Universität-Gesamthochschule Siegen
Fachbereich: Elektrotechnik und Informatik
Fachgruppe: Technische Informatik
Prof. Dr.-Ing. Hans Wojtkowiak

Verbundprojekt DASSY:
Datentransfer und Schnittstellen
für offene integrierte VLSI-Entwurfssysteme

Förderkennzeichen: NT 2860 I6

Schlußbericht

Klaus Quibeldey-Cirkel

14. Januar 1993

Inhaltsverzeichnis

1 Forschungsziele **1**
1.1 Projekt-Meilensteine 3

2 Forschungskontext: VLSI/CAD-Frameworks **5**
2.1 Die W-Fragen 5
2.2 VLSI/CAD-Infrastrukturen 6
2.2.1 Schnittstellen 7
2.2.2 Projektlandschaft 10
2.3 VLSI/CAD-Weltmodelle 12
2.3.1 VLSI-Spezifika 14
2.3.2 Informationsmodellierung 17
2.3.3 Datentransferformate 19

3 Stand der Forschung **25**
3.1 EDIF-Spezifikation für den Austausch von Testdaten 25
3.1.1 Arbeitsphasen 26
3.2 DASSY-Datenmodell: DaDaMo 28
3.2.1 Anforderungsanalyse 28
3.2.2 Modellierungskonzepte für eine allgemeine Werkzeugschnittstelle . . . 36
3.3 DASSY-Prototyp 38
3.3.1 Prototyp-Schema 39
3.3.2 Integration der Siegener Entwurfswerkzeuge 41
3.4 Exkurs: Algebraische Spezifikation einer Werkzeugschnittstelle 43

Literaturverzeichnis **45**
Benutzte Literatur 45
DASSY-B-Hauptdokumente 48
Veröffentlichungen und interne Berichte (Siegen) 48

Kapitel 1

Forschungsziele

„Offene integrierte VLSI-Entwurfssysteme“ setzen vor allem eins voraus: international akzeptierte Industriestandards für CAD-Schnittstellen. Der Status quo zu Beginn des DASSY-Verbundprojekts war gekennzeichnet durch erste Ansätze, zumindest den *Datenaustausch* zwischen den Entwurfswerkzeugen zu vereinheitlichen. Das Austauschformat für elektronische Entwurfsdaten EDIF und die Hardwarebeschreibungssprache VHDL waren (und sind) hier die Hoffnungsträger. Der Status quo am Ende des DASSY-Projekts stellt sich durch konzeptionell und formal standardisierte Richtlinien und Empfehlungen auf internationaler Ebene für den deskriptiven Datentransfer und für den prozeduralen Datenzugriff dar. In der Entwurfsautomatisierung (*Electronic Design Automation*, EDA) ist neben der Informationsmodellierung der Datenaustauschformate zusätzlich die Datenmodellierung für prozedurale Werkzeugschnittstellen getreten.

Die globale Zielsetzung des Verbundprojekts DASSY („Datentransfer und Schnittstellen für offene integrierte VLSI-Entwurfssysteme“ [1]) umfaßt die einheitliche Modellierung aller Entwurfsdaten, die dem konkurrierenden Zugriff der Entwurfswerkzeuge unterliegen und von einem zentralen Datenhaltungssystem verwaltet werden. Dies betrifft den externen Datenaustausch zwischen EDA-Systemen verschiedener Anbieter untereinander sowie die Schnittstellen zwischen EDA-Systemen, ATE-Komponenten (*Automatic Test Equipment*) und ASIC-Herstellern. Als Datentransferformat dient EDIF, für das Erweiterungen (*Test, TCAD, Device Modeling & Verification*) und Verbesserungen hinsichtlich einer eindeutigen semantischen Hinterlegung (Informationsmodelle) in die entsprechenden EDIF-Gremien eingebracht werden [18][19][76]. Zum andern stehen die Konzeption und die prototypische Evaluierung einer standardisierten prozeduralen Werkzeugschnittstelle im Vordergrund. Unabhängig vom eingesetzten Datenhaltungssystem sollen die Daten- und Zugriffsstrukturen der Entwurfswerkzeuge organisiert und verwaltet werden [51][53][54].

Aus dieser Zielsetzung heraus kristallisierten sich die folgenden Schwerpunkte des DASSY-Vorhabens:

- Informationsmodellierung im EDIF-Bereich

EDIF hat sich vom *De-facto-* zum *De-jure-*Austauschformat für elektronische Entwurfsdaten gewandelt [17]. Um den etablierten Standard um weitere Aspekte des Entwurfsprozesses (Test- und Prozeßdaten) zu ergänzen, ohne einer Syntax-Inflation Vorschub zu leisten, sind die bereits eingeführten syntaktischen Konstrukte mit ihrem definitiven Bedeutungsgehalt zu hinterlegen. Die Beschreibung der EDIF-Semantik muß dabei formal erfolgen, um eine rechnerunterstützte Dokumentation und Konsistenzprüfung zu gewährleisten. Das STEP-Produkt EXPRESS [41], eine formale objektorientierte Modellierungssprache, hat sich hier international durchgesetzt.

- Datenmodellierung im Kontext prozeduraler Werkzeugschnittstellen

  Die Schnittstellenproblematik beschränkt sich nicht allein auf die Kommunikation der Entwurfswerkzeuge durch den Austausch formatierter Daten. Sollen heterogene Entwurfswerkzeuge in *datenbank*gestützten Entwurfssystemen von Drittanbietern integriert werden, so müssen die gemeinsamen Entwurfsdaten einzeln oder im jeweils interessierenden Verbund interpretierbar sein. Eine Integration auf der Grundlage von Import/Export-Diensten leistet dies nicht. Um den Aufwand für die Integration von Werkzeugen ohne eigene Datenzugriffsschnittstellen effektiv zu verringern, ist ein allgemeines Datenmodell für datenbankgestützte VLSI-Applikationen zu erstellen. Mit dessen Hilfe können geeignete Datenbankschemata geschrieben werden, mit denen Organisation und Verwaltung der Entwurfsdaten sowie die Operationen auf ihnen einheitlich festgelegt werden.

- Mitarbeit in internationalen Standardisierungsgremien

  Standardisierte Dienstleistungen und Schnittstellen in VLSI-Infrastrukturen sind nur dann von Einfluß und Dauer, wenn sie aktiv von EDA-Herstellern und -Anwendern unterstützt werden. Standardisierungsgremien wie CFI (*CAD Framework Initiative*) und EIA/EDIF (*Electronic Industries Association*), die von führenden Halbleiterkonzernen und Software-Systemhäusern getragen werden, sind hier die erfolgversprechenden Adressaten. Die während der Projektlaufzeit aufgekommene Framework-Diskussion deckt sich in ihrer Thematik und Zielsetzung (standardisierte EDA-Infrastrukturen) mit der des DASSY-Projekts, so daß hier eine weitere Gewichtung der DASSY-Vorschläge erzielt werden konnte.

- Prototypische Evaluierung der DASSY-Konzepte

  Die Akzeptanz der erarbeiteten DASSY-Standardisierungsvorschläge hängt selbstverständlich von deren Effizienz und industriellen Realitätsnähe ab. Eine Prototyp-Demonstration auf den einschlägigen Industriemessen und Konferenzen mit Industriebeteiligung (CeBIT und DAC, *Design Automation Conference*), um die Implementierbarkeit und Leistungsfähigkeit der DASSY-Konzepte zu zeigen, war deshalb unumgänglich.

Die aufgeführten Schwerpunkte im DASSY-Projekt haben organisatorisch und arbeitsteilig zu einer Gliederung in zwei Teilbereiche geführt:

Teil-A: Modellierung, Erweiterungen und Standardisierung des Datentransferformats EDIF

Teil-B: Konzeption, Entwicklung und Standardisierung einer Werkzeugschnittstelle (WZS)

DASSY-Partner				*Teil A/EDIF*	*Teil B/WZS*
TH Darmstadt/GRIS	(GRIS)	:	Prof. Encarnação	√	√
TH Darmstadt		:	Prof. Piloty		√
Universität-GH Duisburg		:	Prof. Hunger	√	
FernUniversität-GH Hagen	(HA)	:	Prof. Schlageter	√	√
TU Hamburg-Harburg	(HH)	:	Prof. Paul	√	
Universität Kaiserslautern	(KL)	:	Prof. Hartenstein	√	
Universität-GH Siegen	(SI)	:	Prof. Wojtkowiak	√	√
CADLAB/Universität-GH Paderborn	(PB)	:	Prof. Rammig		√
CADLAB/SNI AG		:	Dr. Steinmüller		√
GMD St. Augustin		:	Prof. Camposano	√	√

Tabelle 1.1: Projektpartner und ihre Zugehörigkeit zu den DASSY-Themen

1.1 Projekt-Meilensteine

Tabelle 1.1 zeigt die Zugehörigkeit der Projektpartner zu den DASSY-Themen A und B. Eine Arbeitsteilung war zum einen wegen des Gesamtvolumens des Vorhabens unbedingt erforderlich und zum andern auch vielversprechend und effektiv, da die im Hochschulprojekt „Entwurf integrierter Schaltungen“ (E.I.S.) gewonnenen Ergebnisse und Erfahrungen sowie die dort eingeführten und bewährten Kommunikationsstrukturen zwischen den alten und neuen Projektpartnern unmittelbar genutzt werden konnten.

Die Tabelle soll nicht den Eindruck erwecken, daß die zehn Projektpartner Teilaspekte des Gesamtvorhabens isoliert voneinander erforschten. Im Gegenteil: Die fünf Meilenstein-Workshops und die ungezählten gemeinsamen Arbeitstreffen der Teile A und B zwischen den Meilensteinen sorgten für einen regen wissenschaftlichen Meinungsaustausch und für eine effektive Teamarbeit. Gruppen- und Eigenleistungen lassen sich schließlich in einem Verbundprojekt, das seinen Namen verdient, nicht differenzieren. Dem wurde Rechnung getragen, indem die Dokumentation der DASSY-Gesamtergebnisse inhaltlich und editorisch gemeinsam verfaßt wurde.

Wenn wir also im folgenden die DASSY-Forschungsergebnisse der Fachgruppe Technische Informatik der Universität-GH Siegen darstellen, so sollen hier nur die institutsspezifischen Arbeiten hervorgehoben werden, ohne zu verschweigen, daß sie im wesentlichen auf gemeinschaftlich erarbeiteten Konzepten beruhen.

Die Teilergebnisse aus den DASSY-Bereichen „EDIF“ (Teil A) und „Werkzeugschnittstelle“ (Teil B) wurden auf fünf Meilenstein-Workshops dem Projekt-Plenum vorgestellt und ausführlich diskutiert (zuletzt mit Industriebeteiligung). Tabelle 1.2 zeigt die Chronik der DASSY-Meilensteine im Überblick. Die entsprechenden DASSY-Dokumente aus Teil B sind im Anhang referenziert. Die Gesamtdokumentation des DASSY-Projekts wird von der Gesellschaft für Mathematik (GMD), St. Augustin, verwaltet.

Meilenstein	*Koordination*	*Ergebnisse: Teil A/Teil B*
M0 : 1. April 1989	GMD	Teilziele, Arbeitspakete
M1 : 1. April 1990	HH/HA	Anforderungsspezifikation: EDIF, WZS [51][52]
M2 : 1. Jan. 1991	HH/PB	Definition der EDIF-Erweiterungen/Modellierungskonzepte
M3 : 1. Juli 1991	KL/SI	EDIF-Erweiterungen/WZS-Modellierung
M4 : 1. April 1992	KL/GRIS GMD	EDIF-Erweiterungen/DASSY-Datenmodell (DaDaMo) DASSY-Prototyp-Szenario
M5 : 31. Juli 1992	KL GRIS GMD	EDIF-Informationsmodell und Erweiterungen [18][19][76] DaDaMo [53] DASSY-Prototyp [54]

Tabelle 1.2: DASSY-Arbeitsplan

Kapitel 2

Forschungskontext: VLSI/CAD-Frameworks

Der Framework-Technologie wird eine maßgebliche Rolle in der Wachstumsentwicklung des EDA-Marktes im allgemeinen und der ASIC-Industrie im besonderen zugesprochen [2]. Um den Produkt-Entwicklungszyklus weiter zu straffen, werden Soft- und Hardware am Ende der 90er Jahre ausschließlich in standardisierten CAD-Infrastrukturen entworfen. Kürzere Markteinführungszeiten sind nicht mehr allein durch leistungsgesteigerte Werkzeuge möglich, sondern nur noch im standardisierten Verbund aller Entwurfskomponenten: Benutzerführung, Methoden, Werkzeuge und Datenmodelle.

2.1 Die W-Fragen

Das Chip-Zeitalter der Höchstintegration hat ein kaum zu überschauendes CAx-Szenario hervorgebracht, treffend umschrieben durch die Metapher *Islands of Automation* [21]. CA-Entwurfsmethoden für applikationsspezifische integrierte Schaltungen und mannigfaltige CA-Techniken der Entwurfssynthese und -verifikation bilden ein komplexes Instrumentarium. Mit Einführung der *simulierten Netzliste* als formaler Schnittstelle zwischen Chip-Entwerfer und ASIC-Hersteller sind Entwurf und Test in die alleinige Verantwortung des Hardware-Entwicklungsingenieurs übergegangen. Einerseits liegt die erweiterte Kompetenz im Sinne des Entwicklers: als Vorzüge der vorgezogenen Netzlisten-Schnittstelle seien hier nur die verkürzte *Turn-around*-Zeit und die höhere Entwurfssicherheit genannt. Andererseits werden hohe Anforderungen an sein methodisches Systemdenken gestellt: „Welche Entwurfswerkzeuge von welchem Hersteller setze ich wann und in welchem Kontext ein? Wie sind die Schnittstellen definiert? Wie sind die Ein- und Ausgabedaten zu interpretieren?“ An diese W-Fragen knüpft sich weiter die Forderung nach der konsistenten Verwaltung der Mega-Daten von Entwurf und Test.

Was fehlt ist eine pragmatische (gr.-lat. *pragma* = Handlung), in einer standardisierten

Umgebung eingebettete Entwurfsmethodik, die seit der *Introduction to VLSI Systems* von Carver MEAD und Lynn CONWAY [32] vor einem Jahrzehnt erst in jüngster Zeit in neuen Ansätzen verfolgt wird. So hat sich zum Beispiel das Firmenkonsortium CFI zum Ziel gesetzt, bislang isolierte Werkzeuge herstellerübergreifend mit Hilfe eines prozeduralen Schnittstellenmodells zu integrieren [23]; so stehen die massiven Modellierungsaktivitäten in der EDIF-Gemeinde unter dem Anspruch, den Informationsaustausch zwischen EDA-Werkzeugen und Entwurfsstätten zu standardisieren [16]; und so zeichnet sich ein neuer Trend zum *design consulting* mittels wissensbasierter Systeme zur Modellierung von Fachkompetenz ab [35].

2.2 VLSI/CAD-Infrastrukturen

Integrationssysteme, neudeutsch *Frameworks* oder gehoben *(Software-)Infrastrukturen*, machen nur dann einen wirtschaftlichen Sinn, wenn die Dienstleistungen und Schnittstellen Standards genügen, die von der Industrie aktiv unterstützt werden. Zwar haben die Schnittstellenprobleme einen neuen Markt für Softwarehäuser geschaffen, die dezidierte Einzellösungen offerieren, für den EDA-Markt insgesamt aber zeichnet sich ein Sättigungstrend ab: EDA-Anwender investieren zurückhaltender in neue Produkte, die mit eigenen *In-house*-Systemen oder den Werkzeugen von Drittherstellern unverträglich sind oder aber hohe Anpassungskosten mit sich bringen. Die Unterstützung darf sich also nicht nur auf entsprechende Konverter für existierende Produkte beschränken, sondern muß bereits bei der Entwicklung neuer Werkzeuge auf der Grundlage allgemein akzeptierter Industrie-Standards erfolgen.

Die internationalen Bemühungen um langlebige Industrienormen im EDA-Bereich sollen dabei keineswegs die Vielfalt der Produkte schmälern. Im Gegenteil: Die Stagnation des EDA-Marktes und im Gefolge die der ASIC-Industrie soll überwunden werden, indem das Spektrum an alten und neuen Werkzeugen verschiedener Hersteller breitbandig nutzbar gemacht wird. Das Schlüsselwort heißt: *Kompatibilität*. Die Entwicklung des EDA-Marktes ist geprägt durch die enge Kopplung zwischen dem Integrationsgrad der Entwurfswerkzeuge und dem Grad der Standardisierung ihrer Schnittstellen. Der Generationswechsel in der Werkzeugkonfiguration für den industriellen Hardware-Entwurf belegt diesen Trend (Tabelle 2.1).

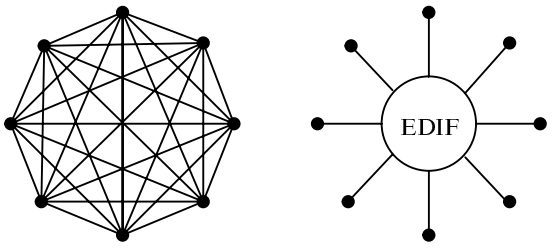

Bild 2.1: Reduktion der Schnittstellenkomplexität von $O(n^2)$ auf $O(n)$

Bild 2.1 verdeutlicht die Schnittstellenproblematik in heterogenen CAD/CAE-Entwurfsum-

Trend	*1. Generation* *70er Jahre*	*2. Generation* *80er Jahre*	*3. Generation* *90er Jahre*
Standardisierung	gering: firmeneigene Datenformate; erste singuläre, nur für bestimmte Werkzeugarten definierte Standards (CIF)	wachsend: hersteller-neutrale Datenformate (EDIF); Einführung der Schnittstelle zwischen ASIC-Entwickler und -Hersteller: *simulierte Netzliste*	dominierend: VLSI/CAD-Schnittstellen für den Datenaustausch und -zugriff weitestgehend standardisiert (EDIF, VHDL, CFI)
Werkzeugintegration	isolierte, nichtkompatible Werkzeuge (Insellösungen, *In-house*-Systeme)	geschlossene, herstellerspezifische *Turn-key*-Systemlösungen	*Frameworks*: offene Infrastrukturen für heterogene Werkzeuge
treibende Kräfte	Werkzeugentwickler vorwiegend aus universitären Forschungseinrichtungen	ASIC-Technologien, CAD/CAE-Anwender	EDA-Industrie und -Anwender, Normungsgremien, Vielfalt an Hardware-Plattformen und Betriebssystemen

Tabelle 2.1: Generationswechsel der Hardware-Entwurfsumgebungen

gebungen am Beispiel des elektronischen Datenaustauschformats EDIF [67]. Die Daten- und Steuerflüsse zwischen den einzelnen Entwurfswerkzeugen bzw. Entwicklungsstätten (dargestellt als n Knoten) sind im allgemeinen bi-direktional. Sie wirken in der Regel wechselseitig auf die Datenhaltungen aller am Entwurfsprozeß beteiligten Komponenten ein. Die gegenwärtig noch dominierenden EDA-Insellösungen lassen folglich den Aufwand für die Kommunikation (Datenkonverter und Tool-Interfaces) quadratisch mit der Vielfalt der Werkzeuge anwachsen: Situation (a). Allein standardisierte Schnittstellen können die Aufwandskosten für kompatible Umgebungen drastisch reduzieren: Situation (b).

