teile ist.

Verheiratung. Die Mikroelektronik ist als Verheiratung von Elektrotechnik und Informatik wichtigster Gegenstand der Technischen Informatik. Systematisch gesehen kann man hier die

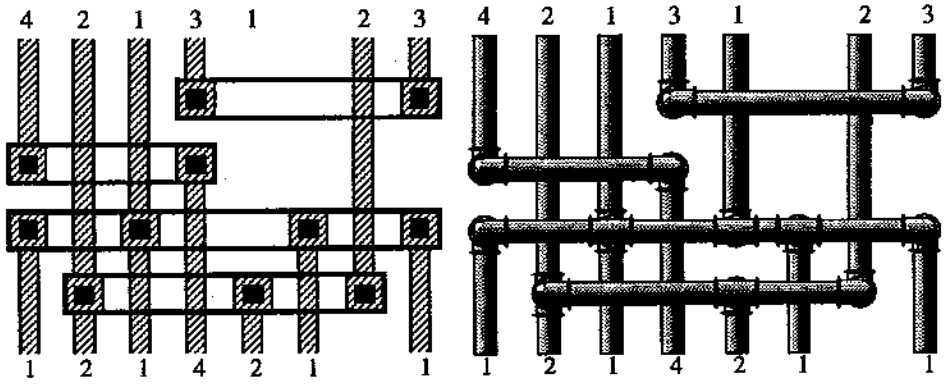

Bild 5.3: Veranschaulichung der Technologie-Unabhängigkeit - Realisierung des Netzwerk n. Bild 14.6 f: a) für hydraulische Logik als Rohrleitungs-Netz, b) für elektrische Logik als planares Leiternetz: Layout für eine integrierte Schaltung; Material-Ebenen: ◻ Metall (horizontal), ▨ polykristallines Silizium (vertikal), anstelle der Fittings: ■ Kontaktloch (schwarz).

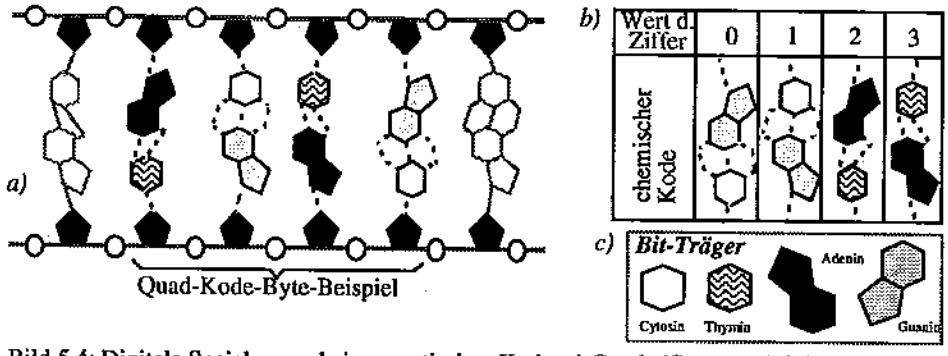

Bild 5.4: Digitale Speicherung beim genetischen Kode: a) Quad- (Quaterny Digit) Kode-Zuweisungs-Beispiel für ein Byte (Abschnitt einer DNS-Spirale), b) chemische Quaternär-Ziffern-Kodierung, c) als Träger des Kodes wirksame Teil-Moleküle.

Arbeitsteilung zwischen Elektrotechnik und Informatik wie folgt charakterisieren (Bild 5.5):

> Die Informatik hat viele Anwender-Disziplinen (Bindestrich-Informatiken und andere), darunter auch die Elektrotechnik .....
>
> Jedoch haben Elektrotechnik und Informatik eine besondere Beziehung zueinander: die Anwender-Beziehung ist wechselseitig, insbesondere auch in der Technischen Informatik

- die Elektrotechnik liefert den physischen Träger,
- die Informatik liefert darauf ausführbare logische Verfahrensweisen.

Die Informatik hat hier eine Doppelrolle: denn sie liefert:

- die logischen Konzepte für "intelligente Produkte",
- die Organisationsverfahren zur Beherrschung der Komplexität im Entwurfsprozeß.

Naiv ausgedrückt besteht in der Mikroelektronik folgende Arbeitsteilung:

- der Elektrotechniker sorgt dafür, daß der Transistor so klein und so schnell ist,
- der Informatiker gibt die Organisationsprinzipien an, mit denen man Tausende oder gar Millionen von Transistoren zu einer sinnvollen Anwendungsfunktion koordiniert.

Der in letzterem Fall gern verwendete Begriff *VLSI-Systeme* soll verdeutlichen, daß ein (insbesondere digitales) Anwendungssystem vor allem auf der Basis des Anwender-Sachverstan-

Aktivität	primäres Ziel	Grundlagen primär
Design komplexer Digitalsysteme	Organisation von Komplexität	Informatik
Design von Analog-Schaltungen	analoge Schaltungstechnik	Elektrotechnik
Design extrem schneller Schaltungen	Optimierung von Interkonnekt und Schaltgeschwindigkeit	Elektrotechnik
Technologie	optimale Transistoren	Elektrotechnik

Bild 5.5: Eine Zusammenfassung der fachlichen Zuständigkeitsbereiche (stark vereinfacht).

