14 Entwurfs-Werkzeuge

Beim Entwurf komplexer Digitalschaltungen, insbesondere hochkomplexer integrierter Schaltungen, werden viele Abstraktionsebenen berührt (vgl. Bild 14.1): nicht-prozedurale Ebenen (vgl. Bild 5.1 bzw.Bild 5.5 oder Bild 14.8) und prozedurale Ebenen (vgl. Bild 14.8) berührt. Viele Entwurfs-Hilfsmittel sind auf eine einzige oder nur wenige Abstraktionsebenen spezialisiert. Darüber hinaus können die Beziehungen zwischen Ebenen und Werkzeugen unterschiedlich sein, je nachdem, ob beispielsweise Synthese oder Überprüfung durchgeführt werden soll. Darüber hinaus haben die verschiedenen Ebenen jeweils ihre speziellen Notationen und Datenformate für die Beschreibung des Entwurfsobjektes.

Aus diesen und anderen Gründen ist die Zahl der Entwurfs-Hilfmittel und der diesen zugrundeliegenden Notationen und Verfahren sehr zahlreich, sodaß es schwierig ist, die Rolle eines einzelnen Werkzeuges klar zu erkennen und in ein verständliches Gesamtbild des Entwurfsprozesses einzuordnen. Um trotz der Größe des Waldes den Standort einzelner Bäume noch erkennen zu können, bedarf es eines transparenten Koordinatensystems. Im folgenden werden neben der Abstraktionsebene als weitere Koordinaten eingeführt: Art der Einbettung des Werkzeuges in die Abstraktionsebenen, Ort der Einbettung, und die Art der Entwurfsdaten. Dieses Kapitel operiert hauptsächlich aus Sicht des Entwurfes von Vollkunden-Schaltungen (vgl. Kapitel 3).

14.1 Arten von Entwurfsdaten

Bevor wir uns nun dem eigentlichen Entwurfsprozeß zuwenden, sollen die dabei zur Anwendung kommenden Werkzeuge (später in Abschnitt 14.2 und 14.3) und hier deren Ein-/Ausgabedaten erläutert werden.Wir können bei Entwurfswerkzeugen und anderen Werkzeugen im Umfeld des Hardware-Entwurfes folgende 3 Grundklassen von Entwurfsdaten unterscheiden:

Beschreibungen (descriptions): Beschreibung oder Spezifikation von Systemen (Entwurfs-Objekte oder bereits implementierte Systeme, z. B. Hardware-Systeme, -Subsysteme oder -Moduln). Eine Beschreibung läßt sich in bis zu drei verschiedenartige Beschreibungsanteile zerlegen [28] (Beschreibungs-Arten, vgl. Bild 14.1): strukturelle Beschreibung (structural description), Verhaltens-Beschreibung (behavioral description), und physische Beschreibung (z. B. Geometrie beim Layout). Die strukturelle Beschreibung spannt eine eigene Dimension auf, z. B. mit Zellen-Hierarchien (s. Bild 14.1). Wichtig bei Beschreibungen ist auch die Unterscheidung zwischen graphischer (Bild 14.2, s. auch Abschnitt 19.4) und textueller Form.

Aktivierungsdaten (activation data): Stimuli (evtl. kombiniert mit Soll-Antworten) zur Aktivierung eines Systems. Beispiele: Testmuster oder "Tests" (Kombination aus Stimuli und Sollantworten). Testmuster können zum echten Test einer physisch vorgegebenen Hardware verwendet werden, sowie auch zur Simulation, bei der eine Beschreibung als Modell des simulierten Objektes dient.

Diagnosen: kommentierende Aussagen (nicht selbst Beschreibungen) über bestimmte Aspekte

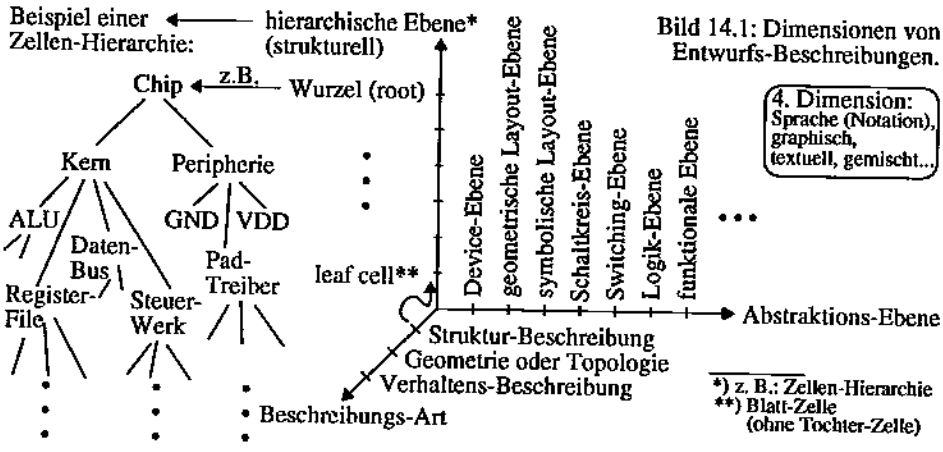


Bild 14.1: Dimensionen von Entwurfs-Beschreibungen.

von Beschreibungen. Beispiele: Prüfungsergebnisse nebst Hinweisen zur Lokalisation entdeckter Fehler, Statistiken etc.

Für alle solche Klassen von Daten sind spezielle Sprachen geschaffen worden, wie beispielsweise Hardware-Beschreibungssprachen (ABL, CVS_BK, KARL, VERILOG, VHDL, etc., vgl. Abschnitt 14.4) und Testbeschreibungssprachen. Diese Sprachen sind meist auf eine oder mehrere bestimmte methodologische Ebenen spezialisiert (vgl. Bild 14.8). Mehrebenen-Sprachen sind oft als Entwurfssprachen praktisch, da man mit solchen gleichzeitig die in einer höheren Ebene dargestellte Spezifikation sowie in einer niedrigeren Ebene das zu entwerfende Objekt beschreiben kann. Dies ist insofern praktisch, als in Zwischenstadien einer beispielsweise manuellen Entwurfsprozedur Spezifikations-Anteile und Objekt-Anteile in der gleichen Beschreibung koexistieren können. In Abschnitt 14.4 werden einige Beispiele von Entwurfsdaten-Formaten eingeführt.

