Reprint from Proceedings 3. ITG/GI/GMM Workshop Hardwarebeschreibungssprachen und Modellierungsparadigmen. Holzhau, Germany, February 26–28, 1997

# Objektorientierte Logiksimulation nach dem VITAL Standard

Josef Fleischmann Rolf Schlagenhaft Martin Peller

Lehrstuhl für Rechnergestütztes Entwerfen
Fakultät für Elektrotechnik und Informationstechnik
Technische Universität München, D–80290 München

**Kurzfassung**

*Für den VHDL-basierten Entwurf von Digitalschaltungen wird durch den VITAL Standard erstmals eine einheitliche Zeitmodellierung für Bibliothekszellen und gleichzeitig eine Beschleunigung der Simulation auf Gatterebene möglich. In dem vorliegenden Beitrag wird die Logiksimulation gemäß VITAL anhand der Pilotimplementierung eines objektorientierten ereignisgetriebenen Simulators erläutert. Erste gute Ergebnisse bezüglich der erzielbaren Beschleunigung gegenüber konventionellen VHDL-Simulatoren werden vorgestellt.*

## 1 Einleitung

Der Entwurf digitaler Schaltungen erfolgt aufgrund der wachsenden Automatisierung der Entwurfswerkzeuge und der stark zunehmenden Verwendung von Hardwarebeschreibungssprachen auf immer höheren Abstraktionsebenen. Trotzdem stellt die Logiksimulation der endgültigen Gatternetzliste vor der Fertigung nach wie vor einen unverzichtbaren Verifikationsschritt dar. Die VHDL-basierte Signoff-Simulation wurde bisher dadurch eingeschränkt, daß kein einheitlicher Standard für die Zeitmodellierung und Bibliothekserstellung in VHDL vorhanden war und außerdem die VHDL-Simulation auf Gatterebene im Vergleich zu speziellen Logiksimulatoren sehr langsam war. Dies führte dazu, daß auf der Gatterebene nicht VHDL, sondern zum Teil Verilog oder spezielle Logiksimulatoren eingesetzt wurden.

In der VITAL Initiative (**V**HDL **I**nitiative **T**owards **A**sic **L**ibraries) einigten sich sowohl Halbleiterhersteller als auch Toolentwickler für den ASIC-Entwurf auf den sogenannten VITAL Standard (IEEE 1076.4) [IEE96], der im Dezember 1995 verabschiedet wurde. Ziel dieses Standards ist es, eine ausreichend genaue Modellierung des Zeitverhaltens für die Signoff-Simulation mit VHDL zur Verfügung zu stellen und dabei eine einheitliche Methodik zur Erstellung von ASIC-Bibliotheken zu festzulegen. Gleichzeitig soll hierbei eine Beschleunigung der VHDL-Simulation auf Logikebene ermöglicht werden.

Seit seiner Verabschiedung wird der VITAL-Standard bereits bei einigen kommerziellen VHDL-Simulatoren berücksichtigt. Das VITAL-package an sich ist mit jedem beliebigen VHDL-Simulator verwendbar. Doch eine Beschleunigung des Simulationsablaufs ergibt sich erst dadurch, daß im Simulatorkern selbst VITAL-Optimierungen eingeführt werden. Das Beschleunigungspotential ergibt sich daraus, daß in VITAL-Modellen nur eine eingeschränkte Untermenge von VHDL verwendet werden darf und die Modelle nach strikten Regeln aufgebaut sein müssen.

Unser Ansatz zur Beschleunigung der Simulation von VHDL-Netzlisten auf Gatterebene unterscheidet sich von dem der kommerziellen Anbieter insofern, als wir nicht von einem vollständigen VHDL-Simulationstool ausgehen und versuchen dieses unter Ausnutzung der VITAL-Beschränkungen für die Netzlistensimulation zu beschleunigen. Vielmehr basiert die Entwicklung des neuen Tools OLIVIA (**O**bjectoriented **L**ogicsimulation **I**mplementing the **VI**T**A**L Standard) auf den dedizierten Logiksimulationstools LDSIM [KA90,Kro89] und OSIM [Sch93]. Im folgenden wird kurz auf die Modellierung gemäß dem VITAL-Standard eingegangen. Anschließend werden einige Einzelheiten über die Implementierung des (in Entwicklung befindlichen) objektorientierten Simulators OLIVIA dargestellt. Weiterhin werden erste durch OLIVIA erzielte Ergebnisse mit simulierten Benchmark-Schaltungen vorgestellt und mit kommerziellen VHDL-Simulatoren verglichen. Schließlich werden die geplanten weiteren Arbeiten auf diesem Gebiet kurz angedeutet.

