Aktive Router: Ein Hardwarekonzept für Storage Area Networks

André Brinkmann, Dominik Langen und Ulrich Rückert
Heinz Nixdorf Institut, Fachgebiet Schaltungstechnik
Universität Paderborn, 33102 Paderborn, Deutschland
Email: {brinkman, langen, rueckert}@hni.upb.de

Kurzfassung

Die Verschaltung von Festplatten zu Feldern zur effizienten Verwaltung von riesigen Datenmengen gewinnt immer mehr an Bedeutung. Durch den Einsatz aktiver Einheiten zum Aufbau von internen Verbindungsstrukturen zwischen den Festplatten ist es möglich, an das System angeschlossene Dateiserver von vielen Basisaufgaben zu entlasten. Diese *aktiven Router* basieren neben einer Routingeinheit und den Schnittstellen zu den Festplatten und den Dateiservern auf einem Mikroprozessor und einer rekonfigurierbaren Einheit. Im Rahmen dieser Arbeit werden mögliche Einsatzgebiete von aktiven Routern, Untersuchungen zu deren Dimensionierung und praktischen Implementierungen der Hardware vorgestellt.

1 Einleitung

In den vergangenen Jahren konnte ein dramatisches Wachstum im Bereich der Festplattenfelder beobachtet werden. Bei einem Anstieg der Speicherdichte von 60% pro Jahr und fallenden Kosten von bis zu 50% pro Jahr und gespeichertem Megabyte verdoppelt sich die Menge des verkauften Festplattenspeichers nahezu jedes Jahr. Den Hauptanteil des Festplattenspeichers machen Festplattenfelder aus, die in lokalen Netzwerken als zentrales Speichermedium dienen. Die zentrale Speicherung der Daten ermöglicht es, die Daten effizient zu verwalten und somit die Kosten für die Speicherung zu senken. Um einen unternehmensweiten Zugriff auf die Daten durch mehrere Server zu ermöglichen, wurden neue Konzepte wie das der Storage Area Networks (SANs) eingeführt. In SANs werden die Speichermedien über ein eigenes, von den lokalen Netzwerken abgekoppeltes Netzwerk miteinander verbunden. Hierdurch wird es möglich, Managementaufgaben, wie das Sichern der Daten auf Bandlaufwerke, ohne Belastung des Unternehmensnetzwerkes durchzuführen.
Die Verwaltung der SAN-Komponenten wird jedoch in der Regel weiterhin durch die zentral angeschlossenen Serversysteme durchgeführt. Die in dem SAN vorhandene Verbindungsinfrastruktur wird nur für das Weiterleiten von Datenpaketen verwendet. Durch die Fortschritte in der Integrationstechnik ist es jedoch möglich, die Routingressourcen innerhalb des Verbindungsnetzwerkes um aktive Einheiten zu ergänzen [2,9]. Im Rahmen des Presto (Paderborn Realtime Storage Network)-Projektes der Universität Paderborn sind verteilte Verfahren für parallele Datenserver entwickelt worden, die eine effiziente Unterstützung der angeschlossenen Einheiten durch den Einsatz von solchen aktiven Routern ermöglichen [3,4,7]. Diese Verfahren umfassen Basisaufgaben wie die Bereitstellung eines virtuellen Adreßraumes über alle Festplatten und Verfahren für die Lastbalanzierung der Anfragen, aber auch Zusatzfunktionalitäten wie das Komprimieren von Datenblöcken und das Wiederherstellen von Datenblöcken nach Festplattenfehlern. Die Verfahren basieren auf der Annahme, das in dem Netzwerk nicht nur Schalter ohne eigene Intelligenz verfügbar sind, sondern daß die Verbindungskomponenten eigenständig Teilaufgaben der Systemverwaltung übernehmen können. Im weiteren Verlauf soll dieses Konzept der aktiven Routingknoten mit einer Übersicht über die zugehörigen Hardwareanforderungen vorgestellt werden.

