18 Entwurfsunterstützung für verteilte heterogene Multimediasysteme

Mirko Benz, Jürgen Haufe, Jan Müller, Lars Reuther,
Jens-Uwe Schlüßler

Zusammenfassung

Technische Systeme werden zunehmend komplexer und heterogener. Sie bestehen aus einer Vielzahl von Subsystemen (Hardware/Software, elektrisch/mechanisch/optisch/..., ...). Dadurch werden die Anforderungen an den Systementwurf deutlich erhöht. Hierfür werden innerhalb des Sonderforschungsbereichs 358 „Automatisierter Systementwurf“ verschiedene Lösungsansätze betrachtet: Methoden zur Beherrschung der Durchgängigkeit der Entwurfsabläufe (ausgehend von einer sehr hohen Abstraktion bis hin zum Anschluss an untere Entwurfsebenen), der Modellierung und Simulation von Systemen aus Komponenten sehr unterschiedlicher physikalischer Domänen sowie der Hardware-/Software-Interaktionen. Die Teilgebiete und die erzielten Ergebnisse werden an Hand eines Demonstrators „Systemunterstützung für komplexe Videokonferenzsysteme“ dargestellt. Der Demonstrator soll als typischer Vertreter eines verteilten heterogenen Multimediasystems als Beispiel für einen realen Systementwurf und die Integration von Systemkomponenten dienen.

18.1 Übersicht

Videokonferenzen in sehr hoher Qualität sind auf heutigen Systemen nur eingeschränkt möglich. Die Audio- und Videoqualität (*Quality of Service*) in einem Videokonferenzsystem wird wesentlich beeinflusst von

- der Effizienz der Protokollimplementierungen in Betriebssystemen,
- der differenzierten Rechenleistung und dem Netzdurchsatz der einzelnen Teilnehmer,
- den begrenzten Speicherressourcen der Server.

Daher wird eine Hardware-Unterstützung für ausgewählte Komponenten eines Videokonferenzsystems untersucht. Zentrale Problemstellung für eine solche Systemunterstützung ist die Verbesserung der Audio-Qualität und der Video-Darstellung und die Reduzierung der Systembelastung. Es wurden hierfür folgende drei Ansätze betrachtet und prototypisch implementiert:

Protokollbeschleunigung zur Erhöhung des Netzdurchsatzes
Flaschenhals für die effektive Nutzung der Leistungsfähigkeit von bereits existierenden und vor allem zukünftigen Hochgeschwindigkeitsnetzen ist die Protokoll-Implementierung in Betriebsyste-

men. Die zur Verfügung stehende Bandbreite wird nur unzureichend genutzt. Es wurde deshalb eine Methodik entwickelt, die eine Hardware-Unterstützung komplexer Rechnernetzprotokolle erlaubt. Dazu wurden verschiedene HW/SW-Architekturen untersucht und bewertet. Ergebnis ist die Beschleunigung des TCP/IP-Protokolls durch den Einsatz einer flexible konfigurierbaren HW-Architektur.

Parallele Rechenfelder für rechenintensive Aufgaben der Bildverarbeitung
In das Videokonferenzsystem werden zur Entlastung der Teilnehmerrechner vorverarbeitete Bildsignale eingespeist. Dafür wurde je eine Anwendung aus dem Bereich der Bildkompression und der Bildrekonstruktion ausgewählt. Eine Spezialkamera enthält einen neu entwickelten Bildsensor mit integriertem analogen Rechenfeld für die sensornahe und verlustleistungsarme Realisierung von Bildverarbeitungsalgorithmen (diskrete Kosinustransformation, räumliche Filterung). Für die Rekonstruktionsalgorithmen zur tomographischen Bildrekonstruktion (gefilterte Rückprojektion, algebraische Rekonstruktion und 2D-Filterung) wurde ein paralleles Rechenfeld als digitaler ASIC entworfen und für die Einspeisung von Bilddaten eines Labortomographen in das Videokonferenzsystem eingesetzt.

Mediendatenverarbeitung in Echtzeit
Es existiert eine Vielzahl von Formaten und Kodierungen für Audio- und Videodaten, die von einem Medienserver unterstützt werden müssen. Die Speicherressourcen eines solchen Servers werden entlastet, wenn die Mediendaten erst auf Anforderung in das gewünschte Zielformat konvertiert werden. Diese Konvertierung erfolgt in Echtzeit unter Berücksichtigung vom spezifisch angeforderten Quality of Service. Dieser Teil des Demonstrators beschäftigt sich mit der Systemunterstützung für diese Echtzeit-Multimediaverarbeitung. Schwerpunkt bildet die Schaffung einer Software-Infrastruktur, welche die Entwicklung von Multimediasystemen, z. B. eines Medienservers, durch geeignete Abstraktionen und Mechanismen zur Ressourcenverwaltung erleichtert.

In den folgenden Abschnitten werden die Arbeiten vor allem an Hand der hardwarebasierten TCP/IP-Beschleunigung demonstriert. Dabei werden die Entwurfsschritte ausgehend von einer Analyse des Ist-Zustandes bis hin zur prototypischen Implementierung dargestellt. Neben den Prototypen entstand eine einheitliche Entwurfsmethodik als Grundlage der Beherrschung des systematischen Entwurfs derart komplexer Systeme. Diese Methodik vereint unterschiedliche kommerzielle wie auch eigens für spezielle Aufgabenklassen im Sonderforschungsbereich entwickelte bzw. angepasste Entwurfswerkzeuge. Die Ergebnisse zu den Aufgaben „parallele Rechenfelder zur Bildverarbeitung“ sowie „Mediendatenverarbeitung in Echtzeit“ ergänzen in weniger detaillierter Form die Übersicht der Arbeiten am Demonstrator aus dem Sonderforschungsbereich 358.