# 2 VITAL Modellierung

Durch den VITAL-Standard wird ein einheitliches Format für die Beschreibung von Bibliothekszellen für die Simulation des Zeitverhaltens in einem VHDL-basierten Entwurfsablauf vorgegeben. Im einzelnen wird hierbei auf folgende Aspekte eingegangen:

- Modellierungsrichtlinien zur Entwicklung von Modellen für ASIC-Zellen und Einschränkung des dabei verwendbaren Sprachumfangs von VHDL
- Timing Routinen zur Beschreibung und Überprüfung des Zeitverhaltens
- Primitive zur Modellierung der Funktionalität
- standardisierte Schnittstelle zur Backannotation mittels SDF (Standard Delay Format)

Aus Abb. 1 wird der prinzipielle Aufbau einer nach VITAL modellierten Bibliothekszelle ersichtlich. Zum einen wird die Schnittstelle und die Bezeichnung der Timing Generics standardisiert, was vor allem eine spätere Backannotation über SDF ermöglicht (Vital Level 0). Zum anderen wird innerhalb der Architecture (Vital Level 1) im einzelnen festgelegt, durch welche Routinen und in welcher Reihenfolge das Zeitverhalten und die Funktion des Modells zu beschreiben sind. Aus dieser strengen Festlegung sowohl der verwendbaren Timing- und Timingcheck-Routinen als auch ihrer logischen Reihenfolge resultiert die Möglichkeit einer effizienten Implementierung zur Beschleunigung der Simulation.

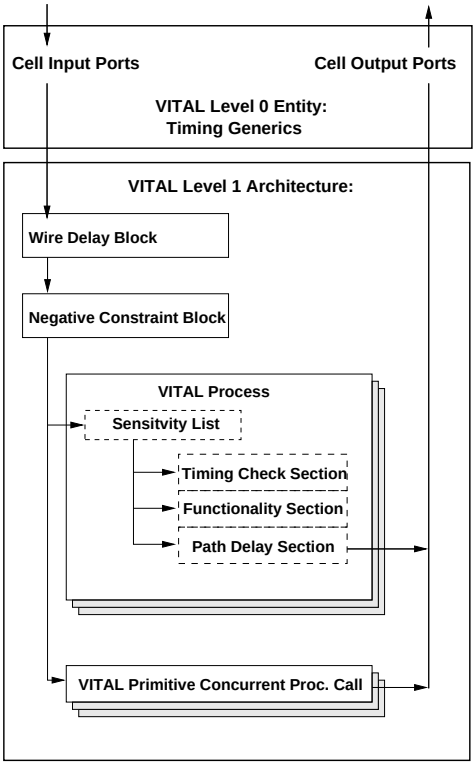


Abbildung 1: Aufbau eines VITAL-Modells [IEE96]

# 3 Implementierung

Als Grundlage zur Implementierung diente der objektorientierte Simulator OSIM [Sch93]. Er enthält Klassen zur ereignisgesteuerten Simulation (Ereignisse, Ereignisverwaltung, Gatterprimitive, Signalmodelle, Ein-, Ausgabe) von Schaltungen nach den in [Kro89] vorgestellten Signal- und Elementmodellen. In OSIM wurde bereits Wert darauf gelegt, Simulationsalgorithmus und Schaltungmodell softwaretechnisch durch objektorientierte Denkweise stark zu entkoppeln. Die während eines Simulationlaufs vorkommenden Objekte lassen sich daher grob in drei Gruppen einteilen:

• **Objekte zur Darstellung der zu simulierenden Schaltung** (Schaltungstopologie): Sie werden dynamisch beim Simulationsstart generiert und stellen die zu simulierende Schaltung dar. Da sich die in VITAL zum Einsatz kommende Schaltungsmodellierung von der in OSIM enthaltenen unterscheidet, konnten die zur Modellierung der Schaltungstopologie vorhandenen Klassen nicht übernommen werden. Einige Basisklassen jedoch waren auch für die Modellierung gemäß VITAL von großem Nutzen.