14.2 Werkzeugklassen nach Anwendungsgebieten

Bild 14.3 führt eine graphische Notation zur Darstellung der Werkzeugklassen ein. Dabei ist die Unterscheidung der 3 Datenarten an Ein- und Ausgängen wichtig (Bild 14.3 a-d): Beschreibungen, Diagnosen, oder Aktivierungsdaten (vgl. Abschnitt 14.1). Bild 14.3 e-j zeigt einige

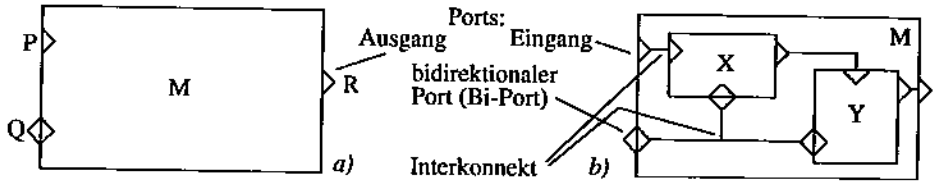


Bild 14.2: Beispiel zur graphischen Form der Struktur-Beschreibung; a) äußere Ansicht (external view), b) innere Ansicht (internal view): Zellen X und Y sind Tochterzellen der Zelle M.

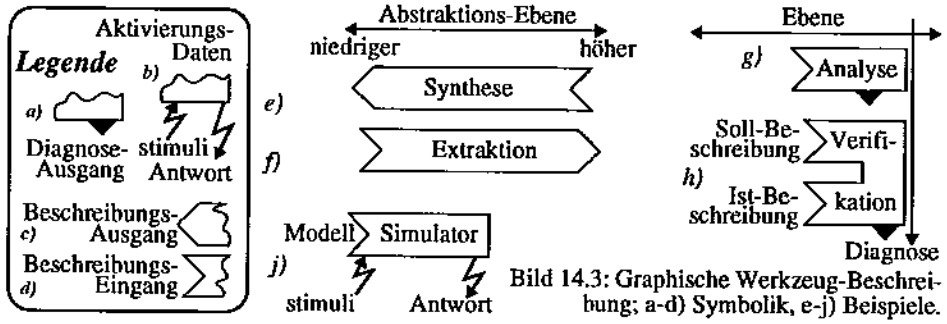


Bild 14.3: Graphische Werkzeug-Beschreibung; a-d) Symbolik, e-j) Beispiele.

Beispiele zur Anwendung dieser graphischen Notation. Die verschiedensten Entwurfswerkzeuge können gemäß den folgenden Anwendungsgebieten klassifiziert werden:

Synthese (Bild 14.3 e): Ableitung einer Beschreibung einer niedrigeren Abstraktionsebene aus einer Beschreibung einer höheren Abstraktions-Ebene. Beispiel: Ableitung eines Schaltplanes (Schaltkreis-Ebene) aus einem Logik-Diagramm (Gatternetz-Ebene). Weitere Beispiele: Kompaktion (durch einen "Kompaktor"), die geometrisches Layout aus symbolischem Layout (z. B. Stick-Diagrammen) ableitet, oder automatische Generierung von Standardzellen-Layout (Beispiel in Bild 14.7) aus der Netzliste (Verdrahtungsplan).

Extraktion (Bild 14.3 f): Ableitung (evtl. auch Rückgewinnung) einer Beschreibung einer höheren Abstraktionsebene aus einer Beschreibung einer niedrigeren Abstraktions-Ebene. Beispiel (Schaltkreis-Extraktor) [54] [68]: Ableitung eines Schaltplanes (Schaltkreis-Ebene) aus dem Layout (Ebene des geometrischen Layout). Weiteres Beispiel (REX [34] [59]): Extraktion von Funktional- und Logik-Beschreibungen aus dem Layout

Prüfung (check, Bild 14.3 g): Prüfung einer Beschreibung einer Abstraktionsebene auf Einhaltung von Entwurfsregeln aus der gleichen Abstraktionsebene. Beispiel: Design Rule Check (DRC): Prüfung von Layout auf Einhaltung des Layout-Regeln. Ergebnis der Prüfung ist eine Diagnostik (o.k., oder im Fehlerfalle die Lokalisation von Fehlerstellen, evtl. auch unter Angabe der verletzten Regel).

Analyse (Bild 14.3 g): ähnlich einer Prüfung. Jedoch werden auch Diagnostiken höherer Abstraktionsebenen gegeben (höher als die Ebene, in der das geprüfte Objekt beschrieben wurde).

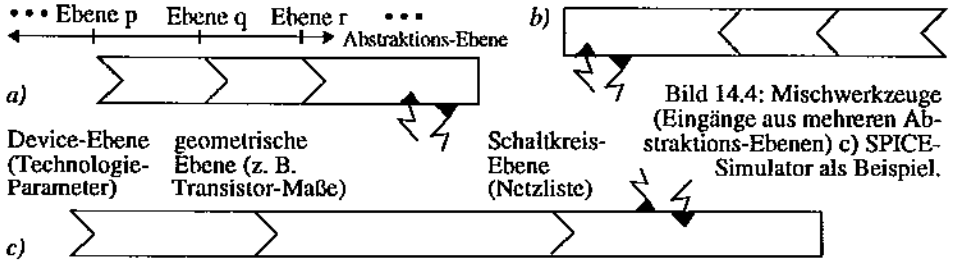


Bild 14.4: Mischwerkzeuge (Eingänge aus mehreren Abstraktions-Ebenen) c) SPICE-Simulator als Beispiel.

Beispiel: electrical rules checker (ERC). Beispielsweise die Diagnostik "Knoten xyz kann nur eine logische Null annehmen" gehört der Gatternetzebene an, wurde jedoch aus der Analyse von Layout (Ebene des geometrischen Layout) gewonnen.

Verifikation (Bild 14.3 h [21] [22]): der Vergleich zweier Beschreibungen (insbesondere aus verschiedenen Abstraktionsebenen stammend). Diagnose: "o.k." (bei Übereinstimmung) oder bei Nicht-Übereinstimmung auch Lokalisierung der Differenzen. Anwendung: Überprüfung, ob ein Entwurf mit seiner Spezifikation übereinstimmt (zwecks Überprüfung der Korrektheit einer Synthese).

