Ein Framework zur verteilten objektorientierten Simulation heterogener Kommunikationssysteme

U. Hatnik, J. Haufe, P. Schwarz*
Fraunhofer Gesellschaft
uwe.hatnik@eas.iis.fhg.de

R. G. Spallek†
Technische Universität Dresden
rgs@ite.inf.tu-dresden.de

Zusammenfassung

Es wird ein Framework für die objektorientierte Simulation bei der Entwicklung komplexer Kommunikationssysteme vorgestellt. Es gestattet die Kopplung verschiedener Simulatoren sowie deren Parallelisierung. Als Schnittstelle wurde der IEEE-Standard OMI (Open Model Interface) eingesetzt. Für die Kommunikation der Objekte untereinander wurden TCP/IP-Sockets, PVM, CORBA und HLA untersucht.

1 Einleitung

Bei der Entwicklung und Optimierung komplexer Kommunikationssysteme spielt die Simulation eine wichtige Rolle und ihre Bedeutung wird in Zukunft weiterhin zunehmen. Dies ist auf die stark steigende Komplexität der Systeme zurückzuführen, wodurch Erfahrungswerte und grobe Abschätzungen der Leistungsmerkmale nicht mehr ausreichen. Beim VLSI-Entwurf, also bei der Entwicklung hochintegrierter Schaltkreise, ist eine simulative Entwurfsunterstützung schon lange unverzichtbar. Auch die Projektierung von Kommunikationsnetzwerken wird heute bereits durch Simulatoren unterstützt, wobei vergleichsweise abstrakte Modelle genutzt werden.
Mit den heute verbreiteten Simulatoren ist es jedoch kaum bzw. nur mit sehr hohem Aufwand möglich, detaillierte Modelle in eine Gesamtsystemsimulation einzubinden. Dies ist durch die Spezialisierung der Simulatoren auf ein bestimmtes Einsatzgebiet bedingt. Beispielsweise nutzen Netzwerk- und Schaltungssimulatoren unterschiedliche Modellierungssprachen, Modellbibliotheken und Analysewerkzeuge.
Die zwei Sichtweisen, abstraktes Gesamtsystem und detaillierte Komponenten, lassen sich heute nur sehr schwer in einer Simulationsumgebung vereinen. Gerade dies ist jedoch von wachsender Bedeutung, da aufgrund der hohen Komplexität moderner Systeme eine separate Betrachtung der einzelnen Komponenten nicht mehr ausreicht. Eine Gesamtsystemsimulation mit Modellen unterschiedlicher Abstraktionsgrade würde nicht nur präzisere Untersuchungen des Systems erlauben, sondern könnte auch als leistungsfähige Testbench bei der Entwicklung einzelner Komponenten dienen.

*Fraunhofer-Institut für Integierte Schaltungen, D-01069 Dresden, Germany
†Fakultät Informatik, D-01069 Dresden, Germany

Aus diesem Grund wurde ein Konzept zur Simulation von Kommunikationssystemen entwickelt, das es erlaubt, Modelle unterschiedlicher Abstraktionsgrade sowie verschiedene Modellierungssprachen und Simulationsalgorithmen zu kombinieren. Das Konzept basiert auf einem objektorientierten Ansatz und ermöglicht die Kombination verschiedener Modellierungssprachen und Simulatoren [HA04, HHS01, Hat05]. Letztere können auf mehrere Rechner verteilt werden und parallel arbeiten, wenn die genutzten Modelle und Algorithmen dies erlauben. Dadurch spielen auch Plattformabhängigkeiten eine untergeordnete Rolle, was ein wichtiger Vorteil des vorgestellten Konzeptes ist. Die durch eine Parallelsimulation eventuell erreichte Simulationsbeschleunigung ist zwar vorteilhaft, jedoch nicht das primäre Ziel. Im Vordergrund stehen hingegen folgende Schwerpunkte:

- Kombination von Modellen unterschiedlicher Abstraktionsgrade
- Offenheit gegenüber unterschiedlichen Modellierungssprachen und Simulatoren
- Nutzung ereignis- und zeitgesteuerter Simulationsverfahren
- Flexibilität bezüglich der einsetzbaren Simulationsalgorithmen
- Wiederverwendung von existierenden Modellen
- Möglichst einfache Einbindung etablierter Simulatoren
- Integration von realen Hardware- und Software-Komponenten
- Verteilte Simulation (Netzwerkkopplung)
- Variable Kommunikationsmöglichkeiten
- Kombination mehrerer Plattformen
- Hohe Flexibilität bei geringer Komplexität

Um diese Eigenschaften auch praktisch realisieren zu können, wurde ein objektorientierter Ansatz gewählt, der im folgenden näher erläutert wird.