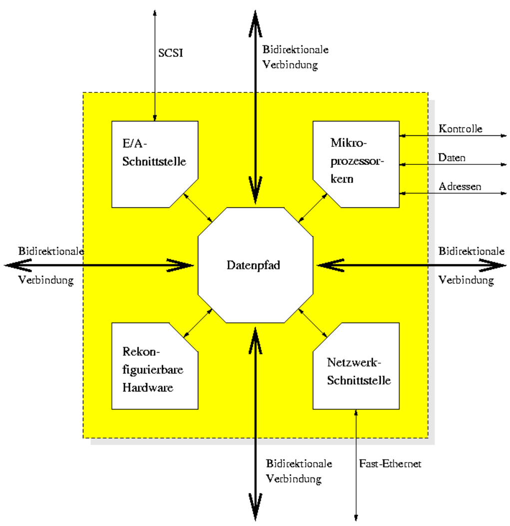

Bild 1 Aufbau eines aktiven Routingknotens

2 Aktive Router

Unser Speichernetzwerk ist modular aus einer Grundkomponente aufgebaut, die im folgenden als aktiver Routingknoten oder kurz als aktiver Router bezeichnet wird (siehe **Bild 1**). Die Kernkomponenten des aktiven Routers sind ein eingebetteter Mikroprozessorkern und eine rekonfigurierbare Einheit. Der Mikroprozessor stellt eine universelle Programmierbarkeit bereit, die durch die dynamische Bereitstellung rekonfigurierbarer Hardware optimiert unterstützt werden kann. Weiterhin verfügt der Baustein über Schnittstellen zu Festplatten und zu lokalen Netzwerken. Um das interne Verbindungsnetzwerk zwischen den Bausteinen aufbauen zu können, verfügt jeder aktive Routingknoten über vier dedizierte serielle Kommunikationsverbindungen. Hierdurch können die aktiven Routingknoten zu beliebigen Netzwerken mit einem maximalen Grad von vier verknüpft werden. Mögliche Netzwerke sind z.B. Gitterstrukturen und Butterfly-Netzwerke. Dabei ist es nicht notwendig, an jeden der Knoten Speichermedien oder Zugänge zu lokalen Netzwerken anzuschließen.
Durch die Fortschritte der Mikroelektronik ist heute möglich, diese Komponenten auf einem Baustein, einem sogenannten System-on-Chip (SoC), zu integrieren. Im folgenden sollen die Aufgaben der einzelnen verwendeten Komponenten, ein Anforderungsprofil und Abschätzungen bzgl. der Integrationskosten auf einem Baustein vorgestellt werden. Die Anforderungen an den Baustein bzgl. des Durchsatzes des internen Verbindungsnetzwerkes und der daraus resultierenden Rechenkomplexität wurden mit Hilfe der Simulationsumgebung SimLab [5] und Implementierungen auf Workstation-Clustern ermittelt. Grundlage der Architekturuntersuchungen sind Entwürfe für FPGAs, Standardzellen- und Vollkundenentwürfe.

2.1 Einbettung eines Mikroprozessors

Durch die Verwendung eines Mikroprozessors in dem aktiven Router wird es möglich, allgemeine Aufgaben von dem Datei-Servern auf das Verbindungsnetzwerk auszulagern und das Netzwerk als aktiven Bestandteil der SAN-Architektur zu nutzen. Im Gegensatz zu üblichen Serverarchitekturen wird dabei nicht davon ausgegangen, daß es sich auf dem aktiven Routingknoten um einen Hochleistungsprozessor handelt, sondern es sollen kleine, einbettbare Komponenten verwendet werden. Durch den Aufbau der Server-Architektur skaliert dabei die Rechenleistung innerhalb des Verbindungsnetzwerkes mit der Anzahl der in dem System befindlichen aktiven Routingknoten.
Der von uns in VHDL entwickelte Mikroprozessor-Kern ist binärkompatibel mit der Motorola M°Core-