Simulation (Bild 14.3 j, s. a. [52]): Bei der Simulation steht kein echtes physisches Objekt zur Verfügung. Das Modell des zu simulierenden Objekts liegt in Form einer Hardware-Beschreibung auf den verschiedenen Ebenen vor. Weitere Eingaben zur Simulation sind die Aktivierungsdaten, d.h. welche Werte wo eingegeben werden und welche Ausgänge und interne Knoten beobachtet werden sollen. Beispiele: Schaltkreis-Simulation, (reine) Logik-Simulation, Timing-Simulation, kombinierte Logik- und Timing-Simulation, RT-Simulation, Mehrebenen- Simulation.

gerichtete Simulation: Die eingegebene Beschreibung des zu simulierenden Objektes enthält nur Elemente, deren Eingänge und Ausgänge als solche bereits im voraus definiert sind (Beispiel: ein logisches Gatter, vgl. Bild 14.5 a-b). Der Simulator kann sich deshalb direkt in einer einseitig verzeigerten Netzlistendarstellung an den Signalflußrichtungen entlanghangeln. Es gibt aber auch kompliziertere Verfahren der Implementierung gerichteter Simulatoren, auf die hier nicht eingegangen wird.

ungerichtete Simulation: Im Gegensatz zur gerichteten Simulation ist hier bei den Elementen der eingegebenen Beschreibung die Signalflußrichtung nicht von vorn herein bekannt. Beispiel (Bild 14.5 c-e): ein Transfer-Transistor oder ein Widerstand (die Signalflußrichtung ergibt sich erst aus der Auswertung der Umgebung dieses Elements. Deshalb sind quasi Relaxations-Verfahren (iterativ!) notwendig.

Test: Ähnlich der Simulation, jedoch tritt ein echtes physisches Objekt (Testobjekt, z. B. eine echte Hardware) an die Stelle einer Beschreibung (Simulations-Modell). Das Testobjekt wird mit Stimuli beaufschlagt. Die darauf vom Testobjekt abgegebenen Antworten (engl.: *responses*) werden mit den Soll-Antworten verglichen. Gebräuchliche Fachbegriffe und Abkürzungen: *DUT* (*device under test*): Testobjekt; Satz von *Testmustern* (*test patterns*): Satz von Paaren, jeweils bestehend aus Stimulus und Sollantwort.

(Automatische) Testerzeugung: Die (automatische) Gewinnung von Testmustern (bzw. auch Aktivierungsdaten für eine Simulation) aus einer Beschreibung. Gebräuchliche Abkürzungen:

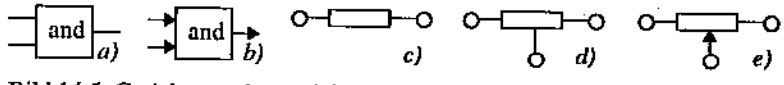


Bild 14.5: Gerichtet und ungerichtet; a) gerichtes Element, b) veranschaulicht (a), c) ungerichtetes Element, partiell ungerichtetes Element, e) zeigt (b).

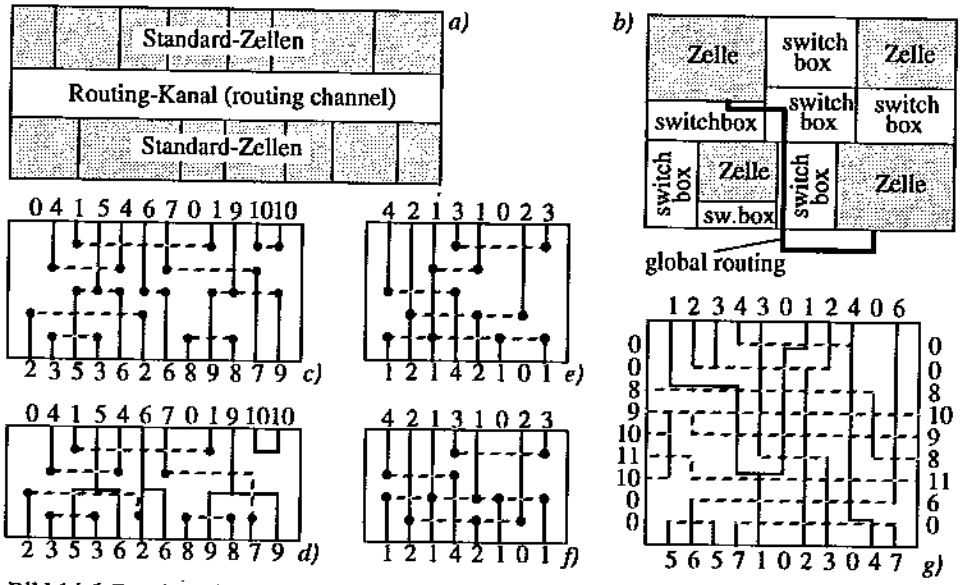


Bild 14.6: Routing: a) *channel routing*, b) *global routing* und *switch box routing*; Algorithmen: c-d) Channel-Optimierung durch Überlagerung der zweiten Ebene, e-f) durch Zusammenführung gleicher Netze (congestion), g) Beispiel eines Switch-box-Problem nach WEAVER [43].

ATPG (*automatic test pattern generation*) [1] [61] [31] [37] [55] [95].

Editor: interaktives (textuelles und/oder graphisches) Werkzeug (Text-Editor bzw. Graphik-Editor) zur "manuellen" Eingabe bzw. Synthese textueller und/oder graphischer Beschreibungen. Beispiel: Layout-Editor (s. dort auf S. 296).

Nachdem nun mit den Dimensionen einer Beschreibung (Abschnitt 14.1) und mit Werkzeugklassen (Abschnitt 14.2) eine Art "Koordinaten-System" eingeführt worden ist, können die im folgenden Abschnitt charakterisierten Beispiele einzelner Werkzeuge auch eingeordnet und gegeneinander abgegrenzt werden. Dieses "Koordinaten-System" soll zur Gewinnung eines geordneten Überblicks verhelfen dadurch, daß die Frosch-Perspektive (folgender Abschnitt) gleichzeitig auch mit der Vogel-Perspektive kombiniert werden kann zur "Giraffen-Perspektive" (vgl. auch Abschnitt 57 und Bild 2.4).

14.3 Beschreibung von Werkzeugen

Die nachfolgenden Abschnitte beschreiben nun eine Auswahl von Werkzeugen die im Entwurfsprozeß angewendet werden. Eine Übersicht dazu gibt Bild 14.8.

Schematics Entry System: bezeichnet ein interaktiv graphisches Eingabewerkzeug für die Erstellung von Netzlisten aus Modul-Bibliotheken und Verdrahtungs-Listen [71].