So konnte zum Beispiel die Basisklasse `SimObj` für alle Komponenten der Schaltungstopologie im Simulator direkt verwendet werden. Sie bildet die Wurzel der mehrstufigen Klassenhierarchie für Topologieobjekte. In `SimObj` sind unter anderem die Methoden zum dynamischen Aufbau des Schaltungsgraphen vor der Simulation und zur Kommunikation zwischen den Komponenten während der Simulation realisiert. Es besteht eine sehr starke Ähnlichkeit zwischen dem natürlichen Signalfluß in einer Schaltung und der Realisierung im Simulator. Diese Methoden werden direkt an die abgeleiteten Schal-

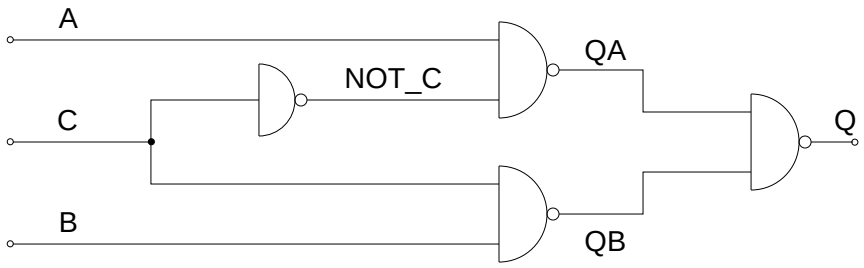


Abbildung 2: Beispielschaltung

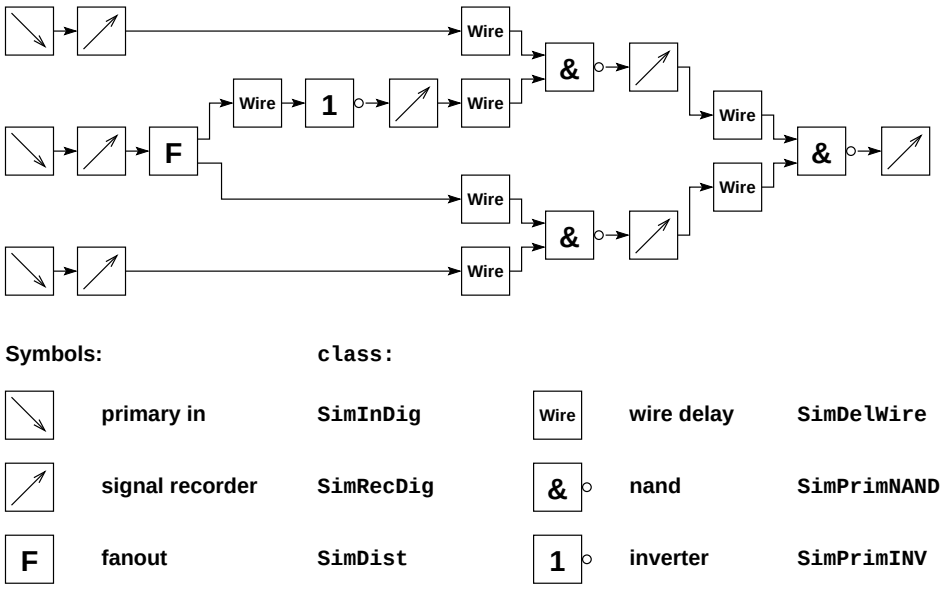


Abbildung 3: Darstellung im Simulator

tungskomponenten vererbt. Die Methode für die Modellauswertung einer Schaltungskomponente wird dagegen in `SimObj` nur virtuell deklariert, da je nach Typ der Komponente verschiedene Aufgaben durchzuführen sind. Für zeitbehaftete Vorgänge können manche Komponenten zum Beispiel Ereignisse erzeugen und später ausführen. Durch Polymorphismus bleibt die Funktionalität der verschiedenen Komponenten vollständig in ihnen selbst verborgen und spielt für die Ereignisverwaltung und den Schaltungsaufbau keine Rolle.