18.2 Hardwarebasierte TCP/IP-Beschleunigung

18.2.1 Motivation und Ansatz

Mit den TCP/IP-Implementierungen im Rahmen von Standardbetriebssystemen – wie Windows NT oder Linux – können moderne Hochgeschwindigkeitsnetze nur zu einem geringen Teil ausgelastet

werden. So wurde für ein Gigabit-Netz (Myrinet – 1.2 Gbit/s) auf Anwendungsebene ein Durchsatz von lediglich 335 Mbit/s gemessen. Speziell angepasste Protokolle ("lightweight protocols") können die Netze deutlich besser auslasten. Allerdings sind sie inkompatibel zu TCP/IP und daher nur eingeschränkt einsetzbar. Weiterhin benötigen diese Ansätze 100% der CPU-Leistung. Somit fehlen Ressourcen zur Anwendungsverarbeitung und für neue Dienste. Da die verfügbare Bandbreite zudem erheblich schneller zunimmt als die zur Verfügung stehende Rechnerleistung, wird sich dieser Trend noch verstärken. Hinzu kommen neue Anforderungen, die durch den Einsatz von fortschrittlichen Diensten, wie „IP Security" oder „Differentiated Services", entstehen. Daraus resultierte die Motivation zur Entwicklung einer Methodik für die Hardwareunterstützung von komplexen Rechnernetzprotokollen. Die wesentlichen Zielstellungen bildeten dabei die Steigerung der Performance unter Beibehaltung der Kompatibilität, die Entlastung der CPU sowie die Schaffung einer flexiblen Architektur zur Integration weiterer Dienste.

Für Standardbetriebssysteme hat sich eine typische Implementierungsarchitektur herausgebildet. Dabei greift die Applikation über das standardisierte Socket-Interface auf die Transportprotokolle zu. Darunter befinden sich die IP-Schicht und die Gerätetreiber sowie das Netzwerk. Die Protokolle werden üblicherweise im Betriebssystemkernel realisiert. Aus dieser Implementierungsarchitektur resultieren mehrere Overheadquellen. Dazu zählen z. B. Kommunikation, Synchronisation, Betriebssystemeinfluss, Kopieroperationen usw. Diese werden bei speziell angepassten Protokollansätzen weitgehend vermieden. Der verfolgte Ansatz sieht zunächst eine Einschränkung auf lokale Hochgeschwindigkeitsnetze vor. Aufgrund der sehr geringen Fehlerrate kommen Teile des TCP/IP-Protokolls in dieser Umgebung nur zu bestimmten Zeitpunkten (z. B. Verbindungsaufbau) oder sehr selten (z. B. Fehlerbehandlung) zum Einsatz. Im Gegensatz dazu wird der Protokollteil für den Nutzdatenaustausch ständig benötigt und unterliegt hohen Performanceanforderungen. Dieser Protokollteil (siehe Abbildung 18-1) wird auch als „Fast Path" bezeichnet.

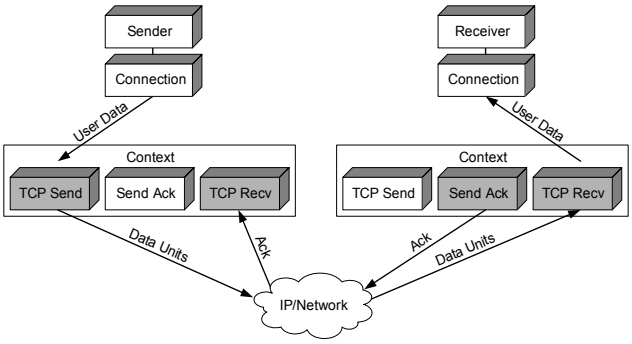

Abbildung 18-1: TCP/IP Fast Path Verarbeitung.

Er enthält keine Funktionen zur Verbindungsverwaltung oder Fehlerbehandlung. Daher weist er eine deutlich geringere Komplexität auf. Dieser Bestandteil kann durch Hardware (einer *Protocol Engine*) realisiert werden, um eine hohe Verarbeitungsgeschwindigkeit zu erreichen. Ein wesentlicher Vorteil dieser Vorgehensweise ist, dass auf existierende Softwareimplementierungen aufgebaut werden kann. Zur Systemintegration ist dabei eine HW/SW-Synchronisation nach erfolgreichem Verbindungsaufbau erforderlich. Dazu werden Kontextinformationen (Protokollvariablen) übergeben und die Protocol Engine konfiguriert. Im Fehlerfall oder zum Verbindungsabbau erfolgt eine Übergabe von der Hardwarepartition zum TCP-Softwarestack zur weiteren Bearbeitung. Die Integration der Protokollbeschleunigung erfolgt für die Anwendung transparent [1].

18.2.2 Entwurfsmethodik

Für eine effiziente hardware-basierte Beschleunigung von Transportprotokollen ist das Zusammenwirken unterschiedlicher Disziplinen wie Protokoll- und Hardwareentwurf, Systemsimulation und Embedded System Design vonnöten, was ein durchgehendes Entwurfskonzept und eine spezifische Methodik zur Umsetzung erforderlich macht [2]. Zunächst wurde eine detaillierte statische und dynamische Analyse des Kommunikationsverhaltens des Linux TCP-Stack durchgeführt. Dabei wurde insbesondere auf benötigte Systemressourcen wie CPU und Hauptspeicher, die Abhängigkeiten bei der Protokollverarbeitung, das Zugriffsverhalten auf Verbindungsdaten sowie auf erforderliche Synchronisationsschritte Wert gelegt. Daraus resultierte eine manuelle Partitionierung des Linux TCP-Stack sowie Ansatzpunkte für eine Hardware/Software Synchronisation.

Kommunikationsplattformen weisen sehr unterschiedliche Anforderungen bez. einer Protokollbeschleunigung auf. Diese resultieren z. B. aus der Anzahl zu unterstützender Verbindungen, zu erzielender Performance, dem Stromverbrauch, dem Realisierungsaufwand, Systemkosten usw. Daher wird es keine universell einsetzbare Lösung geben. Aus diesem Grunde haben wir sehr unterschiedliche Realisierungsarchitekturen für den TCP Fast Path evaluiert – von reiner Software bis zu einer kompletten Hardwarelösung. Dazu zählen embedded RISC Prozessoren, intelligente Netzwerkadapter, Netzwerkprozessoren bis hin zur direkten Hardwareumsetzung als ASIC. Eine jede dieser Architekturen hat ein unterschiedliches Aufwand/Nutzen Verhältnis. Dieses breite Spektrum an untersuchten Zielarchitekturen ermöglicht es daher, eine Protocol Engine bereitzustellen, die auf bestimmte Anforderungen hin optimiert werden kann. Die direkte Hardwareumsetzung erfordert den höchsten Realisierungsaufwand und wird in den folgenden Abschnitten näher beschrieben.

