Aufbau- und Strukturkonzepte einer adaptiven multigranularen rekonfigurierbaren Hardwarearchitektur

Alexander Thomas, Jürgen Becker

Institut für Technik der Informationsverarbeitung (ITIV)
Universität Karlsruhe
Engesserstr. 5
76128 Karlsruhe, Germany
thomas@itiv.uni-karlsruhe.de
becker@itiv.uni-karlsruhe.de

Abstract: Moderne Anwendungsszenarien aus den Bereichen der Multimediaanwendungen und Mobilkommunikation verlangen nach immer leistungsfähigeren Datenverarbeitungsarchitekturen mit immenser Rechenleistung, die durch aktuelle Ansätze wie Mikroprozessoren und DSPs nicht ohne weiteres erreichbar sind. Dieser Beitrag beschreibt ein neues Architekturkonzept aus dem Bereich der rekonfigurierbaren Architekturen [Kr01][Be01][XP02][XI01][AL01]['TR02][AT01][QU01][Go01][Ba01][Zh01], die im Rahmen des DFG Schwerpunktprogramms 1148 „Rekonfigurierbare Rechensysteme“ entwickelt werden und auf die Erforschung und Weiterentwicklung der existierenden array-basierten Ansätze gerichtet ist. Die gewonnenen Erfahrungen und Ergebnisse sollen in eine neue adaptive dynamisch rekonfigurierbare Architektur einfließen, die das Ziel dieses Projekts ist.

1 Einleitung

Die zunehmende Komplexität aktueller Anwendungen beispielsweise aus den Bereichen Multimedia und Mobilkommunikation mit ihren kontroll- und datenlastigen Ausprägungen erfordert stets ausgefeiltere Architekturlösungen, um eine rasche Ausführung zu gewährleisten. Weitaus die meisten Systeme werden heutzutage aus Prozessoren und DSPs zusammengesetzt, welche den Kern zur Datenverarbeitung bilden. Darüber hinaus existieren nur wenige Ansätze, die gegebene Möglichkeiten der modernen Mikroelektronik ausnutzen und neue Wege in der Datenverarbeitung einschlagen.

Konzepte der Mikroprozessoren und DSPs basieren auf sequentieller Ausführung von Instruktionen verbunden mit Lade- und Speichervorgängen von Daten in und aus dem Speicher. Durch diese Vorgehensweise ergeben sich konzeptbedingte Vor- und Nachteile. Die Vorzüge dieses Konzepts liegen in der guten Fähigkeit auf nicht vorhergesehene bedingte Instruktionsabläufe zu reagieren, wobei auch hier Abschläge speziell bei Pipeline-Architekturen in Kauf genommen werden müssen. Größere Pipelinetiefen fordern an dieser Stelle ihren Tribut, da bei einer Verzweigung zuvor falsch gefüllte Pipelinestu-

fen erneut aufgefüllt werden müssen und somit Taktzyklen kosten. In Kombination mit wahlfreiem Speicherzugriff wird eine derartige Architektur in die Lage versetzt, kontrollastige Applikationen hervorragend auszuführen.

Die Nachteile dieser Architektur ergeben sich durch die häufigen Speicherzugriffe beim Instruktioneneinlesen und Daten lesen bzw. schreiben. Diese Strategie führt zwangsläufig zu einem Bandbreitenengpass, da sich dadurch ein reger Austausch zwischen der Datenverarbeitungseinheit und dem Arbeitsspeicher ergibt. In modernen Prozessoren wird diesem Problem durch den Einsatz von Caches entgegengewirkt. Die Caches greifen jedoch nur bei der Verarbeitung von kleinen Datenblöcken und entlasten die meist langsamer getakteten Speichermodule des Arbeitsspeichers, versagen jedoch völlig bei Anwendungen mit großem Datendurchsatz, da durch die limitierte Größe die Caches in diesem Umfeld keine Daten vorhalten können. Auf diese Weise sinkt der effektive Durchsatz der Datenverarbeitungseinheit auf das Niveau des Speicherinterfaces und die Gesamtperformanz des Systems wird limitiert.