Beispielhaft seien hier einige Klassen aufgeführt, die direkt oder über einige Hierarchiestufen von `SimObj` abgeleitet sind: In OLIVIA gibt es Komponenten für Leitungslaufzeiten (`SimDelWire`), Logikgatter (z.B. `SimPrimAND`) und Zustandstabellen (`SimPrimStateTable`), um nur einige zu nennen. Aufgaben innerhalb von Komponenten werden teilweise auch mittels Aggregation (im Gegensatz zu Vererbung) realisiert, wie z.B. Setup- und Holdzeit-Check (`VitalSetupHoldCheck`) und Auswahl der zutreffenden Path Delays (`VitalPathDelay01`).

Abb. 2 und 3 zeigen, wie eine Schaltung durch Komponenten (Objekte von `SimObj` abgeleitet) im Simulator repräsentiert wird. Jedes Quadrat stellt eine `SimObj`-Instanz

dar. Neben den bereits erwähnten Objekten für Leitungslaufzeiten und Logikgatter sind solche zur Stimulation und Aufzeichnung von Signalen und zur Darstellung des Fanouts zu erkennen.

• **Ereignis-Objekte**: In einem ereignisgetriebenen Simulator wird jegliche zeitbehaftete Veränderung in der simulierten Schaltung durch Ereignis-Objekte repräsentiert. Diese werden während der Simulation laufend dynamisch erzeugt, später bearbeitet und schließlich wieder gelöscht. Die verschiedenen Ereignistypen (Primäreingangsänderung, interne Signalveränderung, Ereignisse zur Simulatorsteuerung, ...) bilden ihrerseits eine kleine Klassenhierarchie. In deren Basisklasse sind ausschließlich die Methoden realisiert, die nötig sind, damit die Ereignisverwaltung die Ereignisse aufnehmen und zum Ausführungszeitpunkt an den Simulator übergeben kann. Durch Polymorphismus ist es möglich, daß jedes Ereignis dann bei der Ausführung seine typspezifischen Aufgaben erledigt.

Für die Simulation nach VITAL sind unter anderem aufgrund der andersartigen Signalmodellierung andere Ereignistypen notwendig als für OSIM. Durch Ableitung der neuen Typen von bereits in OSIM vorhandenen abstrakten Basisklassen konnten jedoch große Teile der Ereignisverwaltung direkt verwendet werden.

Bezogen auf Ereignisse besitzt OLIVIA einen Unterschied zu anderen VHDL-Simulatoren. Es kommt nicht der herkömmliche Preemptionsmechanismus von VHDL zum Einsatz, sondern eine Alternative, die im Hinblick auf eine spätere Parallelisierung des Simulators von Vorteil ist. Statt vorige, veraltete Ereignisse beim Eintragen eines neuen Ereignisses in die Ereignisliste zu verdrängen (Preemption), werden diese in OLIVIA erst bei deren Ausführung ignoriert. Dazu ist es allerdings notwendig, zusätzlich die Generierungszeit $t_{Gen}$ eines Ereignisses abzuspeichern. Im VHDL Standard gibt es prinzipiell die beiden Verzögerungsmodelle `TRANSPORT` und `INERTIAL`, die durch unterschiedlich arbeitende Preemptionsmechanismen realisiert sind. Dementsprechend wurden bei OLIVIA auch zwei verschiedene Methoden zum Ignorieren von veralteten Ereignissen implementiert. Welche Methode zur Anwendung kommt, entscheidet der VHDL Verzögerungstyp (`INERTIAL` oder `TRANSPORT`), der ebenfalls in OLIVIA Ereignissen abgelegt ist. Diese veränderte Vorgehensweise hat keinen Einfluß auf die Ergebnisse der Simulation.