Systemsimulation
Die Systemsimulation ermöglicht frühzeitige Aussagen über die Leistungsfähigkeit unterschiedlicher Architekturvarianten der Protocol Engine und ihre Auswirkungen auf das Netzwerk. Weiterhin dient die Simulationsumgebung als Testbench während der Implementierung der Protocol Engine.

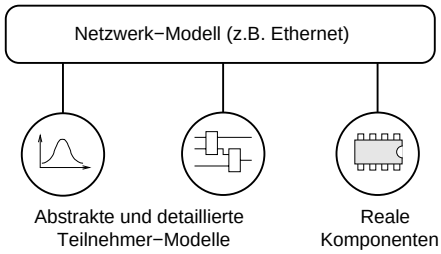

Abbildung 18-2: Top-Level-Sicht eines Systems (LAN-Beispiel).

Abbildung 18-2 zeigt die Top-Level-Sicht eines Systemmodells, das aus einem Netzwerk- und mehreren Teilnehmer-Modellen (Server und Clients des Systems) besteht. Das Netzwerk wird durch seine Topologie und charakteristische Merkmale (z. B. durchschnittliche Bandbreiten, Verzögerungszeiten oder Übertragungsfehlerraten) beschrieben. Die Teilnehmer-Modelle lassen sich in abstrakte und detaillierte Modelle unterteilen. Während abstrakte Modelle durch prinzipielle Parametern (z. B. Verzögerungen, Bandbreite, Paketgröße) charakterisiert werden, verarbeiten detaillierte Modelle reale Nutzdaten und entsprechen sehr exakt der realen Systemkomponente.

Die auf diesem Systemmodell basierende Systemsimulation ermöglicht die Analyse der Abhängigkeiten zwischen den Netzwerk- und Teilnehmer-Parametern. Für unsere Arbeiten standen

hauptsächlich die Auswirkungen unterschiedlicher Architekturvarianten der Protocol-Engine und der Einfluss verschiedener Fehlerraten bei der Datenübertragung im Vordergrund.

Simulationsansatz

Die Simulation komplexer Kommunikationssysteme erfordert eine sehr hohe Flexibilität bezüglich der einsetzbaren Modellierungssprachen und Simulationsalgorithmen. Dies führte uns zu einem objektorientierten Simulationsansatz, der im Folgenden kurz vorgestellt wird. Die objektorientierte Simulation basiert auf der Zerlegung des Gesamtsystems in relativ autonom agierende Teilsysteme. Jedes Teilsystem entspricht dabei einem Objekt, das ein Modell und einen geeigneten Simulationsalgorithmus enthält. Da ein Teilsystem wiederum aus Subsystemen bestehen kann, erlaubt der gewählte Simulationsansatz auch die Komposition eines Objektes aus mehreren anderen Objekten, wodurch eine Objekthierarchie entsteht. Abbildung 18-3 verdeutlicht die Objektstruktur und Objekthierarchie.

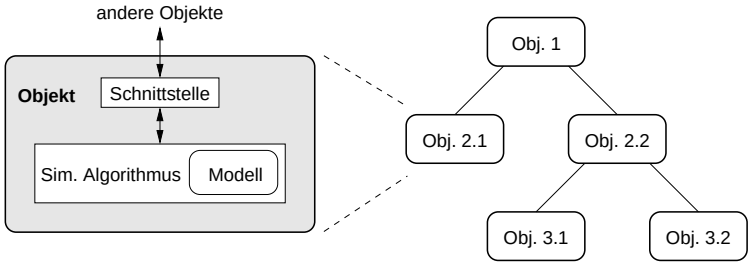

Abbildung 18-3: Objekt-Aufbau und Objekt-Hierarchie.

Ein wesentlicher Vorteil dieses Ansatzes ist die Kapselung aller Implementierungsdetails durch das Objekt, wodurch der Simulationsalgorithmus optimal an das Modell angepasst werden kann und ein „Intellectual Property"-Schutz leicht möglich ist. Der Datenaustausch und die Steuerung der Simulation erfolgen nur über eine fest definierte Schnittstelle, so dass Objekte leicht austauschbar sind und eine hohe Flexibilität erreicht wird. Da jedes Objekt einen eigenen Simulator enthält, bestehen keine prinzipiellen Einschränkungen bezüglich der Modellierungssprache. Ein universeller, sehr komplexer Simulator ist nicht erforderlich. Darüber hinaus ist die Kapselung realer Hard- und Software-Komponenten möglich, ohne dass dies Einfluss auf andere Objekte der Simulationsumgebung hat. Einziger Nachteil ist gegebenenfalls die hohe Anzahl erforderlicher Lizenzen, wenn viele Objekte in einem Simulationslauf gleichzeitig genutzt und diese mit kommerziellen Simulatoren realisiert werden.

Struktur der Simulationsumgebung

Abbildung 18-4 zeigt die Objektstruktur des simulierten Szenarios. Das Top-Level-Objekt enthält das Top-Level-Systemmodell, das mit dem Netzwerksimulator NSv2 [10] simuliert wird. Es beinhaltet neben dem eigentlichen Übertragungsnetzwerk (Ethernet) abstrakte Teilnehmer-Modelle, die direkt vom Netzwerksimulator NSv2 simuliert werden und für eine Netz-Grundlast sorgen.

Bei der Protocol-Engine handelt es sich um ein detailliertes Teilnehmer-Modell, das als VHDL-Beschreibung vorliegt. Es kann also nicht vom Netzwerksimulator NSv2 verarbeitet werden. Entsprechend dem Simulationsansatz wird es daher in einem autonomen Objekt zusammen mit einem geeigneten VHDL-Simulator gekapselt. Die von der Protocol-Engine gelieferten und die zu verarbeitenden Nutzdaten werden wiederum von autonomen Objekten weiterverarbeitet bzw. gene-

riert. Dabei handelt es sich um C-Code, der die Funktionen der Applikationen nachbildet. Mittels eines FPGA-Prototyping-Boards können Komponenten der Protocol-Engine als reale Hardware eingebunden werden [9]. Diese Hardware wird ebenfalls von einem Objekt gekapselt, so dass die Hardware-Implementierung für alle anderen Objekte unsichtbar bleibt. Der Datenaustausch zwischen den Objekten erfolgt auf Basis der „Common Objekt Request Broker Architektur" (CORBA). Die Kopplung ist daher unabhängig von der eingesetzten Programmiersprache und der Systemplattform [7]. Weiterhin ist eine verteilte Simulation möglich, wodurch die Flexibilität des Ansatzes gut unterstützt wird.