Vorlesung	Objekte		Verfahren
	Komplexe	Elemente	
Rechner-organisation	PMS-Elemente* [22][36]	RT-Elemente	Synthese von PMS-Elementen
Digitale Logik	Schaltnetze, Schaltwerke	Gatter	Synthese und Optimierung
Technische Informatik I	Grundschaltungen	"Bauelemente" aus physikalischer und technologischer Sicht	Verstehen der Wirkung elektrotechnischer und physikalischer Effekte
Technische Informatik II**	(viele Abstraktions-ebenen erfaßt)	(viele Abstraktions-ebenen erfaßt)	vom Algorithmus zur planaren Realisierung

*) PMS-Ebene: processor, memory, switch **) Gegenstand dieses Buches

Bild 5.6: Die Mikroelektronik im Grundstudium der Informatik in Kaiserslautern.

des realisiert werden soll, wobei die VLSI-Technik nur Hilfsmittel und Gegenstand von Machbarkeits-Analysen (feasibility studies) sein soll.

Nur Digitalschaltungen. Die hier in diesem Buch behandelte Szene sei auf die Digitaltechnik beschränkt. Für den Informatiker werden die elektrotechnischen Modelle stark vereinfacht (siehe Kapitel 13), sodaß er sich praktisch voll auf den logischen Aspekt und auf organisatorische Probleme konzentrieren kann. Für sehr schnelle Schaltungen wird angestrebt, das Know-how über die doch anspruchsvolleren elektrotechnischen Modelle per Computer in die Entwurfs-Werkzeuge hinein zu transferieren, die durch VLSI-Designer dann weitgehend problemlos als "black box" angewandt werden können. Analog-Schaltungen sind ebenfalls als eine Domäne der Elektrotechnik anzusehen, denn hier wird elektrotechnischer Sachverstand nicht nur für die Technologie, sondern auch noch für die Anwendungsebene benötigt.

Hardware-Entwurfsverfahren. Die Technische Informatik - überwiegender Gegenstand dieses Buches - befaßt sich nur mit Digital-Hardware, mit den Entwurfsverfahren hierzu und in gewissem Grade auch mit dem "Technologie-Transfer" in Anwendungsgebiete hinein. Bei Entwurfsverfahren für digitale Hardware stehen heute diejenigen für hochintegrierte Digitalschaltungen im Brennpunkt des Interesses. Deshalb ist die Mikroelektronik der hauptsächliche Gegenstand des vorliegenden Buches zur Technischen Informatik.

5.2 Methodologische Ebenen der Technischen Informatik

In der Technischen Informatik wird notwendigerweise eine größere Zahl methodologischer Ebenen unterschieden, größer als beispielsweise in der Praktischen Informatik. Jede dieser Ebenen verwendet ihre eigenen, für sie typischen Elemente, Notationen und Methoden und hat quasi eine Schnittstelle zu jeweils ihrer oberen und ihrer unteren Nachbarebene. Bild 5.1 veranschaulicht die Ordnung dieser Abstraktionsebenen in der Hardware bis hinauf zur Monoprozessor-Ebene. Bei Hardware-Synthese-Verfahren werden oft noch höhere Abstraktions-Ebenen (Bild 5.11) miterfaßt (s. Kapitel 19 - 23). Diese Systematik und der Transfer von Beschreibungen zwischen den Ebenen ist Hauptgegenstand diese Buches (letzte Zeile in Bild 5.6).