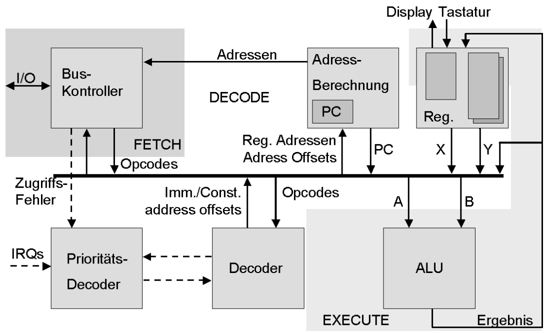

Bild 2 Architektur des Prozessor-Kerns

Architektur (siehe **Bild 2**). Der Prozessor ist ein 32 Bit RISC-Kontroller mit einer festen Instruktionslänge von 16 Bit, bei dem alle Instruktionen mit Ausnahme der Lade-/Speicher-Operationen nur auf den Registern arbeiten. Die Bearbeitung der meisten Befehle braucht einen Takt, Multiplikation und Division benötigen 18, bzw. 37 Takte. Der Prozessor verfügt über zwei Registerbänke mit 16 32-Bit Registern, die alternativ genutzt werden können. Im Supervisor-Modus sind zusätzlich 12 32-Bit Kontrollregister vorhanden, von denen fünf Register frei verfügbar sind und ein Register zur Kontrolle externer Module verwendet werden kann. Der Prozessor verfügt weiterhin über eine 3-stufige Pipeline, die vom Bus-Kontroller gesteuert wird.

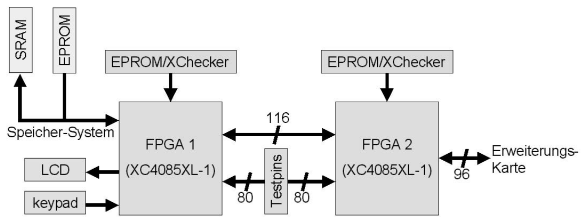

Bild 3 Aufbau der Prototypen-Umgebung

Durch die Verfügbarkeit der VHDL-Beschreibung ist es möglich, den Prozessorkern an die speziellen Randbedingungen im Datenserver-Umfeld anzupassen und z.B. den Befehlssatz entsprechend zu erweitern. In unserem Fall haben wir das Busprotokoll des M°Cores durch das leistungsfähigere Amba AHB-Protokoll ersetzt, um den Anforderungen an modulare SoC-Architekturen mit mehreren aktiven Einheiten gerecht zu werden. Der Prozessorkern wurde im Rahmen der in [6] vorgestellten Prototypenumgebung getestet. Die Prototypenumgebung umfaßt neben zwei Xilinx XC4085-FPGAs u.a. mehrere SRAM und EPROM-Blöcke (siehe **Bild 3**). Ein FPGA dient zur Aufnahme des Mikroprozessorkerns, das andere FPGA steht für applikationsspezifische Erweiterungen zur Verfügung. Mit der einfachen Busschnittstelle verbraucht der Prozessorkern bei einer Taktfrequenz von 2,3 MHz 3036 CLBs bzw. 80% der verfügbaren FPGA Ressourcen.

2.2 Rekonfigurierbare Einheiten

2.2.1 Unterstützung lokalitätsabhängiger Aufgaben

Neben der programmierbaren Komponente wird im Rahmen des Presto-Projektes die effiziente Nutzbarmachung rekonfigurierbarer Konzepte im Bereich der Datenserverarchitekturen untersucht. Ein interessantes Einsatzgebiet für den Einsatz rekonfigurierbarer Hardware innerhalb von Datenservern ist die Unterstützung lokalitätsabhängiger Aufgaben, die durch die im Mikroprozessor bereitstehende Rechenleistung nicht bearbeitet werden können. Lokalitätsabhängig bedeutet hierbei, daß die zugehörige Aufgabenstellung nicht in jedem der aktiven Routingknoten durchzuführen ist, sondern nur auf wenige Knoten innerhalb des Verbindungsnetzwerkes begrenzt ist. Für diese Aufgabenstellungen lohnt die Bereitstellung festverdrahteter Hardwareblöcke innerhalb der aktiven Routingknoten aufgrund des seltenen Einsatzes und der Vielfältigkeit der möglichen Aufgabenstellungen nicht.