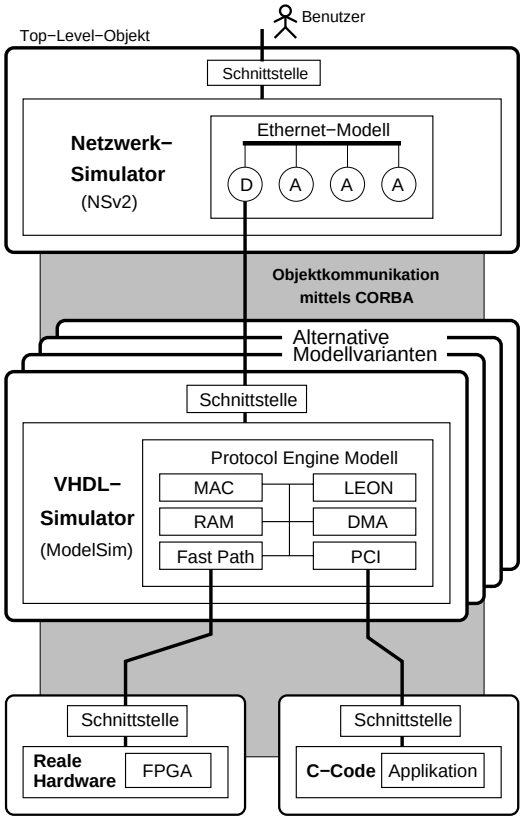

Abbildung 18-4: Objekt-Hierarchie eines Beispielszenarios.

Methodik für den Hardware-Entwurf der TCP/IP Fast Path Unit

Die aus dem TCP/IP-Protokollstack extrahierte Partition der Fast Path Unit umfasst ca. 2000 Zeilen C-Quellcode. Zielstellung ihrer Hardware-Implementierung ist es, zeitkritische TCP/IP-Protokollfunktionen zu beschleunigen. Um den Entwurfsaufwand zu reduzieren, wählten wir den Weg, Methoden der High-Level-Modellierung, der Wiederverwendung und des FPGA-Prototyping für das Entwurfsziel aufeinander abzustimmen.

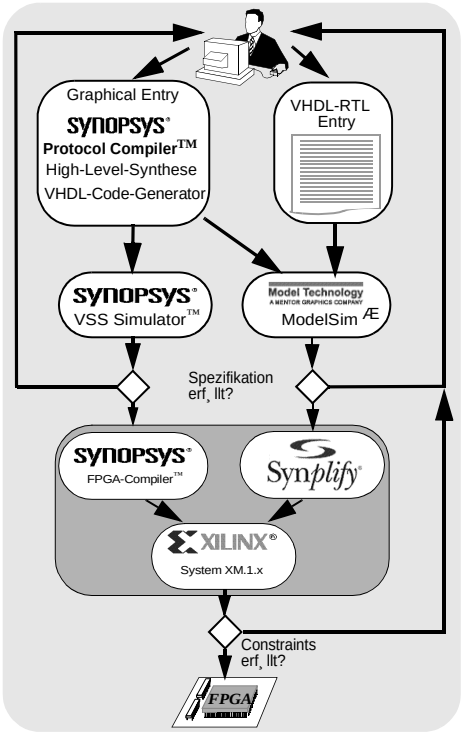

Abbildung 18-5: Erweiterter Entwurfsablauf zum FPGA Prototyping.

Die Analyse der als Hardware zu realisierenden Protokollfunktionen zeigte, dass Entwicklungszeit eingespart werden kann, wenn an Stelle einer textuellen Modellierung – z. B. mittels der Hardware-Beschreibungssprache VHDL – protokollspezifische graphische Entwurfswerkzeuge eingesetzt werden [3]. Deshalb ergänzten wir entsprechend Abbildung 18-5 unseren FPGA-basierten Entwurfsablauf um den Protocol Compiler [4]. Er erlaubt eine graphische Eingabe von Frames und Frame-Operatoren. Diese graphischen Modellierungsbausteine sind der Backus-Naur-Notation sehr ähnlich. Charakteristische Protokollverarbeitungsfunktionen, wie

- Parsen und Reassamblieren von strukturierten Datenströmen
- Synchronisation und Erkennung von Header Substrukturen
- Protokollanpassung

können durch grafische Modellkomposition schnell, übersichtlich und wiederverwendbar beschrieben werden.

Der gewählte Partitionierungsansatz für die TCP/IP Fast Path Unit ist im Abbildung 18-6 dargestellt und gliedert sich in die drei Komponenten Task-Header-Analyse, TCP/IP-Analyse und FIFO-Interface.

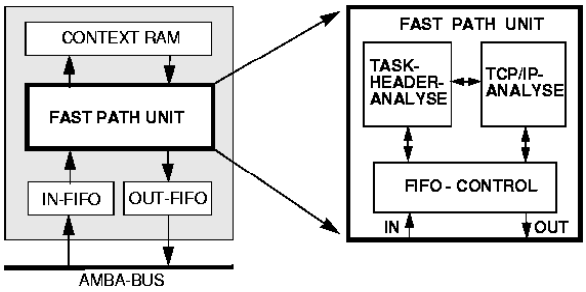

Abbildung 18-6: Komponenten der Fast Path Unit.

Abbildung 18-7 zeigt als Beispiel die graphische Protokoll-Modellierung der Komponente TCP/IP-Analyse in der obersten Hierarchiestufe. Im linken Modellabschnitt ist erkennbar, dass bei erkanntem Pakettyp „network receive" im Rahmen eines Sequential Operators nacheinander die Reference Frames „Ethernet", „IP" und „TCP" abgearbeitet werden. Der Protocol Compiler unterstützt weiterhin eine formale Protokoll-Analyse, eine Validation der Protokollfunktionen mit Backannotation von Simulationsergebnissen in die grafische Modellebene und die automatische Generierung von optimiertem und synthesefähigem VHDL-Code.

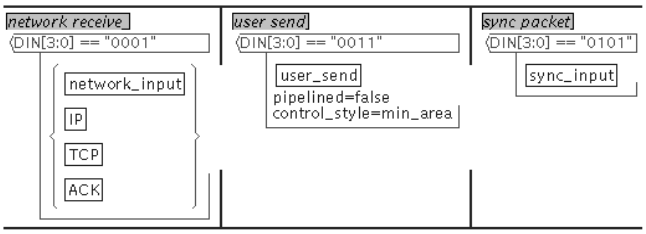

Abbildung 18-7: Graphische Protokollmodellierung.