Innerhalb der applikationsspezifischen integrierten Schaltungen (ASICs) kann der Bandbreitenlimitierung entgegengewirkt werden, indem die erforderlichen Funktionseinheiten in der gewünschten Anzahl implementiert und die Daten während der Verarbeitung von einem Modul zum nächsten weitergereicht werden. Die beim Prozessor erforderlichen Speicherzugriffe würden hier zwischen den Modulen stattfinden und entfallen beim ASICs völlig. Dadurch erreicht der ASIC strukturbedingt einen höheren Ausnutzungsgrad der Funktionseinheiten und benötigt eine niedrigere Ein- und Ausgangsbandbreite. Die Möglichkeit Datenverarbeitung parallel zu betreiben, eben durch die Zusammenschaltung der unterschiedlichen Funktionseinheiten zusammen mit den oben erwähnten Vorteilen, befähigt ASICs zu sehr hoher Leistungsfähigkeit. Die Nachteile dieses Konzepts liegen in der Entwicklung der Schaltkreise und speziell in der teueren Fertigung. Durch die sehr starke Anwendungsabhängigkeit und geringe Flexibilität der ASICs müssen diese stets neu entwickelt werden und treiben auf diese Weise die Kosten eines Systems stark in die Höhe.

Einen ähnlichen jedoch flexibleren Ansatz als bei den ASICs verfolgen die Konzepte der dynamisch rekonfigurierbaren Array-Architekturen. In diesem Bereich werden zwei Typen unterschieden: grobgranulare Architekturen [Kr01][XP01][XP02][XP03] [Be01][Be02][QU01] und feingranulare Architekturen [XI01]. Bei beiden Ausprägungen wird ein Satz aus Funktionseinheiten durch ein rekonfigurierbares Netzwerk verbunden. Sie unterscheiden sich lediglich in der Bitbreite der Datenverarbeitung. Grobgranulare Architekturen arbeiten auf Bitvektorebene und nutzen zur Datenmanipulation arithmetisch logische Einheiten, während feingranurale Architekturen Bitlevel Operationen auf der Basis von Look-Up-Tables unterstützen. Bei beiden Typen wird zur Laufzeit die exakte Funktion innerhalb des Arrays durch die Personalisierung der Funktionseinheiten und des Netzwerkes vorgenommen. Der Vorteil liegt klar auf der Hand: Durch die Abbildung der auszuführenden Algorithmen in die Fläche, wird ein hohes Maß an Parallelität erreicht und wie beim ASIC keine so hohe Bandbreite am externen Interface benötigt, wie das bei einem Prozessor der Fall wäre. Dadurch wird auch bei niedrigeren Frequenzen und Spannungen des Systems die erforderliche Performanz erreicht, was zu einem deutlich niedrigeren Stromverbrauch führt. Natürlich besitzen auch diese Architekturen

mehrere nicht zu vernachlässigende Nachteile. Aus strukturbedingten Gründen und durch den recht einfachen Aufbau der Funktionseinheiten, die bis auf einige wenige Ausnahmen [Be01] nur einen elementaren Funktionsumfang aufweisen, lassen sich kontrollastige Anwendungen nur schwer darauf ausführen. Meist wird das durch eine hohe Anzahl von Rekonfigurationen und einem großen Flächenverbrauch auf dem Array erkauft. Dabei gelangt die erreichbare Performanz in Leistungsbereiche, die in der Praxis nicht akzeptabel ist und den Aufwand für eine derartige Architektur nicht rechtfertigt. An dieser Stelle wäre eine höhere Flexibilität wünschenswert, die es erlaubt auch kontrollbehaftete Anwendungen effizienter auszuführen.