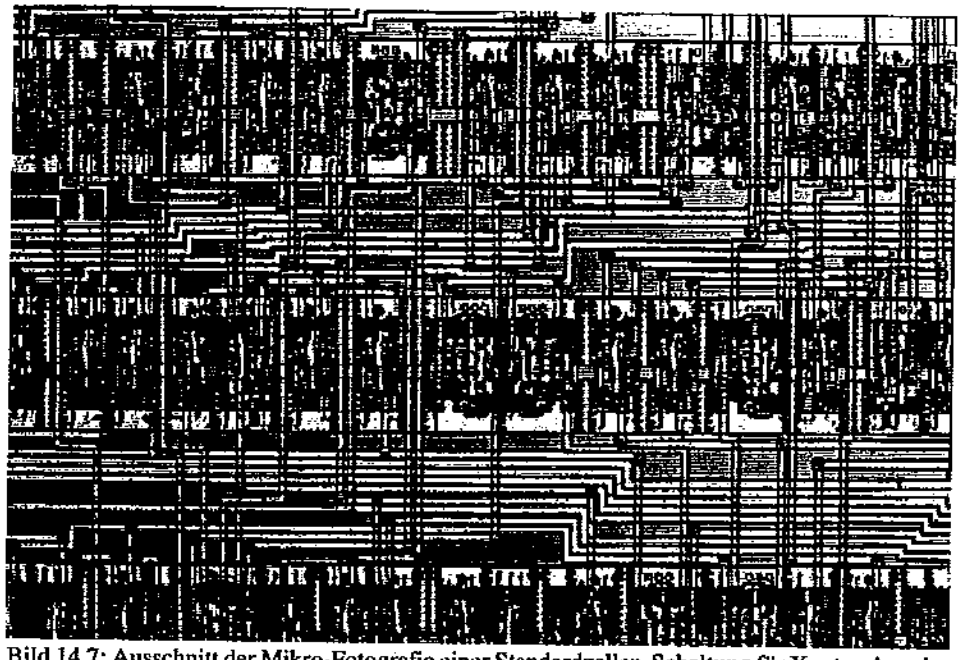


Bild 14.7: Ausschnitt der Mikro-Fotografie einer Standardzellen-Schaltung für Xputer-Anwdg..

Modul-Generator: Programme zur automatischen Erzeugung regelmäßiger Layout-Abschnitte [72], wie beispielsweise Gate-Matrix-Layout [51] (Verfahren nach Uehara-vanCleemput, vgl. Kapitel 18), oder [15], PLAs [64], Weinberger Arrays [100], ALUs [65] etc. Beispiel: ModuleMaker (Fa. CADENCE [63]). Es gibt auch mehr spezialisierte Modul-Generatoren, wie RAM-Generatoren [66] etc.

Routing & Placement-Programm (Programm für Plazierung (von Zellen) und (deren) Verdrahtung, s. a.[14] [47] [88]): Eingabe: eine Zellen-Bibliothek von Moduln (definiert durch Modul-Namen und Ports) und eine Netzliste zur Verbindung der Moduln untereinander. Ein solches Programm plaziert automatisch die Moduln und verdrahtet diese automatisch (möglichst vollständig). Im Gegensatz zu Voll-Kunden-Schaltungen erfordert auf diese Weise generiertes Layout ein Mehrfaches des Flächen-Bedarfs (vgl. z. B. Bild 4.7 mit Bild 4.8 b). Dem steht der Vorteil erheblich kürzerer Entwurfszeit durch Automatisierung gegenüber, vor allem bei sehr komplexen Schaltungen und in noch stärkerem Umfang dann, wenn die Schaltung nur einen geringen Regelmäßigkeits-Faktor erlaubt (vgl. a. Abschnitt 4.2 mit Bild 4.13).

Je nach Modul-Vorrat werden Standard-Zellen-Verfahren (alle Zellen haben die gleiche Höhe: Bild 14.6 a) und Makro-Zellen-Verfahren (fast beliebige Form und Größe der Moduln: Bild 14.6 b) unterschieden. Beispiel: TimberWolf (UC Berkeley) Version MC (Makrozellen) und Version SC (Standard-Zellen) [91]. Bild 14.6 b zeigt eine Aufteilung des Verdrahtungs-Problems in (lokales) switch box routing (s. a. Bild 14.6 g [43]) und global routing (e.g.[48]).

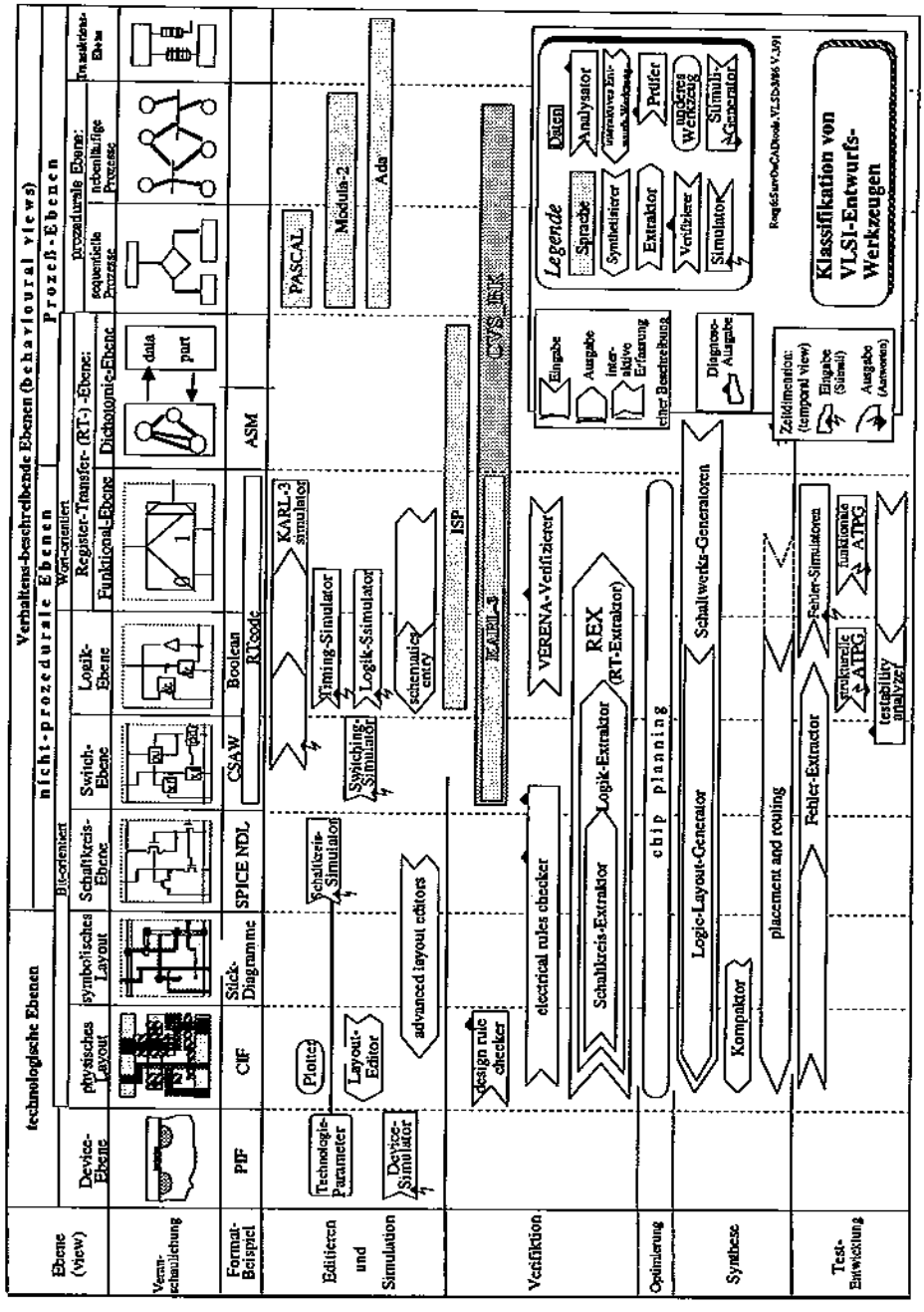


Bild 14.8: Beispiele zur Einordnung von Entwurfs-Werkzeugen, Sprachen und Formaten.

