Workshop der ITG/GI Fachgruppe "Architektur für hochintegrierte Schaltungen" in Zusammenarbeit mit der ITG/GI Fachgruppe "Parallele Algorithmen und Rechnerstrukturen".

2.-3. Juli 1992, Schwäbisches Bildungszentrum Irsee.

Hochintegrierte Schaltungen für Parallelarchitekturen.

Kurzfassung für den Vortrag:

Ein Pipelinebus für Parallelrechner zum schnellen Laden und Entladen von Daten.

Bernhard Lang
TU Hamburg-Harburg
Technische Informatik I
Harburger Schloßstraße 20
2100 Hamburg 90

---

1 Einleitung

Moderne Parallelrechner bieten durch den Einsatz leistungsfähiger Einzelprozessoren hohe Rechenleistungen zur Bearbeitung paralleler Algorithmen. Zur Ausnutzung dieser hohen Rechenleistungen ist die Auslegung der Kommunikationsverbindungen im Rechner von entscheidenter Bedeutung. Zu deren Auslegung müssen verschiedene Anforderungen berücksichtigt werden. Die wichtigsten Anforderungen stellen sich durch die Ausführung paralleler Programme und durch das Laden und Entladen von Massendaten vor und nach dem Programmlauf.

Während der Ausführung paralleler Programme wird ein Verbindungsnetzwerk zwischen den Prozessoren benötigt, welches ein schnelles Austauschen von Zwischenergebnissen ermöglicht. Geeignete Topologien für dieses Netzwerks werden durch die implementierten Algorithmen bestimmt. Modere Parallelrechner bieten zu diesem Zweck fest eingestellte Netze mit kleinem Durchmesser oder schaltbare Verbindungsnetzwerke an [1, 2].

Zur Initialisierung und Terminierung einer parallelen Programmausführung müssen Daten sehr schnell in einen Parallelrechner geladen bzw. aus diesem entladen werden. Dazu müssen während der Initialisierung große Datenobjekte

(Felder, Dateien, Bilder, etc.) auf parallele Prozessoren aufgeteilt oder komplett in jeden Prozessor geladen werden. Diese Vorgänge werden als 'Scattering' und als 'Broadcast' bezeichnet. Zur Terminierung werden errechnete Ergebnisdaten, die verteilt auf parallelen Prozessoren vorliegen, zu einem großen Datenobjekt zusammengefügt. Dieser Vorgang wird als 'Gathering' bezeichnet [3].

Bei der Bewertung von Ausführungszeiten paralleler Algorithmen wird oftmals nur die Ausführung der Programme mit der eingeschlossenen algorithmischen Kommunikation berücksichtigt, nicht jedoch deren Initialisierung und Terminierung [4]. Bei kleinen Datenmengen und langen Berechnungszeiten ist diese Vernachlässigung zulässig. Bei großen Datenmengen und feiner Verteilung der Daten auf die Prozessoren führt diese Vernachlässigung jedoch zu Effizienzen, die in der realen Implementierung nicht erreicht werden.

Die Effizienz der Ausführung eines Algorithmus auf einem Multiprozessorsystem basiert somit auf der parallelen Berechnung und auf der Initialisierung [5]. Insbesondere zur Ausführung von Algorithmen, welche große Datenobjekte bearbeiten, müssen in Parallelrechnern schnelle und flexible Kommunikationsverbindungen zum Laden und Entladen der Massendaten vorgesehen werden.

2 Ein paralleler Bildverarbeitungsrechner

An der TUHH wurde in den vergangenen Jahren der Prototyp eines Parallelrechners entworfen und realisiert, welcher insbesondere die Implementierung komplexer Bildverarbeitungsalgorithmen unterstützt [6, 7]. In seiner Architektur sind die Anforderungen der parallelen Programmausführung sowie der Initialisierung und Terminierung berücksichtigt. Seine Kommunikationspfade sind für sehr hohe Datenraten, wie sie bei der Verarbeitung von Bilddaten aber auch bei anderen Applikationen auftreten, ausgelegt.

Die parallele Programmausführung wird durch die Verwendung von Transputern flexibel ermöglicht. Mit deren integrierten Link-Schnittstellen lassen sich Verbindungsnetze einstellen, die auf jeweilige Algorithmen abgestimmt sind.

Zum schnellen Laden und Entladen von Massendaten besitzt das System eine separate Kommunikationsstruktur, einen schnellen, getakteten Pipelinebus. Er ermöglicht bei voller Übertragungsrate ein 'Broadcast' oder ein beliebig einstellbares 'Scattering' sowie ein 'Gathering' beliebig verteilter Ergebnisdaten.

3 Ein schneller Pipelinebus

Die Funktionalität des Pipelinebusses wird durch Markieren jedes übertragenen Datenworts erreicht. Jedem Datenwort wird ein Adress- und ein Kontrollfeld hinzugefügt. Die Einheit aus Kontrollfeld, Adreßfeld und Datenwort soll im folgenden als Token bezeichnet werden.