Dieses Ziel ist durch die Integration höherer Kontrollintelligenz in die Array-Struktur erreichbar. Die Funktionalität sowohl im Kommunikationsnetzwerk als auch innerhalb der Funktionseinheiten muss vergrößert werden. Die Richtung, die hier angedeutet wird, geht in die Migration der Prozessorsteuereinheit in das Array. Es muss ein optimaler Mittelweg zwischen dem Aufwand und gebotener Flexibilität sowie Performanz gefunden werden. Dieses Papier stellt neue Konzepte für eine neuartige Architektur vor. In diesem Projekt wird versucht, durch Untersuchungen und Erweiterungen entsprechender Array-Strukturkomponenten und aufbauend auf unseren Erfahrungen neuartige Ansätze zu entwickeln, welche im Array zur höheren Leistungsfähigkeit, verbesserter Flexibilität bei gleichzeitig niedrigem Energieverbrauch führen sollen.

2 Architekturkonzept im Überblick

Dieser Abschnitt beschreibt die Überlegungen und Analysen, die wir aufgrund von früheren Erfahrungen und aktuellen Arbeiten auf dem Gebiet der rekonfigurierbaren Architekturen angestellt haben. In den Unterabschnitten werden die Ergebnisse und neue Konzepte dieses Projektes vorgestellt. Die Unterabschnitte sind dabei unterteilt in drei Bereiche: Kommunikationsnetzwerk, Datenpfadstrukturen und Systemanbindung.

2.1 Adaptives Kommunikationsnetzwerk

Ein wichtiger Aspekt der rekonfigurierbaren Array-Architekturen ist die Anbindung der Funktionseinheiten an das Kommunikationsnetz. Hierzu wird ein flexibles Netzwerk benötigt, das auch genügend Kapazität bezüglich der Anzahl der Kanäle bietet. Herkömmliche Ansätze wie das ursprüngliche Kress-Array benutzten zum Aufbau der Kommunikationsstruktur horizontale und vertikale Leitungen. An den Schnittpunkten befinden sich spezielle Module, so genannte Switchboxen, die das Umschalten der Signale zwischen den Leitungen steuern. Die exakten Schalterstellungen werden dabei durch spezielle, zuvor in das Array eingespielte und vorberechnete Konfigurationsvektoren bestimmt. Nun sind in modernen Varianten solcher Arrays jeweils Zwischenregister in den Switchboxen vorgesehen, da die Signale nicht die volle Distanz zwischen zwei Funktionseinheiten innerhalb eines Taktes schaffen können und der Takt nach Möglichkeit nicht durch solche langen und demzufolge kritischen Pfade unnötig gesenkt wird. Durch den Einsatz der Zwischenregister bilden derartige Ketten Pipelinestrukturen, wobei nicht immer zwischen zwei vorkommenden Registern Funktionseinheiten auftreten

müssen. Des Weiteren bedarf es eines passenden Kommunikationsprotokolls, um die Datenkonsistenz im Array zu gewährleisten. Schließlich ist nicht immer davon auszugehen, dass in jedem Takt alle Register innerhalb einer Konfiguration gültige Daten beinhalten.