• **Objekte, die den Simulationsalgorithmus realisieren**: Sie bilden den eigentlichen Simulator und sind zum Großteil statisch. Die beiden wichtigsten unter ihnen sind das Objekt, das die Schaltungstopologie aufbaut und verwaltet, und das Objekt zur Ereignisverwaltung. Diese Objekte stellen die wichtigsten Grundfunktionen für den Simulationszyklus zur Verfügung und sie bleiben während der gesamten Dauer der Simulation bestehen. Das Zusammenspiel zwischen ihnen stellt den eigentlichen Simulatorkern dar. Trotz der Tatsache, daß für die VHDL Simulation ein etwas anderer Simulationsablauf implementiert werden mußte, konnte an dieser Stelle ein großer Teil der in OSIM verwendeten Klassen wieder eingesetzt werden.

Insgesamt läßt sich sagen, daß die Wahl des objektorientierten Ansatzes für OSIM sich bei der Implementierung von OLIVIA als sehr nützlich erwiesen hat. Er brachte sowohl Vorteile bei der Code-Wiederverwendung, als auch bei der Code-Lesbarkeit; dies brachte eine erhebliche Beschleunigung bei der Implementierung des neuen Tools mit sich.

# 4 Ergebnisse

Um die korrekte Funktion des neuen Simulators OLIVIA zu überprüfen und seine Leistungsfähigkeit im Vergleich zu kommerziellen VHDL-Simulationssystemen beurteilen zu können, wurden verschiedene Testschaltungen simuliert. Es wurden hierbei drei kombinatorische Schaltungen aus dem LGSYNT91 Benchmark Set (C17,ALU2 und C6288) und eine lokal entwickelte sequentielle Schaltung (ADD4) verwendet. Die Versuche wurden in erster Linie auf der Rechnerplattform DEC ALPHA 250 4/266 durchgeführt. Zur Stimulation der Beispielschaltungen wurden jeweils Zufallsmuster eingesetzt. Die in Abb. 4 und 5 angegebenen Simulatorlaufzeiten stellen jeweils Durchschnittswerte aus fünf Versuchswiederholungen dar.

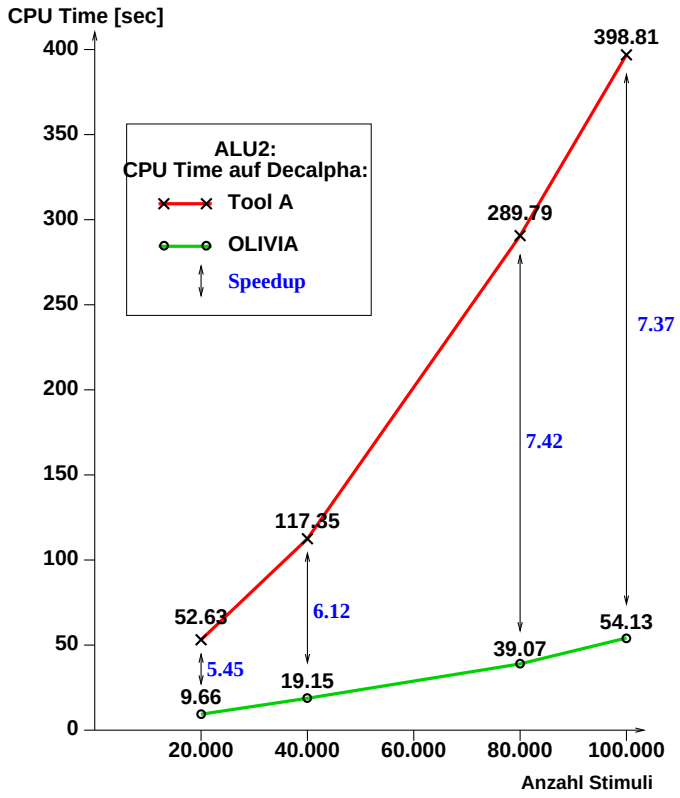


Abbildung 4: Simulationslaufzeiten für verschiedene Anzahl von Eingangsereignissen

Zunächst wurde die Abhängigkeit der Simulatorlaufzeit von der Anzahl der Eingangsereignisse untersucht. Wie in Abb. 4 am Beispiel der ALU2 ersichtlich, läßt sich näherungsweise ein linearer Zusammenhang beobachten. Unsere Versuche haben gezeigt, daß mit zunehmender Schaltungsgröße und Zahl der Stimuli eine Krümmung zu beobachten ist, welche möglicherweise durch Nebeneffekte zu erklären sind, die aus der Größe ($>$ 20 MB) der produzierten Ergebnisdateien resultieren.