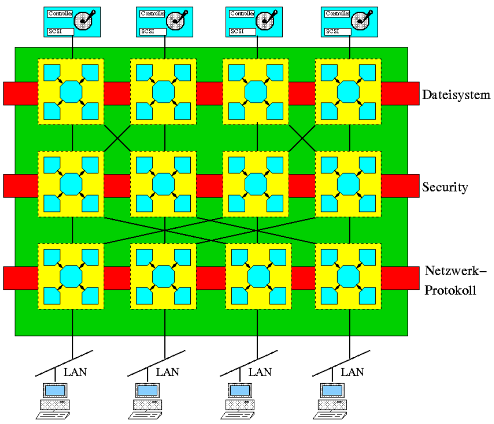

Bild 4 Mehrschichtiges Verbindungsnetzwerk

Besonders geeignet für die Aufgabenverteilung sind mehrschichtige Verbindungsnetzwerke wie das Butterfly-Netzwerk (siehe **Bild 4**). Je nach Lage des aktiven Routingknotens innerhalb des mehrschichtigen Verbindungsnetzwerkes werden der rekonfigurierbaren Einheit unterschiedliche Aufgabenstellungen zugeordnet. So kann z.B. in einem Knoten an einem Zugang zu den lokalen Netzwerken der Mikroprozessor von protokollspezifische Aufgabenstellungen entlastet werden. Sind Sicherheitsaspekte von Bedeutung, wie diese in NASD (Network Attached Secure Disk)-Architekturen [9] oder beim Zugriff auf sensitive Daten im allgemeinen auftreten, kann in einer weiteren Schicht eine Kryptographie-Einheit eingesetzt werden, die eine sichere und schnelle Ver- und Entschlüsselung garantiert.

Im Rahmen des Presto-Projektes haben wir verschiedene Konzepte untersucht, die dem verwendeten Mikroprozessor Teilaufgaben der Komprimierung oder die gesamte Komprimierung von Datenblöcken nach dem Huffman-Algorithmus abnehmen [8,12]. Grundlage der Untersuchungen war der im vorangegangenen Kapitel vorgestellte M°Core-Prozessor und die in [6] vorgestellte Prototypen-Umgebung. Zur Abbildung der Komprimierungshardware steht neben dem FPGA 2 über eine Erweiterungskarte noch SRAM Speicher zur Verfügung.

2.2.2 Einsatz des Huffman-Algorithmus in rekonfigurierbare Hardware

Der Huffman-Algorithmus kann grob in die vier Schritte Aufsummieren der Symbolhäufigkeiten, Bilden des Huffman-Baums, Bilden der Codeworte und Längen und Erzeugen der Codesequenz eingeteilt werden. Im weiteren Verlauf werden für die vier Unteralgorithmen die hier gewählten Hardwareimplementierungen vorgestellt. Dabei wird davon ausgegangen, daß jedes Symbol 8 Bit breit ist und über den Amba AHB Bus vier Symbole gleichzeitig an die Komprimierungseinheit versendet werden. Jeder zu komprimierende Block kann bis zu 32 KByte groß sein.

Für das Aufaddieren der Symbolhäufigkeiten sind mehrere Alternativen denkbar. Das hier gewählte Verfahren arbeitet ohne Registerfelder zur Pufferung der Eingangsdaten, um den Ressourcenverbrauch der Schaltung zu minimieren. Zu jedem der vier Bytes des Eingangsdatum wird ein 16 Bit Zähler aus dem SRAM geladen, inkrementiert und im nächsten Schritt zurückgeschrieben. Für jedes Eingangsdatum werden somit 9 Takte benötigt.

Durch den Einsatz von Registerfeldern gemäß **Bild 5** könnte diese Zeit auf drei Takte pro Eingangsdatum reduziert werden. Für jedes der 256 möglichen Symbole steht hier ein 2 Bit breites Pufferregister zur Verfügung. Die Register werden durch die vier 3-Bit-Addierer gemäß der Häufigkeit des Auftretens in dem Eingangsdatum erhöht. Wird ein Pufferüberlauf bei einem Register erkannt, werden die Adressen der Daten in eins der acht Pufferregister übernommen. Die eigentliche Symbolhäufigkeit wird in dem SRAM-Block gespeichert. Im Durchschnitt muß pro 32-Bit-Eingangsdatum nur ein Lese-Addier-Schreibzugriff auf den SRAM-Speicher vorgenommen werden, um die Registerüberläufe zu sichern. Der Nachteil dieses Verfahrens ist, daß durch die aufwendige Ansteuerung des Registerfeldes 4500 CLBs benötigt werden und sich des weiteren die maximale Taktfrequenz um 1/3 im Vergleich zur ersten Möglichkeit reduziert. Die Größe des Moduls und dessen geringere Taktfrequenz ergeben sich aus den Multiplexer-Strukturen und der notwendigen Erkennung von mehreren gleichen Symbolen in einem 32-Bit Eingabedatum.