2.2.1 Schnittstellen

Mit ihrer massiven Unterstützung für normierte CAD-Infrastrukturen verfolgen die EDA-Anbieter und -Anwender eine von der Phono/Video-Industrie inspirierte *Plug-in-and-play*-Philosophie, die sich ihrem Kompatibilitätsanspruch gemäß folgendes zum Ziel gesetzt hat:

- Austausch und Wiederverwendung gemeinsamer Entwurfsdaten
- Einheitliche Benutzerführung und Methodenkontrolle
- Rechnerunterstütztes Prozeß- und Ressourcen-Management

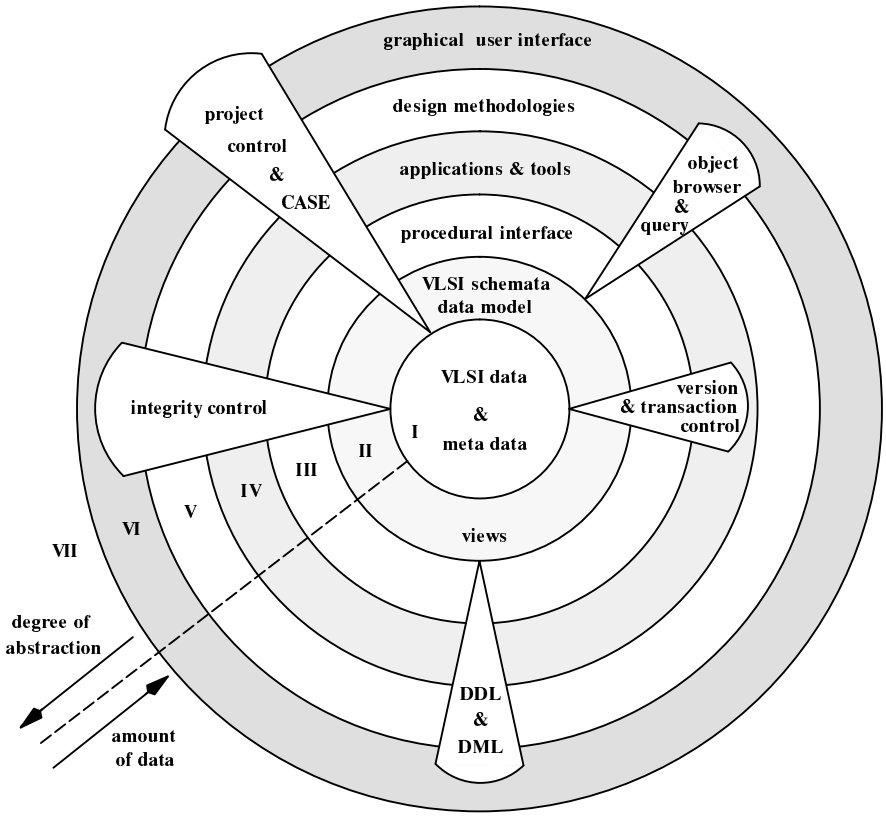

Bild 2.2: Schalenmodell einer VLSI/CAD-Infrastruktur

Bild 2.2 zeigt das Schalenmodell einer idealisierten VLSI/CAD-Infrastruktur. Die Darstellung als isolierende Schichten entspricht der Intention, die Entwurfskomponenten (Benutzerführung, Methoden, Werkzeuge und Daten) klar voneinander zu trennen und ausschließlich kontrollierte Interaktionen über die Schnittstellen zuzulassen.

Zwei gegenläufige Achsen durchziehen die Schnittstellen: Mit zunehmender Abstrahierung nimmt der Detaillierungsgrad der betrachteten Entwurfsdaten ab, verringert sich also das Datenaufkommen. Umgekehrt geht die Konkretisierung der Entwurfsdaten mit einem größeren Aufwand für die Datenhaltung einher. Analog verhält es sich mit der Entwurfskomplexität: Je weiter „außen“ der Entwickler den Entwurfsprozeß steuert und seine Entscheidungen trifft, desto geringer ist seine kognitive Belastung durch die Details. Tabelle 2.2 beschreibt die Schnittstellen im einzelnen.

Schnittstellen sollen „wohl definierte“ und normative Übergänge schaffen und zugleich für übergreifende Interaktionen durchlässig sein. Wir können folgende Dienstleistungen durch mehrere Schnittstellen hindurch unterscheiden:

- Integritätskontrolle: I–VI

Übergänge	*Beschreibung*
I–II	CAD-Rohdaten und Meta-Daten (Daten über Rohdaten) werden vom Datenbanksystem gemäß dem Datenbankschema organisiert und zentral verwaltet. Direkte Zugriffsmöglichkeiten bestehen nicht.
II–III	Datenbankschemata und -teilschemata (*views*), erstellt auf der Grundlage eines objektorientierten Datenmodells, strukturieren die Objekt- und Beziehungstypen des abzubildenden Weltausschnitts.
III–IV	Die prozedurale Werkzeugschnittstelle fungiert als Bindeglied zwischen der statischen Datenhaltung und den dynamischen Einwirkungen der Entwurfswerkzeuge auf den Datenbestand.
IV–V	Das Tool-Management verwaltet die Entwurfswerkzeuge gemäß der gewählten Entwurfsmethodik: Auswahl der Werkzeuge, Generieren von Steuerdateien und Aufrufsequenzen etc.
V–VI	Das Design-Management überwacht den Entwurfsfluß, d. h. die zeitliche Abfolge synthetisierender und verifizierender Prozesse, und koordiniert den Input/Output-Datenfluß an den Schnittstellen der einzelnen Entwurfsabschnitte. Entwurfsentscheidungen können mit Hilfe von Expertensystemen unterstützt werden.
VI–VII	Die grafische Benutzeroberfläche schließlich überbrückt die interaktive Schnittstelle Mensch-Maschine, die nach software-ergonomischen Gesichtspunkten gestaltet ist.

Tabelle 2.2: Schnittstellen in CAD-Infrastrukturen

Die Überwachung der Datenintegrität erstreckt sich über alle internen Schnittstellen einer CAD-Infrastruktur und erfaßt somit die statischen, dynamischen und methodischen Entwurfsaspekte. Eine ausgewogene Kombination von zentralen und verteilten Kontrollmechanismen sorgt für die Einhaltung der Integritätsbedingungen. Ansätze für einen Integritätsmonitor finden sich zum Beispiel in [30].

▷ Projektkontrolle: I–VII

Dem Systementwerfer steht die Möglichkeit offen, jederzeit Status- und Prozeßinformation abzufragen und auszuwerten. Das Tool- und Design-Management vergleicht die aktuellen Konstruktionsdaten mit der Projektspezifikation und schlägt korrigierende Entwurfsentscheidungen vor (Expertensysteme [35]) oder nimmt diese vom Entwerfer entgegen. Zur Verwaltung aller logistischer Daten stehen entsprechende CASE-Werkzeuge zur Verfügung.

▷ Versions- und Transaktionsverwaltung: I–IV

Die für den VLSI-Entwurf charakteristischen Anforderungen an ein Versions- und Transaktionskonzept [27][28] — Protokollierung der zeitlichen und alternativen Entwurfsschritte und Teilergebnisse, Synchronisierung konkurrierender Datenzugriffe — resultieren aus der arbeitsteiligen Durchführung eines Projekts im Team. Versions- und Trans-

aktionsmanagement erfolgen unterhalb der grafischen Benutzeroberfläche und bleiben somit für die Benutzer verborgen. Auswahl und Zusammenstellung der für den Gesamtentwurf relevanten Versionen — die Konfiguration der Einzelentwürfe — stehen unter der Projektkontrolle.

- ▷ Datenbeschreibung und -manipulation: II–VI

  Für die Definition der Datenbankschemata und der Zugriffs- und Verwaltungsoperationen verfügt der Datenbank-Administrator über die von konventionellen Datenbanken bekannten Sprachmittel: DDL und DML (*Data Definition/Manipulation Language*). In CAD-Infrastrukturen aber orientieren sich diese Sprachmittel nicht ausschließlich an einfachen Datenstrukturen wie Records und Tabellen, sondern sind auf die semantisch mächtigeren Organisationsprinzipien des Objekt-Paradigmas ausgerichtet: Klassen, Objekte, Beziehungen und Constraints (siehe auch Anforderungen an *Non-Standard*-Datenbanken in [21]).

- ▷ Objekt/Schema-Browser und Datenabfrage: II–VI

  Zur Visualisierung des aktuellen Datenbestands (Objektebene) und der den Daten zugrundeliegenden Organisationsstrukturen (Schema-Ebene) werden von der grafischen Schnittstelle aus Objekt- und Schema-Browser eingesetzt. Weiterhin stehen die aus der Datenbanktechnologie bekannten prädikatenlogischen Abfragetechniken zur Verfügung, um beispielsweise datensensitive Entwurfsentscheidungen abzusetzen oder Objektmengen nach gemeinsamen Eigenschaften zu qualifizieren.

Derzeit gibt es eine Vielzahl verschiedener VLSI-Frameworks. Sie weichen in einigen wichtigen Aspekten von der Idealvorstellung des Schalenmodells ab und stellen oft nur herstellerspezifische Integrationsrahmen für die eigene Produktlinie dar. Um aber innovative Entwurfswerkzeuge in einen bestehenden Werkzeugverbund aufnehmen zu können (*technology push*) oder mit einem neuen Produkt schnell am Markt zu sein (*market pull*), bedarf es herstellerneutraler Dienstleistungs-Standards.

2.2.2 Projektlandschaft

Daß Frameworks nicht nur ein marketing-wirksames Schlagwort[1] sind, sondern auch einen zentralen Gegenstand der Praktischen und Technischen Informatik darstellen, zeigt die aktuelle Forschungslandschaft. Schätzungen des amerikanischen Instituts für Verteidigungsanalysen (IDA) zufolge, verringern einheitliche Entwurfsumgebungen die Produkt-Entwicklungszeiten um 40 bis 60 Prozent. Das Bostoner Marktforschungsinstitut Technology Research Group (TRG) taxiert den Produktivitätsgewinn auf 30 Prozent [2]. Die aktuelle internationale CAD-Projektlandschaft wurde in erster Linie durch das Wissen um den ökonomischen Faktor „Framework“ motiviert.

[1] Steven SCHULZ in *Electronic Design* [43]: „One of the most fascinating aspects of the CAE industry is how quickly a new buzzword finds its way into vendor's glossy literature. The pamphlets unabashedly claim years of market leadership long before the rest of us have settled on a workable definition for the term.“

Die wichtigsten Framework-Projekte und -Standardisierungsbemühungen lassen sich zwanglos den einzelnen Schnittstellen im Schalenmodell zuordnen:

- *CFI*: II–VII

  Die CAD-Framework-Initiative (CFI) ist ein internationales Industriekonsortium aus EDA-Anbietern und -Anwendern mit dem Ziel,

  > „ ... to develop industry acceptable guidelines for design automation frameworks which will enable the coexistence and cooperation of a variety of tools." [40]

  Das CFI-Konsortium unterhält eine Organisationsstruktur (Kooperationen zwischen den Entwicklungslabors namhafter EDA-Firmen), um neue Standards für CAD-Schnittstellen zu entwickeln oder bestehende zu bewerten. Anschließend sollen die CFI-Richtlinien als Industrienormen vorgeschlagen werden.

- *DoD/EIS*: II–VII

  Das EIS-Programm [36] (Engineering Information Systems) adressiert alle Schnittstellen einer CAD-Infrastruktur. Auch im Rahmen dieses groß angelegten Unterfangens sollen neue Industriestandards entwickelt und/oder existierende evaluiert werden. Um die Brauchbarkeit und Effizienz der vorgeschlagenen Standards und Richtlinien zu demonstrieren, wurden auch hier Industriekooperationen vereinbart: In den Entwicklungszentren wichtiger CAE-Firmen soll parallel und unabhängig voneinander in unterschiedlichen CAD-Entwurfsumgebungen auf der Grundlage der EIS-Standards ein VHSIC-System entwickelt werden. Am Ende der (öffentlichen) Evaluierungsphase werden dann die Ergebnisse in die Normungsgremien der Industrie eingebracht.

- *IEEE/DASS/WGDM*: II–VII

  Die Arbeitsgruppe Design Management (WGDM) des IEEE-Gremiums für Design Automation Standards (DASS) wurde gegründet, um Empfehlungen für den Einsatz von Standards in CAD-Infrastrukturen auszusprechen. Als Grundlage für die IEEE-Richtlinien sollen die Vorschläge und Erkenntnisse aus CFI und EIS dienen.

- *ISO/STEP*: II–III

  Die ISO hat sich das ehrgeizige Ziel gesetzt, existierende nationale und internationale Industriestandards für mechanische, elektrische und elektronische Produktdaten durch einen einzigen Standard zu ersetzen: STEP — Standard for the Exchange of Product Model Data. Das bedeutet, STEP wird eine Vielzahl von Ingenieurdisziplinen umfassen: vom Entwurf integrierter Schaltungen bis zur Konstruktion von Ozeanriesen. Nicht nur die bloßen Fertigungsdaten, sondern auch sämtliche während des Produktlebenszyklus (Entwicklung, Projektierung, Wartung) anfallenden logistischen Daten sollen durch STEP konzeptionell abgedeckt werden und maschinell austauschbar sein.

  Auch wenn der Bogen (zu) weit gespannt sein sollte, hat bereits ein Nebenprodukt der STEP-Aktivitäten in vielen Normungsgremien Einzug gehalten: EXPRESS, eine formale Beschreibungssprache — sowohl textuell als auch grafisch — zur Informationsmodellierung [41]. Mit Hilfe dieses universellen Hilfsmittels läßt sich der statische Informationsgehalt beliebiger Diskursbereiche in Form von Referenzmodellen einheitlich

darstellen und auf Konsistenzaspekte automatisch überprüfen. EXPRESS wird in den Arbeitsgremien von STEP, CFI, IEEE/VHDL, EDIF, EIS und im Forschungsprojekt DASSY eingesetzt.

- *EIA/EDIF*: III–IV

  Die EDIF-Gremien der Electronic Industries Association (EIA) entwickeln und pflegen einen gemeinsamen Standard für den Datenaustausch zwischen EDA-Applikationen [67]. Die derzeitige Version [17] stellt den De-facto-Industriestandard für die Entwurfsbereiche Netzlisten und Schematic-Capture dar. Gegenwärtige Entwicklungsaktivitäten konzentrieren sich auf die Anforderungen aus dem Bereich der gedruckten Leiterplatten (PCB) und der Datenkopplung zwischen Entwurf und Test. Darüber hinaus nehmen die Arbeiten zu einem grundlegenden EDIF-Informationsmodell einen breiten Raum ein [18].

- *ESPRIT/ECIP*: III–IV

  Das European CAD Integration Project (ECIP) ist Teil der ESPRIT-Initiative [9]. Ziel ist die Entwicklung eines formalen konzeptionellen Referenzmodells für alle Arten elektronischer Produkte. Klassifikationsschemata und zugehörige Verzeichnisse der verwendeten Begriffe sollen anhand von effizienten *Benchmarks* auf Vollständigkeit und Widerspruchsfreiheit maschinell überprüfbar sein.

- *ESPRIT/EVEREST*: III–IV

  EVEREST (European Vanguard Effort on Research and Engineering of Systems for Testing) ist ebenfalls Teil der ESPRIT-Initiative [42]. Wie der Titel schon ankündigt, zielt EVEREST speziell auf den Test als eine Sicht des Entwurfsraums. In diesem Projekt wird die Spezifikation eines Modells zur Repräsentation der Testinformation gefördert, wobei das Testmodell als Grundlage für die Erweiterung von EDIF dient [76].

Auf dem Weg zu industrieweit anerkannten CAD-Standards leisten die oben skizzierten Framework-Projekte (einschließlich DASSY) und Normungsgremien die entscheidende Arbeit und bieten zugleich die Foren für EDA-Produktanbieter und -anwender, um die Entwicklung und Etablierung der Framework-Technologie aktiv voranzutreiben.

2.3 VLSI-Weltmodelle

Eine effiziente Kommunikation zwischen heterogenen Werkzeugen erreicht man nur durch Integration und nicht allein durch Import/Export-Dienste auf Dateibasis. Um aber Entwurfswerkzeuge integrieren zu können, bedarf es vor allem einer *schematisierten Modellvorstellung* über den Informationspool, der während des Entwurfsprozesses auf allen Darstellungsebenen angereichert wird und der das Entwurfsobjekt mit seiner Entstehungsgeschichte vollständig zum aktuellen Zeitpunkt beschreibt. Da Werkzeuge „werkeln“ sollen, d. h. den Informationspool verändern und erweitern, müssen neben den statischen Aspekten vordringlich auch die dynamischen Zugriffsmöglichkeiten modelliert werden.

Die Generalanforderungen an die Modellierung eines derartigen Informationspools nennen wir „Weltmodell“ in Anlehnung an den Begriff des „Datenmodells“ in der Datenbanktechnologie. Wie Datenmodelle (zum Beispiel das relationale) stellen Weltmodelle die konzeptionellen Werkzeuge zur Modellierung eines interessierenden Weltausschnitts zur Verfügung. Im Mittelpunkt der datenintensiven Framework-Technologie stehen also zwei Facetten der Informationsverarbeitung:

1. Modellieren und Schematisieren der Entwurfsinformation: Datenrepräsentation
2. Konsistentes Fortschreiben und Abfragen des Datenbestands: Datenverwaltung

Um Systeme zu modellieren, setzen wir Mittel der Abstraktion ein, — Konzepte und Formalismen, die die Entwurfskomplexität reduzieren und beherrschbar halten [69]. Die Framework-Technologie steht hierbei in der Tradition der Soft- und Hardware-Abstraktionen [44][45]. Als interdisziplinäre Technologie — nicht zuletzt treten der Faktor *Mensch* und seine kognitiven Grenzen in den Vordergrund — macht sie sich das Wissen und die Erfahrungen aus der Methodenentwicklung in der Informatik zunutze [11]. Die Abstraktionen stehen für die Eckpfeiler einer fortschreitenden Disziplinierung des Entwurfsprozesses. Ob *System on Chip* oder *Programming in the Large*, die Parallelen sind unverkennbar (Tabelle 2.3).