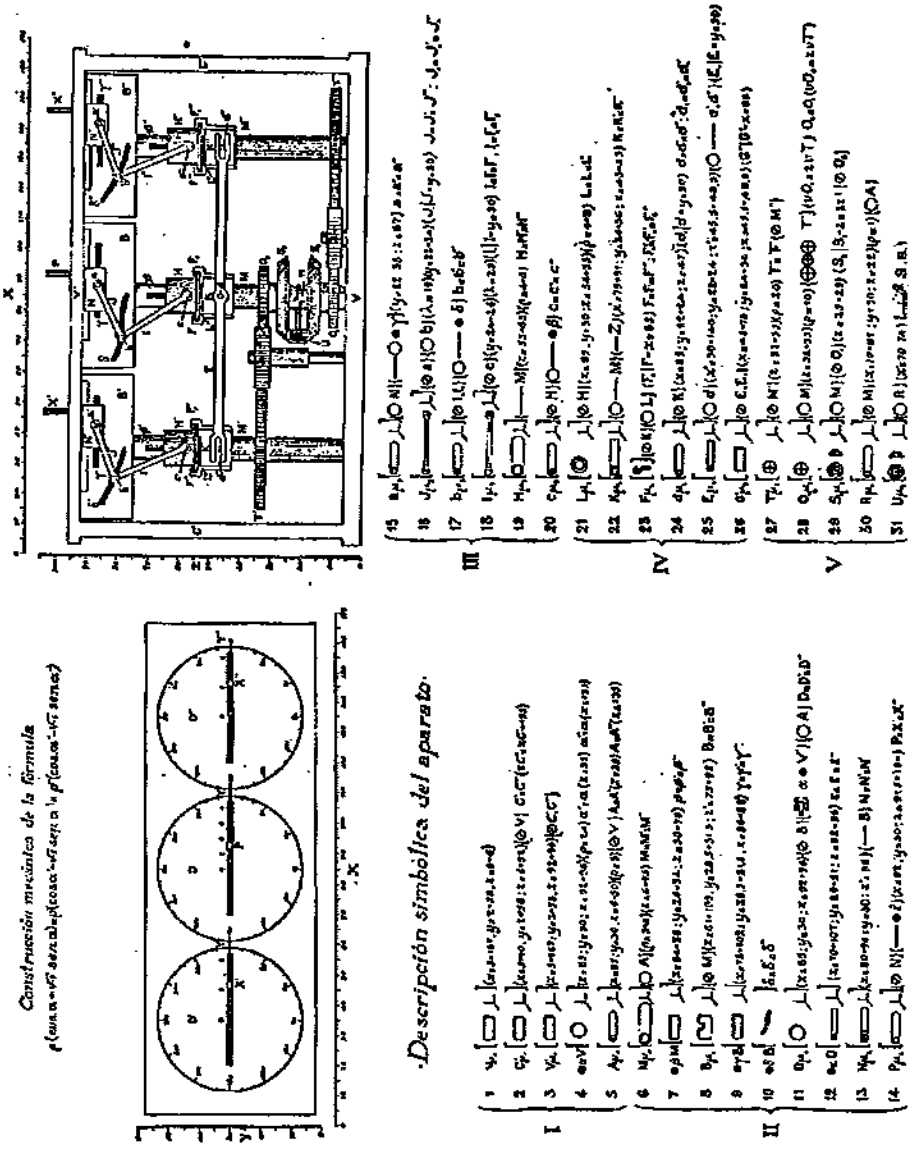

Bild 5.7: Hardware-Beschreibungssprache (Skizze) f. mechanische Rechner (Torrez Y Quevedo).

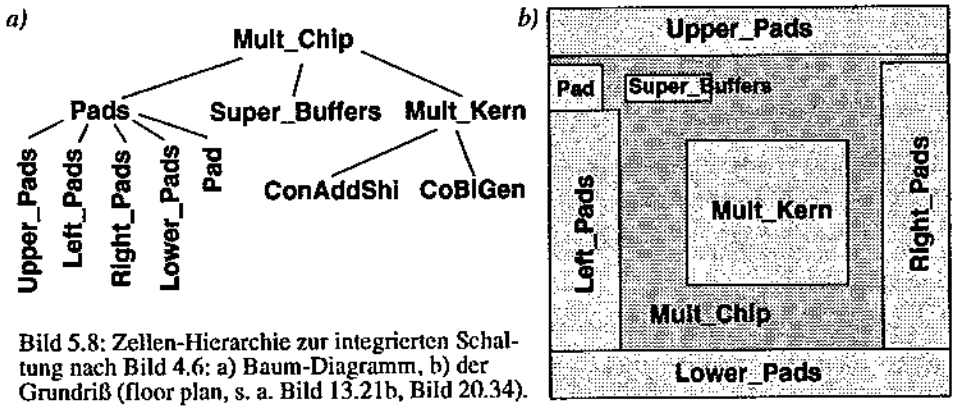

Bild 5.8: Zellen-Hierarchie zur integrierten Schaltung nach Bild 4.6: a) Baum-Diagramm, b) der Grundriß (floor plan, s. a. Bild 13.21b, Bild 20.34).

Zellen-Hierarchien. Bei komplexen Schaltungen wird die Notation jeder Ebene - wie bei Software - hierarchisch partitioniert[1], wie das Beispiel in Bild 5.8 veranschaulicht. Der Raum der Beschreibungen durch die verschiedenen Abstraktions-Ebenen hindurch wird dadurch zweidimensional. Hierarchien sind übrigens auch bei CAD-Systemen anderer Anwendungsgebiete üblich, wobei im Maschinenbau beispielsweise meist das Chassis die Wurzel ist.

5.2.1 Nicht-prozedurale Beschreibungen

Wir wollen uns zuerst den Hardware-nahen nicht-prozeduralen Ebenen zuwenden, die in Bild 5.10 charakterisiert werden [11]. Zum Vergleich mit Darstellungen anderer Ebenen ist in der zweiten Spalte jeweils das gleiche Objekt gezeigt: ein vollständig dekodierter Zwei-Wege-Multiplexer in nMOS-Technologie (diese wird in Kapitel 11 eingeführt). Beschreibungen höherer nicht-prozeduraler Ebenen können Quellen für die Synthese sein [1] [2] [3], aber auch für Verifikation [5] [6] oder Testerzeugung [30] [35] [37] [38]. Wir beginnen unten. Die drei untersten Ebenen sind elektrotechnischer Natur und sind durch die Technologie bestimmt, über das Repertoir an Materialien sowie Typ und Form der verfügbaren Transistoren.

Physisches Layout. Bild 5.10 f zeigt das physische Layout, welches der geometrischen Ebene angehört. Hier werden nur planare geometrische Figuren dargestellt, jedoch keinerlei Verhaltens-Modelle. Auch hier ist eine Hierarchisierung möglich, die allerdings nur durch die Rechner-interne Datenstruktur verkörpert wird. Bei der graphischen Ausgabe des Layouts über Bildschirm oder Papier wird die Hierarchie durch den Zusatz von Begrenzungs-Boxen und Namen dargestellt (vgl. Bild 5.8 b). Spätestens bei der Fertigung geht diese Hierarchie verloren. Aber auch manche CAD-Programme ebnen die Hierarchie ein. Mit Layout-Entwurf befaßt sich Kapitel 12 .

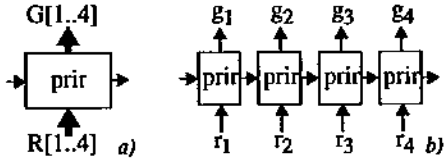

Bild 5.9: Datenpfad-Formate am Beispiel des Prioritäts-Operators prir: a) Wort-Format, b) Darstellung im Einzelbit-Format.

1) Zur Vermeidung eines Chaos muß die Hierarchie-Struktur auf allen Abstraktionsebenen einheitlich sein.