2 Der objektorientierte Ansatz

Der objektorientierte Ansatz ermöglicht die Zusammenstellung eines Simulationsszenarios aus mehreren Objekten. Jedes Objekt entspricht dabei einem Teilsystem und besteht aus dem entsprechenden Modell, dem zu dessen Simulation erforderlichen Simulator und einer Objektschnittstelle. Ein Objekt kann weitere Objekte instantiieren, die es selbst verwalten und steuern muß. Dadurch bildet sich eine Objekthierarchie, in der jedes Objekt immer nur die selbst erzeugten Objekte kennt und von genau einem übergeordneten Objekt unmittelbar gesteuert wird, siehe Abbildung 1. Der verfolgte Ansatz ermöglicht die Ausnutzung aller objektorientierten Basiskonzepte, erzwingt aber keine objektorientierte Implementierung der Objekte[1]. Dadurch lassen sich bereits existierende Simulatoren einbinden, auch wenn sie nur als ausführbares Programm vorliegen, wie es bei kommerziellen Simulatoren meist der Fall ist.

[1] Durch die Kapselung des gesamten Simulators eines Objektes können alle objektorientierten Konzepte außer Klasse und Vererbung auch in diesem Fall ausgenutzt werden.

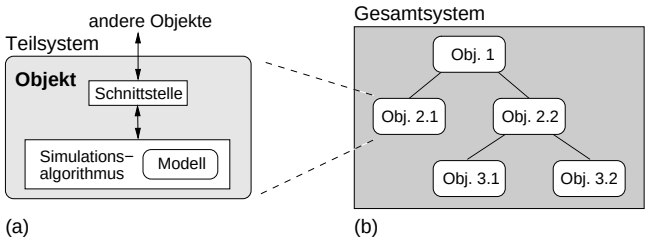

Abbildung 1: Struktur eines Objektes (a) und Objekt-Hierarchie (b)

Es wurden vier Objektvarianten untersucht, dazu zählen der bereits erwähnte objektinterne Simulationsalgorithmus, die Kopplung mit externen Simulatoren sowie die Einbeziehung externer Softwarekomponenten und realer Hardware.
Der vorgestellte objektorientierte Simulationsansatz bietet folgende Vorteile:

- Für jedes Teilsystem kann die am besten geeignete Modellierungssprache genutzt werden. Dies kann eine objektorientierte Sprache sein, muß es jedoch nicht.
- Der Simulationsalgorithmus eines Objektes kann optimal an das Modell angepaßt werden, da er nicht universell (für alle Objekte) eingesetzt werden muß.
- Objekte können leicht angepaßt, ausgetauscht und wiederverwendet werden, da die Implementierung gekapselt wird.
- Bereits existierende Modelle können leicht in eine Simulationsumgebung eingebunden werden, sofern ein Simulator inklusive einer Schnittstelle zur Verfügung steht.
- Objekte können externe Simulatoren, reale Softwarekomponenten des Systems und reale Hardware-Komponenten kapseln und diese somit in die Simulation integrieren.
- Ein Objekt bzw. sein Modell kann bei Bedarf partitioniert werden, so daß aus den Partitionen neue Objekte entstehen, die jeweils die geeignetsten Abstraktionsgrade, Beschreibungssprachen und Simulatoren nutzen können.
- Objekte können verschiedene Plattformen nutzen und über ein Netzwerk gekoppelt werden, wobei verschiedene Realisierungsvarianten der Objekt-Kommunikation einsetzbar sind.
- Die Realisierung einer verteilten Simulation ist aufgrund der autonomen Objekte leicht möglich.
- Die Objektstruktur eines generierten Szenarios ermöglicht eine gute Überschaubarkeit und relativ geringe Komplexität der Simulationsumgebung.