Abstraktion	*Hardware*	*Software*
Modularisierung	Bausteinprinzip: Bibliothekszellen, Steuer- und Operationswerke, Speichermodule, Boards etc.	Blockstrukturen: Funktionen, Prozeduren, abstrakte Datentypen, Klassen, Objekte
Hierarchische Strukturierung	Abstraktionsebenen (*system, algorithmic, micro-architectural, logic, circuit*), Sichten (*behavioral, structural, physical*) [22]	*stepwise refinement* [37][50], Haupt- und Unterprogramme, Klassenhierarchien und Vererbung [34]
Formalisierung	Hardware-Beschreibungssprachen, Datentransferformate	Algebraische Spezifikation, automatische Programmierung

Tabelle 2.3: Disziplinierung des Entwurfsprozesses

Wir entnehmen dieser Methodenentwicklung eine übergreifende Gesamtausrichtung, eine Grundströmung, die sich durch mehrere Informatik-Disziplinen zieht — Künstliche Intelligenz (KI), Datenbankforschung und Programmiermethodik:

Konzeptionelles Modellieren

Vor der Implementierung steht die Spezifikation und vor der Spezifikation das „Begreifen“ eines Problems, das kognitive Erfassen des interessierenden Weltausschnitts. Bis dato erfolgen

alle drei Phasen getrennt — *konzeptionell* getrennt. Die Beschreibungsmittel der Problemanalyse, der Spezifikation und der Implementierung sind in der Regel nicht kongruent. Die „semantische Kluft“ zwischen ihnen ist sprichwörtlich. Konzeptionelles Modellieren ist der Versuch, alle Phasen des Systementwurfs begrifflich durchgängig zu unterstützen.

Ohne es deutlich herauszustellen, betreiben die verschiedenen Informatikforschungen eine vergleichbare Methodik der Komplexitätsbewältigung, nämlich die Modellbildung auf konzeptioneller Ebene. Das augenfällig Gemeinsame ist die Suche nach einer genügend präzisen Notation für das Inventar eines Weltausschnitts:

- Wissensrepräsentation in der KI [5]
- Semantische Datenmodellierung in der Datenbankforschung [15]
- Formale Spezifikation in der Programmiermethodik [20][47]
- Informationsmodellierung in den CAD-Standardisierungsgremien [9][18][36]

Auch wenn es sicherlich Unterschiede in der Zielsetzung gibt, so sind doch die Methoden vergleichbar. Die interdisziplinären Konferenzen und Veröffentlichungen in der Informatik zeugen davon: [7][33][39][47].

Frederick BROOKS (vielzitierter Autor des metaphorischen Aufsatzes *No Silver Bullet: Essence and Accidents of Software Engineering* [8]) zählt die konzeptionelle Modellierung zu den hoffnungsvollen Kandidatinnen für eine *silver bullet* gegen ausufernde, monströse Softwareprojekte:

> „Fashioning complex conceptual constructs is the *essence*; *accidental* tasks arise in representing the constructs in language. Past progress has so reduced the accidental tasks that future progress now depends upon addressing the essence.“ [8]

David HAREL relativiert den BROOKSschen Pessimismus, indem er dazu aufruft, die verschmähten *accidential tasks* in der Systemmodellierung — die Repräsentation — nicht gering zu schätzen. Der Trend gehe zu „visualisierten“ und „ausführbaren“ Systemmodellen, die das konzeptionelle Konstrukt sukzessiv und teilweise automatisiert in implementierbare Modelle (Programme und Datenbankschemata) überführen [25].

2.3.1 VLSI-Spezifika

Wie sieht nun das *conceptual construct* des VLSI-Entwurfs aus? Welche Anforderungen von seiten der Entwurfswerkzeuge und der Datenhaltung werden gestellt? Von der Grundlage der Erkenntnisse aus [3][6][21][51] läßt sich der folgende Anforderungskatalog zusammenstellen, der nicht notwendigerweise auf VLSI-Applikationen beschränkt ist, sondern auch für andere *Non-Standard*-Datenbankanwendungen, wie Maschinenbau und Geo-Datenbanken, gilt:

Generalisierung/Spezialisierung

Die Welt ist vernetzt, auch die technische. Technische Objekte hierarchisch ordnen zu können, setzt einen Klassifikationsmechanismus voraus. Als mächtiges intellektuelles Modellierungsmittel dient hier die Abstraktion, die auf „crisply-defined conceptual boundaries relative to the perspective of the viewer" führt [4]. Global schlägt sich die Generalisierung/Spezialisierung in der Schichtung der VLSI-Abstraktionsebenen nieder: *system*, *algorithmic*, *micro-architectural*, *logic* und *circuit levels* nach GAJSKI [22]; lokal in der hierarchischen Begriffsbildung auf Klassenebene (in EDIF beispielsweise im geometrischen Konstrukt *figure* als Oberbegriff von *circle*, *rectangle*, *polygon* u.a.).

Wesentlich für Klassenhierarchien ist deren semantischer Zusammenhalt, d. h. die *kind-of*-Beziehungen der Klassen untereinander. Sie implizieren die Möglichkeit der Vererbung/Ableitung von Klasseneigenschaften, was methodisch als Strukturierungsmittel zur Komplexitätsbewältigung und implementierungstechnisch als *code sharing* in Programmen und Datenbanken genutzt wird. Oberklassen repräsentieren generalisierte Abstraktionen. Unterklassen stellen Spezialisierungen ihrer Oberklassen dar: Verhaltens- und Strukturmuster der Oberklassen werden — eventuell modifiziert — übernommen und eigene hinzugefügt. Generalisierung/Spezialisierung gestattet somit ein Höchstmaß an Ausdrucksökonomie.

Objektausprägung

Der Übergang von der systemanalytischen Phase — der Klassenfindung und -bildung — zum Entwurf und zur Implementierung erfolgt über die *Instanziierung*[2] der Klassen, d. h. über die Objektausprägung oder Exemplarbildung. Dieser Teilungsmechanismus (*instance-of*-Beziehungen zwischen Objekt- und Klassenwelt) unterstützt die Software-Wiederverwendung entscheidend und ist für zellenbasierte Entwurfstechniken (Standardzellen- und Gate-Array-Entwurf) charakteristisch. Exemplare besitzen eine eindeutige Identität und können beliebig oft durch Referenzieren der zugehörigen Objektklassen für eine Applikation verfügbar gehalten werden — bei *generischen* Klassen auch unter Angabe von Parametern. Beispiel: Logik-Gatter und Register-Transfer-Bausteine mit wahlfreier Anzahl der Eingänge oder *Fan-out*-Treiberfähigkeit.

Aggregation

Innerhalb der Objektwelt bedarf es eines Konstruktionsmechanismus, der den Aufbau komplexer Objekt-Verbände durch *part-of*-Beziehungen ermöglicht: zusammengesetzte und geschachtelte Objekte gleicher oder unterschiedlicher Klassenzugehörigkeit. So aggregiert eine *Netzliste* funktionale Schaltungskomponenten (elektrische Anschlüsse) zu einem Objekt-Verband, der als Ganzes verwaltet wird: vernetzte Objektreferenzen. Mit der Art der Aggregation eng verbunden ist die auf ihr definierte operative Semantik, d. h. zum Beispiel die

[2] Aus dem Englischen *instantiating* von vielen Autoren auf diese Weise eingedeutscht.

Frage, ob aggregierte Objekte (Unterobjekte) beim Löschen des Aggregats (Oberobjekts) mit gelöscht werden sollen. Im DASSY-Datenmodell wurden hierfür VLSI-spezifische Beziehungstypen vordefiniert, die vom Anwender mit eigener operativer Semantik angereichert werden können [53].

Modellierbarkeit passiver und aktiver Eigenschaften

Struktur und Verhalten der Objekte müssen gleichermaßen und gemeinsam beschreibbar sein, um die statisch-dynamischen Aspekte innerhalb der Objektmengen — die Aktionen und Reaktionen auf den Objektstrukturen — zu erfassen. Diese Forderung geht über die Leistungen *semantischer Datenmodelle* hinaus, beispielsweise des Entity-Relationship-Modells von Peter CHEN [10] und seiner zahlreichen Erweiterungen, die sich im allgemeinen auf die statische Strukturbeschreibung beschränken.

Abstrakte Datentypen

Diese Anforderung geht zurück auf die Beschränkung konventioneller Datenhaltungssysteme auf eine Handvoll Standard-Datentypen, wie Ganz-, Festkommazahlen und Zeichenketten. Abstrakte Datentypen erhöhen die Integrität des Datenbestands: Operationen und Daten sind an bestimmte Datentypen gebunden. Implementierungsabhängige Datenstrukturen und der zugehörige Operationscode bleiben dem Anwender verborgen: Prinzip des *information hidings* [37]. Beispiel für einen abstrakten Datentyp: *Punktlisten* zur Polygon-Beschreibung, wie sie von Layout-Editoren im allgemeinen verwendet werden, mit Operationen, die die Selbstüberschneidungsfreiheit des Polygons überprüfen und Koordinaten-Transformationen erlauben [6].

Sichten/Beschreibungsdomänen

Entwurfswerkzeuge arbeiten nur auf bestimmten Repräsentationen eines Entwurfsobjekts: Schaltkreis-Extraktoren benötigen eine andere Objektbeschreibung (Maskengeometrien und Entwurfsregeln) als Digital-Simulatoren (Netzlisten und Verhaltensbeschreibungen). GAJSKI [22] versteht hierunter die Verhaltens-, Struktur- und physikalische Domäne. EDIF unterscheidet zehn verschiedene *viewTypes*, die hauptsächlich die Entwurfsbereiche Konnektivität, Geometrie und Dokumentation — teils überlappend — abdecken [67].

Objekt-Versionierung

Das Archivieren der Entwurfsentscheidungen und der daraus resultierenden Entwurfsobjekte erfordert einen Mechanismus, der die zeitlichen und alternativen Abfolgen in der Objektentwicklung verwaltet: Versionen und Alternativen [27]. Versionierung ist von maßgeblicher

Bedeutung bei der Koordinierung eines Team-Projekts. Damit eng verknüpft sind Entwurfs-Transaktionen, ein Erfordernis der Datenhaltung zur Konsistenzsicherung bei konkurrierendem Datenzugriff und Systemfehlern.

Interaktionsmechanismen

Konventionelle Techniken des Informationsaustauschs, wie der *Event/Trigger*-Mechanismus [29] in Datenbanken, setzen Kommunikationsmittel und -pfade zwischen den Entwurfsobjekten voraus. Eine wesentliche Intention des Informationsaustauschs liegt in der Sicherung der Datenintegrität. Die Korrektheit der Daten muß unter den jeweiligen Aspekten von Sichten, Versionen, Transaktionen, Aggregation und Typisierung gewährleistet sein.

Es ist kein Zufall, daß die Anforderungen an die Modellierbarkeit der CAD-Entwurfsobjekte inhaltlich von den zentralen Konzepten der Objektorientierung [4][34] erfüllt werden. Zum einen resultieren diese Anforderungen — analog zur Evolution des Objekt-Paradigmas — aus den systemtheoretischen Erkenntnissen der Komplexitätsbewältigung. Zum andern unterstützt der interdisziplinäre Charakter des Objekt-Paradigmas unseren intuitiven Zugang zu komplexen Sachverhalten und vereint zugleich traditionelle und innovative Techniken des ingenieurmäßigen Systementwurfs. Eine Gegenüberstellung zwischen VLSI/CAD-Spezifika und Objekt-Modell wäre folglich nahezu deckungsgleich.

2.3.2 Informationsmodellierung

Um das Weltmodell einer Applikation aufzustellen, müssen wir zunächst die statischen Strukturen erfassen — eine Momentaufnahme des Informationsgehalts untersuchen. Zum statischen Anteil eines Weltmodells gelangen wir über die in den letzten Jahren aufgekommene formale Informationsmodellierung.

Ursprünglich als formale Sprache für die Definition eines allgemeinen Produktmodells entwickelt, avancierte EXPRESS innerhalb weniger Jahre zum Standard-Medium der Informationsmodellierung in CFI und anderen Framework-Initiativen. EXPRESS knüpft unmittelbar an die Methoden und Techniken der semantischen Datenmodellierung an, wie sie beispielsweise in den erweiterten Entity-Relationship-Modellen [10] oder in der Modellierungssprache IDEF1x [26] entwickelt wurden. Im Vordergrund steht dabei die formale Darstellung von CAD-Daten zusammen mit deren genauen Bedeutung und Bezug untereinander.

Was ist nun EXPRESS? Zunächst die Negativ-Definition: EXPRESS ist keine Programmiersprache, auch wenn sie ausführbare Konstrukte enthält. Sie ist keine Datenbank-Abfragesprache, auch wenn sie SQL-ähnliche Elemente besitzt. Sie ist auch nicht eine komplexe Datenstruktur im Sinne von Records, Feldern oder Tabellen. Nun die Positiv-Definition: EXPRESS ist eine normative Modellierungssprache, textuell und grafisch. Sie formalisiert (a) die Beschreibung der Objekte eines Diskursbereichs (*universe of discourse*) und deren Beziehungen untereinander und (b) die Beschränkungen (*constraints*), denen die strukturellen

Eigenschaften unterliegen.

Warum ist Informationsmodellierung[3] im Framework-Bereich angesagt? Die Gründe sind zwingend:

- Konsensfindung unter Experten verschiedener Entwurfsbereiche über den Bedeutungsinhalt der Entwurfsdaten: normative Entwurfssemantik
- Kommunikationsmittel, um ein Problem zu analysieren und eine Lösung zu spezifizieren
- Grundlage für die Implementierung als Datenaustauschformat und Datenbankschema: Referenzmodell
- konzeptioneller Unterbau von CAD-Frameworks und deren Schnittstellen

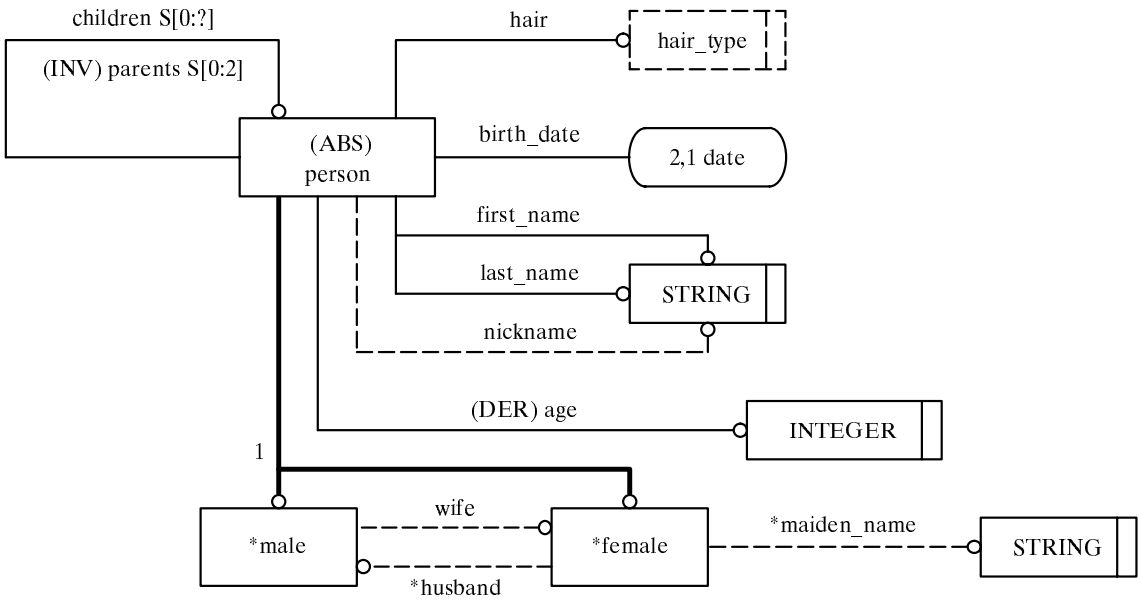

Bild 2.3: EXPRESS-G Beispiel

Das Inventar der VLSI-Weltmodelle verlangt eine diskrete Natur der Darstellungsmittel. Darüber hinaus erhöht eine grafische Form die Anschaulichkeit und unterstützt das Abstraktionsvermögen des Betrachters. Bild 2.3 gibt ein illustratives Beispiel für die Informationsmodellierung in EXPRESS-G, der grafischen Untermenge von EXPRESS [41].

[3]Informations- und konzeptionelle Modellierung sind nicht synonym, obwohl sie oft einander gleichgesetzt werden. Erstere weist eine rein strukturelle Zielrichtung auf (die statische Beschreibung eines Diskursbereichs), letztere eine strukturell-operative Zielrichtung (statische und dynamische Beschreibung).

Das im Bild 2.3 gezeigte Modell beschreibt das konzeptionelle Konstrukt `Person`. Eine `Person` besitzt gewisse Eigenschaften, wie Vor- und Zunamen, optional einen Rufnamen, Haarfarbe, Geburtsdatum und daraus abgeleitetes (DERived) Alter. Eine `Person` ist eine abstrakte (ABStract) Oberklasse in dem Sinne, daß es kein direktes Exemplar von ihr gibt. Erst die weitere Spezialisierung von `Person` führt auf „instanziierbare" Unterklassen, nämlich männliche und weibliche Personen. Zwischen letzteren können monogame Beziehungen bestehen, was dann zusätzliche Eigenschaften in der Namensführung der Eheleute mit sich bringt (die Modellierung des aktuellen deutschen Namenrechts dürfte sich aufwendiger gestalten). Eine `Person` kann natürlich Kinder haben, die wiederum Personen mit obigen Eigenschaften und Beziehungen sind. Der Rollenname der Eltern/Kind-Beziehung gibt Auskunft über die jeweilige Sichtweise.

Daß einige Eigenschaften bestimmten (nur in der textuellen Form von EXPRESS sichtbaren) Beschränkungen unterliegen, wird durch einen vorangestellten Stern (*) vermerkt. Einheiten (*entities*), die eine Menge von konzeptionellen oder realen Objekten mit gleichen Eigenschaften repräsentieren, werden in EXPRESS-G durch Rechtecke gekennzeichnet. Beziehungen sind durch Linien zwischen den Rechtecken erkennbar. Strichstärke und Strichart stehen dabei für Art und Stärke der Beziehung.