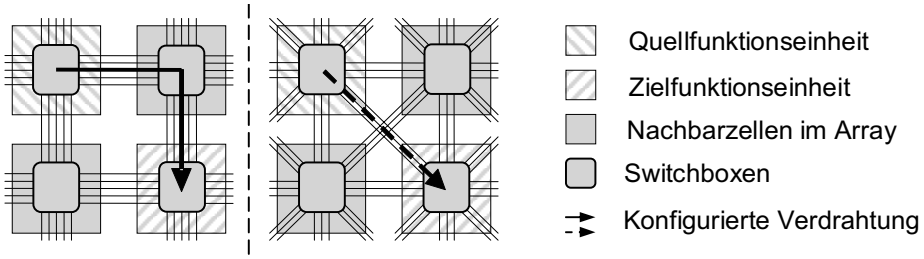

Abbildung 1: Diagonales Routing trägt zur Verkürzung der Latenzzeiten bei

Unserer Ansicht nach sind die horizontalen und vertikalen Kommunikationsleitungen nicht ausreichend, um einen effizienten Datenaustausch zu realisieren. Ein Beispiel für eine günstige Realisierung hierfür ist der Datentransport in diagonaler Richtung (siehe Abbildung 1). Möchte man in einem Kress-Array Daten diagonal routen, so müssen, um von einer Funktionseinheit ihren diagonalen Nachbarn zu erreichen, bereits zwei Hops in Kauf genommen werden. Dies kostet beim Einsatz von Zwischenregistern bereits zwei Takte, obwohl die eigentliche Distanz nur eine Einheit beträgt. Das Hinzufügen von einfachen diagonalen Verbindungen erhöht die Komplexität der benötigten Switchboxen auf mindestens das Doppelte, da die Gesamtzahl der Richtungen von vier auf acht steigt. Unser Vorschlag, um diesem Problem abzuhelfen, ist der Einsatz von sechs Routingrichtungen pro Switchbox. Die gesamte Struktur des Arrays würde auf diese Weise einen hexagonalen Charakter bekommen (siehe Abbildung 2). Durch diese Änderung würde sich die Anzahl der direkten Nachbarn zwischen den Zellen erhöhen und das zuvor angesprochene Problem beseitigt werden. Innerhalb der neuen Struktur ergibt sich für jede Zelle eine Gruppe von sechs Nachbarzellen mit der Erreichbarkeit von einem Takt, eine weitere Gruppe mit zwölf Zellen mit der Erreichbarkeit von zwei Takten, usw. (siehe Abbildung 3). Jede Gruppe, angefangen bei der ersten und gefolgt von der nächst größeren, schließt die Bezugszelle hexagonal ein. Durch diese praktische Eigenschaft des hexagonalen Aufbaus wird die Erreichbarkeit einer jeden Zelle und somit die Latenzzeit zwischen den Funktionseinheiten im Array homogenisiert und gleichzeitig im Durchschnitt verkürzt.

Wie bereits oben beschrieben werden in den Switchboxen zur Personalisierung der Schalterstellungen zur Weiterleitung der Daten Konfigurationsvektoren benötigt. Der Umfang der Konfigurationsdaten kann je nach Länge der Leitungen enorm sein. Diese Daten werden im Zuge der Anwendungsentwicklung entweder vom Entwickler manuell erstellt oder von entsprechenden Tools, wie Compiler oder Mapper, aus einer abstrakteren Beschreibung generiert. Diese Daten benötigen im Arbeitsspeicher zusätzlichen Platz. Da alle Funktionseinheiten in einem rekonfigurierbaren Array durch Koordinaten charakterisiert sind, wäre an dieser Stelle die Ausnutzung der Koordinateninformationen vorstellbar, um der Hardware eigenständiges Routing zu ermöglichen. Zu diesem Zweck

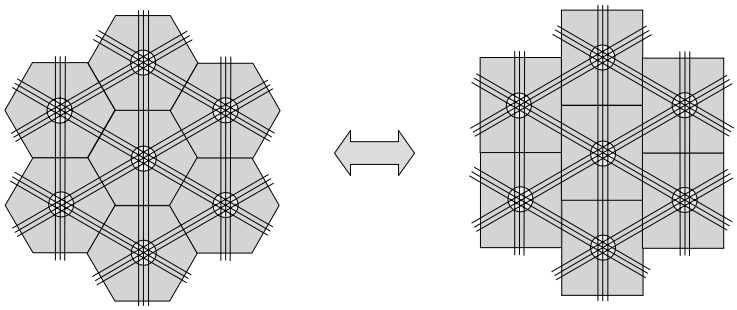

Abbildung 2: Hexagonaler Aufbau (links logisch; rechts physikalisch)

müssen die Switchboxen um einen Routingcontroller erweitert werden, der genügend Intelligenz aufweist, um diese Aufgabe zu bewältigen.