Bei Beschränkung auf Standard-Zellen (Bild 14.6 a und Bild 14.7) kann weniger schwieriges channel routing angewandt werden (Bild 14.6 c-f).

Design Rule Checker (DRC. s. a. [80]): Ein Prüfprogramm, welches geometrisches Layout akzeptiert und Verletzungen geometrischer Entwurfsregeln diagnostiziert, meist nach Polygon-orientierten Verfahren arbeitend. Beispiele: EDGE (Fa. CADENCE [69]). Die meisten DRCs erlauben das Einlesen neuer Entwurfsregeln über einen sogenannten "technology file". Beispiel für einen Gitter-basierten DRC: das PISA-System (Univ. Kaiserslautern) für Lambda-basierte Entwurfsregeln [33] [60] (s. a. Bild 12.28).

Electrical Rule Checker (ERC): Layout akzeptierendes Diagnoseprogramm zur Lokalisierung von Kurzschlüssen, Unterbrechungen, Tautologien, falschen Inverter-Verhältnissen, etc. Ein solches Programm verwendet intern einen Switching-Extraktor, dessen Ausgabe dann analysiert wird. Beispiel: ERC (Fa. CADENCE [70]).

Kompaktor (s. a. [80]): Eine Form von automatischem Layout-Generator. Eingabe: symbolisches Layout (wie beispielsweise Sticks [101], maschinenlesbare Stick-Diagramme, vgl. Bild 12.4 b) oder eine geometrisch interpretierte Netzliste, Ausgabe: geometrisches Layout. Beispiele: CABBAGE [41], REST [56], SLICS [16], oder SPARCS [6].

Layout-Editor: meist sehr komfortables Graphik-Programm zum interaktiven Entwerfen von geometrischem Layout [67]. Frühe Beispiele sind: der ICARUS-Editor [12] (Bild 13.24 b zeigt einen Plot), KIC2 [45] [46] und Caesar [82]. Magic [83] ist ein hochentwickelter Layout-Editor, der ganze Figur-Kombinationen Layoutregel-geprüft einsetzen kann.

CAD-Framework: Ein CAD-Framework stellt dem CAD-Anwender für alle vorhandenen Werkzeuge eine einheitliche Benutzer-Oberfläche und eine einheitliche Datenhaltung (meist in Form einer komfortablen Entwurfs-Datenbank) zur Verfügung. Wichtiges Kriterium eines CAD-Framework ist, daß es sich um ein offenes System handelt, in das eigene Werkzeuge oder Werkzeuge anderer Hersteller voll integriert werden können. Beispiel: FALCON ([23], Mentor Graphics) oder Opus ([73], CADENCE).

Logik-Synthese: Programm zur automatischen Gewinnung (Generierung) von Netzlisten (z. B. für das Standard-Zellen-Verfahren, siehe Routing & Placement) aus Logikdiagrammen oder Logik-Tabellen. Beispiel: MIS-II [4] (UC Berkeley).

Logik-Extraktor: Programm zur automatischen (Rück-)Gewinnung eines Logik-Diagrammes bzw. logischen Gleichungssystems aus Layout oder evtl. einer etwas höheren Abstraktionsebene (z. B. [59], [85]).

Switching Network Extraktor: Programm zur automatischen (Rück-)Gewinnung einer Beschreibung der Switching-Ebene aus Layout, wie beispielsweise [53].

Schaltkreis-Extraktor: Programm zur automatischen (Rück-)Gewinnung eines Schaltplanes aus Layout. Beispiele: [85] oder PDExtract ([68], CADENCE).

Schaltkreis-Simulator: Analog-Simulatoren, die über Differentialgleichungen implementiert

sind [58] [94] [97] [98]. Firmen haben oft ihre eigenen nur intern verwendeten Simulatoren. **Eingabe:** eine Netzliste der Schaltkreis-Ebene (z. B. SPICE NDL [58], vgl. Bild 14.4, s. auch Kap. 13). Die Bauelemente innerhalb einer solchen Netzliste sind durch elektrische und geometrische Parameter spezifiziert, wie individuelle Transistor-Abmessungen, Werte von Widerständen und Kapazitäten. Hinzu kommen als globale Parameter die Technologie-Parameter des vorliegenden Fertigungsprozesses (wie Oxidschicht-Dicken, Dotierungs-Parameter u. a. m.), die der Informatiker meist "quasi unbesehen" von der Technologie-Seite übernimmt. Für Stimuli ist bei dieser Klasse von Simulatoren meist kein "separater Eingang" vorhanden. Meist werden Befehle, welche Impulsformen beschreiben und Spannungswerte etc. angeben, die in einem eigenen Abschnitt an die Netzliste angehängt werden. Für die Ausgabe der Antwortsignale (Analogbereich) ist meist ein Graphik-Unterprogramm vorhanden. Anwendungs-Beispiele für SPICE sind u. a. zu finden in [19] [86].

Logik-Simulator: ein gerichteter Simulator, der von einer Beschreibung der Gatternetz-Ebenen ausgeht [57]. Häufig sind die Stimuli aus einem separaten File eingebbar. Meist haben solchen Simulatoren auch eine interaktive Betriebsart, in welcher der Benutzer direkt mit dem Simulations-Modell experimentieren kann. Beispiele: HILO [75] (Fa. GenRad), Quicksim [76] (Fa. Mentor Graphics).

Timing-Simulator: Simulator, der von einem Zeit-Kontinuum ausgeht und auch den genauen Zeitpunkt des Eintreffens von Ereignissen (Pulsflanken) und die Länge von Impulsen angibt.

Timing Analyzer: ein Analyseprogramm zur Auffindung zeitkritischer Pfade aus einer Hardware-Beschreibung (Netzliste) [44] [84].