Will man die Informationsstrukturen in VLSI-Schnittstellen modellieren, so sind grundsätzlich zwei funktionale Komponenten zu unterscheiden:

1. Werkzeug-Schnittstellen
2. Datenaustausch-Schnittstellen

Erstere sind prozedural und im Zugriff on-line. Von der Konzeption her stellen sie abstrakte Datentypen dar: Das Werkzeug kennt nur die Operatoren zum Abfragen und Manipulieren des Datenbestands. Wie der Datenzugriff organisiert ist, wie beispielsweise die Datenstrukturen oder der Mechanismus, um die Objektidentität zu gewährleisten, realisiert sind, bleibt dem Werkzeug verborgen. Datenaustausch-Schnittstellen dagegen sind deskriptiv und arbeiten in der Regel off-line: Der Datentransfer erfolgt geschlossen in einer formatierten ASCII-Datei. EDIF und VHDL sind hier die Standardformate.

Für beide Aspekte einer CAD-Schnittstelle müssen geeignete Sprachkonzepte angeboten werden. EXPRESS mangelt es derzeit noch an Ausdrucksmitteln für Modell*verhalten*. Prozedurale Schnittstellen lassen sich (noch) nicht beschreiben. Von Seiten der DASSY-Projektpartner wurden bereits entsprechende Erweiterungsvorschläge in die EXPRESS-Normungsgremien getragen — mit guter Resonanz.

2.3.3 Datentransferformate

Die Notwendigkeit, die Syntax-Inflation einzudämmen — den Informationsgehalt der Entwurfsdaten zu modellieren bevor die Form der Daten festgelegt wird —, wurde in den internationalen Standardisierungsgremien erst in letzter Zeit erkannt. Im Bestreben, Form

und Inhalt, also Syntax und Semantik, nicht zu vermischen, nimmt das Austauschformat für elektronische Entwurfsdaten EDIF eine Vorreiterrolle ein. Um die von vielen Anwendern prognostizierte Syntax-Krise zu überwinden, hat das EDIF Technical Committee bereits 1989 entschieden, daß zukünftige Erweiterungen des Standards nur noch auf der Grundlage eines wohlfundierten Informationsmodells zugelassen werden. Als formales Beschreibungsmittel wurde hierfür EXPRESS ausgewählt.

Aber nicht nur der Informationsgehalt zukünftiger Erweiterungen, sondern auch die Semantik der aktuellen EDIF-Version soll im nachhinein modelliert werden. Die spärlichen und zum Teil widersprüchlichen Erläuterungen im EDIF-Referenzhandbuch bieten für das Informationsmodell nur Anhaltspunkte, aber keine konsistente Gesamtbeschreibung. Wie sehr der Interpretationsbedarf angewachsen ist, zeigt der Umfang des fünfbändigen Sammelwerks „EDIF Questions and Answers“ von 1988 bis 1991. In der Sache „EDIF-Version 2.0.0“ war folglich *Reverse Engineering* vonnöten. Hierzu wurde Ende 1989 eine „EDIF European TSC Information Modelling Working Group“ (EDIF IMG) ins Leben gerufen, die sich ausschließlich mit der Informationsmodellierung der EDIF-Domäne befaßt (Näheres im folgenden Unterkapitel). Die Domäne elektronischer Entwurfsdaten wurde ansatzweise auch in anderen Gremien und Projekten modelliert: *ECAD Domain Model* (EIS-Projekt [24]), *Conceptual Model* (ECIP [9]) und *Test Model* (EVEREST [76]).

Im wesentlichen dreht es sich bei elektronischen Austauschformaten um die *Repräsentation* von (a) Grafik, wie Symbolen, Stromlaufplänen und Maskenlayouts, (b) logischer und struktureller Konnektivität[4] und (c) Signalmustern, wie Simulations- und Testdaten.

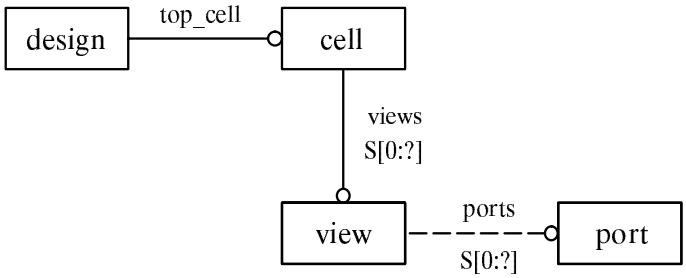

Bild 2.4: Ausschnitt aus dem EDIF-Informationsmodell: Zellen und Sichten

Die EXPRESS-G-Modelle in den Bildern 2.4 und 2.5 zeigen ausschnittsweise einige Arbeitsergebnisse der EDIF-IMG. Bild 2.4 gibt die klassische Sichtenbetrachtung einer Zelle wieder und steht für die Über-Alles-Modellierung der EDIF-Welt. Bild 2.5 gibt ein Beispiel für das *Reverse Engineering* syntaktischer Konstrukte. Hier wurde versucht, das syntaktische Dickicht der grafischen EDIF-Primitive zu lichten und eine übersichtliche Darstellung zu erreichen. Die textuelle und grafische Gesamtdokumentation der EDIF-Informationsmodelle

[4] Logische Konnektivität meint Signale, definiert als die Menge aller elektrisch verbundener Anschlüsse; strukturelle Konnektivität meint Netze, verstanden als die räumlich-strukturelle Implementierung eines Signals.

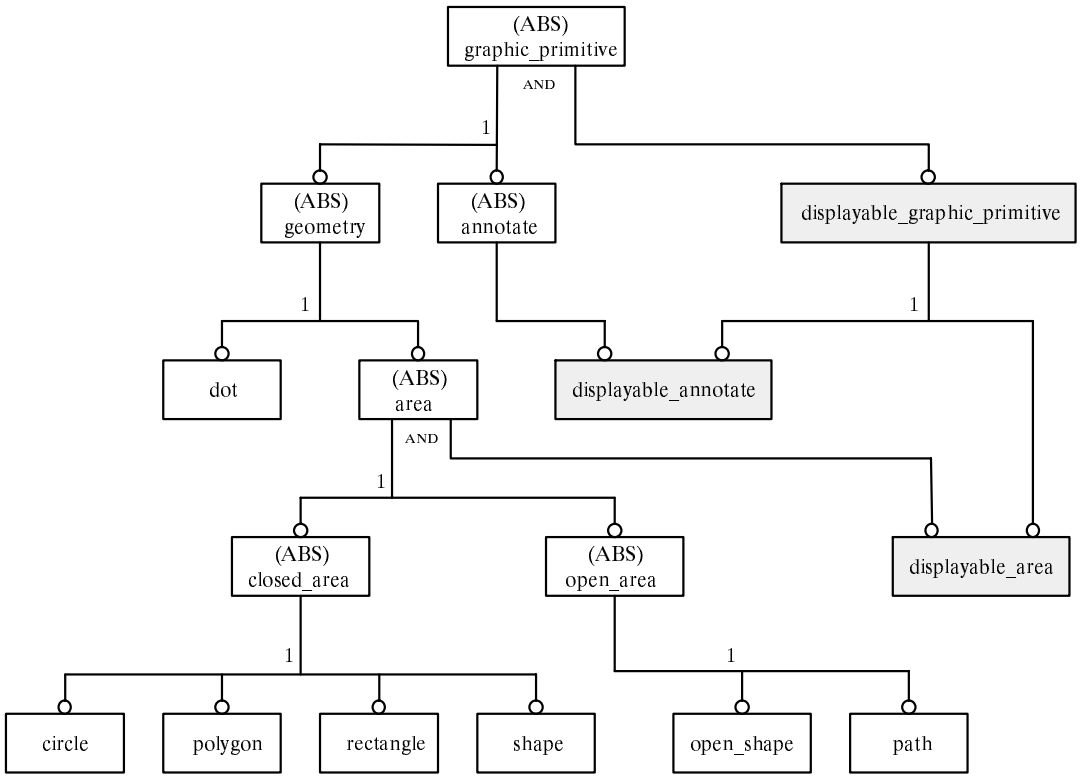

Bild 2.5: Typhierarchie der grafischen Grundelemente in EDIF

kann [18] entnommen werden.

Zusammengefaßt können wir die Motivation für die Informationsmodellierung von EDA-Austauschformaten wie folgt skizzieren:

- Sie führt auf ein syntax-unabhängiges Modell der durch das Format bezeichneten CAD-Objekte,
- formalisiert die Zusammenhänge und Restriktionen in der Objektwelt,
- spielt beispielsweise die Rolle eines Referenzmodells, das den inhaltlichen Vergleich zweier formatierter Datenbeschreibungen erlaubt (trotz der syntaktischen Ungleichheit läßt sich durch Abbilden der jeweiligen Informationsmodelle auf das Referenzmodell die semantische Äquivalenz überprüfen) und
- sie sollte als Ausgangspunkt und Grundlage für die Syntax-Definition dienen.

In diesem Sinne nützt die Informationsmodellierung im Kontext des Datenaustauschs sowohl den Entwicklern als auch den Anwendern der Framework-Technologie.

EDIF European TSC Information Modelling Working Group

In seiner Präsentation des DASSY-Projekts auf dem EDIF-Forum in Königswinter im Oktober 1989 regte Wolfgang Wilkes die Einrichtung einer Arbeitsgruppe „EDIF and Data Modelling“ an. Die Initiative wurde in der EDIF-Gemeinde positiv aufgenommen. Hilary Kahn, Vorsitzende des EDIF European Technical Sub-Committees (TSC), etablierte daraufhin die *EDIF European TSC Information Modelling Working Group*, kurz IMG genannt, deren konstituierende Sitzung im Dezember 1989 an der Manchester University stattfand. Seit ihrer Gründung wurden zahlreiche intensive Arbeitstreffen in Hagen und Manchester abgehalten. Die Universität-GH Siegen gehört zu den Gründungs- und aktiven Mitgliedern der IMG, die sich personell größtenteils aus DASSY-Projektpartnern zusammensetzt. Zum derzeitigen Stand sind folgende Universitäten und Firmen an den Arbeiten der IMG aktiv beteiligt:

Büro für Informatik und Datenverarbeitung	:	Gerold Alfs
Fujitsu, England	:	Richard Henderson
Manchester University, England	:	Rachel Lau
FernUniversität-GH Hagen	:	Gerhard Scholz, Wolfgang Wilkes
TH Darmstadt	:	Max Ungerer
ICL, England	:	Alan White
Universität-GH Siegen	:	Klaus Quibeldey-Cirkel

Die Liste der Institute und Unternehmen, die ihr Interesse an den Arbeiten und Ergebnissen der IMG ausdrücklich bekundet haben, ist um ein Vielfaches länger [18].

Zielsetzung und Arbeitsphasen Das in der Satzung definierte globale Arbeitsziel — „towards an information model underlying EDIF 2 0 0 as it is defined in the Reference Manual and in the Questions and Answers“ — reflektiert die Bemühungen des EDIF Technical Committees (TC), den semantischen Gehalt der EDIF-Konstrukte in der aktuellen Version und in Vorschlägen für zukünftige Erweiterungen zu identifizieren und formal zu beschreiben. Als Beschreibungssprache entschied man sich für EXPRESS, der von Douglas Schenck im Rahmen der STEP-Initiative entwickelten formalen Notation zur Informationsmodellierung. Das erste Teilziel der Arbeitsgruppe bestand folglich darin, die Anwendbarkeit von EXPRESS zur Darstellung der EDIF-spezifischen Modellierungskonzepte, die sich hinter der Syntax des Datenaustausch-Formats verbergen, zu evaluieren. Dies wurde exemplarisch an den EDIF-Konstrukten *figure* und *figureGroup* praktiziert [60]. Das Ergebnis war überwiegend positiv. Entdeckte Defizite in der Ausdrucksmächtigkeit von EXPRESS wurden durch aktive Mitarbeit in den STEP-Normungsgremien behoben und notwendige Erweiterungen dort eingebracht [66].

Das globale Arbeitsziel wurde in mehreren Phasen angenähert:

1. Die einerseits aus der praktischen Evaluierung der EXPRESS-Beschreibungsmöglichkeiten und anderseits aus der Anforderungsanalyse zur DASSY-Werkzeugschnittstelle gewonnenen Erfahrungen und Erkenntnisse versetzten die Mitglieder der Arbeitsgruppe

rasch in die Lage, ein Grobmodell der in EDIF darstellbaren Objekt-Kategorien und -Beziehungen aufzustellen.

2. Ein erster konkreter Modellierungsvorschlag lag dem EDIF TC bereits im Sommer 1990 vor: „EXPRESS Model of the Overall Structure of EDIF". Die Rückmeldungen waren äußerst erfreulich. Weitere Semantik wurde alsbald in den Vorschlag aufgenommen: komplexe *Netze* und *Ports*. Das Modell wurde daraufhin auf dem EDIF Forum in Warrington, England, und auf der EDIF World in New Orleans der EDIF Gemeinde vorgestellt. Durch die DASSY-Kooperation in der europäischen Sektion der CFI, hier insbesondere im TSC *Design Representation* (CFI DR), konnte die IMG ihre Vorstellungen in einem weiteren internationalen Normengremium einbringen. Auch dort wird an einem Informationsmodell für VLSI-Entwurfsdaten gearbeitet. Da sich zu diesem Zeitpunkt die Arbeiten der IMG und der CFI DR thematisch überschnitten, waren Anknüpfungspunkte gegeben, die eine solide Zusammenarbeit erlaubten.

3. Parallel zu den konzeptionellen Arbeiten in der IMG wurden einzelne EDIF TSCs bei der Modellierung ihrer Erweiterungsvorschläge unterstützt. So war beispielsweise die Gesamthochschule Siegen aktiv an der Definition einer Testsicht für EDIF beteiligt (in Personalunion zwischen den EDIF TSCs Test und IMG). Die testspezifischen Modellierungsaspekte wurden auf dem EDIF Forum in Warrington erstmals vorgestellt [69]. Der Übergang von der konventionellen Syntaxorientierung zur Informationsmodellierung wurde endgültig vollzogen. Das in EXPRESS beschriebene Test-Modell wurde und wird auf breiter Front dem internationalen Fachpublikum vorgestellt: EDIF World, ITC, ETC und DAC.

Kapitel 3

Stand der Forschung

Im folgenden wurde versucht, die Eigenleistungen der Siegener Fachgruppe innerhalb des Verbundprojekts herauszustellen (bezüglich Eigen- versus Gruppenleistung siehe auch die Anmerkungen im einleitenden Kapitel). Im wesentlichen sind dies Arbeiten zu:

1. Informationsmodellierung im Testbereich
2. Datenmodellierung im Kontext prozeduraler Werkzeugschnittstellen
3. Prototyp-Implementierung
4. Algebraische Spezifikation einer Werkzeugschnittstelle

Eine umfassende Darstellung der erzielten Ergebnisse kann den DASSY-Hauptdokumenten entnommen werden.

3.1 EDIF-Spezifikation für den Austausch von Testdaten

Das Ziel der Arbeiten in diesem Projektteil war es, Beschreibungsmöglichkeiten für Testdaten in das Datenautauschformat EDIF einzubringen. Obwohl die Möglichkeiten von seiten des Projekts voll ausgeschöpft wurden, konnte das Ziel nicht während der Projektlaufzeit erreicht werden. Ein wesentlicher Grund liegt vor allem in dem Aufwand, der mit dem Wechsel der prinzipiellen Arbeitsmethodik in den EDIF Technical Committees verbunden war.

Bestand die Vorgehensweise zunächst darin, Änderungen an der EDIF-Syntax vorzuschlagen, so wurde die Methode Ende 1989 auf die Modellierung der Informationen umgestellt [68]. Dies bedeutete zunächst, daß wegen der zu durchlaufenden Lernkurve und der zu bereinigenden Unklarheiten in der Version 2 0 0 eine Verzögerung eintrat. Durch die Unterstützung aus dem

DASSY-Projekt (zum Teil in Personalunion mit der IMG) konnte diese Phase auch innerhalb des Test TSC deutlich verkürzt werden.

Zum jetztigen Zeitpunk ist absehbar, daß die Integration von Testdaten in die EDIF-Hauptversion auf der Basis des entworfenen Informationsmodells deutlich leichter und schneller vonstatten gehen wird als die alte Praxis der Syntax-Orientierung. Die Arbeiten werden im ersten Quartal 1993 abgeschlossen sein. Der Beitrag des DASSY-Projekts, insbesondere die bereitgestellten Möglichkeiten, durch persönliche Kontakte die Zusammenarbeit mit der EDIF-Organisation in Amerika zu intensivieren, ist nicht zu unterschätzen.

3.1.1 Arbeitsphasen

Im folgenden wird die im Projekt geleistete Arbeit in vier Phasen dargestellt, die inhaltlich die wichtigsten Schritte umreißen. Daß sich jeweils eine Dauer von ungefähr einem Jahr ergibt, ist Zufall.

- **Phase I:** Mitte 1988 bis Mitte 1989

  Die Arbeiten beginnen damit, daß die in Version 2 0 0 vorgesehenen syntaktischen Elemente erweitert werden. Die Prämisse des EDIF Technical Committee (TC) ist, die Syntax von Version 2 0 0 nur minimal zu verändern, so daß die neue Version so weit wie möglich syntaxkompatibel zur alten wird.

  Die Arbeit wird vorwiegend in Europa durchgeführt. Ein amerikanischer Teil des EDIF Test Technical Subcommittee (TTSC) existiert formal, aber wegen organisatorischer Probleme ist dessen Beitrag gering.

  Das technische Spektrum, das vom TTSC bearbeitet wird, ist sehr breit. Schwerpunkte sind formatorientierte Testmuster sowie Fehlerverzeichnisse.

- **Phase II:** Mitte 1989 bis Mitte 1990

  Um die Diskussionen im TTSC effektiver zu gestalten, wird eine eigene Methode zur Datenmodellierung entwickelt, sogenannte "Bubble Diagrams". Nachdem die Modellierungssprache EXPRESS sowie deren graphische Repäsentation EXPRESS-G vorliegt, wird diese Sprache eingesetzt. Gründe dafür sind die Leistungsfähigkeit der Sprache und die Tatsache, daß EXPRESS als formales Beschreibungsmedium für alle Teile der STEP-Initiative verlang wird.

  Der amerikanische Teil des TTSC formiert sich unter Bill Sebesta, IBM. Nun beginnt eine lange, fruchtbare Zusammenarbeit.

  Das Datenmodell in der Version 1.4.10 wird herausgegeben (*Blue Book* [57]). Stilistisch ist es noch sehr an die vorher entwickelte Syntax angelehnt. Es besteht im wesentlichen aus Vorschlägen des europäischen Teils, ist aber mit dem amerikanischen Teil abgesprochen.