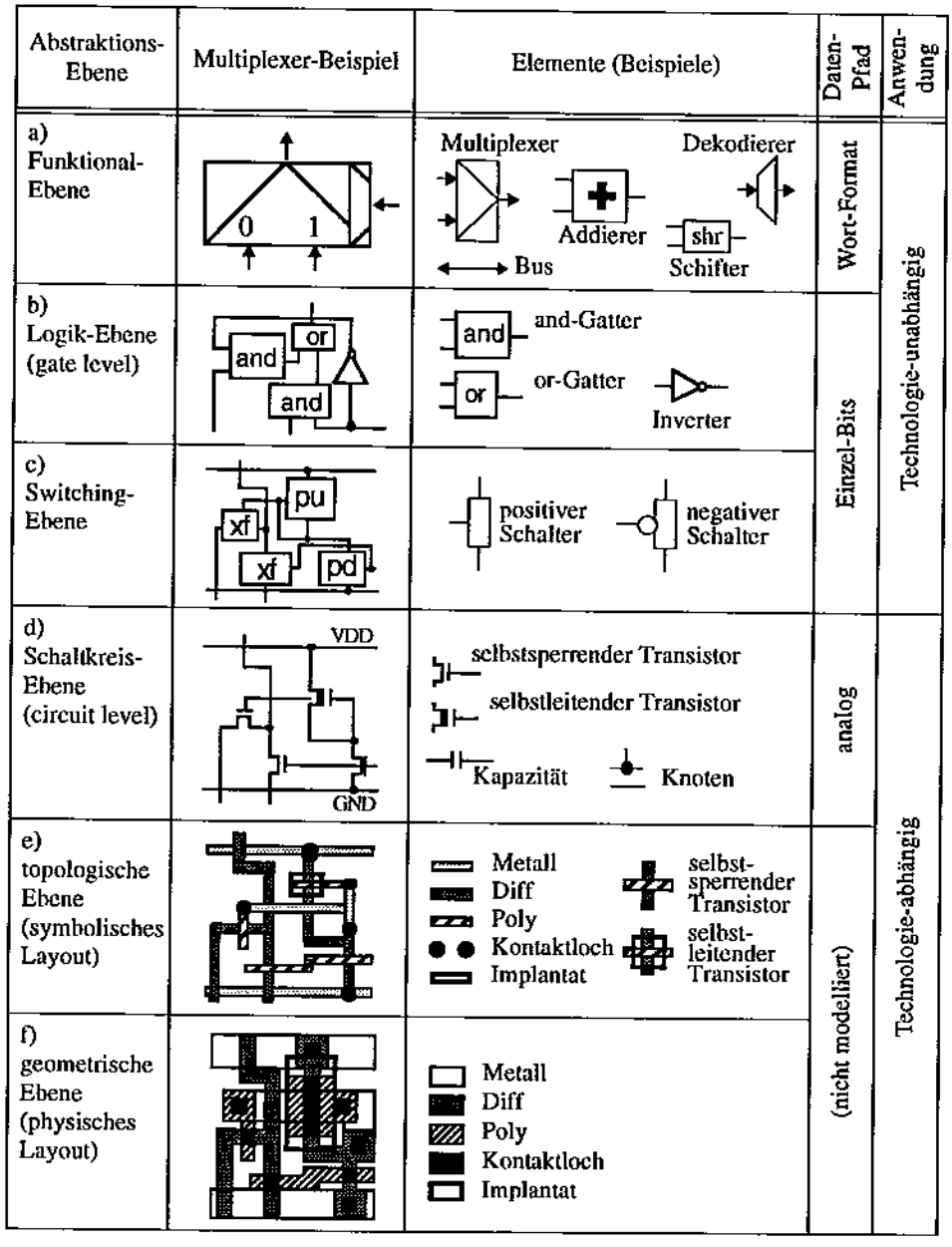

Bild 5.10: Veranschaulichung der Darstellung in den nicht-prozeduralen Abstraktionsebenen.

Symbolisches Layout. Bild 5.10 e zeigt das symbolische Layout, welches der topologischen Ebene angehört. Der Unterschied zum Layout besteht darin, daß keine Vermaßung angegeben wird. Insbesondere beim strukturierten VLSI-Entwurf (vgl. Kapitel 4 und 19 - 22) auftretende Probleme der räumlichen Anordnung von Bauelementen, Leitungen und Anschlußpunkten sind unter Ausschluß der Vermaßung oft viel einfacher zu lösen. Eine weithin bekannte Notation dieser Ebene sind die sogenannten Stick-Diagramme [29] (siehe auch Bild 12.4 b). In der vorliegenden Ebene erfolgt z. B. die Optimierung von CMOS-Standardzellen (Kapitel 18).

Die Schaltkreis-Ebene (Bild 5.10 d) ist die oberste unter den Technologie-abhängigen Abstraktions-Ebenen. Für Digitalsysteme wird diese eingeführt in Kapitel 6 . Die analogtechnische Schaltkreis-Simulation wird in Abschnitt 13.4.1 kurz eingeführt.

Die Switching-Ebene (Bild 5.10 c) ist die unterste Technologie-unabhängige Abstraktions-Ebene [26]. Abstrakte Schalter modellieren echte Bauelemente unabhängig davon, ob deren Realisierung mit bipolaren Transistoren oder MOS-Transistoren, durch opto-elektronische oder mechanische Schalter, fluidisch, pneumatisch oder sonstwie erfolgt (s. Kapitel 7 und 8).

Die Logik-Ebene (Bild 5.10 b, engl.: *gate level*) befaßt sich mit Schaltnetzen und (fest-verdrahteten) Schaltwerken. Deren Grundlagen werden in diesem Buch beim Leser vorausgesetzt. Einschlägige Einführungen genügen (beispielsweise in Kaiserslautern die Vorlesung "Digitale Logik", 2-stündig im 1. Semester des Grundstudiums. Eine tabellarische Zusammenfassung der wichtigsten Boole'schen Regeln für Optimierungszwecke ist Bild 7.3.Wenige Anwendungen solcher Regeln finden sich in Kapitel 7 .