Mit dem auf unserem Prototype-Board befindlichen FPGA XCV 1000 von Xilinx [5] stehen 12288 Slices und damit ausreichend Hardware-Ressourcen für die Implementierung der Hardware-Partition zur Verfügung. Deshalb nutzten wir primär Varianten der Modellpartitionierung und Experimente mit verschiedenen Prinzipien hierarchischen Zustandskodierung für eine Optimierung des Zeitverhaltens aus. Die untersuchten Modellierungs- und Synthesevarianten sind in [6] ausführlich dargelegt. Die daraus resultierenden Ergebnisse sind in Tabelle 18-1 zusammengefasst.

Tabelle 18-1: Ergebnisse der Synthesevarianten.

Virtex XCV1000	FPGA gesamt	Synopsys FPGA Compiler		Synplicity Synplify
		Optimierungsziel		
		Min. Area	Min. Delay	
Anzahl der Slices	12288	2872	2838	3339
max. Taktfrequenz in MHz		16	18	25

18.2.3 Implementierung

Die Fast Path Unit verarbeitet TCP-Datenpakete direkt. Zur Systemintegration sind zusätzlich Pufferspeicher sowie eine Schnittstelle zum Netz und zur Kommunikation mit dem Endsystem erforderlich. Weiterhin wurde eine Komponente zur Steuerung des Kontroll- und Datenflusses benötigt. Dafür wurde die durch die ESA (European Space Agency) zur Verfügung gestellte LEON Sparc System-on-Chip Plattform genutzt. Diese beinhaltet u. a. eine Sparc-kompatible CPU, eine On-Chip-Busarchitektur (AMBA) sowie ein Speichersubsystem. Diese Plattform liegt als VHDL-Code vor und wird durch eine GNU C Umgebung unterstützt [8]. Darauf basierend wurde die Fast Path Unit an den AMBA-Bus angebunden sowie Komponenten für die Netzwerkkommunikation (Fast Ethernet) und zur Hostanbindung integriert. Dafür kommt ein FPGA-Prototyping-Board mit einem Xilinx Virtex XCV1000 zum Einsatz. Bei diesem FPGA Typ werden 21% der Ressourcen für die Fast Path Unit und 25% für den Leon Sparc Kern benötigt. Die Auslastung des Block-RAMs beträgt 56%.

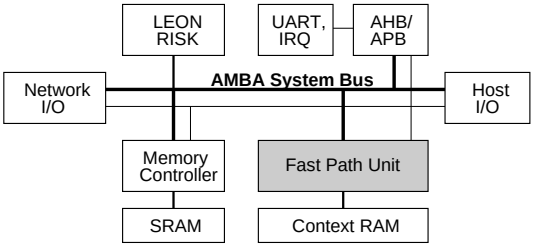

Abbildung 18-8: Systemarchitektur der Protocol Engine.

Mit dem realisierten Prototyp (siehe Abbildung 18-8) konnte ein Durchsatz von 48 Mbit/s erzielt werden. Wesentliche Begrenzungen dabei sind das verwendete Netz (max. 100 Mbit/s), die geringe Taktfrequenz sowie die “shared memory”-Kommunikation mit der Host-CPU. Ungeachtet dessen konnte bereits mit diesem Prototyp der Nachweis für die grundsätzliche Tauglichkeit des gewählten Ansatzes erbracht werden. Für die Fast Path Unit allein sind in der Simulation deutlich höhere Datenraten erzielbar. So wurde bei einem Takt von 100 MHz ein Durchsatz von bis zu 2800 Mbit/s erreicht.

18.2.4 Methodische Ergebnisse

Durch die Realisierung des Prototyps konnte ein durchgehender Designflow für die Hardwarebeschleunigung von komplexen Kommunikationsprotokollen erprobt werden. Dieser reicht von der Protokollanalyse über die Hardware/Software-Partitionierung und Systemsimulation bis zur Realisierung und Evaluierung mehrerer Architekturvarianten. Besonders hervorzuheben ist dabei eine systemweite Leistungsbewertung, eine verteilte heterogene Systemsimulation sowie die automatisierte Hardware-Synthese von Protokollkomponenten. Im Ergebnis konnte dadurch nachgewiesen werden, dass sich Durchsatz und Verzögerung durch den vorgestellten Ansatz erheblich verbessern lassen. Weiterhin besteht aufgrund des breiten Spektrums untersuchter Architekturen eine große Flexibilität und Skalierbarkeit, um auf bestimmte Zielumgebungen abgestimmte Protokollbeschleuniger zu realisieren. Dabei bietet die gewählte Architektur Leistungsreserven für neue Applikationen und Netze.

18.3 Parallele Rechenfelder für die Bildverarbeitung

Viele Anwendungen der Bildverarbeitung stehen vor der Herausforderung, umfangreiche Datenmengen in sehr kurzer Zeit und mit geringer elektrischer Leistungsaufnahme zu verarbeiten. Beispielsweise müssen beim Einsatz der Computertomographie für die Materialforschung zeitlich sehr schnell aufeinander folgende Vorgänge analysiert und visualisiert werden. Bei der Implementierung von rechenintensiven Signalverarbeitungsalgorithmen für die Kommunikationstechnik sind weitere Ressourcenbeschränkungen wie Chipfläche oder Verlustleistung zu berücksichtigen. Die Verarbeitung der Daten mit Hilfe paralleler Rechenfelder ist ein Ansatz, mit dem diese Anforderungen erfüllt werden können. Im Rahmen der Arbeiten im Sonderforschungsbereich 358 wurden deshalb Entwurfsmethoden entwickelt, mit denen ein vorgegebener Algorithmus in eine optimale parallele Architektur überführt werden kann (siehe Abbildung 18-9). Dabei werden Beschränkungen berücksichtigt, die allgemein durch die verfügbaren Ressourcen wie Verlustleistung, Chipfläche, Anordnung der Prozessorelemente, Bandbreite der Interfaces und verfügbare Funktionselemente vorgegeben sind. Die Erprobung dieser Entwurfsmethoden erfolgte an zwei Chipimplementierungen.