- **Phase III:** Mitte 1990 bis Mitte 1991

Die Modelle werden überarbeitet. Das Ergebnis ist eine deutliche Abstraktion und damit ein wichtiger Schritt zu einem Informationsmodell. Außerdem wird die Prämisse des TC, möglichst nahe an Version 2 0 0 zu bleiben, fallengelassen. Der Grund ist, daß mit der Arbeit der IMG am Informationsmodell für Version 2 0 0 so viele Inkonsistenzen sichtbar wurden, daß die nächste Hauptversion ohnehin nicht mehr auf syntaktischer Ebene kompatibel bleiben kann. Dies befreit das TTSC von zahlreichen Beschränkungen.

Die Zusammenarbeit mit der amerikanischen Gruppe verbessert sich zusehends. Ein wesentlicher Grund sind die regelmäßigen persönlichen Kontakte, zu denen vor allem das DASSY-Projekt beigetragen hat. Auch die Rolle der amerikanischen Gruppe ändert sich: Während zunächst die in Europa erarbeiteten Vorschläge nur kommentiert wurden, wird jetzt aktiv an eigenen Vorschlägen gearbeitet.

Die wesentliche technische Weiterentwicklung besteht in der Einführung der Arraystruktur für die wichtigsten Komponenten des Tests: Zyklendauer, Formate, Testmuster und die Zuordnung von elektrischen Eigenschaften zu den logischen Werten. Daraus resultiert eine einheitliche Darstellung der Testinformation. Die Umstrukturierung ist eine direkte Folge der höheren Abstraktion durch den Übergang zur Informationsmodellierung. Das Ergebnis der Arbeiten ist die Version 2 0 32 (*Green Book* [58]), das dem TC vorgestellt wird.

Um die entwickelten Konzepte zu überprüfen, wird aus einer Zwischenversion ein prozedurales Interface erzeugt. Dieses Format wird *Test Specification Format* (TSF) genannt, um es eindeutig von EDIF zu unterscheiden. Im EVEREST-Projekt [42] entstehen eine Reihe von Programmen, die TSF lesen und schreiben können. Die Erfahrungen bei der Implementierung und der anschließenden Testphase fließen in die weitere Entwicklung der EDIF-Testsicht ein.

- **Phase IV:** Mitte 1991 bis Mitte 1992

  Das EDIF TC ist durch den enormen Aufwand für die Hauptversion nicht in der Lage, das Model 2 0 32 direkt zu bearbeiten. Die Zeit wird genutzt, um mit der Version 2 0 300 einen ersten vollständigen Vorschlag zu erarbeiten. Die Dokumentation besteht aus einer Einführung, dem Informationsmodell und einem Syntaxvorschlag.

  Das Informationsmodell wird ein weiteres Mal überarbeitet, wobei ein großer Schritt in Hinsicht auf die zur Integration in die Hauptversion notwendige Abstraktion vollzogen wird. Es befindet sich damit auf der gleichen Abstraktionsebene wie die Hauptversion.

  Im amerikanischen Teil wird intensiv an Tests von Parametern und an komplexen Testmustern für DFT-unterstützte Systeme gearbeitet. Die im Herbst 1992 vorgestellte Version 2 0 301 enthält die Erweiterungen in bezug auf den Test von Parametern.

  Bei Ende des Projekts zeichnet sich ab, daß das EDIF TC im Herbst damit beginnen wird, Testinformation in die Hauptversion zu integrieren, so daß die Version 3 0 0, die im April 1993 herausgegeben wird, auch die grundlegenden Anforderungen des digitalen Tests abdecken wird.

An die vierte Phase anschließend wird die Testerweiterung in die Hauptversion integriert. Die Arbeiten werden im März 1993 abgeschlossen sein.

Das DASSY-Projekt hat einen wesentlichen Beitrag zum Fortschritt der Arbeiten geleistet. Die Verzögerungen gegenüber den Planungen zu Projektbeginn liegen zum einen in der grundlegenden Änderung der Arbeitsmethode. Andererseits darf bei einer internationalen Zusammenarbeit nicht vergessen werden, daß der Arbeitsablauf nicht an einer Stelle allein geplant wird, sondern durch Konsens in einer Hierarchie von Komitees festgelegt wird, wobei dem Einfluß des Einzelnen enge Grenzen gesetzt sind. Unter Beachtung dieser beiden Aspekte kann das Projekt hinsichtlich der Modellierung der Testinformation uneingeschränkt als erfolgreich bezeichnet werden.

3.2 DASSY-Datenmodell: DaDaMo

3.2.1 Anforderungsanalyse

Ziel war es, die operationalen Anforderungen an eine EDIF-orientierte VLSI-Werkzeugschnittstelle aus der Analyse der Arbeitsweise der Entwurfswerkzeuge abzuleiten. Um dieses Ziel koordiniert und mit einheitlicher Methodik zu erreichen, wurde folgender Weg beschritten:

1. Tabellarische Zusammenstellung der bei den Projektpartnern verfügbaren Werkzeuge gruppiert nach Entwurfsaufgaben wie Schematic Entry, Digital-, Analog-, Fehler- und Device-Simulation, Testmustererzeugung und Layouterstellung
2. Aufstellen eines formalen Kriterienkatalogs zur Anforderungsanalyse ausgewählter Werkzeuge
3. Untersuchung der Quellprogramme der Entwurfswerkzeuge anhand des Kriterienkatalogs (Ist-Zustand)
4. Katalogisierung der Ist-Anforderungen in einheitlicher Notation unter Berücksichtigung wünschenswerter Anforderungen aus der Sicht des Werkzeugbenutzers (Soll-Zustand)

3.2.1.1 Anforderungsanalyse der Werkzeugsicht „Schematic Entry“

Im Rahmen des Arbeitsplans untersuchten die Siegener Projektpartner das toolspezifische Anforderungsprofil der Entwurfsebene „Schematic Entry“. Primärer Gegenstand der Untersuchung waren die am Siegener Institut entwickelten und in Praktika erprobten Werkzeuge zur graphischen Schaltplan-Erfassung und Netzlisten-Extraktion: ICE (Integrated Circuits Editor [63]) und NESSI (Netlist Extraction from Structured Schematic Input [64]). Als weitere Fallstudien wurden auch die Schematic-Werkzeuge der anderen Projektpartner auf der Grundlage der Programmspezifikationen berücksichtigt.

Die folgende Anforderungsanalyse der Werkzeugsicht „Schematic Entry“ [65] fußt im wesentlichen auf dem Kombinationskonzept der Stromlaufplan-Erfassung, d. h. auf dem CAD/CAE-Verbund von hierarchischen Graphikeditoren und Extraktoren.

Die hierarchische Schaltplanerfassung und ihre logische Extraktion begünstigen ein objektorientiertes Kapseln, Speichern und Manipulieren der Entwurfsdaten. Die Entwurfsobjekte weisen zum einen eine vielfältige individuelle Attributsstruktur auf: flache und verschachtelte Attribute; zum andern sind sie in der Regel in ihrem Aufbau komplex: sie setzen sich aus weiteren Objekten zusammen. Die Unterobjekte werden dabei nicht explizit mit ihren zugehörigen Attributmengen und ihrer eigenen Aufbaustruktur eingefügt. Stellvertretend stehen hier lediglich Verweise (Referenzen) auf die zugeordneten Objektklassen, von denen sie typische Attribute und die Objektstruktur erben.

Für die CAD/CAE-Werkzeuge, die auf der Entwurfssicht „Schematic Entry“ arbeiten, ergab die Untersuchung ihrer internen Datenhaltungen die folgende Klassifizierung ihrer Objekte und Operationen.

Objekte Prinzipiell kann die Klassifizierung der Entwurfsobjekte nach Eingabe- und Ergebnisdaten erfolgen. Die Programmeingaben für die Schaltplanerfassung und deren Logikextraktion umfassen geometrische Primitiv-Objekte mit einfachen deskriptiven Eigenschaften, die einer formalisierten Beschreibung unterliegen. Die Ergebnisdaten der Logikextraktion sind dagegen in der Regel komplex strukturiert, d. h. sie setzen sich aus mehreren Komponenten zusammen und tragen als Informationseinheiten die elektotechnische und logische Semantik der Schaltplankomponenten.

Eingabeobjekte Die hier untersuchten Werkzeuge beschreiben die graphische Repräsentation der Schaltungskomponenten im *Caltech Intermediate Format* (CIF), das ursprünglich von Mead und Conway für die Beschreibung von Maskengeometrien eingeführt wurde [32]. Als primitive geometrische Konstrukte stehen zur Verfügung: *polygon, box, wire* und *roundflash*. CIF-Objekte können bestimmten Ebenen, sogenannten *layers*, zugeordnet werden. Das *symbol*-Konstrukt erlaubt das Gruppieren zusammengehöriger Geometrien, die dann als Einheit (Makro, *symbol*) aufrufbar sind. Der Aufruf *(call)* eines *symbols* kann über eine Transformation der Koordinaten erfolgen: Translation des Symbolursprungs, Spiegelung an den Koordinatenachsen und Rotation der x-Achse.

Extrahierte Objekte Die mit dem Graphikeditor erstellten Diagramme besitzen ausschließlich geometrische und textuelle Informationen, tragen aber keine funktionale Bedeutung. Erst die Logikextraktion und die Verbindungsanalyse liefern die funktionalen Schaltungskomponenten und deren logischen Verbindungen. Die Extraktion unterscheidet die folgenden Objekte nach ihrer Granularität:

- Gesamtnetz, Teilnetze und Schaltungsknoten
- einfache und gebündelte Verbindungsleitungen
- uni- und bidirektionale Anschlüsse
- Gattersymbole

Für die Erstellung einer simulator-spezifischen Netzliste müssen zusätzlich elektrische und logische Parameter aus den Zell-Bibliotheken extrahiert werden.

Operationen Bei der Differenzierung der in den CAD/CAE-Werkzeugen bereits implementierten bzw. aus Benutzersicht als wünschenswert postulierten Operationen ist die Selektivität des Zugriffs ein wesentliches Kriterium. Hier kann zwischen den graphischen Objekten, die im Graphikeditor erzeugt und manipuliert werden, und den logisch abstrahierten Schaltplanobjekten, die aus der Graphikextraktion resultieren, unterschieden werden.

Zugriffe auf graphische Objekte Da der Benutzer immer nur einen begrenzten Ausschnitt des graphischen Datenbestandes eines Entwurfsobjektes gleichzeitig überschauen und bearbeiten kann, müssen ihm geeignete selektive Zugriffsoptionen zur Verfügung stehen. Die Ausschnittsbegrenzung ist zum einen technischer Art: Auflösungsvermögen und Abmessung des Graphikmonitors bzw. des Papierausdrucks, zum andern geht sie auf die Tradition des Reißbrett-Konstruierens zurück: Erstellen eines Schaltplans in vorgegebenen Seitenformaten. Also sollte die Werkzeugschnittstelle auf der Schematic-Ebene Ausschnittsabfragen nach folgenden Selektionskriterien ermöglichen:

- Schaltplanseiten (*page*-spezifische Objekte)
- Graphikebenen (*layer*-spezifische Objekte)
- Fensterausschnitt (*bounding-box*-Objekte)

Zugriffe auf Schaltplanobjekte Die in den Laborpraktika und zum Teil in der Industriepraxis gewonnenen Erfahrungen beim Entwurf integrierter Schaltungen zeigen, daß auch auf der logischen Darstellungsebene der Schaltung erweiterte On-line-Zugriffe erforderlich sind. Komfort und Effizienz der interaktiven Entwurfsarbeit können durch schaltplanorientierte Objektzugriffe wesentlich gesteigert werden. Ohne den Anspruch auf Vollständigkeit diene die folgende Liste der Selektionskriterien:

- zell-spezifische Unterobjekte bei vorgegebener Aufruftiefe
- primäre Signale
- benutzer-definierte Bezeichner für Signale, Netze, Anschlüsse und Bausteine
- vom Extraktor vergebene Bezeichner
- Verbindungspunkte, die durch Leitungskreuzungen impliziert werden
- Teilnetze, z. B. technologie-spezifische Netze wie tri-state und wired-or
- technologie-spezifische Bausteine: LSTTL, FAST usw.

Weiterhin scheinen Zugriffe auf Schaltplanobjekte sinnvoll, die erst bei einer Revision der Schaltung nach der Logiksimulation und Testbarkeitsprüfung lokalisierbar sind. Beispielsweise ergäbe die Menge aller nicht testbaren Schaltungsknoten ein wichtiges Selektionskriterium, weitere sind denkbar.

Operationen zur Konsistenzsicherung Die Konsistenzsicherung der Entwurfsdaten auf der Schematic-Ebene muß sowohl über spezielle externe Prüfroutinen, *Electrical-Rule Checker*, als auch durch effiziente Konsistenzmechanismen innerhalb der Werkzeugschnittstelle gewährleistet werden. Inwieweit die Einhaltung der Entwurfsregeln durch Datenbankoperationen gesichert werden kann, hängt entscheidend von der Mächtigkeit des semantischen Datenmodells der Werkzeugschnittstelle ab.

Da Logikextraktoren auf mehreren Repräsentationsebenen arbeiten (Darstellung der Schaltung als graphisches Symbol, Schaltplan und Netzliste), muß die Korrektheit des Gesamtbestandes der Extraktionsdaten hinsichtlich ebenen-übergreifender Konsistenzregeln gewährleistet sein. Neben einfachen Konsistenzprüfungen, die im wesentlichen die Historie der Objektreferenzen auswerten (*define-before-use*-Regel), sind algorithmisch komplexere erforderlich. Beispielsweise wäre eine allgemeine Konsistenzaussage über die Repräsentationen ein und derselben Schaltung auf unterschiedlichen Entwurfssichten wünschenswert, z. B. auf den EDIF-Sichten *schematic* und *netlist*. Die Werkzeugschnittstelle sollte entsprechende Konsistenzaussagen ermöglichen.

3.2.1.2 Anforderungen aus dem Entwurfsprozeß

Parallel zur Analyse der Werkzeuganforderungen an eine Datenbankschnittstelle wurde in einer zweiten Arbeitsgruppe der Entwurfsprozeß untersucht. Als Vorgehensweise wurden zwei Ansatzpunkte vorgeschlagen:

1. Analyse und Auswertung der in der Literatur beschriebenen allgemeinen Entwurfsmethoden hinsichtlich ihrer Anforderungen an eine zentrale Datenhaltung
2. Zusammenstellung pragmatischer Anforderungen aufgrund eigener Erfahrungen im Chip-Entwurf der Mitglieder der Arbeitsgruppe

Eine detaillierte Zusammenstellung der Endergebnisse wurde am ersten Meilenstein vorgetragen und allen DASSY-Partnern in Form eines Berichts vorgelegt [52]. Im folgenden soll die Arbeit der Gruppe skizziert werden.

Entwurfsprozeß Der Entwurf einer intergrierten Schaltung ist ein komplexer Prozeß, der durch eine Reihe von Randbedingungen besondere Anforderungen und Ansprüche an die Datenhaltung stellt. Im Unterschied zu klassischen Datenbankanwendungen arbeiten bei einem Chip-Design

- arbeitsteilig eine in ihrer Zusammensetzung möglicherweise häufig wechselnde Gruppe von Personen
- mit einer großen Anzahl verschiedener Werkzeuge
- über einen relativ langen Zeitraum (Wochen, Monate) hinweg
- in einem iterativen Prozeß

und produzieren damit eine große Menge Information verschiedenster Art und Struktur (Verhaltensbeschreibungen in Klartext, logische Information in speziellen Simulatorsprachen, geometrische Daten etc.). Daraus ergibt sich die Forderung nach einer geeigneten Objektverwaltung, die Forderung nach einer die Einzelprozesse synchronisierenden und koordinierenden Ablaufverwaltung sowie die Forderung generell vorläufige und damit inkonsistente Zwischenergebnisse zu speichern und den verschiedenen Benutzern zugänglich zu machen.

Objektverwaltung Als Objekte in diesem Kontext sollen Gebilde innerhalb der Datenhaltung bezeichnet werden, die aufgrund bestimmter Kriterien (Zugriffsoperationen, topologischer Aufbau etc.) das Merkmal der Eigenständigkeit besitzen und einen Namen haben. Objekte können hierarchisch strukturiert sein, daß heißt aus anderen Objekten bestehen. Objekte können in verschiedenen Varianten auftauchen, die als Versionen bezeichnet werden. Damit soll sowohl die historische Entwicklung eines Objekts erfaßt werden als auch eine Art Verwandschaftsverhältnis zwischen zwei Objekten verdeutlicht werden können. Objekte sollen nach bestimmten Attributen gruppiert und zu neuen Objekten zusammengefaßt werden können (Bibliotheksverwaltung).

Verschiedene Sturkturierungsebenen (Abstraktionsebenen, Sichten) für Objekte sollen von der Werkzeugschnittstelle unterstützt werden. Beziehungen zwischen Objekten sollen sowohl innerhalb einer Strukturierungsebene als auch orthogonal dazu definiert werden können. Daraus ergibt sich unmittelbar die Forderung nach der Verwaltung von einfachen und komplexen Konsistenzbedingungen. Als Beispiel für eine einfache Konsistenzbedingung soll hier die Plausibilitätsprüfung von Objektattributen bezüglich vorgegebener Regeln angeführt werden, die von der Werkzeugschnittstelle selbst unterstützt werden kann. Als Prüfung einer komplexen Konsistenzbedingung wird beispielsweise der Verifikationslauf eines Simulators bezeichnet, der außerhalb der Schnittstelle stattfindet, jedoch von der Datenhaltung verwaltet wird (Simulatorlauf wird angestoßen und Ergebnis gespeichert).

Ablaufverwaltung Zusätzlich zur reinen Verwaltung von Daten soll die Werkzeugschnittstelle einen gleichzeitigen und geregelten Zugriff mehrerer Benutzer oder externer Werkzeuge auf diese Daten unterstützen. Dazu müssen Mechanismen bereitgestellt werden, die

- nur autorisierten Benutzern und Benutzergruppen Zugang zu den entsprechenden Design-Daten verschaffen

- durch Verwaltung von Zugriffsrechten lediglich verschiedene Objekthierarchien bzw. -sichten sichtbar machen
- den gleichzeitigen Zugriff mehrerer Benutzer auf ein und dasselbe Objekt synchronisieren und dadurch Fehler verhindern, die durch paralleles Arbeiten entstehen können
- die Einhaltung einer vorgegebenen Reihenfolge von Entwurfsschritten überwachen und unterstützen (Design-Management)
- bei Systemabstürzen Datenbestände in einem gewissen Rahmen restaurieren können
- den informellen Datenaustausch (z. B. von Zwischenergebnissen) zwischen verschiedenen Benutzern unterstützen

Alle erwähnten Forderungen wurden daraufhin überprüft, inwieweit sie oder zumindest Teilaspekte davon in klassichen Datenbanksystemen und vorhandenen Frameworks Berücksichtigung finden. Hier mußte festgestellt werden, daß speziell die Randbedingungen Unterstützung von Benutzergruppen, lange Transaktionsszeiten und die damit verbundene Verwaltung von inkonsistenten Zwischenzuständen sowie die Unterstüzung von Design-Management nur ansatzweise bzw. lediglich in nicht implementierten Konzepten zu finden sind.