In Abb. 4 sind außerdem bereits die Geschwindigkeitsunterschiede zwischen dem Simulator OLIVIA und dem kommerziellen TOOL A[1] zu erkennen: Mit dem Tool OLIVIA

[1]Aus lizenzrechtlichen Gründen wird auf die Angabe der genauen Produktbezeichnung beim Vergleich mit kommerziellen VHDL-Simulatoren verzichtet

wurde bei der sequentiellen Schaltung ALU2 je nach Anzahl der Stimuli eine Erhöhung der Simulationsgeschwindigkeit um das fünf- bis siebenfache erzielt.

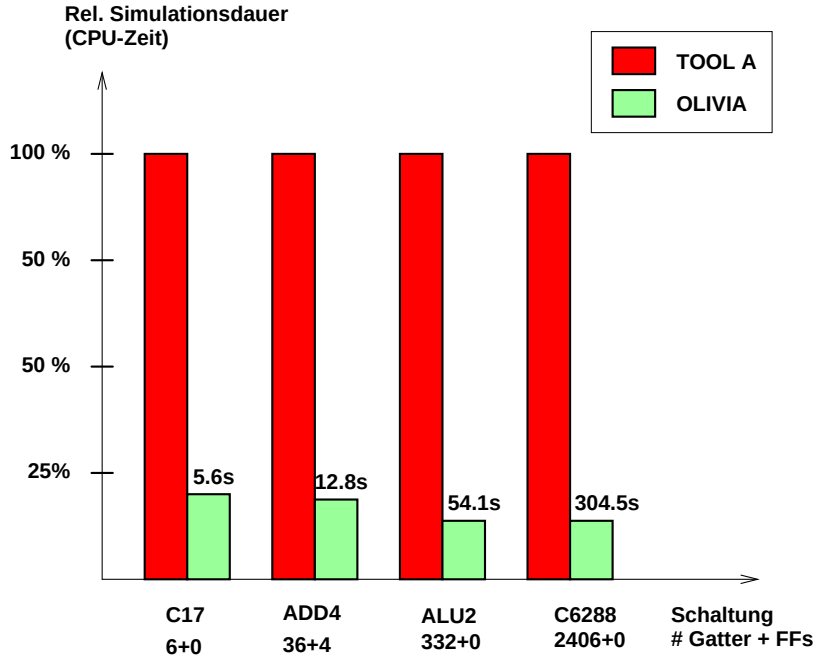


Abbildung 5: Simulationslaufzeiten

Im Überblick sind unsere ersten Ergebnisse mit den derzeit simulierten Benchmark-Schaltungen in Abb. 5 wiedergegeben: Es zeigt sich ein sehr beachtlicher Speedup für alle bisher simulierten Schaltungen. Der Geschwindigkeitsvorsprung von OLIVIA scheint außerdem mit zunehmender Schaltungsgröße und einer zunehmenden Anzahl von Stimuli eher noch zuzunehmen. Um dies noch genauer zu untersuchen, wird derzeit daran gearbeitet, noch weitere Simulationen mit größeren Benchmark-Schaltungen durchzuführen. Weiterhin werden derzeit noch Vergleiche mit optimierten kommerziellen VHDL/VITAL-Simulatoren durchgeführt, darunter auch der derzeit schnellste VHDL-Simulator. Hier wurde ebenfalls eine Steigerung der Simulationsgeschwindigkeit um das zwei- bis dreifache erzielt. Die genaue Ursache für diesen Fortschritt läßt sich nicht exakt feststellen. Ein möglicher Grund ist, daß bei unserem Ansatz nicht ein bestehender VHDL-Simulator ergänzt wurde, sondern die durch Verwendung von VITAL möglichen Optimierungen direkt im Simulationskern eines neuen Simulators berücksichtigt wurden.