Mehrebenen-Simulator: Simulator, der eine Mehrebenen-Sprache für Beschreibungen akzeptiert. Beispiele: KARL-3-Simulator [99] [62], Thermis-Simulator [9] (nicht-prozedurale RT- Ebene, Gatternetz-Ebene und Pseudo-Switching-Ebene), VERILOG [96] [29] [74], VHDL [7] [50] [2] [49] (IEEE-Standard, allerdings nicht unumstritten).

RT-Simulator: Simulator, der Beschreibungen der RT-Ebene (Register-Transfer-Ebene) akzeptiert. In der Register-Transfer-Ebene werden vorzugsweise Datenpfad-Strukturen beschrieben, die aus Registern und Schaltnetzen (Funktionen und anderen Operatoren) bestehen, was in Bild 14.9 veranschaulicht wird.

Switching-Simulator: eine spezielle Form von nicht-gerichtetem Logik-Simulator (s. [5], [13], [86] et al., es werden meist nur folgende logische Signalwerte unterschieden: 0, 1, *(hochohmig), ?(undefiniert)). Beispiel: MUSA (UC Berkeley [92]). Die Zeitskala unterscheidet nur diskrete Zeitschritte. Switching-Simulatoren akzeptieren als Eingabe Netzlisten der Switching-Ebene. Die Ausgabe wird durch Relaxations-Verfahren ermittelt.

Hierarchie: Heutzutage werden Zellen-Hierarchien, allein schon aus Komplexitätsgründen, praktisch von allen Werkzeugen und Datenformaten unterstützt, weshalb dies bei der detaillierten Behandlung nicht besonders erwähnt wird. Werkzeuge, die nur mit "flachgeklopften" Entwürfen (die Hierarchie ist aufgelöst) umgehen können, sind im allgemeinen nur für kleine Entwurfs-Probleme einsetzbar und finden deshalb kaum Verwendung. Von manchen Werk-

zeugen wird eine "flache" Darstellung als Internformat verwendet, meist unsichtbar für den Benutzer.

Diese Liste von Entwurfs-Hilfsmitteln ist bei weitem nicht vollständig. Es werden lediglich einige exemplarische Beispiele gegeben, welche in erster Linie der Erläuterung des Koordinatensystems zur Ordnung der Werkzeuge und Datenformate dient. Eine gute (wenngleich nicht die allerneuste) Übersicht über Methoden (weniger über kommerziell angebotene Software) bietet die Buchreihe [81] mit den Bänden [11] [40] [90] [80] [102] [20] [28].

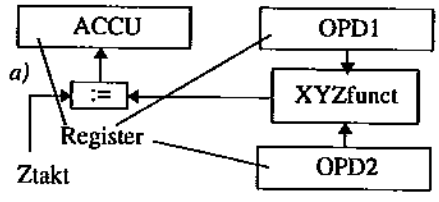
14.4 Entwurfs(daten)-Sprachen und -Formate

Bei der Vielfalt der existierenden Entwurfs-Werkzeuge ist es klar, daß es fast ebenso viele Entwurfsdaten-Formate gibt. Um die Anzahl der Entwurfs-Datenformate zu minimieren, hat sich die Industrie auf die Benutzung von Standards (EDIF, VHDL) oder zumindest auf Quasi-Standards (z.B. SPICE-Netzlisten) geeinigt.

EDIF: "Electronic Design Interchange-Format" (ANSI-Standard - Version 2 0 0): ein umfassendes Format zur Beschreibung von Netzlisten, symbolischem Layout, physical Layout, Schematics, PCB-Layout, geometrische Entwurfsregeln, Logische Modelle (ANSI steht für: American National Standards Institute) u. v. a. m. EDIF ist für den Datenaustausch zwischen verschiedenen Werkzeugen und Entwurfs-Umgebungen vorgesehen, jedoch nicht als eine "Benutzer-Schnittstelle" [27] [10]

VHDL: Eine auf Betreiben des IEEE standardisierte Mehrebenen-Hardware-Beschreibungssprache, die als Benutzerschnittsstelle aber weniger gut geeignet ist. Diese Sprache ist m. E. eher eine (Hardware/Software-) System-Beschreibungssprache, fast ein Ada-Dialekt [7] [50] [2] [49]. Die Semantik von VHDL ist durch den Standard nicht erfaßt, sondern wird durch zusätzliche "Packages" definiert (was Anlaß zu Kompatibilitäts-Problemen ist).

CIF (Caltech Intermediate Form): im universitären Bereich weit verbreitetes Format (insbesondere: Austausch-Format) für geometrisches Layout (Caltech: California Institute of Technology, Pasadena, Kalifornien) [54] (s. a. [86] und [88]). Viele Silicon-Broker (z. B. auch der MOSIS-Dienst [8] und das Eurochip-Projekt [78] (sowie das inzwischen leider eingestellte E.I.S.-Projekt [25]) akzeptieren (bzw. akzeptierten) auch in CIF eingereichte Entwürfe für die



b)

al Ztakt do ACCU := XYZfunct (OPD1, OPD2);

Bild 14.9: Register-Transfer; a) Datenpfad-Blockdiagramm, b) textuelle Beschreibung.

Fertigung.

GDS-II: in der Industrie am weitesten verbreitetes Format für geometrisches Layout (Industrie-Quasi-Standard)

Calma: Format für geometrisches Layout (Industrie-Quasi-Standard)

SPICE NDL: Von SPICE und manchen anderen Werkzeugen unterstütztes Netzlistenformat [58] (s. a. [86] u. [19]).

Waves: Testdatenformat (Quasi-Standard, im IEEE-Standardisierungs-Prozeß) [77].

BLIF (Berkeley Logic Interchange Format): Format der Logik-Ebene als Quellen- und Objekt-Format einer Reihe experimenteller Entwurfs-Werkzeuge insbesondere der University of California at Berkeley (UCB) [93].

Die obige Liste erhebt bei Weitem keinen Anspruch auf Vollständigkeit. Einige der obigen Formate sowie eine Anzahl anderer Formate werden beispielsweise in [89] kurz eingeführt.

14.5 Literatur

[1] G. Alfs, R. W. Hartenstein, A. Wodtko: The KARL/KARATE System: Automatic Test Pattern Generation Based on RT-Level Descriptions, Proceedings of the International Test Conference, pp. 230- 235, 1988

[2] J. Bhasker: VHDL Primer; Prentice-Hall, 1992

[3] A. Bonomo, G. Girardi, L. Lavagno, R. Hartenstein, R. Hauck: Semantic Specification of **CVS_BK** Language (**CVS Behavioral Karl**); ESPRIT / CVS report, CSELT, Torino, Italy /Informatics Dept., Univ. Kaiserslautern, 1987

[4] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, A. Wang: MIS: A Multiple-Level Logic Optimization System, IEEE Tr-CAD. vol. CAD-6, no. 6, pp. 1062-1081, 1987.