Konsistenzbedingungen Ziel jeder zentralen Datenhaltung ist es, die Konsistenz der gespeicherten Datenbestände zu gewährleisten. Dabei ist in diesem Zusammenhang zwischen einer sogenannten „statischen“ Konsistenz und einer „operationalen“ Konsistenz zu unterscheiden. Die statische Konsistenz läß sich, wie weiter oben beschrieben durch einfache Konsistenzbedingungen ausdrücken, und kann innerhalb der Datenhaltung überprüft werden. Die operationale Konsistenz wird außerhalb der Datenhaltung überprüft und muß daher verwaltet werden. Dazu ist es erforderlich

- unterschiedliche Grade der Konsistenz auszudrücken (z. B. Layout eines Objekts ist verifiziert hinsichtlich logischer Beschreibung, aber nicht hinsichtlich Design-Rules).
- Konsistenzbedingungen auf unterschiedliche Art zu formulieren (Programme, Boolesche Ausdrücke im Datenmodell)
- auf Konsistenzverletzungen unterschiedlich zu reagieren (z. B. Festlegung der Maßnahmen durch den Designer)

Der Literatur konnte ein Über-Alles-Modell des Enturfsprozesses entnommen werden, das die iterativen Entwurfsschritte der Analyse, Synthese und Evaluierung sehr anschaulich wiedergibt und sich mit den in der Arbeitsgruppe vorhandenen Entwurfskenntnissen und -erfahrungen am ehesten deckt: das Y-Diagramm nach Daniel GAJSKI [22].

Makroskopisches Modell des Entwurfsprozesses

Der Anspruch des Y-Diagramms — „a comprehensive view of VLSI CAD tools“ — wurzelt in dem Bedürfnis, die Hauptströmungen in der Chip-Konstruktion der 80er Jahre übersichtlich zu erfassen. Je nach dem Grad der Automatisierung und der Art der Wissenseinbringung unterscheiden GAJSKI und KUHN drei Verfahrenslandschaften:

- *Entwurfsunterstützung*

  Alle Entwurfsentscheidungen obliegen dem Entwerfer. CAD-Werkzeuge wirken ausschließlich unterstützend: „efficient paper and pencil“.

- *Expertensysteme*

  Verlagerung der Entwurfsunterstützung von den mechanischen zu den intelligenten Aspekten des Entwurfs — Expertenwissen als Quelle wird in Form von Entwurfsregeln in Wissensbanken abrufbar gespeichert (effizient für die Entwurfsprüfung, noch in den Anfängen bezüglich Entwurfssynthese und -planung).

- *Entwurfsautomation*

  In der Zielsetzung konträr zum CAD-Ansatz: statt zu unterstützen, wird der teilweise oder gänzliche Verzicht auf den Entwerfer angestrebt (z. B. automatisches Plazieren und Entflechten von Layout-Zellen, Silicon-Compiler).

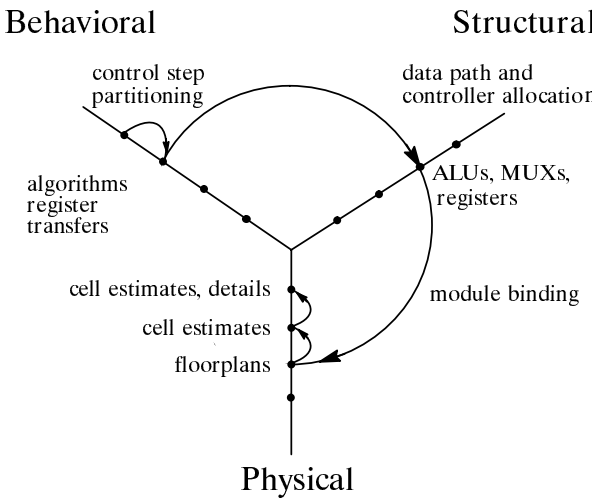

Bild 3.1: Y-Diagramm mit Anwendungsbeispiel nach THOMAS und WALKER

Der Entwurfsprozeß wird schematisch als Achsen-Modell dargestellt: Die Achsen spannen den Entwurfsraum in drei komplementäre Entwurfsbereiche auf, als Sichten oder Domänen bezeichnet. Die *Verhaltenssicht* ist auf die Funktionalität, die *Struktursicht* auf die logische Struktur und die *physikalische Sicht* auf die physikalische Implementierung gerichtet. Andere

Aspekte[1] können unter diesen Domänen subsumiert werden. Entlang der Raumachsen erfolgt von außen nach innen die Entwurfsverfeinerung auf Ebenen zunehmender Detaillierung.

Das Y-Diagramm soll also eine Antwort geben auf die Frage: „Wie werden VLSI-Werkzeuge eingesetzt, um eine Entwurfsspezifikation auf höherer Ebene zu verifizieren und in eine Implementation auf niedriger Ebene zu überführen?“ Einerseits sollen die Verfahrensketten (Synthese, Analyse und Verifikation) bereits in der Entwurfsplanung veranschaulicht werden, und zwar im gesamten Entwurfsraum sichtenorientiert und unter Berücksichtigung der geltenden Entwurfsabstraktion. Andererseits soll das Y-Modell die Mechanismen und Interaktionen *realer* Entwurfssysteme transparent machen. Bild 3.1 illustriert letzteres für das von THOMAS und WALKER angegebene Beispiel der automatischen Datenpfad-Synthese [46].

Als „inter domain links“ verlaufen die Repräsentationsebenen konzentrisch um den Ursprung des Y-Diagramms. GAJSKI spezifiziert einen Standardsatz von fünf Abstraktionsebenen[2] und nennt exemplarisch einige Entwurfsobjekte der jeweiligen Sicht (Tabelle 3.1).

Levels of Abstraction	Domains of Description		
	behavioral	*structural*	*physical*
system	performance specs.	CPUs, memories, controllers, busses	physical partitions
algorithmic	algorithms	hardware modules, data structures	clusters
micro-architectural	register transfers, state sequencing	ALUs, MUXs, registers, microsequencer	floorplans
logic	Boolean equations	gates, flip-flops, cells	cell, module plans
circuit	transfer functions, timing	transistors, wires, contacts	layout

Tabelle 3.1: VLSI-Schichtung versus Entwurfssichten nach GAJSKI

Mittels einer graphischen Notation lassen sich die Entwurfswerkzeuge nach Art und Wirkungsbereich im Entwurfsraum lokalisieren. Synthetisierende Werkzeuge werden als gerichtete Übergänge notiert: aus der Verhaltens- in die Struktur-Domäne (im Bild 3.1 „data path and controller allocation“), aus der Struktur- in die physikalische Domäne („module binding“) oder von einer höheren Abstraktionsebene in eine niedrigere gleicher oder benachbarter Domäne („control step partitioning“). Umgekehrte Übergänge beziehen sich auf entsprechende analysierende Werkzeuge (z. B. Schaltkreis-Extraktoren). Eine Schleife (Anfangs- und Endpunkt

[1] RAMMIG regt zum Beispiel die Erweiterung des Y- zum X-Diagramm an, um testspezifische Aspekte im Entwurfsraum aufzuzeigen [38].

[2] Konrad WEBER schlägt für die Schichtung der Abstraktionsebenen den Begriff der „VLSI-Stratifikation“ vor und spricht in diesem Zusammenhang von *stratifizierten* Entwürfen. [48]

fallen zusammen) repräsentiert ein Analysewerkzeug, das ausschließlich auf einer Ebene arbeitet (Simulatoren). Darüber hinaus erlauben Einstiegs- und Zielpunkt der Verfahrenskette das Identifizieren der ursprünglichen Entwurfsspezifikation (hier die algorithmische Verhaltensbeschreibung) und der erreichten Implementierungstiefe (Masken-Layout). Ein weiteres Charakteristikum des Y-Modells ist die „non-isomorphe“ hierarchische Dekomposition der Entwurfskomponenten: *1-zu-n*-Abbildungen zwischen Verhaltens-, Struktur- und physikalischen Entwurfsobjekten. Im Beispiel von Bild 3.1: Für eine gegebene Menge von Register-Transfer-Beschreibungen existieren mehrere abstrakte Implementierungen in der Struktur-Domäne. „Choosing one of these alternatives is exploring the design space.“ [46]. Weiterhin erlaubt das Modell die Lokalisierung von Entwurfseinschränkungen, wie beispielsweise Flächenbedarf und Verlustleistung.

3.2.2 Modellierungskonzepte für eine allgemeine Werkzeugschnittstelle

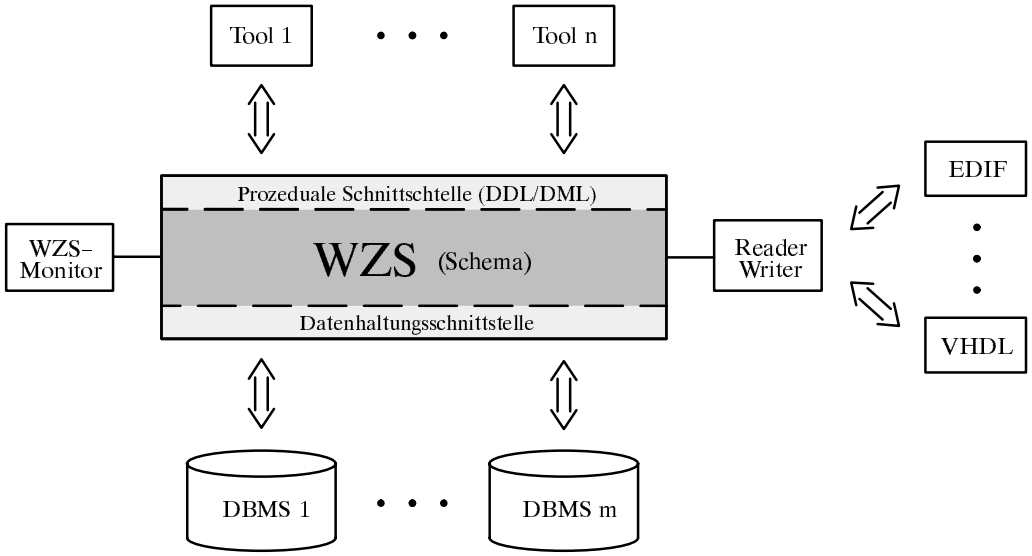

Bild 3.2: Architektur der DASSY-Werkzeugschnittstelle (WZS)

Wir haben bereits darauf hingewiesen: In einem VLSI/CAD-Framework erfolgt die Schnittstellenanpassung nicht allein dadurch, daß Entwurfsdaten als *blobs*[3] transferiert werden. Vielmehr muß auch die Möglichkeit bestehen, diese Daten einzeln und im interessierenden Zusammenhang zu interpretieren und zu manipulieren. Mit anderen Worten: Entwurfswerkzeuge sollen über die Framework-Schnittstellen miteinander operieren und gegebenenfalls austauschbar sein, um so die Integrationskosten gering zu halten. Prozedurale Werkzeugschnittstellen leisten das Gewünschte. Bild 3.2 zeigt beispielsweise den Aufbau einer solchen, wie sie im DASSY-Projekt entwickelt wurde.

[3] *binary large objects*

In Anlehnung an die Terminologie der Datenbankler sprechen wir hier von einem *Datenmodell*[4], das es für prozedurale Interfaces zu entwerfen gilt. Die Aufgabenstellung ist ähnlich wie bei der geschilderten Informationsmodellierung für Daten-Austauschformate, nur daß hier der dynamische Aspekt des Datenzugriffs hinzukommt. Wir sprechen deshalb auch von *konzeptionellen Modellen* in Abgrenzung zu den Informationsmodellen, die ausschließlich die statischen Aspekte eines Weltausschnitts repräsentieren.

Prozedurale Werkzeugschnittstellen und ihre konzeptionelle Modellierung sind Brennpunkte der Entwicklungsaktivitäten in CFI und anderen Framework-Initiativen [23]. Im folgenden wollen wir stellvertretend für die Anforderungen aus diesem Bereich das DASSY-Datenmodell (*DaDaMo*) vorstellen [53].

DaDaMo ist ein effizienter Konzept-Mix aus der semantischen und objektorientierten Datenmodellierung. Seine Implementierbarkeit wurde bereits prototypisch nachgewiesen und seine Relevanz für die Framework-Standardisierung international anerkannt. DaDaMo beinhaltet erweiterte Konzepte für die Definition und Manipulation komplexer Entwurfsdaten:

- Entwurfsdaten, wie Schaltungskomponenten und Verbindungsstrukturen, werden grundsätzlich als *Objekte* im Sinne des Objekt-Paradigmas modelliert (Datenkapselung, Klassenzugehörigkeit) und gehandhabt (abstrakte Datentypen, klassenbezogene Methoden). Die statischen Eigenschaften eines Objekts werden durch seine Attribute, die strukturellen durch seine (attribuierbaren) Beziehungen zu anderen Objekten und die dynamischen durch benutzer-definierte Operationen beschrieben. Die Definition von Constraints erlaubt darüber hinaus, die Objekteigenschaften individuell zu restringieren, um den besonderen Anforderungen der Entwurfswerkzeuge zu genügen. Der Konsistenzmechanismus in DaDaMo basiert auf dem Constraint-Konzept. Die Constraints werden ereignisabhängig ausgelöst (*Event*→ *Trigger*) und durch die Ausführung benutzerdefinierter Methoden evaluiert.

- DaDaMo unterstützt die Modellierung von VLSI-spezifischen Weltausschnitten, indem es die multiple Vererbung der Objekteigenschaften zuläßt und die Beziehungstypen frei definierbar hält. Welche Semantik eine Beziehung trägt — sowohl deskriptiv als auch operativ (Rollenfunktion) — und welche Objektstrukturen darstellbar sind — vernetzt oder verschachtelt —, bestimmt der Anwender.

  Die operative Semantik der Beziehungen wird *aktiv* unterstützt. Im Gegensatz zu anderen Vorschlägen drückt DaDaMo Beziehungen nicht allein durch die Angabe von Objektreferenzen aus, verbirgt das, was auf den Beziehungen geschehen soll, nicht in unzugänglichem Operationscode oder beläßt es gar auf bloße Hinweise durch entsprechende Bezeichner. Beziehungstypen, die oft im Mikroelektronik-Entwurf benötigt werden, sind in DaDaMo bereits vordefiniert:

  - *relate*
    symmetrischer Beziehungstyp (nur ein Rollenname für beide Richtungen), ohne

[4] Peter LOCKEMANN in [31]: „So wie ein Werkzeugkasten bestimmt, welche physikalischen Modelle gebaut werden können, so legt auch ein Datenmodell prinzipiell fest, wie immaterielle Modelle aussehen müssen."

besondere operative Semantik, dient der Navigation auf assoziierten Objektmengen durch Funktionen wie *set_relation, get_relation, modify_relation* und *remove_relation*

- *contain*
  abgeleitet vom Typ *relate*, beschreibt die Beziehung zwischen einem übergeordneten Container-Objekt, dem Aggregat, und den aggregierten inneren Objekten, wobei der Zugriff auf das Container-Objekt auch den Zugriff auf die inneren Objekte und deren Beziehungen impliziert (durch Überschreiben von *get_relation*)
- *shared contain*
  abgeleitet vom Typ *contain*, wobei aber die Existenz der inneren Objekte vom gesamten Aggregat abhängt: wird das Container-Objekt gelöscht, so auch seine Unterobjekte (durch Überschreiben von *remove_relation*), es sei denn, sie sind gleichzeitig innere Objekte anderer Container-Objekte
- *strong contain*
  abgeleitet vom Typ *shared contain*, wobei ein inneres Objekt durch spezielle Klassen-Constraints ausschließlich nur einem Container-Objekt zugeordnet ist und somit grundsätzlich mit diesem gemeinsam gelöscht wird

Über die voreingestellten Beziehungstypen können weitere definiert werden. Diese leiten sich stets aus den vordefinierten ab und werden durch benutzereigene Methoden in ihrer operativen Semantik spezifiziert.

- Abgeleitete Attribute verringern die Datenredundanz und erhöhen die Datenkonsistenz. Konkrete Entwurfsentscheidungen können durch abgeleitete Attribute auf der Objektebene, d. h. wertemäßig zur Laufzeit der Applikation, automatisch weitergegeben werden. Dies vereinfacht das Management für eine Objektversionierung erheblich [49]. Die für die aktuelle Ableitung eines Attributwerts von dem eines anderen Objekts erforderlichen Berechnungsformeln werden im Schema angegeben. VLSI-Beispiel: kapazitive Belastung eines Schaltungsknotens durch die Eingänge der aktuell verdrahteten Zellen.
- DaDaMo bietet nach außen zwei Sprachschalen für die Schemabeschreibung an. Zum einen steht eine EXPRESS-orientierte Schale zur Verfügung, um die in EXPRESS geschriebenen Informationsmodelle nahtlos in die Werkzeugschnittstelle zu integrieren, und zum andern eine an C++ angelehnte Sprachschale, um der Tatsache Rechnung zu tragen, daß der Großteil aller industriellen Framework-Applikationen in C bzw. C++ geschrieben ist.

3.3 DASSY-Prototyp

Durch die Entwicklung eines Prototypen der in Bild 3.2 skizzierten DASSY-Werkzeugschnittstelle sollte gezeigt werden, daß die in der Anforderungsanalyse gewonnenen Anforderungen an eine offene, integrierte VLSI-Entwurfsumgebung prinzipiell realisierbar sind. Zu diesem Zweck wurden zu Projektbeginn die unterschiedlichen Werkzeuge der Projektpartner analysiert und deren Anforderungen an ein gemeinsam nutzbares Datenmodell spezifiziert. Hierbei wurden die Konzepte des DASSY-Datenmodells DaDaMo als Basis verwendet.

Ziel war es, zum Projektabschluß einen Prototypen der Werkzeugschnittstelle zu präsentieren, mit dessen Hilfe unterschiedliche Entwurfswerkzeuge ihre Design-Daten auf beliebig austauschbaren Datenbanken verwalten können. Zudem sollte es möglich sein, über diesen Mechanismus einen Datenaustausch zwischen den verschiedenen Werkzeugen durchzuführen. Die DASSY-Werkzeugschnittstelle wurde auf den Datenbanksystemen ORACLE, OCT, POSTGRES und IREEN implementiert, um die verschiedenen dabei auftretenden Aspekte und Probleme zu beleuchten.