Der objektorientierte Ansatz wird durch eine Objekt-Schnittstelle und verschiedene Kommunikationsvarianten ergänzt, die in den folgenden Abschnitten kurz vorgestellt werden.

3 Die Objekt-Schnittstelle

Um den vielen existierenden und meist proprietären Schnittstellen keine weitere hinzuzufügen, wurde das standardisierte *Open Model Interface* (kurz OMI) [IEE98] untersucht und aufgrund seiner Eignung als Objektschnittstelle eingesetzt. Dabei handelt es sich um eine Schnittstelle, die prinzipiell dem objektorientierten Ansatz folgt und durch den lEEE-Standard 1499 spezifiziert wird. Das OMI ist zur Kopplung von Logiksimulatoren (VLSI-Entwurf) mit funktionalen Modellen gedacht, d.h. mit direkt ausführbaren Algorithmen. Als Ergebnis der durchgeführten Untersuchungen zeigte sich, daß es sich aufgrund seiner Struktur auch sehr gut für den hier vorgestellten Ansatz eignet. Gewöhnliche Modell-

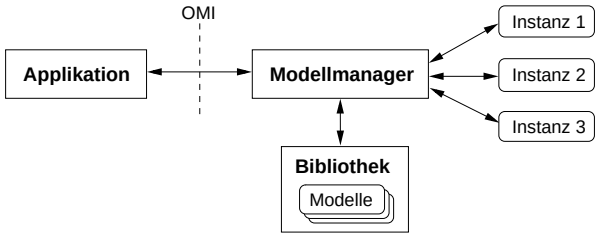

Abbildung 2: Grundstruktur des OMI

schnittstellen realisieren eine direkte Kopplung von Modell und Simulator. Im Gegensatz dazu sieht die Spezifikation des OMI eine zusätzliche Instanz zwischen Modell und Simulator, den OMI-Modellmanager, vor. Abbildung 2 verdeutlicht diese Grundstruktur. Die OMI-Spezifikation definiert die Schnittstelle zwischen Applikation (Simulator) und Modellmanager, jedoch nicht die Kopplung von Modellmanager und den Modellen, Modellbibliotheken und Modellinstanzen. Diese Eigenschaft paßt sehr gut in das verfolgte Konzept, da somit eine Adaption unterschiedlicher Schnittstellen möglich ist. Im Rahmen des vorgestellten Ansatzes entsprechen die Instanzen sowie die OMI-Applikation den Objekten.
Der Modellmanager verwaltet die verfügbaren Objekte sowie deren Kopplung. Bei Bedarf kann er weitere Funktionen übernehmen, z. B. eine Konvertierung von Datentypen, die Anpassung von Simulationsalgorithmen oder die Bereitstellung von Modellinformationen. Die oftmals rudimentäre Funktionalität von proprietären Simulator-Schnittstellen läßt sich so ergänzen, ohne daß dies für die OMI-Applikation sichtbar ist, was dem erwünschten Geheimnisprinzip entspricht. Abbildung 3 zeigt eine mögliche Strukturvariante, die auch im entwickelten Framework implementiert wurde.

4 Die Objekt-Kommunikation

Der genutzte OMI-Standard geht davon aus, daß die Modellfunktionen als Objektcode oder Bibliothek eingebunden und direkt aufgerufen werden. Dies setzt aber voraus, daß alle Prozesse in einem gemeinsamen Prozeßraum laufen. Eine verteilte Simulation ist so jedoch nicht möglich und wird in der OMI-Spezifikation auch nicht erwähnt.