[5] R. Bryant: MOSSIM: A logic-level simulator for MOS LSI; MIT Lab for Computer Science, MIT, Cambridge, Mass, Sept. 1979

[6] J. L. Burns, A. R. Newton: SPARCS anew constrained based IC symbolic layout spacer; Transactions IEEE Custom Integrated Conference pp 534-539, May 1986

[7] D.R. Coelho: The VHDL Handbook, Kluwer Academic Publishers, 1989.

[8] D. Cohen: MOSIS - User Interface for Silicon Foundries; in: [38]

[9] M. Doshi, R. B. Sullivan, D. Schuler: Thermis Logic Simulator - A Mixed Mode, Multilevel, Hierarchical, Interactive Digital Circuit Simulator, Proc. 21st DAC, 1984.

[10] EDIF Steering Commitee: EDIF - Electronic Design Interchange Format Version 2 0 0; EIA Electronic Industries Association; Washington D.C., 1988.

[11] W. Engl: Process and Device Modeling; North Holland, Amsterdam, 1986

[12] D. Fairbairn, J. Rowson: ICARUS: An Interactive Integrated Circuit Layout System; Proc. 15th DAC, June 1978

[13] W. Fichtner, M. Morf: VLSI CAD Tools and Applications; Kluwer, 1987

[14] W. Fischer, R. Schüfny; MOS-VLSI-Technik; Akademie-Verlag, Berlin 1987

[15] P. Gee, M. Y. Wu, S. M. Kang, I. N. Hajj: A Metal-metal matrix cell generator for multilevel metal MOS technology; Integration 9 (1990) S. 25 - 47

[16] D. Gibson, S. Nance: SLIC - Symbolic Layout of Integrated Circuits; 13th DAC, 1976

[17] G. Girardi, R. Hartenstein, U. Welters: ABLED: a RT level Schematic Editor and Simulator user Interface; Int'l EUROMICRO Symposium; Brussels, Belgium, 1985

[18] G. Girardi, R. Hartenstein, U. Welters: ABL - an interactive graphic user interface in microelectronics; in (Hrsg.: J.Encarnaçao): CAD-Schnittstellen und Datentransfer-Formate im Elektronik-Bereich; Springer-Verlag, Berlin / Heidelberg / New York 1986

[19] L. A. Glasser, D. W. Dobberpuhl: The Design and Analysis of VLSI Circuits; Addison-Wesley, 1985

[20] S. Goto: Design Methodologies; North Holland, Amsterdam, 1986

[21] W. Grass, N. Schielow: Automatische Verifikation logischer Entwürfe; Informatik-Fachberichte 84; Springer-Verlag, 1984, p.113-126

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

[23] D. S. Harrison, A. R. Newton, R. L. Spickelmeier, T. J. Barnes: Electronic CAD frameworks; Proc. IEEE, Vol. 78, No. 2, 1990.

[24] R. Hartenstein: KARL reference manual; CVT report, Univ. Kaiserslautern, 1986

[25] R. Hartenstein: Shared Cultures: CIF Library, Starting Frames and Scalable Design Rules; [42]

[26] R. Hartenstein: Ausbruch der E.I.S.-Zeit, Computer Magazin, März 1986

[27] R. Hartenstein: EDIF; Kurs-Handout, IT Press Verlag, Bruchsal 1989

[28] R. Hartenstein: Hardware Description Languages; North Holland, Amsterdam, 1987

[29] R. Hartenstein: Einführung in den VLSI-Entwurf; Skriptum, Universität Kaiserslautern, 1992, auch als Buch: IT Press Verlag (in Vorbereitung)

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

[31] R. Hartenstein: The History of KARL; Bericht, Univ.Kaiserslautern 1993

[32] R. Hartenstein: KARL and ABL; in (ed.: J. P. Mermet): Fundamentals and Standards in Hardware Description Languages; Kluwer Academic Publishers, Boston, 1993

[33] R. Hartenstein, R.Hauck, A. Hirschbiel, W. Nebel, M. Weber: PISA, a CAD Package and Special Hardware for Pixel Oriented Layout Analysis, Dig.Techn.Pap. ICCAD 1984.

[34] R. Hartenstein, W. Nebel: Functional Design Verification by Register Transfer Net Extraction from Integrated Circuit Layout Data; Proc. IEEE COMP EURO, Hamburg, 1987

[35] R. Hartenstein, E. von Puttkamer: KARL - a Hardware Description Language as a part of a CAD tool for VLSI; CHDL'79, Int'l Symp. on Computer Hardware Description Languages and their Applications, Palo Alto, CA, USA, 1979; IEEE New York, 1979

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

[37] R. Hartenstein, A Wodtko: Automatic generation of Functional Test Patterns from RT language source; Int'l EUROMICRO Symposium; Brussels, Belgium, 1985

[38] R. Hartenstein, K. Woelcken: Proc. European Conf. on Customer / Vendor Interfaces in Microelectronics (EURO CVIM), Kaiserslautern 1986; GMD, St. Augustin,1987

[39] H. Hedengran: A Parser for KARL 2, internal report; Dept. of Applied Electronics, The Royal Inst. of Technology, Stockholm, Sweden, 1980

[40] E. Hörbst: Logic design and Implementation; North Holland, Amsterdam, 1986

[41] M. Y. Hsueh: Symbolic Layout and Compaction of Integrated Circuits; Report no. UCB/ERL/M79/80, University of California, Berkeley, CA, 1979