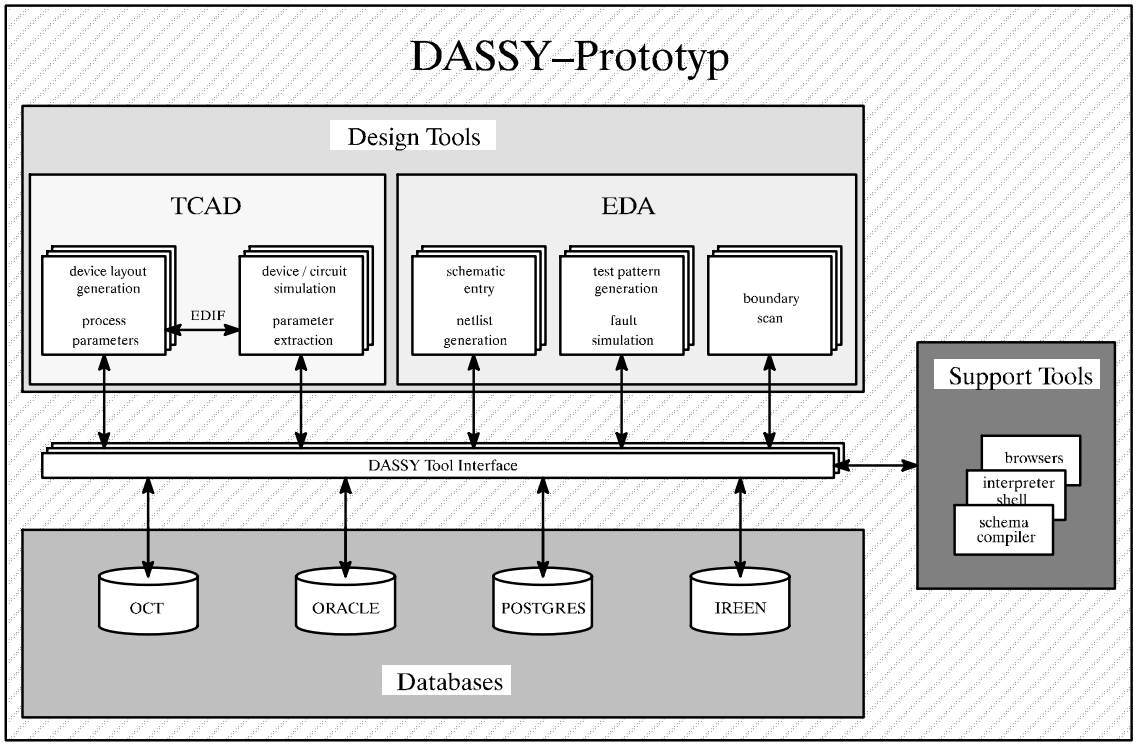

Bild 3.3: DASSY-Prototyp: Szenario

Die integrierten Entwurfswerkzeuge decken einen weiten Bereich von Entwurfsschritten ab. Alle von den Projektpartnern integrierten Werkzeuge haben dabei Zugriff auf die Datenbank über die DASSY-Werkzeugschnittstelle. Zur Integration der Werkzeuge waren unterschiedliche Strategien erforderlich, je nach Aufwand, Verfügbarkeit des Source-Codes oder bereits im Werkzeug existierenden Standardschnittstellen für Datenhaltung und Datenaustausch.

3.3.1 Prototyp-Schema

Die beiden von der Siegener Fachgruppe im Rahmen des DASSY-Prototypen angepaßten Design-Werkzeuge ICE und NESSI unterstützen lediglich Teile des in Bild 3.4 dargestellten Schemas.

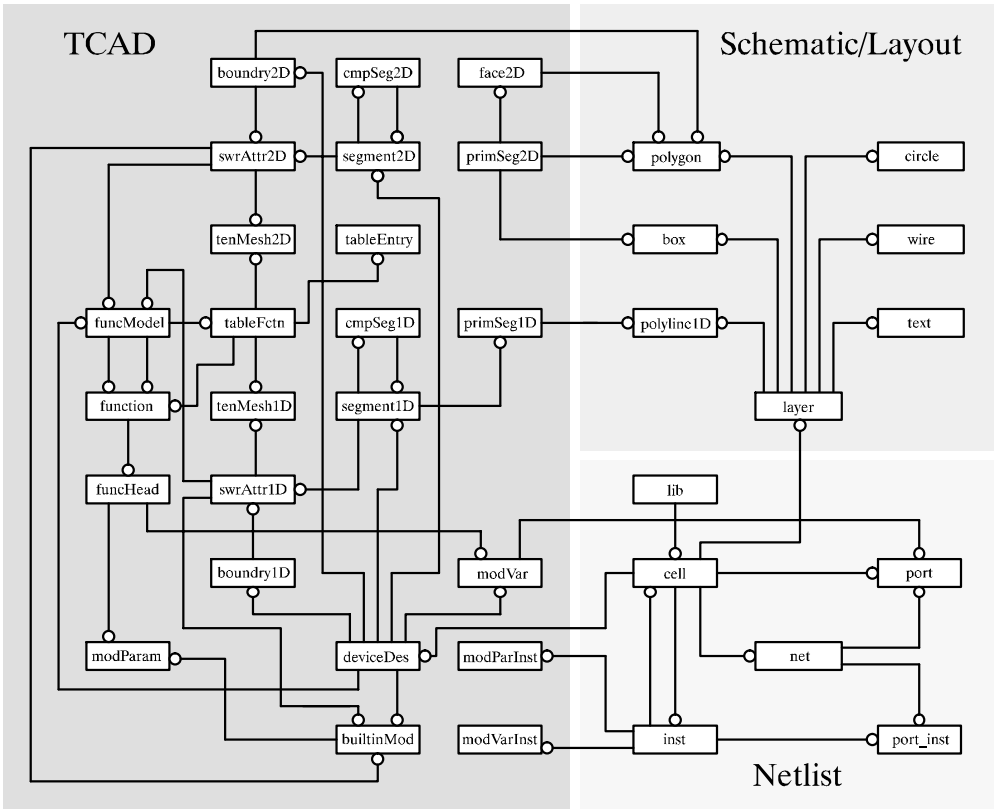

Bild 3.4: DASSY-Prototyp: Datenbankschema in *EXPRESS-G*-Notation

So unterstützt der Grafikeditor ICE neben den gemeinsamen Entities 'lib', 'cell' und 'inst' nur den Bereich Schematic/Layout. Soll mit ICE ein Schaltplan oder Maskenlayout bearbeitet werden, so ist vom Benutzer der Name seines Schaltungsentwurfs anzugeben. Diesem entspricht im Datenmodell eine Zellbibliothek (lib), die alle Teilschaltungen (cells) umfaßt. Die meisten Teilschaltungen sind als Instanzen (inst) in anderen Teilschaltungen enthalten und bilden somit eine Zellenhierarchie.

Neben den Instanzen untergeordneter Zellen enthält eine Schaltplan- oder Layout-Zelle in der Regel eine Menge geometrischer Entities (polygon, box, circle, wire, text), die verschiedenen Ebenen (layer) zugeordnet sind. Beim Maskenlayout entspricht jede Ebene einem Fertigungsschritt, während bei Schaltplänen unterschiedliche Ebenen für Gattersymbole, Verbindungsleitungen und -knoten, Ports, Busse etc. vorgesehen werden.

Wird mit ICE eine Hierarchie von Schaltplan-Zellen erstellt und gemäß dem zuvor skizzierten Schema in der Datenbank abgelegt, so ist der Postprozessor NESSI in der Lage, diese rein geometrischen Daten zu analysieren und entsprechende Netzlisteninformationen zu erzeugen. Für jedes Symbol (inst) werden dabei die entsprechenden Symbolanschlüsse (port_inst) ermittelt und durch Analyse der Leitungen zu Netzen (net) verbunden. Symbole der primären

Ein- und Ausgänge werden als Zellenanschlüsse (ports) interpretiert.

3.3.2 Integration der Siegener Entwurfswerkzeuge

Als Beitrag der Siegener Fachgruppe wurden zwei an diesem Institut entwickelte Entwurfswerkzeuge in das DASSY-Projekt eingebracht: Bild 3.5. Diese beiden Werkzeuge, der Grafikeditor ICE und der Netzlistenextraktor NESSI, werden im folgenden näher vorgestellt (siehe auch den Anhang zu [54]).

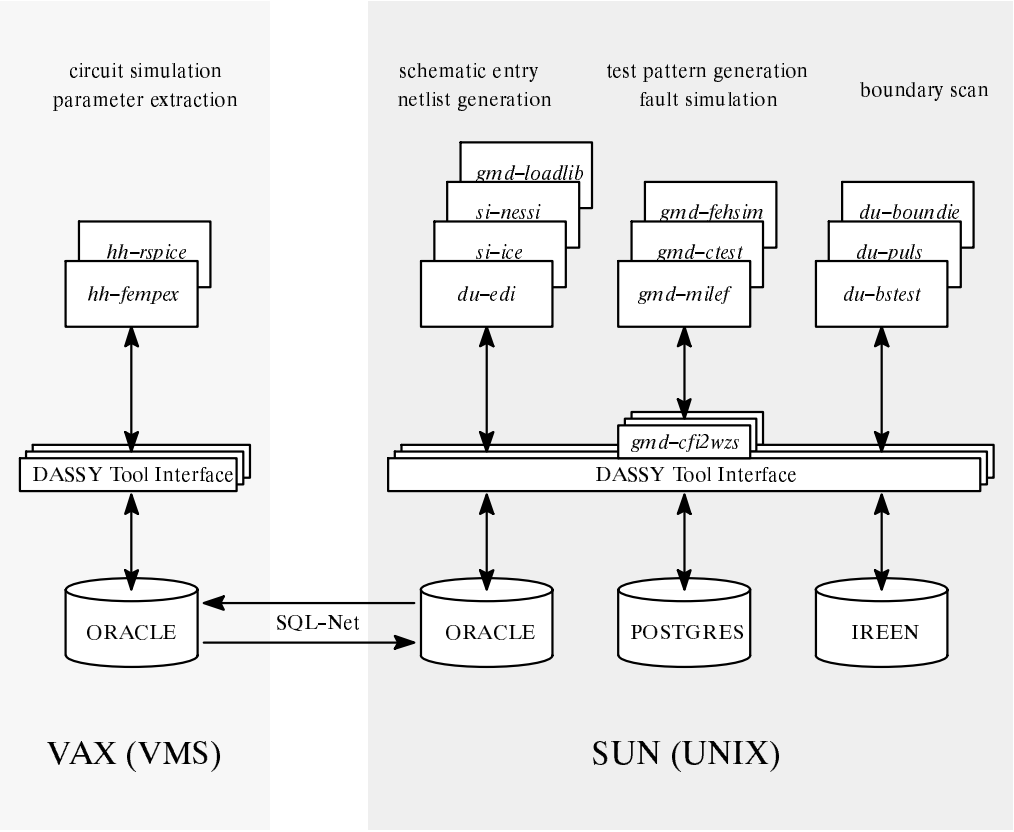

Bild 3.5: DASSY-Prototyp: Werkzeuganpassung

Da beide Werkzeuge im Source-Code vorlagen, wurde in beiden Fällen das Konzept der *White-Box*-Integration angewandt: beide Werkzeuge wurden um Funktionen erweitert, die neben der ursprünglichen externen Datenhaltung im File-System dem Anwender auch eine gleichwertige externe Datenhaltung in einer Datenbank ermöglichen. Für den Zugriff auf die Datenbank wurde, gemäß den oben aufgeführten Zielen, die DASSY-Werkzeugschnittstelle verwendet.

Die Routinen der internen Datenhaltung sowie der Bearbeitung dieser internen Datensätze blieben dagegen unverändert. Da das DASSY-Prototypen-Schema die Datenmodelle beider Werkzeuge vollständig umfaßt, war eine *1-zu-1*-Abbildung zwischen interner und externer Datenhaltung sowohl beim Laden als auch beim Sichern der Daten möglich.

Für den Datentransfer zwischen externer Datenbank und interner Datenhaltung über die Werkzeugschnittstelle wurden drei unterschiedliche Verfahren diskutiert:

- vollständiges Laden bzw. Sichern eines Designs

  Bei diesem Verfahren wird die vollständige 'cell library', d. h. alle zu dem Schaltungsentwurf gehörenden Zellen, beim Programmstart geladen und beim Programmende zurückgeschrieben. Explizite Befehle ermöglichen es dem Benutzer, auch während einer Sitzung Zwischenergebnisse zu sichern bzw. eine andere Schaltung zu laden und weiterzubearbeiten.

  Dieses Verfahren hat den Nachteil, daß der Benutzer beim Programmstart und -ende längere Wartezeiten in Kauf nehmen muß. Zudem werden viele Daten unnötigerweise transferiert.

- bedarfsweises Laden bzw. Sichern einer Zelle

  Beim Programmstart wird nur die vom Benutzer angegebene Zelle (Teilschaltung, Schaltungsmodul) geladen. Während der Bearbeitung dieser Zelle werden, bei Bedarf, nur die darin instanziierten Zellen nachgeladen. Nicht benötigte Zellen (z. B. nicht verwendete Schaltungssymbole aus einer Symbolbibliothek) bleiben unberührt. Lediglich durch die Bearbeitung veränderte Zellen werden beim Programmende in die Datenbank zurückgeschrieben.

  Dieses Verfahren hat, wie auch das zuvor genannte, den Nachteil, daß das Anwendungsprogramm eine eigene, interne Datenverwaltung durchführen muß. Die Datenbank wird wie ein Dateisystem behandelt, die Vorzüge einer zentralen Datenbank mit all ihren Konsistenzsicherungsmechanismen werden nicht genutzt.

- direkter Zugriff auf Zellenkomponenten

  Ein direkter Zugriff auf jedes einzelne, in der Datenbank gespeicherte, Objekt wäre das Ideal einer integrierten Datenhaltung im Sinne der DASSY-Projektziele. Die interne Datenhaltung des Werkzeuges könnte entfallen oder lediglich als cache (zur Beschleunigung des Datenzugriffs) beibehalten werden. Jede Änderung in einer bearbeiteten Schaltung wäre direkt auch in allen anderen Werkzeugen verfügbar, die gleichzeitig dieselbe Zelle bearbeiten. Die Datenbank stellt dabei die Konsistenz der Entwurfsdaten sicher.

  Leider ließ sich ein direkter Zugriff im Rahmen des DASSY-Prototypen nicht realisieren, da das oben vorgestellte Prototypen-Schema nur die Modellierung statischer Daten vorsieht, d. h. Objekte werden nur in einem festen, endgültigen Zustand beschrieben. Von einem spezifischen Werkzeug benötigte temporäre Attribute (wie z. B. ein select flag oder ein already processed flag) werden vom DASSY Datenmodell nicht zur Verfügung gestellt.

Die Umwidmung der internen Datenhaltung zum cache ließ sich leider ebenfalls nicht realisieren, da die Werkzeugschnittstelle dem Anwendungsprogramm keine Information über den Ursprung eines aus der Datenbank gelesenen Objektes mitliefert. Ein während der Bearbeitung verändertes Objekt läßt sich daher in der Datenbank nicht mehr lokalisieren und direkt (write through cache) aktualisieren.

Aus den genannten Gründen wurde in beiden Werkzeugen der Datenbankzugriff über die DASSY-Werkzeugschnittstelle so ausgelegt, daß ein bedarfsweises Laden bzw. Sichern von Zellen durchgeführt wird.

3.4 Exkurs: Algebraische Spezifikation einer Werkzeugschnittstelle

Parallel zum prozeduralen Ansatz für die Modellierung einer Werkzeugschnittstelle wurde alternativ die Methode der algebraischen Spezifikation verfolgt [20]. Die folgenden Ausführungen entstammen den noch laufenden Arbeiten zu dieser Thematik [77].

Ein Schwerpunkt beim Aufbau einer operationalen Schnittstelle liegt in der Erfassung und Beschreibung der Semantik von Standardoperationen, die den Zugriff auf die in der Datenhaltung gespeicherten Daten abwickeln, sowie der Definition einer Operationsklasse für ein abstraktes Datenmodell. Diese Arbeit stellt ein komplexes softwaretechnisches Unterfangen dar. Es ist deshalb notwendig, formale Spezifikationsmethoden zu verwenden, die einerseits die Unabhängigkeit von der Datenrepräsentation gewährleisten, es andererseits aber ermöglichen, zu einem frühen Zeitpunkt einen Prototypen der spezifizierten Schnittstelle zu erzeugen.

Es existieren verschiedene Methoden zur formalen Spezifikation. Für die Spezifikation der Schnittstelle wurde zuerst der algebraische Ansatz verwendet. Darüber hinaus wurden auch Untersuchungen für weitere Spezifikationsmethoden durchgeführt, um festzustellen, inwieweit die Softwareentwicklung mit Hilfe der formalen Spezifikation praktisch ausgeführt werden kann, und inwieweit die Korrektheit der einzelnen Entwicklungsschritte formal verifizierbar ist.

Da durch die formale Spezifikation keinerlei Festlegungen bezüglich der Datenrepräsentation getroffen wird, kann gezeigt werden, daß die Operationen einer Werkzeugschnittstelle auch *top down* — d. h. ohne Kenntnis eines eventuell zugrundeliegenden Datenbanksystems bzw. Datenmodells — entworfen werden können. Wie eine solche Spezifikation in eine effiziente Implementierung überführt wird, wurde im Rahmen dieser Arbeiten behandelt.

Eine formale Beschreibung ist zwingend notwendig, will man unmißverständlich kommunizieren. Sie kann mit Hilfe eines Interpreters (z. B. ACT-ONE) dazu dienen, die Konsistenz der Datenbasis und die Korrektheit der Änderungen durch Manipulationsoperationen unabhängig von der Datenrepräsentation nachzuweisen. Der hier verfolgte algebraische Ansatz konzentriert sich auf die Analyse und Spezifikation der operationalen Schnittstelle zwischen

der Datenhaltung und den Entwurfswerkzeugen im VLSI-Bereich. Im Rahmen der Arbeiten wurde für alle Entwurfsebenen ein Satz von Standardoperationen spezifiziert, mit denen auf die in der Datenhaltung gespeicherten Daten zugegriffen wird.