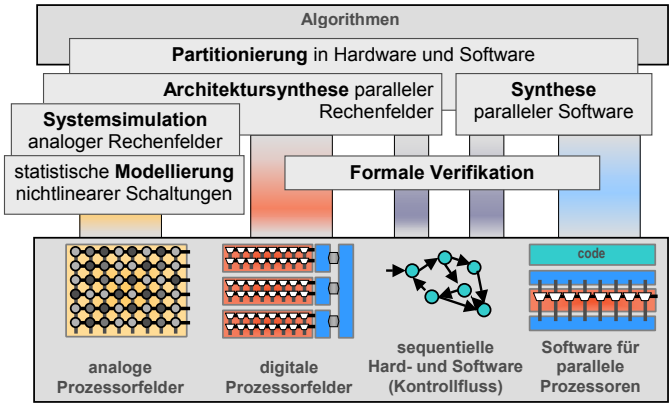

Abbildung 18-9: Systementwurf paralleler Rechenfelder.

Architektursynthese
Auf dem Gebiet der Architektursynthese wurden neue Methoden und Werkzeuge für einen automatisierten Entwurf von parallelen Rechenfeldern (RF) entwickelt. Insbesondere wurden für den Schritt der Bestimmung von Allokation (räumliche Abbildung der einzelnen Verarbeitungsschritte auf die Elemente des RF) und Scheduling (Bestimmung eines zeitlichen Ablaufs der Verarbeitung) im Entwurfsfluss von RF neue Optimierungsprobleme entworfen, die in einem breiteren Umfang als bisher Ressourcenbeschränkungen einbeziehen [11]. Diese Ressourcenbeschränkungen führen dazu, dass unter Beibehaltung des Zieles der möglichst schnellen Verarbeitung (minimale Latenz) andere Kriterien wie Größe der RF-Partitionen, Auswahl der am besten geeigneten Funktionseinheiten der Prozessoren des RF und der Erhalt von Freiheitsgraden für die spätere Partitionierung einbezogen

werden. Ähnliche Methoden werden auch für das Software-Pipelining, d. h. die Abbildung des Algorithmus auf die Funktionseinheiten *eines* Prozessors, genutzt [12].

Für den Schritt der RF-Partitionierung wurden die Methoden der Partitionierung und Relokalisierung entwickelt [14], die bei der Abbildung des Algorithmus auf ein Rechenfeldfeld reduzierter Größe Implementierungs- und Interfacebeschränkungen berücksichtigen. Dies ermöglicht die Anpassung an Restriktionen der Außenwelt und ein Speichermanagement auf Systemebene. Die Methoden wurden in einem Entwurfsprogramm implementiert, das als Eingabe einen C-ähnlich notierten Algorithmus verwendet, und mit dem die Synthese einer Rechenfeldarchitektur interaktiv oder automatisch vonstatten geht. Mit Hilfe der genannten Methoden wurde ein paralleles Rechenfeld für die Rekonstruktion tomographischer Bilddaten [17] entworfen und als Chip-Implementierung "RecoChip" (siehe Tabelle 18-2) in ein Rekonstruktionssystem [13] integriert.

Tabelle 18-2: Parameter und Eigenschaften der Chipentwürfe.

	Bildsensor mit integriertem hochparallelen analogen Rechenfeld (VISP 128)	RecoChip
Chipparameter	0,6µm AMS CMOS-Technologie Fläche: 2,5 x 4,5 mm^2 Pixelfläche: 12 x 12 μm^2 Verlustleistung bei 5 V: 250 mW (Rechenfeld) 25 mW (ADC)	0.35µm CMOS-Technologie Fläche: 21.34 mm^2 500 000 Gatter
Komponenten	64 x 128 Pixelzellen 64 x 8 analoge Prozessorelemente	32 x 2 digitale Prozessoren Befehls-Scheduler Parallele/Serielle I/O (á 32 bit)
Algorithmen	2D-Filter diskrete Kosinustransformation (DCT)	Gefilterte Rückprojektion (FBP) Algebraische Rekonstruktionstechnik (ART) 2D-Filter
Besonderheiten	integrierter Bildsensor	kaskadierbar (x-, y-Richtung), programmierbar für weitere Algorithmen

Systemsimulation und statistische Modellierung analoger Rechenfelder

Bei der analogen Realisierung paralleler Rechenfelder unterliegt das reale Verhalten der einzelnen Prozessorelemente (PE) zufälligen und systematischen Abweichungen vom idealen Verhalten. Die Ursache liegt in den Parameterstreuungen der Einzelbauelemente der Prozessorelemente. Eine vollständige Netzwerksimulation des analogen Rechenfeldes auf Transistorniveau ist nicht möglich. Die Aufgabe besteht darin, aus der Transistorschaltung des Prozessorelementes ein Verhaltensmodell abzuleiten, das die vielfältigen statistischen Einflüsse in wenigen Modellparametern zusammenfasst. Dafür wurden zwei unterschiedliche Methoden angewendet [15]:

1. Die auf einem linearen Fehlermodell beruhende Methode der Varianzanalyse. Zur Unterstützung dieser Methode ist eine Abschätzung des Fehlers möglich, der durch die Linearisierung entsteht.
2. Monte-Carlo-Analyse. Aus mehreren Netzwerk-Simulationsrechnungen für ein Prozessorelement wird ein Verhaltensmodell mit statistischen Parametern gewonnen. Unterstützt wird diese Methode durch eine Abschätzung des notwendigen Stichprobenumfangs.

Die Ergebnisse dieser Verhaltensmodellierung des Prozessorelementes sowie die aus der Architektursynthese gewonnene Topologie sind der Ausgangspunkt der High-Level-Systemsimulation des Rechenfeldes [16]. Jede der Komponenten des Rechenfeldes liegt als objektorientierte Beschreibung (C++) vor. Die Simulationsumgebung enthält in einer Klassenbibliothek den Simulationskern, Elemente für die Auswahl von Schaltungsparametern (z. B. statistische Parameter) sowie Funktionen für die Bilddatenmanipulation und -bewertung. Mit Hilfe der vorgestellten Methoden war es möglich, den Einfluss von Parameterstreuungen der Fertigungstechnologie auf das Verhalten eines CMOS-Bildsensors mit integriertem hochparallelen analogen Rechenfeld (siehe Tabelle 18-2) bereits in der Entwurfsphase zu bestimmen.

18.4 Mediendatenverarbeitung in Echtzeit

Der dritte Teil des Demonstrators beschäftigt sich mit der Systemunterstützung für Echtzeit-Multimediaverarbeitung. Schwerpunkt bildet die Schaffung einer Software-Infrastruktur, welche die Entwicklung von Multimediasystemen, z. B. eines Medienservers, durch geeignete Abstraktionen und Mechanismen zur Ressourcenverwaltung erleichtert.