[42] 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
[43] R. Joobani, D. Siewiorek: WEAVER: A knowledge-based routing expert; 2nd DAC 1985
[44] N. P. Jouppi: Timing Analysis and Performance Improvement of MOS VLSI Designs: IEEE Trans. CAD 6,4 (July 1987)
[45] K. Keller, A. R. Newton: KIC2: A Low Cost Interactive Editor for Integrated Circuits Design; IEEE Compcon 1982
[46] K. Keller, G. Billingsley: KIC2: Integrated Circuit Layout Program; University of California, Berkeley, CA, April 1983
[47] R. Kolla, P. Molitor, H. G. Osthof: Einführung in den VLSI-Entwurf; Teubner, 1989
[48] C. Lee: An algorithm for path connection and its application; IRE Trans. Computers 1961
[49] S. S. Leung, M. A. Shanblatt: ASIC System Design with VHDL; Kluwer, 1989
[50] R. Lipset, C. Schäfer, C. Usserey: VHDL Hardware Description & Design, Kluwer 1989
[51] A. D. Lopez, H. F. Law: A Dense Gate Matrix Layout Method for MOS VLSI, IEEE Journal of Solid State Circuits, vol. SC-15, no. 4, pp. 736-740, 1980.
[52] P. Marwedel: Synthese und Simulation von VLSI-Systemen; Hanser, 1993
[53] S. McCormic: EXCL: A Circuit Extractor for IC Designs, Proc. 21st DAC; 1984.
[54] C. Mead, L. Conway: Introduction to VLSI Systems; Addison Wesley; 1980
[55] 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
[56] R. Mosteller: REST - A leaf cell design system; in: VLSI'81, Academic Press, 1981
[57] S. Murai, H. Matsushita, K. Enomoto: Logic Simulation Programs, in [40]
[58] L. W. Nagel: SPICE: A Computer Program to Simulate Semiconductor Circuits, University of California Berkeley, Electronic Research Lab, Report ERL-M 520, 1975.
[59] W. Nebel: REX - Automated Extraction of RT-Level Descriptions from Integrated Circuit Layout Data; Dissertation, Universität Kaiserslautern, 1986
[60] W. Nebel: CAD-Entwurfskontrolle in der Mikroelektronik; Teubner, Stuttgart 1985
[61] N. N.: CVT-TIGER Algorithms and Tool Description, rep., CSELT Torino Italy, 1985.
[62] N. N.: KARL-3 Reference Manual, Universität Kaiserslautern, 1986
[63] N. N.: Design Synthesis, Volume 1 & 2, Reference manual; CADENCE Design, 1989.
[64] N. N.: ES2 ECPD15 & ECPD12 Library Databook, Chapter 4: Compiled PLA Megacells; European Silicon Structures Limited, 1992
[65] N. N.: ES2 ECPD15 & ECPD12 Library Databook, Appendix B: Compiled 2901 Megacells; European Silicon Structures Limited, 1992
[66] N. N.: ES2 ECPD15 & ECPD12 Library Databook, Chapter 2: Compiled RAM Megacells; European Silicon Structures Limited, 1992
[67] N. N.: Physical Design, Reference Manual; CADENCE Design Systems, Inc.; 1989.
[68] N. N.: EDGE Physical design Verification, Reference manual, Chapter 5: PDextract (Connectivity Extraction); CADENCE Design Systems, Inc.; 1989.
[69] N. N.: EDGE Physical design Verification, Reference manual, Chapter 4: PDcheck (Design Rule Checking); CADENCE Design Systems, Inc.; 1989.
[70] N. N.: EDGE Physical design Verification, Reference manual, Chapter 7: ERC (Electrical Rules Checking); CADENCE Design Systems, Inc.; 1989.
[71] N. N.: Design Entry, Reference manual; CADENCE Design Systems, Inc.; 1989.
[72] N. N.: Design Synthesis Volume 1, Reference manual, Chapter 4: ModuleMaker Library (Standard Cell Generator); CADENCE Design Systems, Inc.; 1989.

[73] N. N.: Design Framework II, Reference Manual; CADENCE Design Systems, Inc.; 1991
[74] N. N.: VERILOG-XL Users Guide; CADENCE Design Systems, Inc.; 1990.
[75] N. N.: System HILO Reference Manual; GenRAD, Limited, 1988.
[76] N. N.: Quicksim Users Manual; Mentor Graphics Corp., 1990.
[77] N. N.: Tekwaves 1.0, Users Manual; Tektronix Inc., 1990.
[78] N. N.: Eurochip Services (Academic); VLSI Des. Train'g Action; CEC, DG XIII, Sept 1993
[79] N. N.: Eurochip Services for Industry; VLSI Des. Train'g Action; CEC, DG XIII, Sept 1993
[80] T. Ohtsuki: Layout Design and Verification; North Holland, Amsterdam, 1986
[81] T. Ohtsuki: Advances in CAD for VLSI; North Holland, Amsterdam, 1986 - 1987 (7 Bände: [11] [40] [90] [80] [102] [20] [28])
[82] J. K. Ousterhout: Caesar: an Interactive Editor for VLSI; VLSI Design 2,4 (4th Qu.1981)
[83] J. K. Ousterhout et al.: Magic: A VLSI Layout System, Proc. 21th DAC 1984
[84] J. K. Ousterhout: CRYSTAL, Octools Manual Pages, University of California Research Laboratory, April 1989.
[85] G. Pelz, U. Roettcher: Pattern Matching and Refinement Hybrid Approach to Circuit Comparison; IEEE Trans. on CAD, 1994
[86] H.-U. Post: Entwurf und Technologie hochintegrierter Schaltungen; Teubner, 1989
[87] G. Roos, E. Bernath, T. Büchner, S. Rust, T. Schwederski: Dedicated VLSI Processors for Image Processing; GME-Conf. on VLSI; Baden-Baden 1991; VDE-Verlag, 1991
[88] W. Rosenstil, R. Camposano: Rechnergestützter Entwurf hochintegrierter MOS-Schaltungen; Springer-Verlag, 1989
[89] S. M. Rubin: Computer Aids for VLSI Design; Addison-Wesley, 1987
[90] A. Ruehli: Circuit Analysis, Simulation and Design; North Holland, Amsterdam, 1986
[91] C. Sechen, A. Sangiovanni-Vincentelli: Timberwolf 3.2: A New Standard Cell Placement and Global Routing Package, Proc. 1986 Design Automation Conf., June 1986.
[92] R. B. Segal: MUSA, Octools Manual Pages, University of California Research Laboratory, April 1989
[93] R. B. Segal: Berkeley Logic Interchange Format (BLIF), University of California Research Laboratory, April 1989.
[94] H. Sibbert: DOMOS: A Nonlinear Transient Simulation and Optimization Program for Integrated MOS Circuits, User Manual, Universität Dortmund, 1982.
[95] I. Stamelos, M. Melgara, M. Paolini, S. Morpurgo, C. Segre: A Multi-Level Test Pattern Generation and Validation Environment; International Test Conference, 1986
[96] D. Thomas, P. Moorby: The VERILOG Hardware Description Language, Kluwer, 1991.
[97] P. W. Tuinenga: SPICE -. A Guide to Circuit Simulation and Analysis Using PSPICE; Prentice-Hall, 1992
[98] A. Vladimirescu, S. Liu: The Simulation of MOS integrated circuits using SPICE2; Memorandum no. UCB/ERL M80/7, Electronics Research Laboratory, University of California, Berkeley, Febr. 1980
[99] B. Weber: Implementierung eines KARL Simulators, Dipl.Arb, Univ. Kaiserslautern 1981
[100] A. Weinberger: Large Scale Integration of MOS Complex Logic: A Layout Method; IEEE Journal of Solid State Circuits, Vol. SSC-2, pp.182-190; DEDC. 1967
[101] J. Williams: Sticks - A New Approach to LSI Design; Master's Thesis, MIT, June 1977
[102] T. Williams: VLSI Testing; North Holland, Amsterdam, 1986