Die Probleme, die in diesem Zusammenhang zu lösen sind, ähneln sehr stark der Problematik des Internetroutings. In diesem Fall kommt jedoch der homogene und statische Aufbau des Arrays der Machbarkeit zu Gute. Die benötigten Algorithmen sollten wesentlich einfacher aufgebaut sein. IP-Router im Internet besitzen Tabellen mit Routinginformationen. Zum einen sind diese Tabellen recht umfangsreich und müssen darüber hinaus während des Betriebes aufgebaut und aktualisiert werden. Diese Vorgänge sind allerdings in einem statisch aufgebauten Array nicht notwendig, da sich die Struktur und Position der Teilnehmer nicht ändert. Es ist davon auszugehen, dass entsprechend aufgebaute FSMs innerhalb der Switchboxen mit eincodierten Information über die eigene Position und die Ausprägung des Arrays in der Lage sein sollten Routingaufgaben zu übernehmen. In diesem Zusammenhang ist es jedoch weiterhin notwendig, die Kommunikationsmittel, die benötigt werden, um Switchboxen kommunizieren zu lassen, festzulegen. Lässt man Routingcontroller innerhalb der Switchboxen Informationen austauschen, so wird kein Vorhalten der Informationen, beispielsweise über die Verfügbarkeit der Switchboxen, notwendig. Dies trägt dazu bei, wertvolle Fläche auf einem Chip einzusparen. Passende Algorithmen müssen hierfür in weiteren Arbeiten untersucht und bewertet werden. Abbildung 3 zeigt ein Beispiel für das adaptive Routing zur Laufzeit. Das Umgehen der belegten Kanten wird hierbei automatisiert vollzogen. Der Clou dabei ist eben, dass die Belegung der Ressourcen von fremden Konfigurationen stammen kann und somit der exakte Status der Belegung dem Mapper zur Compilezeit nicht bekannt sein muss.

Ein weiterer wichtiger Punkt im Bereich der Kommunikationsstrukturen betrifft die Trennung der Nutzdatennetzwerke von Konfigurationsnetzwerken. In Architekturen wie XPP [XP02] oder DReAM [Be01] wird zur Übertragung von Konfigurationen in das Array ein eigens dafür vorgesehenes Netzwerk eingesetzt. Dieses Konzept verfolgt den Zweck ungestörtes Übertragen von Konfigurationen unabhängig von der Auslastung des Nutzdatennetzwerkes zu sichern. Allerdings bedeutet das, dass hier eine Spezialisierung des Netzwerkes erreicht wird, welche nur einem Zweck dient und bei Bedarf nicht anderweitig einsetzt werden kann. Darüber hinaus wird das Konfigurationsnetz meist aus einer Quelle versorgt [XP03] und ist somit einer weiteren zu vermeidenden Bandbreiten-

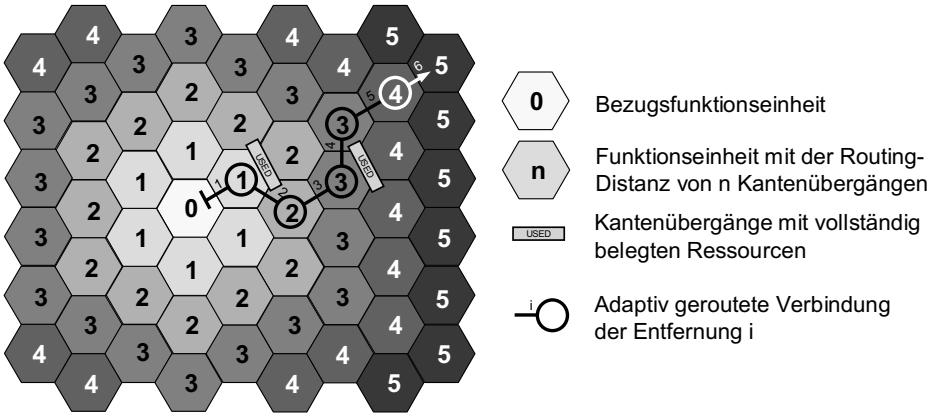

Abbildung 3: Hexagonaler Array-Aufbau in Kombination mit adaptivem dynamischem Routing

limitierung unterworfen, welche eine minimale Rekonfigurationszeit festlegt und bei Bedarf nicht beschleunigt werden kann.