Bei der Übertragung eines Datenstroms kann diesem synchron ein Markierungsstrom aus Adreß- und Kontrollfeldern zugefügt werden, der die Verteilung der Daten steuert. Mit identischer Markierung aller Worte eines Datenstroms wird ein 'Broadcast' realisiert; mit einem Markierungsstrom, dessen Adreßfelder die gewünschte Verteilung eines Datenstroms auf parallele Prozessoren be-

schreiben, läßt sich ein schnelles 'Scattering' durchführen. Durch Stimulieren paralleler Prozessoren mit einem Markierungsstrom, dessen Adreßfelder die Verteilung von Ergebnisdaten beschreiben, ist weiterhin ein synchrones 'Gathering' möglich. Die parallelen Prozessoren müssen dann — stimuliert durch diese Markierungen — einen Strom der Ergebnisdaten ausgeben. Zur Unterscheidung der verschiedenen Adreßinterpretationen dienen die Kontrollfelder.

Zur Steuerung des Datenstroms nach obigem Schema benötigt jeder Prozessor im parallelen System ein Pipelinebusinterface. Dieses Interface empfängt ankommende Token und vergleicht deren Adreßfelder mit einer eigenen Pipelinebusadresse. Gemäß der Interpretation, die auf Kontrollteil und Adreßvergleich basiert, werden diese entweder in den Prozessor eingelesen, stimulieren eine Datenausgabe des Prozessors oder werden an den folgenden Prozessor in der Pipeline weitergereicht.

4 Realisierungen des schnellen Pipelinebusses

Erste Realisierungen des schnellen Pipelinebusses basieren auf diskret aufgebauten endlichen Automaten, welche die beschriebenen Verarbeitungen in einem Taktzyklus vornehmen. Der Hardwareaufwand dieser Realisierungen ist hoch. Die Taktrate des Busses beträgt 10 *MHz*, was bei einer Wortbreite von einem Byte zu einer Übertragungsrate von 10 *MByte/sec* führt.

Eine neue Realisierung des Busses basiert auf modernen FPGA-Bausteinen. Durch Pipelinisierung des ursprünglichen Entwurfs kann das Interface in einen einzigen LCA-Baustein XC3090 abgebildet werden [8]. Die Adaption des zugehörigen Prozessors erfolgt über FIFOs und Video-RAM Speicher. Deren Kopplung benötigt einen weiteren LCA-Baustein XC3064. Mit dieser FPGA-Realisierung wird der Hardwareaufwand deutlich verringert bei gleichzeitiger Steigerung des Bustakts um ca. Faktor 2.

Für einen vielfältigen Einsatz als Standardinterface bietet sich die Realisierung als ASIC-Baustein an. Nach vorsichtigen Schätzungen kann die obige FPGA-Realisierung des Interface einschließlich FIFOs und Kopplungslogik in einem Baustein untergebracht werden. Dabei ist aufgrund der Pipelinisierung eine weitere Steigerung des Taktes und eine Steigerung der Datenwortbreite möglich. Ein erster Entwurf dieses Pipelinebus-ASIC ist in Arbeit.

Literatur

[1] Oliver A. McBryan and Eric F. Van de Velde: Matrix and Vector Operations on Hypercube Parallel Processors. Parallel Computing, Nr. 5, Seiten: 117-125, 1987.

[2] Falk D. Kübler: A cluster-oriented Architecture for the Mapping of Parallel Processor Networks to High Performance Applications. In: Handouts, Economical Parallel Processing, 31. Mai-1. Juni 1988, Bern.

[3] Youcef Saad and Martin H. Schultz: Data communication in parallel architectures. Parallel Computing, Nr. 11, Seiten: 131-150, 1989.

[4] Horace P. Flatt and Ken Kennedy: Performance of Parallel Processors. Parallel Computing, Nr. 12, Seiten: 1-20, 1989.

[5] Hans Burkhardt, Bernhard Lang und Michael Nölle: Aspects of Parallel Image Processing Algorithms and Structures. In H. Burkhardt, Y. Neuvo und J.C. Simon, Editoren, From Pixels to Features II, Parallelism in Image Processing, ESPRIT BRA 3035 Workshop, Bonas, September 1990.

[6] Bernhard Lang: Ein paralleles Transputersystem zur digitalen Bildverarbeitung mit schneller Pipelinekopplung. In *11. DAGM-Symposium Mustererkennung, Hamburg*, Seite 372–379, Berlin Heidelberg, 1989. Springer Verlag.

[7] Bernhard Lang: Digitale Bildverarbeitung mit kombinierter Pipeline- und Parallel-Architektur. Dissertation TU Hamburg-Harburg, 1991.

[8] Xilinx Inc: The Programmable Gate Array Data Book, 1988.