Bei der Analyse verschiedener VLSI-Werkzeuge stellen sich die folgenden Fragen:

- Welche Informationsstruktur ist für VLSI-Entwurfsumgebungen typisch?
- Welche Operationen mit welcher Funktionalität sind für eine integrierte Entwurfsumgebung erforderlich?
- Welche typischen Probleme entstehen bei der Integration von Werkzeugen?

Aus Zeitgründen wurde die Analyse nur für eine Teilmenge der Entwurfswerkzeuge vorgenommen, doch werden dabei alle wesentlichen Anforderungen des Entwurfsprozesses erfaßt. Als Fallstudien wurden die Anforderungen auf der VLSI-Entwurfsebene „Schematic Entry“ sorgfältig analysiert [63][64][65].

Die gewonnenen Anforderungen wurden in eine formale Spezifikation umgesetzt, repräsentiert durch Signaturen zur Beschreibung der Syntax und durch Axiome zur Festlegung der Semantik. Entsprechend dem algebraischem Ansatz werden Entwurfsobjekte als abstrakte Datentypen aufgefaßt. Die Syntax der zu definierenden Datentypen wird durch Sorten (Datenbereiche) und Operationssymbole formuliert. Die Operationssymbole umfassen dabei alle Möglichkeiten des Benutzers, auf Daten zuzugreifen. Die zugehörigen Gleichungen geben an, welche dieser Zugriffe gleiche Werte liefern. Dadurch wird die Semantik der Operationen definiert. Durch Anwendung des Konstrukts „Kombination“ der Spezifikationssprache ACT-ONE wird es möglich, zu vorhandenen oder an anderer Stelle definierten Spezifikationen weitere Sorten, Operationen und Gleichungen hinzuzunehmen. Dadurch können die Beziehungen zwischen den verschiedenen Entwurfsobjekten explizit formuliert werden. Mit Hilfe des Parametrisierungskonzepts werden viele Basis-Datentypen als dynamische Datentypen definiert, wodurch die Flexibilität der Werkzeugschnittstelle gewährleistet werden kann. Auch die Formulierung von Integritäts -und Konsistenzbedingungen wird durch die Tatsache, daß die Semantik der Operationen durch Axiome unabhängig von einer Implementierung spezifiziert wird, wesentlich erleichtert.

Auf diesen Ergebnissen aufbauend stellte sich die Frage, wie das Konzept der abstrakten Datentypen und die formale Spezifikationsmethode benutzt werden können, um die Integration von Entwurfswerkzeugen zu unterstützen. Es wurde hierzu ein Ansatz vorgeschlagen, bei dem die angestrebte operationale Schnittstelle von der Anwendungsseite, d. h. von den Werkzeugen her untersucht wird.

Es wurden sowohl die für VLSI-Anwendungen geeigneten Datenstrukturen als auch die darauf ausführbaren Operationen formal spezifiziert. Das Ergebnis der Spezifikation bildet eine auf semantischem Niveau in der Mächtigkeit an VLSI-Anwendungen orientierte Schnittstelle, die eine einheitliche Operationsklasse, über die die Werkzeuge auf die Daten zugreifen, zur Verfügung stellt und komplexe Konstruktions-Datentypen erlaubt.

Literaturverzeichnis

Benutzte Literatur

[1] Abel, Elfriede:
Verbundprojekt DASSY.
In: 13. E.I.S.-Zeitung, Hrsg.: Gesellschaft für Mathematik und Datenverarbeitung GMD, Birlinghoven, 1990, S. 7

[2] Bhat, Jayaram; Taku, Fumio:
Frameworks: EDA-Technologie für die 90er Jahre.
In: Elektronik, Jg. 39, H. 21, 1990, S. 88–94

[3] Batory, D. S.; Kim, W.:
Modelling Concepts for VLSI CAD Objects.
In: ACM Transactions on Database Systems, Bd. 10, 1985, S. 322–346

[4] Booch, Grady:
Object Oriented Design with Applications.
The Benjamin/Cummings Publishing Company, Redwood City, 1991

[5] Brachman, Ronald J.; Levesque Hector J. (Hrsg.):
Readings in Knowledge Representation.
Morgan Kaufmann Publishers, San Mateo, Kalifornien, 1985

[6] Brauer, Johannes:
Datenhaltung in VLSI-Entwurfssystemen.
B. G. Teubner, Stuttgart, 1990

[7] Brodie, Michael L.; Mylopoulos John; Schmidt, Joachim W. (Hrsg.):
On Conceptual Modelling.
Springer-Verlag, New York, 1984

[8] Brooks, Frederick P.:
No Silver Bullet: Essence and Accidents of Software Engineering.
In: Computer, Jg. 20, H. 4, 1987, S. 10–19

[9] Chalmers, David L.; Meys, Frans A. C.:
Successful Cad Integration Needs a Standard Conceptual Model.
In: Proc. of the third European EDIF Forum, Bonn/Königswinter, 1989, S. II-6–II-10

[10] Chen, Peter P.:
The Entity-Relationship Model: Toward a Unified View of Data.
In: ACM Transactions on Database Systems, Jg. 1, H. 1, 1976, S. 9–36

[11] Claus, Volker:
Entwicklung von Informatik-Methoden.
In: Handbuch der Modernen Datenverarbeitung, Jg. 26, H. 150, 1989, S. 26–44

[12] Coad, Peter; Yourdon, Edward:
Object-Oriented Analysis.
Prentice Hall, New Jersey, 1991

[13] Coad, Peter; Yourdon, Edward:
Object-Oriented Design.
Prentice Hall, New Jersey, 1991

[14] Daniell, James Donald:
An Object Oriented Approach to CAD Tool Control.
Dissertation, Carnegie Mellon University, Pittsburgh, 1989

[15] Dittrich, Klaus R.:
Objektorientierte Datenbanksysteme.
In: Informatik Spektrum, Springer Verlag, Jg. 12, H. 4, 1989, S. 215–218

[16] EDIF:
Fourth European EDIF Forum: Session I: Information Modelling.
In: Proc. of the 4th European EDIF Forum, Warrington, GB, 1990, S. 3–43

[17] EDIF Steering Committee:
Electronic Design Interchange Format EDIF, Version 2 0 0, EIA/ANSI Standard 548.
Electronic Industries Association, Washington D. C., 1990

[18] EDIF European TSC Information Modelling Working Group:
IMG Information Model of EDIF Version 2 0 0, Version 1.0.
Hrsg.: Wilkes, Wolfgang, FernUniversität Hagen, 1992

[19] EDIF TSC Device Modeling & Verification:
EDIF Device Descriptions, V 2 0 5.
Hrsg.: Blödel, Joachim, Universität Kaiserslautern, 1991

[20] Ehrig, Hans-Dieter; Gogolla, Martin:
Algebraische Spezifikation abstrakter Datentypen.
Teubner-Verlag, 1989

[21] Encarnação, José L.; Lockemann, Peter C.:
Engineering Databases: Connecting Islands of Automation Through Databases.
Springer-Verlag, Berlin Heidelberg, 1990

[22] Gajski, Daniel D.; Kuhn, Robert H.:
Guest Editors' Introduction: New VLSI Tools.
In: Computer, Jg. 16, H. 12, 1983, S. 11–14

[23] Goldfein, Arnold; MAZDRA, Ron; Brevard, Laurence:
Information Model and (Example) Programming Interface.
CAD Framework Initiative Inc., Boulder, CO, 1990

[24] Gottlieb, P.; Kumar, A.; Maffit, R. B.:
EIS Engineering Information Model: EDIF Domain Model.
Electronic Technology Division, Microelectronics Center, 1989

[25] Harel, David:
Biting the Silver Bullet: Toward a Brighter Future for System Development.
In: Computer, Jg. 25, H. 1, 1992, S. 8–20

[26] *Information Modeling Manual IDEF1 - Extended (IDEF1x).*
D. Appleton Company, Manhatten Beach, CA., 1985

[27] Katz, Randy H.
Toward a Unified Framework for Version Modeling in Engineering Databases.
In: ACM Computing Surveys, Jg. 22, H. 4, 1990, S. 375–408

[28] Kim, W.; Lorie, R.; McNabb, D.; Pfouffe, W.:
Nested Transactions for Engineering Databases.
In: IBM Research Reports, RJ 3934, 1983

[29] Kotz, Angelika M.:
Triggermechanismen in Datenbanksystemen.
Springer-Verlag, Berlin Heidelberg, Informatik-Fachberichte 201, 1989

[30] Lipeck, Udo W.:
Dynamische Integrität von Datenbanken.
Springer-Verlag, Berlin Heidelberg, Informatik-Fachberichte 209, 1989

[31] LOCKEMANN, Peter C.:
Konsistenz, Konkurrenz, Persistenz — Grundbegriffe der Informatik?.
In: Informatik Spektrum, Springer Verlag, Jg. 9, 1986, S. 300–305

[32] MEAD, Carver; CONWAY, Lynn:
Introduction to VLSI Systems.
Addison-Wesley Publishing Company, Reading, Mass., 1980

[33] MEERSMAN, R. A.; SERNADAS A. C. (Hrsg.):
Data and Knowledge.
Proceedings of the Second IFIP 2.6 Working Conference on Database Semantics, Elsevier Science Publishers B. V. (North-Holland), Amsterdam, 1988

[34] MEYER, Bertrand:
Object-Oriented Software Construction.
Prentice Hall, 1988

[35] MILLER, Ann:
From Expert Assistant to Design Verification: Applications of AI to VLSI Design.
In: SOUTHEASTCON '89, Energy and Information Technologies in the Southeast, IEEE, New York, 1989, S. 406–410

[36] NACLERIO, Nicholas J.:
Standards for Engineering Information Systems.
In: Proc. of the third European EDIF Forum, Bonn/Königswinter, 1989, S. II-1–II-5

[37] PARNAS, David L.:
On the Criteria to be Used in Decomposing Systems into Modules.
In: Communications of the ACM, Jg. 15, H. 12, 1972, S. 1053–1058

[38] RAMMIG, Franz J.:
Systematischer Entwurf digitaler Systeme.
B. G. Teubner, Stuttgart, 1989

[39] RAMMIG, Franz J. (Hrsg.):
Tool Integration and Design Environments.
Elsevier Science Publishers B. V. (North-Holland), Amsterdam, 1987

[40] RHYNE, Tom:
The MCC CAD Framework Laboratory.
Vortrag auf dem 3. European EDIF Forum, Bonn/Königswinter, 1989

[41] SCHENCK, Douglas:
EXPRESS Language Reference Manual.
McDonnell Aircraft Company, St. Louis, MO., 1990

[42] SCHNEIDER, Birger:
Testing Tomorrow's Technology.
ESPRIT 2318, Elektronik Centralen, Hørsholm, 1991

[43] SCHULZ, Steven E.:
Frameworks: Debunking the Myths.
In: Electronic Design, Jg. 39, H. 16, 1991, S. 71–80

[44] SÉQUIN, Carlo H.:
Managing VLSI Complexity: An Outlook.
In: Proceedings of the IEEE, Jg. 71, H. 1, 1983, S. 149–166

[45] SHAW, Mary:
Abstraction Techniques in Modern Programming Languages.
IEEE Software, Jg. 1, H. 4, 1984, S. 10–26

[46] THOMAS, Donald E.; WALKER, Robert A.:
A Model of Design Representation and Synthesis.
In: Proc. of the 22nd ACM/IEEE Design Automation Conf., 1985, S. 453–459

[47] WAND, Yair:
A Proposal for a Formal Model of Objects.
In: Object-Oriented Concepts, Databases, and Applications, Hrsg.: Kim, Won; Lochowsky, Frederick H., Addison-Wesley Publishing Company, Reading, Mass., 1989, S. 537–559

[48] Weber, Konrad:
Eine Konzeption zur Automatisierung stratifizierter Entwürfe digitaler Systeme basierend auf sukzessiver Detaillierung objektorientierter Modelle.
Dissertation, TU München, 1985

[49] Wilkes, Wolfgang:
Instance Inheritance Mechanisms for Object-Oriented Databases.
In: Int. Workshop on Object-Oriented Databases, Lecture Notes in Computer Science, Springer-Verlag, Bd. 334, 1988

[50] Wirth, Niklas:
Program Development by Stepwise Refinement.
In: Communications of the ACM, Jg. 14, H. 4, 1971

DASSY-B-Hauptdokumente

(Die vollständige Dokumentenliste des DASSY-Projekts wird von der GMD verwaltet.)

[51] DASSY-Projektpartner:
Anforderungen an eine Werzeug-Schnittstelle aus der Sicht der Tools.
Bericht zum 1. DASSY-Meilenstein-Workshop, GMD, St. Augustin, 1990

[52] DASSY-Projektpartner:
Anforderungen an eine Werkzeugschnittstelle aus der Sicht des Entwurfsprozesses.
Bericht zum 1. DASSY-Meilenstein-Workshop, GMD, St. Augustin, 1990

[53] DASSY-Projektpartner:
DaDaMo: The DASSY Data Model (DaDaMo).
Version 2.0, GMD, St. Augustin, 1992

[54] DASSY-Projektpartner:
The DASSY Prototype.
Version 1.0, GMD, St. Augustin, 1992

Veröffentlichungen und interne Berichte (Siegen)

[55] Crusan, Todd; Pyron, Carol; Wahl, Michael:
A Proposed Industry Standard for Test Data Interchange.
Proceedings of First International Workshop on the Economics of Design and Test, Austin, Texas, 1991

[56] EDIF TSC Test:
EDIF Test View, EDIF V 2 0 301.
Hrsg.: Wahl, Michael G., Universität Siegen, 1992

[57] EDIF TSC Test:
Test Description, Version 1.4.10.
Hrsg.: Sebesta, William W; Wahl, Michael G., Universität Siegen, 1992

[58] EDIF TSC Test:
Information Model of the EDIF Test Extension, EDIF V 2 0 32.
Hrsg.: Pyron, Carol; Wahl, Michael G., Universität Siegen, 1991

[59] Pfeifer, Johannes:
Geometrische Transformation im Schema des DASSY-Prototypen.
Interner Bericht, Universität Siegen, FB 12 Technische Informatik, Januar 1992

[60] Pfeifer, Johannes; Quibeldey-Cirkel, Klaus; Zhou, Yuan :
Partielles EDIF-Datenmodell in EXPRESS am Beispiel von „figureGroup/figure“.
Interner Bericht, Universität Siegen, FB 12 Technische Informatik, Februar 1990

[61] PFEIFER, Johannes; WAHL, Michael G.; WOJTKOWIAK, Hans:
Data Exchange Formats for Testing.
Proceedings of EUROMICRO, Köln, 1989 and North Holland, Microprocessing and Microprogramming, H. 27, 1989, S. 687–694

[62] PYRON, Carol; WAHL, Michael:
EDIF Test: The Upcoming Standard for Test Data Transfers.
Proceedings of IEEE International Test Conference, Baltimore, MD, 1992

[63] QUIBELDEY-CIRKEL, Klaus; ZHOU, Yuan :
Untersuchung des Grafikeditors ICE anhand eines Kriterienkatalogs über Zugriffscharakteristika.
Interner Bericht, Universität Siegen, FB 12 Technische Informatik, Februar 1990

[64] QUIBELDEY-CIRKEL, Klaus; ZHOU, Yuan :
Untersuchung des Netzlisten-Extraktors NESSI anhand eines Kriterienkatalogs über Zugriffscharakteristika.
Interner Bericht, Universität Siegen, FB 12 Technische Informatik, Februar 1990

[65] QUIBELDEY-CIRKEL, Klaus:
Anforderungsanalyse der Werkzeugsicht „Schematic Entry".
Interner Bericht, Universität Siegen, FB 12 Technische Informatik, April 1990

[66] QUIBELDEY-CIRKEL, Klaus:
Structure of the EXPRESS Data Dictionary.
Positionspapier, Universität Siegen, FB 12 Technische Informatik, Juni 1990

[67] QUIBELDEY-CIRKEL, Klaus; WAHL, Michael G.:
The Electronic Design Interchange Format EDIF: Capabilities and Evaluation.
In: Proc. of the Int. Conf. on the Factory of the Future to Realise CIM, Intertechno '90, Hrsg: Horváth, Imre; Lehotzky, József, Verlag: Gépipari Tudományos Egyesület, Budapest, 1990, S. 515–524

[68] QUIBELDEY-CIRKEL, Klaus; WAHL, Michael G.:
What's on in the EDIF Test Are(n)a?.
In: Proc. of the 4th European EDIF Forum, Manchester, GB, 1990, S. 92–101

[69] QUIBELDEY-CIRKEL, Klaus; WOJTKOWIAK, Hans:
Modellierung des VLSI-Entwurfsprozesses: Rückschau und Ausblick.
In: Entwurf Integrierter Schaltungen, 5. E.I.S.-Workshop, TU Dresden, Hrsg: Kaesser, Augustin W., Gesellschaft für Mathematik und Datenverarbeitung, GMD-Studie Nr. 188, 1991, S. 23–34

[70] QUIBELDEY-CIRKEL, Klaus:
CAD-Frameworks: Die Probleme jenseits der Werkzeuge.
Interner Bericht, Universität Siegen, FB 12 Technische Informatik, Juli 1992

[71] SEBESTA, William V.; WAHL, Michael:
The State of the EDIF Test Evaluation – Activities of the EDIF Test Technical Subcommittee.
Proceedings of EDIF Forum, Bonn, 1989. S. IV/57–65

[72] SEBESTA, William V.; VERHELST, Bas; WAHL, Michael:
Development of a New Standard for Test.
Proceedings of EDIF World, New Orleans, 1990, and IEEE International Test Conference, Washington, D.C., 1990, S. 988–993

[73] SEJTHEN, Niels-Henrik; PLASSART, Alain; PYRON, Carol; VANDELOO, Paul; VERHELST, Bas; WAHL, Michael:
EDIF as a Standard Test Specification Format.
Proceedings of Fifth European EDIF Forum, La Grande Motte, 1991

[74] VANDELOO, Paul; VERHELST, Bas; WAHL, Michael:
EDIF as a Standard Test Specification Format.
Proceedings of European Test Conference, München, VDE-Verlag, 1991, S. 201–207 und ESPRIT Conference, Brussels, 1991

[75] WAHL, Michael G.:
State of the EDIF Test Extension – A Report about the Activities of the European EDIF Test Working Group.
European EDIF Forum, Amsterdam, 1988 , S. SIM 4–10

[76] Wahl, Michael:
EDIF-Test: Der neue Standard für den Austausch von Testinformationen.
Productronic, Hüthig-Verlag, München, November 1990, S. 64–72

[77] Zhou, Yuan; Brauer, Johannes:
Formal Specification of EDIF Semantics.
Proceedings of Fifth European EDIF Forum, La Grande Motte, 1991