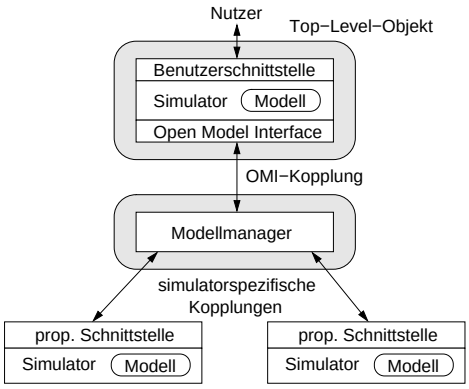

Abbildung 3: Eine mögliche Strukturvariante

Aufgrund des flexiblen Ansatzes, speziell der zugrundeliegenden Struktur, ist dennoch eine verteilte Simulation realisierbar. Der Modellmanager kann geeignete Kommunikationsmechanismen nutzen, um mit entfernten Objekten über ein Netzwerk zu kommunizieren, ohne daß dies für die OMI-Schnittstelle eine Rolle spielt. Die Applikation kann alle OMI-Funktionen nutzen, wobei die Funktionsaufrufe vom Modellmanager in geeigneter Weise umgesetzt und über ein Netzwerk übertragen werden. Diese Netzwerkkopplung bleibt der Applikation völlig verborgen.
Der erforderliche Aufwand und die Effizienz einer solchen Kopplung hängen nicht nur von der genutzten Software (z. B. TCP/IP-Sockets, PVM, CORBA, HLA), sondern auch von der Struktur der Kopplung ab. Dies ist darauf zurückzuführen, daß die Applikation direkt, d. h. ohne ein dazwischenliegendes Netzwerk, mit dem Modellmanager kommunizieren kann. Der Modellmanager braucht nur die Funktionen und Daten über ein Netzwerk weiterzuleiten, die tatsächlich von den Instanzen benötigt werden. Alle anderen Funktionen werden lokal im Modellmanager behandelt. Weiterhin kann er die Kopplung über das Netzwerk optimieren, z. B. indem er alle Eingangsdaten einer Instanz bei der Applikation erfragt und dann gemeinsam zur Instanz überträgt. Aufgrund dieser Vorteile kam diese Variante im implementierten Simulationsframework zum Einsatz.
Bezüglich der Objekt-Kommunikation wurden vier Varianten untersucht, basierend auf TCP/IP-Sockets, der *Parallel Virtual Machine*, der *Common Object Request Broker Architecture* und der *High Level Architecture*. Diese Varianten werden genutzt, um den Modellmanager mit den Modellinstanzen (Objekten) zu koppeln. Entsprechend dem OMI-Ansatz wird dazu eine weitere, selbst entwickelte Schnittstelle genutzt, wobei der Modellmanager die Anpassung zum OMI übernimmt.
Der höhere Komfort von PVM sowie des CORBA- und HLA-Systems sind gegenüber der Socket-Schnittstelle mit einem internen Overhead verbunden, z. B. durch die Anpassung der Zahlendarstellung, die RPC-Funktionalität und die transparente Kommunikation. Wie sich zeigte, fällt dieser jedoch geringer aus als erwartet, so daß die Untersuchung der Kommunikationsvarianten unter anderem zu folgenden Ergebnissen führte.

Im Performance-Vergleich ist die HLA-Kopplung die langsamste Variante. Da die Vorteile der HLA in dem verfolgten Ansatz nicht nutzbar sind, besteht kein Grund, diese zu favorisieren. Eine Ausnahme wäre die Einbeziehung einer bereits existierenden HLA-Simulation in eine Simulationsumgebung entsprechend des hier verfolgten Ansatzes.
Das CORBA-System erweist sich als vorteilhaft, wenn für die Objekt-Kommunikation ein möglichst geringer Implementationsaufwand betrieben werden soll. Dabei sind jedoch die in der vorgelegten Arbeit näher betrachteten Eigenschaften der Client/Server-Architektur zu berücksichtigen. Auch die leichte Einbeziehung bereits existierender Server-Objekte ist als Vorteil dieser Variante zu bedenken.
Wenn CORBA's Client/Server-Architektur eine unerwünschte Einschränkung darstellt, der Implementationsaufwand der Socket-Kopplung jedoch vermieden werden soll, bietet die PVM-Variante einen guten Kompromiß. Sie ermöglicht bei mittlerem Aufwand eine plattformunabhängige Kopplung ohne sonstige Nachteile.
Die Socket-Variante sollte genutzt werden, wenn kein PVM- oder CORBA-System verfügbar ist oder genutzt werden soll (z. B. aus Sicherheits- oder Kostengründen). Der hohe Implementationsaufwand gegenüber den anderen Varianten relativiert sich aufgrund der Wiederverwendbarkeit. Desweiteren hat der Entwickler die volle Kontrolle über die selbst realisierte Funktionalität, was mitunter von Vorteil ist. Auch wenn seltene Plattformen genutzt werden sollen, ist die Socket-Variante vorzuziehen, da eine derartige Schnittstelle von jedem modernen Betriebssystem bereitgestellt wird.
Als zusammenfassendes Ergebnis kann gesagt werden, daß bei der Auswahl einer Kommunikationsvariante weniger die Performance, sondern vielmehr die effektivste und in den jeweiligen Rahmenbedingungen geeignetste Lösung gewählt werden kann. Die vom Modellmanager realisierte Anpassung der Schnittstellen kann bezüglich der Performance vernachlässigt werden, da die Netzwerkkopplung deutlich langsamer ist.