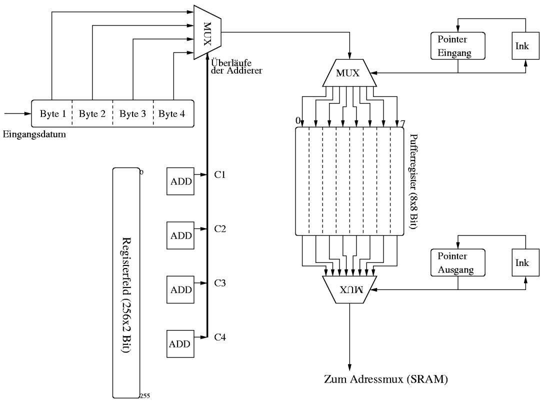

Bild 5 Pufferregister

Nach dem Aufsummieren der Symbolhäufigkeiten kann der Codebaum aufgebaut werden, der die Symbole des Codealphabetes als Blätter enthält. Der Aufbau des Codebaums beginnt mit dem Erstellen einer Liste, in der die Symbole und deren Wahrscheinlichkeiten abgelegt sind. Hierbei kann von der unsortierten Liste, die die Auftrittshäufigkeit der einzelnen Symbole enthält, ausgegangen werden. Aus dieser Liste werden jeweils die beiden Minima gestrichen und als Nachfolger eines noch zu bildenden Knotens in den Codebaum eingesetzt. Der neue Knoten wird in die Liste der Wahrscheinlichkeiten integriert und erhält als Summe die Wahrscheinlichkeiten der gelöschten Minima. Enthält die Liste nur noch ein Element, wird dieses als Wurzelelement des Baumes gewählt.

Der Berechnungsaufwand dieses leicht umzusetzenden Verfahrens wächst quadratisch in der Symbolanzahl. Alternativ hierzu wurde ein Heap-Sort integriert, der in O(*n log n*) arbeitet. Die maximale Taktfrequenz des gesamten Komprimierers sinkt jedoch bei Einsatz eines Heap-Sorts auf einem 4085XL-1 FPGA von 12 MHz auf 6 MHZ, der Ressourcenbedarf bleibt in etwa konstant bei 1200 CLBs.

Bei der Zuweisung der Codewörter wird der Baum von der Wurzel aus durchlaufen. Erreicht der Durchlauf ein Blatt, kann der Symbolname als Adresse und die Symbolkodierung als zu speicherndes Datum betrachtet werden. Die Eingangsdaten, die zur Häufigkeitsbildung genutzt wurden, können nun aus dem SRAM ausgelesen werden und in linearer Zeit kodiert werden.

Der Vergleich der Leistungsfähigkeit der Huffman-Komprimierung in Hardware und Software wird im folgenden unter der Annahme geführt, daß beide Einheiten mit derselben Taktfrequenz arbeiten können. Diese Annahme ist für Systeme, in denen der Prozessor als ASIC ausgeführt und der Huffman-Komprimierer auf einen FPGA abgebildet wird, natürlich in dieser Form nicht direkt haltbar. Bedenkt man jedoch, daß die Abbildung der Hardwareeinheit auf einen aktuellen Xilinx Virtex FPGA bereits eine maximale Taktfrequenz von 36 MHz hat und nur 10% der Ressourcen des Bausteins beansprucht und weiterhin FPGAs mit immer größeren Speicher auf den Bausteinen ausgeliefert werden, kann durch die parallele Instanziierung mehrerer Komprimierer dieser Nachteil wieder ausgeglichen werden.