# 5 Zusammenfassung und Ausblick

In diesem Beitrag wird zum ersten Mal die Neuimplementierung des VITAL Standards in einem objektorientierten ereignisgetriebenen Simulators für die Simulation des Zeitverhaltens auf Gatterebene vorgestellt. Der Simulator OLIVIA wird derzeit für den Einsatz in einem VLSI-Entwurfspraktikum entwickelt und zeigt folglich gewisse Einschränkungen im Vergleich zu einem kommerziellen Simulationstool. So wurde zum Beispiel die Bibliothek der benötigten VITAL-Modelle (einfache logische Primitive, Flip-Flops) direkt

im Simulator implementiert; das Einlesen einer beliebigen VITAL-Bibliothek ist derzeit noch nicht möglich. Einige VITAL Konstrukte, wie z.B. *Path Conditions* zur Angabe mehrerer Verzögerungszeiten pro Pfad sowie *VitalResolve* zur Behandlung von Bussignalen, wurden beim konzeptionellen Entwurf berücksichtigt, werden aber derzeit noch nicht eingesetzt. Für die bisher untersuchten Schaltungen zeigt der Simulator OLIVIA beachtliche Geschwindigkeitsvorteile gegenüber konventionellen VHDL/VITAL-Simulatoren bei identischen Simulationsergebnissen.

Laufende und geplante Arbeiten beschäftigen sich mit folgenden Themen:

- Evaluierung der Performance und Vergleich mit kommerziellen VITAL-optimierten Simulatoren unter Einsatz größerer Benchmark-Schaltungen
- Weiterentwicklung des Frontends, um das Einlesen von hierarchischen strukturellen VHDL-Beschreibungen zu ermöglichen
- Implementierung von Methoden zur SDF-Backannotation von layoutabhängigen Verzögerungsdaten

Des weiteren wurde während der Implementierung von OLIVIA bereits ein besonderes Augenmerk auf die spätere Parallelisierbarkeit gelegt. Ein Parallelisierungskonzept für die parallele VITAL-Simulation auf einem Workstation-Netzwerk ist bereits angedacht und stützt sich auf die Erfahrungen, die bei der Parallelisierung von OSIM zu OPSIM gemacht wurden [Gan94].

# Literatur


[Gan94] GANZ, ANDREAS: *Objektorientierte verteilte Simulation*. Diplomarbeit, Lehrstuhl für Rechnergestütztes Entwerfen, Technische Universität München, 1994.

[IEE96] IEEE STANDARDS BOARD: *IEEE Standard for VITAL Application-Specific Integrated Circuit (ASIC) Modeling Specification*, 1996. IEEE 1076.4-1995.

[KA90] KRODEL, H. T. and K. J. ANTREICH: *An accurate model for ambiguity delay simulation*. In *European Conference on Design Automation (EDAC)*, pages 563–567, Glasgow, 1990.

[Kro89] KRODEL, HANS THOMAS: *Verfahren zur Logiksimulation komplexer digitaler Schaltungen mit flexibler Modellierung*. Doktorarbeit, Lehrstuhl für Rechnergestütztes Entwerfen, Technische Universität München, 1989.

[Lev95] LEVIA, OZ: *Introduction to asic cells modeling with vital*. In *VHDL Forum for CAD in Europe*, April 1995. Tutorial.

[Pel96] PELLER, MARTIN: *Logiksimulation mit VITAL*. Diplomarbeit, Lehrstuhl für Rechnergestütztes Entwerfen, Technische Universität München, 1996.

[Sch93] SCHLAGENHAFT, ROLF: *Objektorientierter Simulator für Logikschaltungen*. Diplomarbeit, Lehrstuhl für Rechnergestütztes Entwerfen, Technische Universität München, 1993.

[Sch95] SCHULZ, STEVEN E.: *Introduction to vital '95*. In *Mentor User's Group Meeting*, 1995. Tutorial Presentation.