Die Funktional-Ebene (nicht-prozedurale) Register-Transfer-Ebene (Bild 5.10 a) ist die oberste unter den nicht-prozeduralen Ebenen mit speichernden und nicht-speichernden Elementen wie Register, Registerfelder, Verzögerungsglieder, relationelle und arithmetische Operationen sowie andere Wort-orientierte Funktionen, wie beispielsweise Verdrahtungs-Operatoren ([9] [14], beispielsweise auch in Bild 20.16 und Bild 20.30 - Bild 20.32). Einzelbits werden meist zu Bit-Vektoren, also zu Binärworten zusammengefaßt (veranschaulicht am Beispiel des Prioritäts-Operators in Bild 5.9, wo Bit-Scheiben zu einem Operator *prir* zusammengesteckt werden, wodurch die einzelnen Anforderungs-Bits r1 bis r4 zum Anforderungsvektor R[1..4] zusammengefaßt werden wie die Zuteilungsbits g1 bis g4 zu einem Zuteilungsvektor G[1..4]).

Misch-Notationen. Hardware-Beschreibungs-Sprachen (CHDLs, engl.: *Computer Hardware Descriptive Languages*) [9] [23] [14] sind meist mehrere Abstraktions-Ebenen umfassende Misch-Notationen. Dadurch werden solche Sprachen mehr oder minder gut auch als Entwurfs-Kalkül anwendbar, u. a. dadurch, daß Entwurfsproblem und Lösung in der gleichen Sprache darstellbar sind, ebenso wie Zwischenresultate gemischt aus Problemteilen und halbfertiger Lösung. Insbesondere für den strukturierten Entwurf sind graphische Misch-Notationen [2] [4] [14] als sehr mächtige Entwurfswerkzeuge implementiert worden. Besonders effektiv sind dabei solche Misch-Notationen, die auch topologische Strukturen mit Darstellungen mehrerer anderer Ebenen mischen können [4] [23] [31] [32].

Jo-jo-Spiel. Hauptanliegen dieses Buches ist die Vermittlung der Fähigkeit zum Herumwandern im Entwurfsraum, bei Synthese (abwärts), Extraktion, Überprüfung und Fehlerbeseitigung (aufwärts), und auch bei der Entwicklung von Entwurfs-Werkzeugen und -Umgebungen (aufwärts und abwärts) oder beim Vordringen in die dritte Dimension (vgl. Bild 14.3 und

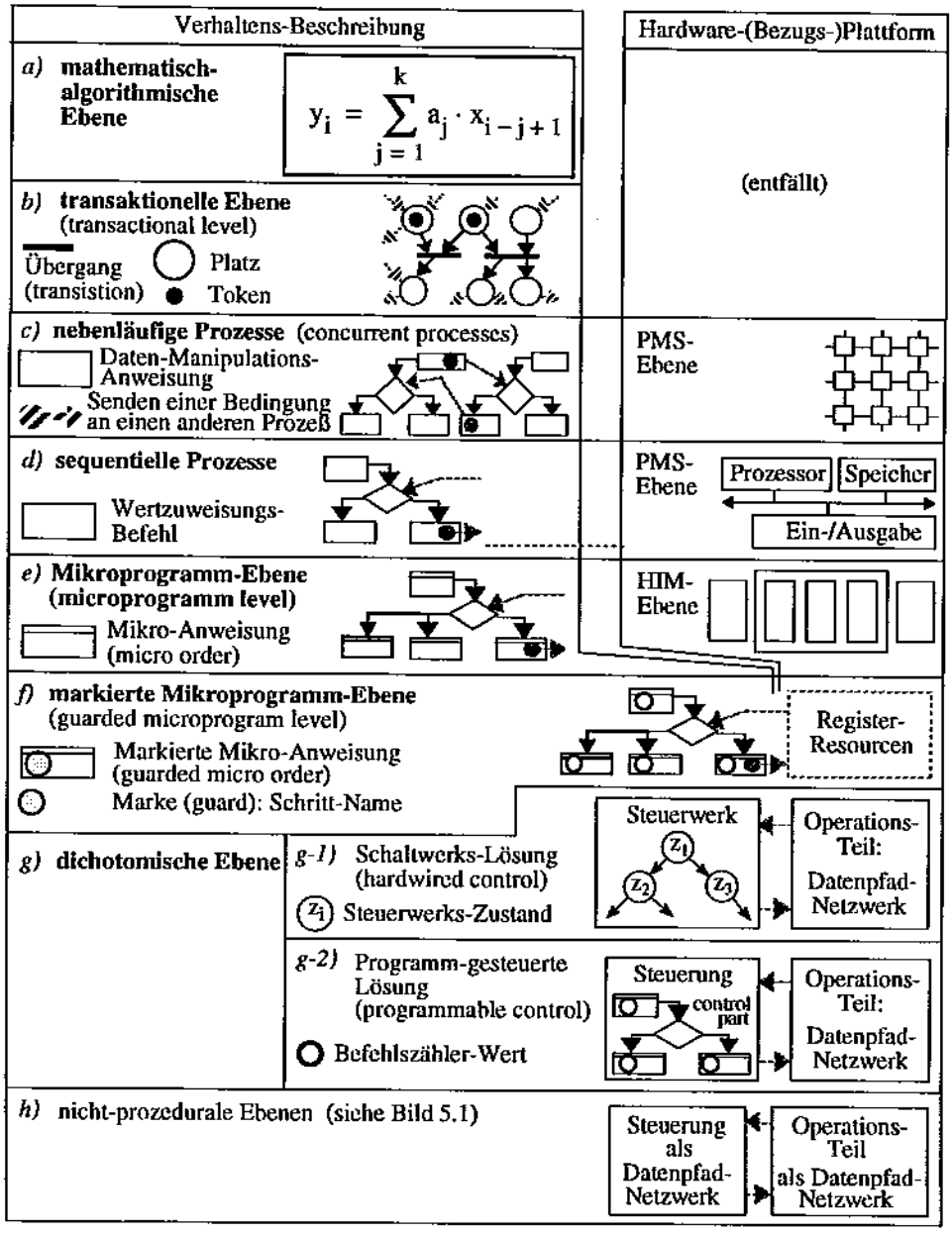

Bild 5.11: Die prozeduralen Abstraktions-Ebenen aus der Sicht der Technischen Informatik.