Die Verarbeitung von Multimediadaten stellt eine Reihe von Anforderungen an das Betriebssystem sowie die verwendete Software-Architektur:

- Die Verarbeitung der Mediendaten muss in der Regel in Echtzeit erfolgen. Durch die Bildrate bei Videoströmen oder die Sample Rate bei Audioströmen sind Zeitschranken definiert, bis zu denen die Verarbeitung beendet sein muss.
- Durch den Einsatz in heterogenen Systemen sind flexible Systeme erforderlich. Unter anderem müssen verschiedene Formate der Mediendaten und unterschiedliche Konfigurationen (Bandbreiten, Bildgeometrie, ...) berücksichtigt werden.
- Algorithmen zur Medienverarbeitung besitzen häufig einen hohen Ressourcenbedarf (Rechenleistung, Speicherkapazität, ...) Dadurch sind effiziente Verwaltungsverfahren notwendig, die eine möglichst hohe Ausnutzung der zur Verfügung stehenden Ressourcen ermöglichen.

Im Rahmen der Arbeiten des SFB wurde zur Lösung dieser Aufgabenstellung am Beispiel eines Medienservers ein formatunabhängiges Speichersystem auf Basis eines Echtzeit-Betriebssystems realisiert [19][20][21]. Dem System liegen zwei Ideen zu Grunde: die Verarbeitung von Mediendaten durch Konverterketten sowie die Berücksichtigung stochastischer Eigenschaften bei der Ressourcenverwaltung.

Konverterketten bilden das Modell für Anwendungen. Abbildung 18-10 stellt den prinzipiellen Aufbau einer Konverterkette dar. Ein Konverter verarbeitet einen Eingangsdatenstrom und erzeugt einen Ausgangsdatenstrom, der wiederum Eingangsdatenstrom eines weiteren Konverters ist. Die Datenströme zwischen den Konvertern werden durch *Schwankungsbeschränkte Periodische Datenströme* [22] beschrieben. Diese Beschreibung charakterisiert die Datenströme anhand von Volumen- und Zeiteigenschaften. Aus der Kenntnis dieser Eigenschaften und des Verhaltens der Konverter lassen sich die Ressourcen bestimmen, die zur Bearbeitung der Datenströme benötigt werden.

Der Einsatz von Konverterketten ermöglicht die Konstruktion von flexiblen Systemen zur Mediendatenverarbeitung. Durch die generische Definition der Schnittstellen zwischen den einzel-

nen Konvertern und zwischen Konvertern und der Ressourcenverwaltung ist das einfache Austauschen bzw. das einfache Umkonfigurieren einer Konverterkette auch zur Laufzeit gewährleistet.

Viele der in den Konverterketten eingesetzten Algorithmen besitzen einen hohen Bedarf an Rechenzeit und anderen Ressourcen, wie z. B. Netzwerk- oder Festplattenbandbreite. Dadurch sind besonders effiziente Verfahren zum Scheduling dieser Ressourcen erforderlich. Die bekannten Echtzeit-Schedulingalgorithmen verwenden jedoch meistens Worst-Case Annahmen über das Laufzeitverhalten der Algorithmen, was eine schlechte Auslastung der Ressourcen zur Folge hat.

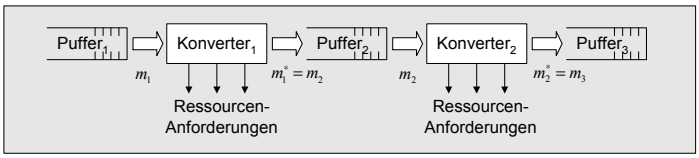

Abbildung 18-10: Medienverarbeitung durch Konverterketten; Die Konverter stellen eine Abbildung eines Eingangsdatenstroms m auf einen Ausgangsdatenstrom m^* dar.

Eine der Hauptursachen für die Verwendung von Worst-Case Abschätzungen ist die Annahme, dass das Scheduling alle gesetzten Zeitschranken erfüllen muss. Gerade Algorithmen zur Medienverarbeitung sind jedoch in der Lage, nicht eingehaltene Zeitschranken in bestimmten Grenzen zu tolerieren. So ist es z. B. für die Darstellung eines Videos in den meisten Fällen akzeptabel, einige wenige Bilder nicht anzuzeigen. Diese Eigenschaft kann ausgenutzt werden, indem den Anwendungen die Ressourcen mit einer *Mindestqualität* bereitgestellt werden. Eine Mindestqualität bedeutet z. B. für das CPU-Scheduling, dass einer Anwendung garantiert wird, dass in einem bestimmten Prozentsatz ihrer Verarbeitungsperioden ausreichend Rechenzeit zur Verfügung steht um die geforderten Zeitschranken einzuhalten. Mit diesen Voraussetzungen kann für die Einplanung der benötigten Ressourcen eine Beschreibung des tatsächlichen Verhaltens der Ressourcen verwendet werden, z. B. empirisch ermittelte Häufigkeitsverteilungen. Durch den Einsatz dieses *Quality Assuring Scheduling* [23] kann eine deutlich Verbesserung der Ressourcenauslastung erzielt werden. So konnte z. B. für eine Festplatte bereits bei einer Reduzierung der geforderten Qualität auf 97% nahezu die gesamte Bandbreite ausgenutzt werden. Dies entspricht einer Verbesserung der Auslastung der Festplatte um einen Faktor von 2,5 gegenüber einer Einplanung basierend auf Worst-Case Annahmen.

18.5 Zusammenfassung

Ausgehend von einer software-basierten Implementierung eines Videokonferenzsystems als einem typischen Vertreter für heterogene verteilte Multimediasysteme wurden ausgewählte Aspekte des Systementwurfs an Komponenten des Videokonferenzsystems bearbeitet. Die wesentliche Zielrichtung dabei war, sehr rechenintensive Prozesse in Spezial-Hardware auszulagern und zusätzliche Flexibilitat durch die Einführung von dedizierten Servern zur Formatkonvertierung zu erlangen. Insbesondere erfolgten Arbeiten zur Protokollbeschleunigung durch Hardware, zur digitalen und analogen Bildsignalvorverarbeitung und zur Echtzeitkonvertierung von Mediendaten auf einem Medienserver. Für die Umsetzung der dafür angewandten Algorithmen in eine Implementierung

wurden neuartige Lösungsansatze und Entwurfsmethoden entwickelt. Zur Validation der dargestellten Lösungen erfolgte eine prototypische Integration in ein Videokonferenzsystem. Dabei konnte eine erhebliche Verbesserung der Darstellungsqualität erreicht werden. Darüber hinaus ergeben sich zahlreiche weitere Anwendungsszenarien.