Diese Überlegungen haben uns dazu bewogen ein universelles Kommunikationsnetzwerk zu implementieren. Die Einsparungen, welche sich durch das Wegrationalisieren des spezialisierten Konfigurationsnetzwerkes ergeben würden, lassen sich in das universelle Kommunikationsnetz investieren und genügend Ressourcen für beide Datentypen, sowohl Nutzdaten als auch Konfigurationen, realisieren. In Verbindung mit mehrfach implementierten Bridges zur Anbindung an den Systembus, die in folgenden Abschnitten näher erläutert werden, lässt sich auf diese Weise stark parallelisiertes und leistungsfähiges Konfigurationenmanagement realisieren. Dazu ist eine entsprechende Erweiterung des Kommunikationsprotokolls nötig, um die gemeinsame Nutzung des Netzwerkes zu ermöglichen.

2.2 Multigranulare Datenpfadstrukturen

Bisherige Lösungen im Bereich der rekonfigurierbaren Systeme setzen einfach aufgebaute Funktionseinheiten ein. Dabei kommen meist arithmetisch logische Einheiten (ALUs) zum Einsatz, die elementare Operationen bieten und ähnlich wie in den Switchboxen durch einen Konfigurationsvektor zur Laufzeit charakterisiert werden. Die Konfigurationsdaten werden als Op-Codes den ALUs zugeführt und selektieren auf diesem Weg die gewünschte Operation in der betreffenden Funktionseinheit. Durch dieses einfache Prinzip lassen sich sehr effiziente Strukturen im Array programmieren, die zur Ausführung von datenlastigen Algorithmen hervorragend geeignet sind. Werden jedoch einfache bedingte Verzweigungen benötigt, so reicht diese Struktur nicht aus.

Die Firma PACT XPP Technologies AG geht an dieser Stelle mit der XPP Architektur [XP02][XP03] einen neuen und interessanten Weg. Zum Übertragen von Kontrollinformationen wurde in der XPP ein eigens dafür bestimmtes Event-Netzwerk vorgesehen. Dieses Netzwerk besteht aus Ein-Bit-Bussen, welche Ergebnisse von Bedingungen übertragen können. Diese Informationen können innerhalb der XPP zur Steuerung von Mul-

tiplexern und Demultiplexern genutzt werden und erweitern die Architektur zur Unterstützung von Kontrolloperationen. Dies befähigt die Architektur natürlich nicht zur Ausführung von stark kontrollastigen Anwendungen, jedoch können damit bereits relativ günstige Kontrollmechanismen aufgebaut werden.

Der Aufbau einer XPP Funktionseinheit, so genanntes „processing array element“ (PAE) besteht innerhalb dieser Architektur aus drei Objekten. Zwei äußere Objekte eines PAE, die Backward und Forward Register (BREG und FREG), welche neben einfachen logischen und arithmetischen Operationen auch Datentransferaufgaben in vertikaler Richtung übernehmen. Komplexere Operationen wie Additionen und Multiplikationen werden innerhalb einer speziell dafür ausgeprägten ALU-PAE im mittig platzierten ALU-Objekt ausgeführt. Diese drei beschriebenen Objekte in einer PAE werden oben und unten durch horizontale Busse verbunden und tauschen auf diesem Wege innerhalb und außerhalb des PAE Daten aus. Diese Struktur erlaubt bereits den Aufbau komplexer Strukturen in der XPP und leitet unserer Meinung nach die richtige Richtung in der Entwicklung im Bereich der rekonfigurierbaren Arrays ein.

Die Tendenz in der Forschung und Entwicklung muss also die Richtung komplexerer Funktionalität der Funktionseinheiten einschlagen. Es ist unschwer zu erkennen, dass rein array-basierte Lösungen in diesem Bereich keine praktischen Vorteile für Anwendungsunterstützung bringen. Kontrollastige Algorithmen lassen sich auch auf einer XPP Architektur realisieren, wenn auch nur mühsam. Sie benötigen dafür sehr viel Fläche und nutzen die gebotene Parallelität einer solchen Architektur nur ungenügend aus. Häufig ist dafür ein zu häufiges Rekonfigurieren erforderlich, was die Leistungsfähigkeit drastisch senkt.