5 Ein experimentelles Framework

Zur Überprüfung der Erwartungen wurde ein prototypisches Framework entwickelt, das die Simulatoren NS-2 [Uni], Ptolemy Classic [BHLM94] und ModelSim [Men01] nutzt. Darüber hinaus wurde reale Systemsoftware und ein FPGA-Board eingebunden. Das Ergebnis ist ein leistungsfähiges Framework, das bereits eine Vielzahl von möglichen Anwendungsfällen abdeckt und sich leicht erweitern läßt. Abbildung 4 zeigt die Struktur des experimentellen Frameworks. Wie bereits erläutert wurde, befindet sich das *Open Model Interface* zwischen dem Top-Level-Objekt (OMI-Applikation) und dem Modellmanager. Das Top-Level-Objekt steuert demzufolge die mit dem Modellmanager gekoppelten Objekte, wobei die Ereignisausführung mit Hilfe eines konservativen Simulationsalgorithmus synchronisiert wird [Hat05]. Entsprechend dem OMI-Ansatz ist die Schnittstelle zwischen dem Modellmanager und den OMI-Instanzen nicht vom Standard vorgeschrieben. Daher kommt an dieser Stelle eine selbst entwickelte Schnittstelle zum Einsatz. Zur Objekt-Kommunikation können die vorgestellten Varianten kombiniert werden.
Als Top-Level-Simulatoren wurden das Framework Ptolemy Classic und der Netzwerksimulator NS-2 gewählt, da diese mit ihren Eigenschaften und Funktionsumfängen (Biblio-

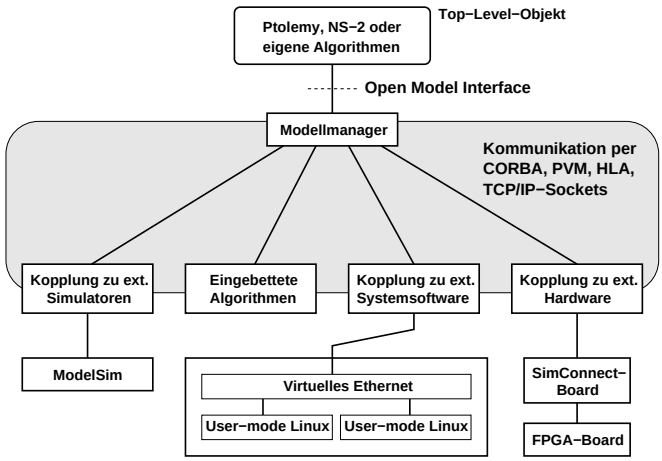

Abbildung 4: Struktur des realisierten Frameworks

theken, Tools etc.) viele Einsatzfälle abdecken. Ein weiterer Vorteil dieser Simulatoren ist die freie Verfügbarkeit der Software inklusive des Quelltextes, so daß eigene Komponenten ergänzt werden können und keine Lizenzgebühren anfallen. Reichen die Möglichkeiten der Top-Level-Simulatoren nicht aus, z. B. weil VHDL-Modelle hinzukommen, können weitere Objekte in die Simulation eingebunden werden. Untersucht wurde der VHDL/Verilog-Simulator ModelSim[Men01], eine User-mode Linux Umgebung[Dik] und die Kopplung mit einem FPGA-Board [HHS00].