Danksagung
Diese Arbeit entstand im Rahmen des durch die Deutsche Forschungsgemeinschaft geförderten Sonderforschungsbereiches 358 „Automatisierter Systementwurf“. Neben den Autoren haben noch folgende Personen an der Realisierung aktiv mitgewirkt, denen wir hiermit herzlich danken: F. Binkowski, K. Feske, J. Grusa, U. Hatnik, M. Hosemann, R. Lehmann, H. Overbeck, J. Schönherr.

Literatur

[1] Benz, M. „An Architecture and Prototype Implementation for TCP/IP Hardware Support“, TERENA Networking Conference, Antalya (2001)
[2] Benz, M., Hatnik, U., Feske, K., Schwarz, P. „TCP/IP Protocol Engine System Simulation. PROMS 2001 – Protocols for Multimedia Systems, Enschede, Niederlande (2001)
[3] Seawright, A. et al. „A System for Compiling and Debugging Structured Data Processing Controllers“, EURO-DAC'96, Geneva, September 16-20 (1996)
[4] SYNOPSYS „V2000.05 Protocol Compiler User`s Guide“, Synopsys Inc. (2000)
[5] XILINX „Xilinx Data Book“, Xilinx, Inc., San Jose, CA, www.xilinx.com (2001)
[6] Grusa, J. „FPGA-Prototypenentwicklung eines Hardwarebeschleunigers für TCP/IP-Protokoll-Komponenten“, Diplomarbeit HTW Dresden, Fachbereich Elektotechnik (2001)
[7] Hatnik, U., Haufe, J., Schwarz, P. „Object Oriented System Simulation of Large Heterogeneous Communication Systems“, in Merker, R., Schwarz, W. (Hrsg.) „System Design Automation, Fundamentals, Principles, Methods, Examples“, pp.185-194, Kluwer Academic Publishers (2001)
[8] Gaisler, J. „LEON Sparc Processor“, www.gaisler.com, 2001
[9] Hatnik, U., Haufe, J., Schwarz, P. „An innovative approach to couple EDA tools with reconfigurable hardware“, 10th International Conference on Field Programmable Logic and Applications, FPL, Villach (Austria), August 2000, 826–829
[10] Fall, K. Varadhan, K. „ns Notes and Documantation“, www.isi.edu/nsnam/ns (2000)
[11] Fimmel, D. „Generation of scheduling functions supporting LSGP-partitioning.” In Proc. IEEE Int. Conf. on Application Specific Systems, Architectures and Processors 2000, pages 349–358, Boston, July 2000.
[12] Fimmel, D., Müller, J. „Optimal Software Pipelining under Resource Constraints.” International Journal of Foundations of Computer Science, vol. 12, no. 6, pp. 697–718, 2001
[13] Müller, J. Kelber, J., Schaffer, R., Kortke, M., Merker, R. „A Hardware accelerator for tomographic reconstruction and 2D-filtering.” In Proc. SCI, Orlando, Florida, 2001.
[14] Eckhardt, U., Merker., R. „Hierarchical algorithm partitioning at system level for an improved utilization of memory structures.” IEEE Transactions on CAD, 18(1):14 24, January 1999.

[15] Graupner, A. Schwarz, W., Lemke, K. Getzlaff, S., Schüffny, R. „Statistical Analysis of Analogue Structures." In R. Merker and W. Schwarz, editors, System Design Automation – Fundamentals, Principles, Methods, Examples, pages 164-175. Kluwer Academic Publishers, 2000.

[16] Henker, S. Graupner, A., Getzlaff, S., Schreiter, J., Schüffny, R. „A New Hierarchical Simulator for Highly Parallel Analog Processor Arrays." In R. Merker and W. Schwarz, editors, System Design Automation – Fundamentals, Principles, Methods, Examples, pages 176-182. Kluwer Academic Publishers, 2000.

[17] Schmitt, T., Fimmel, D., Kortke, M., Merker., R. „Parallel processor arrays for tomographic reconstruction algorithms." In F. Pichler, R. Moreno-Diaz, and P. Kopacek, editors, Computer Aided Systems Theory-EUROCAST99, volume 1798 of Lecture Notes in Computer Science, pages 127 141. Springer, Berlin Heidelberg, March 2000.

[18] S. Getzlaff, J. Schreiter, A. Graupner, and R. Schüffny. „A System-On-Chip Realization of a CMOS Image Sensor with Programmable Analog Image Preprocessing." In Proc. IEEE International Symposium on Circuits and Systems ISCAS'2001, 2001

[19] Lindner,W., Berthold, H., Binkowski, F., Heuer, A., Meyer-Wegener K. „Enabling Hypermedia Videos in Multimedia Database Systems Coupled with Realtime Media Servers", in Int. Database Engineering and Application Symposium IDEAS 2000, Yokohama, Japan

[20] Härtig, H., Baumgartl, R., Borriss, M., Hamann, Cl.-J., Hohmuth, M., Mehnert, F., Reuther, L., Schönberg, S., Wolter, J. „DROPS – OS Support for Distributed Multimedia Applications", in Eight ACM SIGOPS European Workshop 1998, Sintra, Protugal

[21] Härtig, H., Reuther, L., Wolter, J., Borriss, M., Paul, T. „Cooperating Resource Managers", in Proceedings of the Fifth IEEE Real-Time Technology and Applications Symposium (RTAS), June 1999, Vancouver, Canada

[22] Hamann, Cl.-J., Märcz, A., Meyer-Wegener, K. „Buffer Optimization in Realtime Media Streams Using Jitter-Constrained Periodic Streams", Technical Report SFB358-G3-01/2001, TU Dresden

[23] Hamann, Cl.-J., Löser, J., Reuther, L., Schönberg, S., Wolter, J., Härtig, H. „Quality Assuring Scheduling – Using Stochastic Behaviour to Improve Resource Utilization", in 22th IEEE Real-Time Systems Symposium RTSS 2001, London, UK