Die sequentielle Verarbeitung von Instruktionen in einem Prozessor ist hier wesentlich günstiger und erlaubt einfache und relativ unkomplizierte Verzweigungsoperationen. Begünstigt wird diese Eigenschaft durch den wahlfreien Speicherzugriff, der einem Prozessor zur Verfügung steht. Die Steuerung der Speicherzugriffe ist hierbei sowohl instruktions- als auch datenbedingt. Natürlich ist es völlig ausgeschlossen, dass jede Funktionseinheit Zugriff auf den Hauptspeicher erhält. Zum einen würde das eine gewaltige Bandbreite des Hauptspeichers voraussetzen und zum anderen einen hohen Verdrahtungsaufwand nach sich ziehen, ganz abgesehen von langen Signallaufzeiten. Ein kleiner interner Speicher innerhalb der Funktionseinheiten wäre jedoch vorstellbar, um Mikroprogramme lokal auszuführen. Die resultierende Flexibilität der Funktionseinheiten könnte anpassungsfähige Konfigurationen ermöglichen, die einen echten Fortschritt Richtung Ausführung kontrollastiger Algorithmen darstellen würde.

Ein weiterer Aspekt ist im Zusammenhang mit den Datenpfaden zu beleuchten: die Verarbeitungsgranularität. Hier stellt sich die Frage, ob eine homogene Granularität innerhalb einer Architektur zur gewünschten Anwendungseffizienz führen kann. Die Antwort auf diese Frage ist natürlich davon abhängig, in welchem Bereich diese Architektur eingesetzt wird. Werden jedoch Anwendungen mit verschiedenen Ausprägungen ausgeführt, so ist die Antwort klar nein. Aus diesem Grund ist die Entwicklung eines Datenpfades mit multigranularen Eigenschaften zwingend notwendig. Dieses Ziel kann im ersten Schritt erreicht werden, in dem mehrere Module mit unterschiedlichen Granulari-

täten in einem Datenpfad eingesetzt werden. Um den Nutzungsgrad der Module möglichst hoch zu halten, wird eine Datenpfadstruktur benötigt, die den gleichzeitigen Einsatz dieser Module ermöglicht. Hierzu würde sich eine ringförmige Struktur aus einem Satz an Registern und Modulen eignen (siehe Abbildung 4). Wie der Abbildung entnommen werden kann, sind zwei universelle CrossBars erforderlich, um die große Flexibilität zu gewährleisten. Im Allgemeinen sollte sich jedoch diese als ideal anzusehende Struktur verkleinern lassen, da sicherlich nicht alle Schaltkombinationen der CrossBars erforderlich sind. Es muss hier untersucht werden, welche sinnvollen Subsets der allgemeinen Datenpfadstruktur nützlich sind.

Aus der ringförmigen Struktur des Datenpfades ergeben sich allerdings auch zusätzliche Eigenschaften des Arrays, denn im Grunde entspricht ein derartiger Aufbau des Datenpfades einem lokalen Subnetz, welches eine zweite Hierarchieebene im Array repräsentiert. Wie oben bereits beschrieben, ist es durch die kompakte Struktur des Datenpfades möglich, komplexe Kontrollstrukturen zu integrieren. Über das globale Netzwerk lassen sich mehrere Subnetze verbinden und Kontrollinformationen austauschen. Diese Vorgehensweise würde komplexe Anwendungen ermöglichen. In Verbindung mit LUT-Modulen ließen sich beispielsweise auf eine einfache Weise FSM-Funktionalitäten in den Datenpfad konfigurieren.