6 Zusammenfassung

Die Ergebnisse der Arbeiten liefern einen Beitrag zur Entwicklung fortschrittlicher Simulationsumgebungen, die den ständig steigenden Anforderungen bei der Entwicklung komplexer Kommunikationssysteme gewachsen sein müssen. Es zeigte sich, daß der objektorientierte Ansatz viele Vorteile mit sich bringt. Insbesondere die Kombination von Geheimnisprinzip und hierarchischer Objektstruktur führt zu einer hohen Flexibilität und leichten Erweiterbarkeit, bei einer relativ geringen Komplexität der Simulationsumgebung. Die an die Objektschnittstelle gestellten Anforderungen konnten mit dem *Open Model Interface* erfüllt werden. Durch den flexiblen Ansatz dieser Schnittstelle können nicht nur funktionale Modelle in eine Simulation eingebunden, sondern auch komplexe Simulatoren sowie verschiedene Soft- und Hardware-Komponenten in einer verteilten Simulation kombiniert werden. Weiterhin sind vier Kommunikationsvarianten (TCP/IP-Sockets, PVM, CORBA und HLA) bezüglich ihrer Eignung und Performance untersucht worden. Es hat sich gezeigt, daß bei der Auswahl einer Variante nicht die Performance, sondern sonstige Eigenschaften, z. B. Verfügbarkeit auf bestimmten Plattformen oder Client-Server-Prinzip, im Vordergrund stehen können. Dieses unerwartete Ergebnis ist auf vergleichbare Werte bei

Performance-Messungen zurückzuführen. Dabei wurde auch deutlich, daß der durch den Modellmanager des OMI verursachte Overhead in Vergleich zur Kommunikationszeit vernachlässigt werden kann. Die Vorteile der gewählten Lösung müssen also nicht mit einem hohen Performance-Verlust erkauft werden. Diese Ergebnisse sind sehr erfreulich, da sie mit dem grundlegenden Ziel, auf Basis der Anforderungen die Simulationsumgebung zu erstellen (anstatt mit einem restriktiven Framework trickreich die Simulation zu ermöglichen) einhergehen.
Zusammenfassend kann gesagt werden, daß das implementierte Framework die an das Konzept gestellten Erwartungen bestätigt. Damit steht ein leistungsfähiges Konzept zur Simulation von komplexen Kommunikationssystemen zur Verfügung, das sich mit seinen positiven Eigenschaften deutlich von den bereits existierenden Simulatoren abhebt.

Literatur

[BHLM94] BUCK, J., S. HA, E. LEE und D. MESSERSCHMITT: *Ptolemy: A Framework For Simulating and Prototyping Heterogeneous Systems*. Int. Journal of computer Simulation, 4(2):155–182, 1994.

[Dik] DIKE, J.: *The User-mode Linux Kernel Home Page*. URL: http://user-mode-linux.sourceforge.net.

[HA04] HATNIK, U. und S. ALTMANN: *Using ModelSim, Matlab/Simulink and NS for Simulation of Distributed Systems.* In: *Proc. IEEE PARELEC*, Seiten 114–119, Dresden, 2004.

[Hat05] HATNIK, U.: *Ein objektorientierter Ansatz zur verteilten Simulation von digitalen Kommunikationssystemen*. Dissertation (eingereicht), Technische Universität Dresden, 2005.

[HHS00] HATNIK, U., J. HAUFE und P. SCHWARZ: *An innovative approach to couple EDA tools with reconfigurable hardware*. In: *10th International Conference on Field Programmable Logic and Applications, FPL 2000*, Seiten 826–829, Villach, Austria, August 2000. Springer Verlag.

[HHS01] HATNIK, U., J. HAUFE und P. SCHWARZ: *Object Oriented System Simulation of Large Heterogeneous Communication Systems*. In: *System Design Automation - Fundamentals, Principles, Methods, Examples*, Seiten 185–194. Kluwer Academic Publishers, March 2001.

[IEE98] IEEE COMPUTER SOCIETY: *IEEE Std 1499-1998, Standard Interface for Hardware Description Models of Electronic Components*. Institute of Electrical and Electronics Engineers, New York, December 1998.

[Men01] MENTOR GRAPHICS: *User Manual ModelSim*, 2001. http://www.model.com.

[Uni] UNIVERSITY OF SOUTHERN CALIFORNIA, INFORMATION SCIENCES INSTITUTE: *The Network Simulator - ns-2*. URL: http://www.isi.edu/nsnam/ns/.