Bild 14.8) zur Testerzeugung [30] [35] [37] [38], Verifikation [5] [6], Simulation ([19] [28] und Abschnitt 13.4.1) oder Überprüfung auf Einhaltung der Layout-Regeln und elektrische Regeln (Kapitel 12). Hierbei soll die Ordnung der Abstraktions-Ebenen und die hierarchische Strukturierung als Landkarte und Koordinatensystem bei diesem Jo-jo-Spiel (engl.: *yo-yo*)als Kompaß dienen. Details, insbesondere zum strukturierten VLSI-Entwurf [3] [16] zeigen Kapitel 19 und 20 und andere.

5.2.2 Prozedurale Beschreibungen

Dieser Abschnitt setzt die Klassifikation der Abstraktions-Ebenen fort [11] mit prozeduralen Beschreibungen (Bild 5.11, Folgen von Ereignissen, mit oder ohne Angabe der Zustandsvariablen). In den unteren Ebenen haben wir oft eine Dichotomie von gesteuertem Teil und Steuerwerk angewandt. In Abstraktions-Ebenen weiter oben verschwindet die Hardware ganz aus der eigentlichen Beschreibung und bleibt nur als Semantik einer (Programm-)Sprache im Hintergrund. Es erhebt sich die Frage, wozu prozedurale Beschreibungen, eigentlich eine Domäne der Software, hier in der Hardware dienen sollen. Diese dienen hier meist als Notation zur Formulierung des Entwurfsproblems, wie beispielsweise zur automatischen oder zumindest formalen Ableitung einer Hardware aus einem Programm (bei der High-Level-Synthese, siehe Kapitel 23), oder zu deren Ableitung aus der mathematischen Formel oder einer Notation, die einer höheren Programmiersprache ähnlich ist (beispielsweise für systolische Arrays [20], siehe Kapitel 22), als auch zur mehr informellen Ableitung aus einer Programm-Notation, wie beispielsweise bei der Parallelisierung der inneren Schleife (Kapitel 19 und 20). Weiterhin wichtig sind prozedurale Notationen natürlich auch beim Hardware/Software-Ko-Design [15].

Dichotomische Ebene. Ab dieser Ebene aufwärts wird die Zerlegung der Hardware in eine Dichotomie aus Steuerteil und gesteuertem Teil formalisiert (Bild 5.11 g), während dies in darunterliegenden Abstraktionsebenen (beispielsweise Bild 5.11 h) nur eine triviale Partitionierungs-Maßnahme war. In der dichotomischen Ebene erfolgt eine echte Mischdarstellung derart, daß das Steuerwerks-Verhalten (Folgen von Ereignissen mit Zustands-Variable) abstrakt beschrieben wird durch Zustandsgraphen oder Übergangstabellen [10]. Der gesteuerte Teil (Datenpfade, ALU o. ä.) hingegen erscheint hier in der niedrigeren funktionalen Abstraktions-Ebene (Bild 5.10 a). Diese Dichotomie ist deshalb sinnvoll, weil der Entwurf [10] und die automatische Synthese von Steuerwerken [41] ein relativ ausgereiftes eigenes Gebiet, ja zusammen mit den Industrie-Steuerungen fast ein eigener Wirtschaftszweig ist. Zur Beschreibung und Synthese von Steuerwerken gibt es sogar zwei theoretisch äquivalente Subkulturen: die sogenannten fest verdrahteten Steuerungen (Bild 5.11 g-1, engl.: *hardwired control*), auch Schaltwerke genannt (F.S.M., engl.: *finite state machine*), sowie die sogenannten (Speicher-) programmierbaren Steuerungen, die nach Art des von-Neumann-Modell in einen Abwickler (engl.: *sequencer*) und einen Programmspeicher zerfallen Bild 5.11 g-2).

Mikroprogramme. Die Ebene der Mikroprogramme (Bild 5.11 e) kann als eine Abstraktion der dichotomischen Ebene angesehen werden. Hier wird die Architektur mikroprogrammierbarer Prozessoren definiert. Die Beziehungen zwischen zwei Sprach-Ebenen, nämlich der Maschinen-Programm-Ebene und der Mikroprogramm-Ebene, werden hier durch eine interpretative Hardware bestimmt [8]. Eine strukturierte Formalisierung dieser interpretativen Beziehungen ist durch das HIM-Schema angegeben [22] [13]. Eine Hardware-nähere Form von

Mikroprogrammen sind die markierten Mikroprogramme (Bild 5.11 f, engl.: *guarded microprograms*), die als Zwischennotation in der High-Level-Synthese auftreten (Kapitel 23).