Die Bearbeitungszeiten sind sowohl von der Anzahl der möglichen und verwendeten Symbole, als auch von der Größe der Datenblöcke abhängig. Praktisch läßt sich bei Datenblöcken der Größe 20 KByte durch den Hardware-Entwurf der Faktor 25 gewinnen. Dabei können bei einer Taktfrequenz von 1 MHz bis zu 1,2 MBit/s komprimiert werden. D.h., daß durch den Einsatz eines Virtex FPGAs mit einer Taktfrequenz von 36 MHz und mit zwei instanziierten Komprimierungseinheiten bereits bis zu 86 MBit/s komprimiert werden können, was den Anforderungen an die Online-Komprimierung eines Fast-Ethernet-Kanals entspricht. Werden die Baumbildung und die Baumausgabe in Software durchgeführt, kann man des weiteren bei einem Zeitgewinn um den Faktor 7 gegenüber der Software-Lösung die Hälfte des Flächenbedarfs einsparen. In diesem Fall wird jedoch der Prozessor nicht komplett entlastet und eine Skalierung des Ansatz auf mehrere parallele Blöcke ist nicht möglich.

2.3 Schnittstellen zu den Festplatten und den lokalen Netzwerken

Bestimmendes Element bei der Auslegung der aktiven Routingknoten sind die Schnittstellen zu den Festplatten und zu den angeschlossenen Dateiservern. Durch die Übertragungsbandbreite dieser Komponenten wird die notwendige Rechengeschwindigkeit der auf dem aktiven Routingknoten anfallenden Aufgaben sowie die auf den internen Verbindungen notwendige Übertragungsbandbreiten festgelegt.

Als Anschlußmedium für die Dateiserver wurden Fast-Ethernet-Verbindungen ausgewählt. Sie bieten zum einen bei großen SANs mit räumlich verteilten Dateiservern besser Eigenschaften als herkömmliche SCSI-Verbindungen und werden durch neu aufkommende Techniken wie SCSI-über-TCP/IP immer attraktiver. Alternativ hierzu hätten auch Fibre-Channel-Schnittstellen gewählt werden können. Diese hätten das Einsatzgebiet der aktiven Routingknoten jedoch auf den Anschluß an reine Datei-Server eingeschränkt, die Verwendung als Network Attached Storage wäre in dieser Form nicht möglich.

Im Rahmen des Presto-Projektes wurde die MAC-Schicht des IEEE 802.3-Standards in VHDL implementiert. Zum Anschluß an umgebende Einheiten wurde auch hier der Amba AHB-Bus verwendet. Die Kommunikation zwischen dem Businterface und der Transmit-, bzw. Receive-Einheit findet über Dual-Ported RAMs statt. In der aktuellen Version verfügt der Kontroller noch nicht über eine eigene DMA-Einheit, eingegangene Pakete werden über Interrupts oder Polling erkannt. Die Programmierung des Ethernet-Kontrollers findet über einen Registersatz statt, der auch eine Schnittstelle zu den MII Management Registern der physikalischen Schicht des IEEE 802.3-Standards bereitstellt.

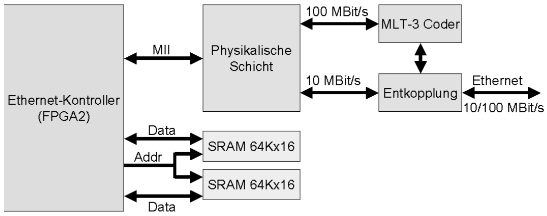

Bild 6 Integration des Ethernet-Kontrollers in die Prototypenumgebung

Der Fast-Ethernet Controller wurde ebenfalls in der in [6] vorgestellten Prototypenumgebung neben dem M°Core-Prozessorkern auf einem Xilinx XC4085XL FPGA abgebildet. Dabei werden 22% der verfügbaren CLBs des FPGAs verwendet. Diese Umgebung wurde um eine Erweiterungskarte ergänzt, die die physikalische Schicht des Ethernet-Protokolls und Paketspeicher enthält (siehe **Bild 6**). Die Übertragungsrate ist dabei durch den niedrig getakteten Prozessorkern auf 10 MBit/s begrenzt. Das System wurde in einer heterogenen Umgebung aus Workstations und PCs mit TCP/IP getestet.
Neben der Integration des Fast-Ethernet-Kontrollers ist die Bereitstellung eines Fast-SCSI-Anschlusses mit einer Datenübertragungsrate von 160 MBit/s geplant. Es ist nur sinnvoll, die Geschwindigkeit des SCSI-Anschlusses zu erhöhen, wenn entweder mehrere Fast-Ethernet-Ports pro aktiven Router bereitgestellt werden oder ein Gigabit-Ethernet-Modul in den aktiven Routingknoten integriert wird.