Sequentielle Programme. Ab dieser Ebene (Bild 5.11 d) ist der Verhaltens-Notation im allgemeinen keine Hardware-Beschreibung mehr beigegeben und steht allenfalls als Semantik der vorliegenden (Programmier-)Sprache im Hintergrund. Die Dichotomie ist ab hier also wieder aufgehoben. Das Hardware-seitige Gegenüber dieser Abstraktions-Ebene ist oft informell. Ein etwas formelleres Beispiel ist die PMS-Notation ([36], PMS steht für "Processor, Memory, Switch"). In der Mikroelektronik-Anwendung treten sequentielle Programme als Notation zur Formulierung des Entwurfsproblems auf, etwa zur automatischen Ableitung einer Hardware bei der High-Level-Synthese (Kapitel 23). Oft erfolgt eine mehr informelle Ableitung aus einer Programm-Notation, wie beispielsweise bei der Parallelisierung der inneren Schleife (siehe Kapitel 19 und 20).

Nebenläufige Programme. Hier (Bild 5.11 c) liegen in gleichzeitig ausführbare Prozesse gegliederte Programme vor zwecks beschleunigter Ausführung auf Mehrrechnersystemen. Hierbei entsteht implizit ein System miteinander kommunizierender Steuerwerke, da jeder Rechner sein eigenes Steuerwerk hat. In der Mikroelektronik werden in dieser Ebene auch Hardware-Entwurfsprobleme formuliert, teilweise auch mit Notationen, die eher einer sequentiellen höheren Programmiersprache ähnlich sind. Ein Beispiel ist die automatische Synthese systolischer Arrays [20] (siehe auch Kapitel 22).

Transaktionelle Notationen. In dieser Ebene (Bild 5.11 b) wird die Unabhängigkeit nebenläufiger Prozesse insofern aufgegeben, als nicht mehr jeder Prozessor sein eigenes Steuerwerk hat. Diese Ebene kann Ausgangspunkt von Hardware-Synthese sein, wie etwa bei konzentrierten nebenläufigen Steuerungen (beispielsweise die Hardware-Realisierung interpretierter Petri-Netze [18] (siehe Kapitel 24). Hierbei können mehrere asynchrone Hardware-Resourcen mit parallel verlaufenden Ereignisfolgen durch eine gemeinsame konzentrierte Steuerung koordiniert werden [34].

Die mathematische Formel (Bild 5.11 a) oder eine andere Notation dieser Abstraktions-Ebene ist oft indirekter Ausgangspunkt einer Hardware-Entwurfs-Anstrengung. Direkt auf formale Weise erfolgt hieraus der Entwurf von systolischen Arrays [20] (siehe Kapitel 22).

Misch-Notationen. Es ist eine Reihe von Hardware-Beschreibungssprachen implementiert worden die als Misch-Notationen sowohl nicht-prozedurale Ebenen als auch prozedurale Abstraktions-Ebenen umfassen (wie z. B.: [11] [25]). Diese können schon fast CAD Frameworks sein [14] [23] und dabei verschiedene Arten von CAD-Werkzeugen unterstützen bei Aufwärts- und Abwärts-Transformationen sowie Transformationen in die dritte Dimension (Bild 14.3 und Bild 14.8). Wichtig sind Misch-Notationen auch für Hardware/Software-Ko-Design [15].

5.3 Zusammenfassung

Ein Ziel dieses Kapitels ist es, mit die Ordnung der Abstraktions-Ebenen als eine Hilfe darzustellen zur Orientierung in der Vielfalt der Notationen und Entwurfs-Werkzeuge. Aus dieser Blickrichtung soll auch die Rolle von Algorithmen für Verfahren des Entwurfsprozesses (EDA-Algorithmen) beleuchtet werden. Ein vorrangiges Ziel ist es, sozusagen "Jo-jo-Fähigkeiten" zu vermitteln: Beschreibungen aufwärts und abwärts zu transformieren - von einer me-

thodologischen Ebene in andere Ebenen abzubilden abwärts für die Synthese - aufwärts für Extraktion und Verifikation. Auch diese Fähigkeit ist hilfreich für die Transparenz des Entwurfsprozeß und in der Praxis, wie z. B. bei der Lokalisierung und Behebung von Fehlern in der Software, als auch beim Entwurfsobjekt selbst.

5.4 Literatur

[1] G. Arato, R. Manione: PSICO: a System for Automatic Layout Synthesis; report, CSELT, Torino, Italy, 1986.

[2] A. Bonomo, M. Italiano, L. Lavagno, M. Maggiulli, M. Melgara, M. Paolini, I. Stamelos: BACH (Behavioral-Level Automated Compilation of Hardware): An Integrated ASIC Synthesis System; Proc. ESPRIT Technical Week, Brussels, Belgium 1988

[3] A. Bonomo, G. Bussolino, G. Girardi, M. Italiano: From Structured RT Description to Floor-Plan; report; CSELT, Torino, Italy, 1986 - also: Proc. EUROMICRO Symposium, Portsmouth, UK, 1987; North Holland, Amsterdam, 1986

[4] G. Girardi, R. Hartenstein, U. Welters: KARL (textual) and ABL (graphic): A User/Designer interface in microelectronics; in (Hrsg.: J. Encarnaçao): CAD-Schnittstellen und Datentransfer-Formate im Elektronik-Bereich; Springer-Verlag, 1986.

[5] W. Grass, N. Schielow: Verena: A Program for automatic verifications of the register transfer description; IFIP Int'l Symp. CHDL'85; Tokyo, Japan, 1985.

[6] W. Grass, R. Rauscher: A Practicable Strategy for the Verification of Interactive Microprogram Transformations; EUROMICRO'85, Brussels, Belgium; North Holland, 19

[7] H Grünbacher, R. Hartenstein: FPGAs - Architectures and Tools for Rapid prototyping; Springer-Verlag, Heidelberg, 1993

[8] R. Hartenstein: Microprogramming Concepts - a Step towards Structured Hardware Design; Micro 7 (1974), ACM New York, 1974.

[9] R. Hartenstein: Fundamentals of Structured Hardware Design: A Design Language Approach at Register Transfer Level; North Holland; Amsterdam/New York, 1977.

[10] R. Hartenstein: Basics of Structured Design Methodologies: Data Path and Finite State Machine; in: [27]

[11] R. Hartenstein: Classification of Hardware Description Languages; in [12]

[12] R. Hartenstein: Hardware Description Languages; Elsevier, Amsterdam, 1987.

[13] R. Hartenstein: Hierarchies of Interpreters for Modelling Complex Digital Systems; Proc. GI 3rd Ann. Conf. Hamburg 1973; Springer-Verlag, 1973

[14] R. Hartenstein: The History of ABL and KARL; Bericht; Univ. Kaiserslautern 1993

[15] R. Hartenstein: Hardware / Software Ko-Design; (W. Moore, W. Luk, eds.:) Proc. 3rd Int'l Workshop on Field-programmable Logic and Applications, Oxford, UK, Sept 1993

[16] R. Hartenstein, K. P. Bastian, W. Nebel: VLSI-Algorithmen: innovative Schaltungstechnik statt Software; GME-Tagung Baden-Baden, 1985, VDE-Verlag, Berlin 1985

[17] R. Hartenstein, R. Hauck: Functional Extraction from Personality Matrixes of MOL (Matrix-Oriented Logic) Circuits; IFIP CHDL'87, Amsterdam, 1987

[18] R. Hartenstein, A. Hirschbiel, M. Weber: Patil Array: a self-prioritizing array to implement Petri nets; Int'l EUROMICRO Symposium; Brussels, Belgium, 1985.

[19] R. Hartenstein, K. Lemmert: KARL-3 reference manual; Univ. Kaiserslautern, 1984.

[20] R. Hartenstein, K. Lemmert: SYS^3: A CHDL-based Systolic Synthesis System; Proc. IFIP CHDL'89, Washington, D.C., Elsevier, 1989.

[21] R. Hartenstein, W. Nebel: Shifting Functional Design Verification Towards RT Level by Automatic Register Transfer Net Extraction; IFIP CHDL'87, Amsterdam, 1987

[22] R. Hartenstein, M. Ryba: Computer Structure Partitioning Schemes; in (ed.: D. Edwards): Design Methodologies for VLSI and Computer Architecture, North Holland,1988

[23] R. Hartenstein, U. Welters: Higher Level Simulation and CHDLs; in (eds.: W. Fichtner, M. Morf): Proc. IFIP Summer School on VLSI Design, Beatenberg, Switzerland, 1986; Kluwer Publishing Co., 1986

[24] R. Hartenstein, R. Zaks: Microarchitecture of Computer Systems; North Holland 1975

[25] R. Hauck: CVS_BK User Guide, CVT report, Univ. Kaiserslautern, 1988

[26] J. P. Hayes: A unified switching theory with applications to VLSI design; Proc. IEEE, vol. 70, Oct 1982

[27] P. Jespers, C. Sequin, F. van de Wiele: Design Methodologies for VLSI Circuits; Proc. NATO-ASI "Very Large Scale Integration", Louvain-la-Neuve 1981, Noordhoff & - Stijthoff, Rockville, Maryland, 1981

[28] L. Lavagno, R. Manione: SMART, SiMulator At Register Transfer level; report, CSELT, Torino, Italy, 1987.

[29] C. Mead, L. Conway: Introduction to VLSI systems; Addison-Wesley, 1980

[30] S. Morpurgo, A. Hunger, M. Melgara, C. Segre: RTL Validation of VLSI: An Integrated Set of Tools for KARL; IFIP Int'l Symposium on Computer Hardware Description Languages, (CHDL'85); Tokyo, Japan, 1985

[31] V. G. Moshnyaga, H. Onodera, K. Tamaru, H. Yasuura: A Language for Designing Data-Path Module Generators; IEEE Int'l Design Workshop "Russian Workshop'92", Moscow, Russia; also: IFIP WG 10.5 Workshop on Synthesis, Generation and Portability of Library Blocks for ASIC Design, Grenoble, France, Mar 1992

[32] V. G. Moshnyaga, H. Yasuura: A Language for Designing Module Generators; Proc. SASIMI'92 - Synthesis and Simulation Meeting and Int'l Exchange, Kobe, Japan, 1992

[33] N. N.: Gemeinsame Stellungnahme der Fakultätentage Elektrotechnik und Informatik zur Abstimmung ihrer Fachgebiete im Bereich Informationstechnik, Informatik-Spektrum 14,3 (Juni 1991), p. 163 - 167.

[34] S. S. Patil: Micro-control for Parallel Asynchronous Computers; in: [24]

[35] C. Segre, S. Morpurgo: Proposed Algorithm for a Testability Analysis Tool based on KARL; CVT report, Ing. Olivetti S.p.A., Ivrea, Italy, 1984

[36] D. P. Siewiorek, C. G. Bell, A. Newell: Computer Structures: Principles and Examples; McGraw-Hill, New York et al. 1982

[37] I. Stamelos, M. Melgara: CVT TIGER: a RT-level Test Pattern Generation and Validation Environment; report, CSELT, Torino, Italy, 1985

[38] I. Stamelos, M. Melgara, M. Paolini, S. Morpurgo, C. Segre: A Multi-Level Test Pattern Generation and Validation Environment; International Test Conference, 1986

[39] S. Wendt: Nichtphysikalische Grundlagen der Informat(ionstechn)ik; Springer-Verlag 1989