2.4 Aufbau des internen Verbindungsnetzwerkes

Der Aufbau der Verbindungen zu den Festplatten und den lokalen Netzwerken ist durch die Spezifikationen der zugehörigen Protokolle in weiten Teilen vorgegeben. Ein freier Parameter ist der Aufbau des Verbindungsnetzwerkes zwischen den aktiven Routingknoten.
Die Verschaltung der Routingknoten bestimmt bereits zu einem großen Teil die notwendige Übertragungsbandbreite der internen Verbindungen, die bei einer optimalen Auslastung der Festplatten und Anschlüsse zu den Abnehmern auftreten. In diesem Abschnitt sollen Simulationsergebnisse vorgestellt werden, die Aussagen über die notwendige Dimensionierung der internen Verbindungen machen und es soll der Entwurf eines Transceivers vorgestellt werden, der unsere Anforderungen bzgl. der Integration in den aktiven Routingknoten erfüllt.

2.4.1 Anforderungen an die internen Verbindungen

Bei den folgenden Untersuchungen der notwendigen Übertragungsbandbreiten der internen Verbindungen wird davon ausgegangen, daß die Bandbreiten zu den Festplatten und den lokalen Netzwerken ausreichend groß sind, daß diese nicht den Engpaß bei den Untersuchungen darstellen. Für beide Anschlußarten wird von einer Übertragungsrate von 2 GBit/s ausgegangen. Durch in [3] vorgestellten randomisierten Zugriffsstrategien wird sichergestellt, daß die Verteilung der Anfragen gleichmäßig über die Festplatten erfolgt und ein pseudo-zufälliges Anfragemuster entsteht. Die durchschnittliche Paketgröße beträgt 65.536 Byte und es können in jedem Puffer bis zu 18 Pakete gespeichert werden. Es werden bei der Wegewahl kürzeste Wege und als Switchingstrategien FIFO, LIS, ELIS und das Growing Rank Protokoll verwendet [11]. Jede der Verbindungen des aktiven Routingknotens hat eine Bandbreite von 500 MBit/s. Hier vorgestellte Netzwerke sind ein Butterfly-Netzwerk mit 192 Knoten und ein vollbesetztes Gitter mit 32 Knoten.
Ein Indikator für die maximale Auslastbarkeit eines Netzwerkes ist die Latenzzeit, bis eine Anfrage an das System beantwortet wird. In **Bild 7** ist die durchschnittliche Latenzzeit für verschiedene Injektionsraten in das System für ein Butterfly-Netzwerk angegeben. Hieraus ist zu erkennen, daß für das vorgegebene Butterfly-Netzwerk bei maximal 750 MBit/s die Latenzzeiten für alle untersuchten Switching-Protokolle stark ansteigen. Bei einer geforderten Auslieferungsrate des Netzwerkes von 100 MBit/s pro ausliefernden aktiven Routingknoten bedeutet dieses, daß für das Butterfly-Netzwerk nur eine interne Übertragungsrate von mindestens 75 MBit/s bereitgestellt werden muß. Hierbei ist zu beachten, daß nicht an jedem Knoten eine Festplatte hängt und die Daten über zwei Verbindungen pro Knoten abfließen können. Beim 8x4 Gitternetzwerk steigt die erforderliche Bandbreite bereits auf 200 MBit/s pro Verbindung an.

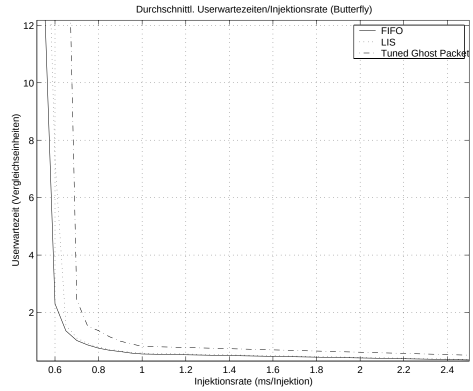

Bild 7 Latenzzeiten im Butterfly-Netzwerk in Abhängigkeit von der Injektionsrate

2.4.2 Aufbau der internen Verbindungen

Für den Aufbau des internen Verbindungsnetzwerkes wurde ein serieller Transceiver entwickelt, der als analoge Komponente im Vollkundenentwurf realisiert wurde [10]. Der Transmitter basiert dabei auf einer PLL, die auf Basis eines 10 Bit Eingangsdatenstroms einen seriellen Ausgangsstrom und einen zugehörigen Takt erzeugt. Der Receiver wandelt diese Daten mit Hilfe einer DLL wieder in parallele Daten zurück. Der Transceiver wurde im Rahmen des Europractice-Programms in einer 0,6 µm Technologie gefertigt. Der Baustein hat eine maximale Übertragungsrate von 900 MBit/s bei einer Versorgungsspannung von 3,3 Volt und einer Stromaufnahme von 90 mA. Der eigentliche Kern des Transceivers hat eine Fläche von nur 410 x 200 µm^2 für den Receiver und von 530 x 320 µm^2 für den Transmitter. Auf dem Baustein sind zu Testzwekken zwei Transceiver integriert, die gemeinsam mit den Anschlußpads eine Fläche von 10 mm^2 einnehmen. Durch seine kompakten Ausmaße und seinen geringen Stromverbrauch ist er optimal für eine spätere Integration in den aktiven Router geeignet.

Diese Arbeit wurde teilweise durch den DFG Sonderforschungsbereich 376, Projekt C5, und die Infineon Technologies AG gefördert.

3 Literatur

[1] ARM: AMBA™ Specification (Rev. 2.0), 1999

[2] D.S. Alexander, M.W. Hicks, P. Kakkar und A. Keromytos: The SwitchWare Active Network Implementation. In: The 1998 ACM SIGPLAN Workshop on ML, 1998

[3] P. Berenbrink, A. Brinkmann und C. Scheideler: Design of the PRESTO Multimedia Storage Network. In: Proceedings of the Workshop on Communication and Data Management in Large Networks (INFORMATIK 99), 1999

[4] P. Berenbrink, A. Brinkmann und C. Scheideler: Distributed Path Selection for Storage Area Networks. In: Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA´2000), 2000

[5] P. Berenbrink, A. Brinkmann und C. Scheideler: SimLab - A Simulation Environment for Storage Area Networks, Technischer Bericht, Universität Paderborn, 2000

[6] A. Brinkmann, D. Langen und U. Rückert: A Rapid Prototyping Environment for Microprocessor based System-on-Chips and its Application to the Development of a Network Processor. In: Proceedings of the 10th International Conference on Field Programmable Logic and Applications (FPL-2000), 2000

[7] A. Brinkmann, K. Salzwedel und C. Scheideler: Efficient, Distributed Data Placement for Storage Area Networks. In: Proceedings of the 12th ACM Symposium on Parallel Algorithms and Architectures (SPAA´00), 2000

[8] F. Gebauer.: Ressourceneffiziente Implementierung der Huffman-Komprimierung. Diplomarbeit, Fachgebiet Schaltungstechnik, Universität Paderborn, 2000

[9] G. Gibson, D. Nahle und K. Amiri: File Server Scaling with Network-Attached Secure Disks. In: Proceedings of the ACM International Workshop on Measurement and Modeling of Computer Systems (Sigmetrics ´97), 1997

[10] I. Hehemann: PLL/DLL basierte Daten- und Taktrückgewinnungsschaltungen in CMOS-Technologie. Diplomarbeit, Fachgebiet Schaltungstechnik, Universität Paderborn, 2000

[11] C. Scheideler und B. Vöcking: From Static to Dynamic Routing: Efficient Transformations of Store-and-Forward Protocols. In: Proc. of 31st ACM Symposium on Theory of Computing (STOC), 1999

[12] R. Sedgewick: Algorithmen in C++, Addison Weseley Publishing Company, 1992