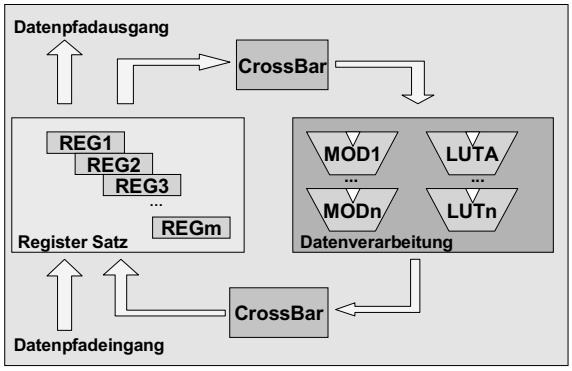

Abbildung 4: Ringförmige multigranulare Struktur des Datenpfades

Im Laufe dieses Projekts muss untersucht werden, ob die hier vorgestellten Datenpfadkonzepte sich sinnvoll und vor allem günstig in mikroelektronischen Schaltungen realisieren lassen und sich die gewünschte Flexibilität im Array bestätigt. Zweifelsohne geht die Richtung der Entwicklung in diesem Bereich in die Erweiterung der Funktionseinheiten bzw. Datenpfade, welche von uns in diesem Projekt verfolgt wird.

2.3 Systemanbindung

Ein weiterer wichtiger Aspekt im Bereich der rekonfigurierbaren Arrays ist die Anbindung an den Systembus. Hier kommen überwiegend einfach aufgebaute Bridges zum Einsatz mit Master und Slave Interfaces. Die Funktionalität beschränkt sich meist auf die Umsetzung der array-internen Protokolle auf den Systembus. Dabei arbeiten diese Kom-

ponenten relativ unselbständig. Slaves erwarten Daten von masterfähigen Systembusteilnehmen und können nicht eigenständig auf dem Bus interagieren. Die Einleitung der Transfers können Slaves im günstigen Fall über eigene Interruptleitungen beim Systemcontroller anmelden und bekommen auf diese Weise die Aufmerksamkeit der Systemkomponenten.

Bei masterfähigen Bridges verhält sich der Sachverhalt etwas anders. Zur Einsparung der Fläche auf dem Chip werden diese Komponenten ebenfalls sehr einfach gehalten und können nicht selbständig Adressen generieren, um auf dem Systembus autonom aufzutreten. Diese Aufgabe wird vom Array übernommen und über array-interne Adressgeneratoren mit nötigen Daten versorgt. Dieses Vorgehen ermöglicht jedoch ebenso wie der unflexible Charakter des Arrays nur relativ homogene Adressen zu erzeugen, was wiederum das Anwendungsfeld stark einschränkt.

Unsere Ideen gehen hier einen neuen Weg. Wir legen der Entwicklung des neuen Ansatzes die Vorstellung zugrunde, den Datenverarbeitungsteil und den Steuerungsteil eines Prozessors trennen zu können. Dabei wird der Datenpfad im Array realisiert. Die Steuerung der Verarbeitung hingegen wird in die Bridge der Systemanbindung verlagert. Sequenzer bzw. Adressgeneratoren, Instruktionsinterpretation und die daraus folgende Beeinflussung der Datenverarbeitung soll hierbei innerhalb der Bridge realisiert werden. Zur Steuerung des Ablaufs der Konfigurationswechsel und Lesen bzw. Schreiben der zu verarbeitenden Daten soll ein passender Instruktionssatz definiert werden. Somit erhält eine derartig implementierte und erweiterte Bridge einen Ablaufplan zur Steuerung der Array-Abläufe und die Fähigkeit zum autonomen Operieren, was zu einer Entlastung des Systemcontrollers führt und einen enormen Performanzschub verspricht. Das Resultat wäre ein stark parallelisiertes System, welches eine gute Ausnutzung der vorliegenden Ressourcen bietet.

Wie bereits im Abschnitt 2.1 beschrieben, wäre eine universelle Realisierung des Kommunikationsnetzes eine nützliche Alternative zu Standardlösungen von einem separaten Konfigurationsbus. Mit Hilfe der erweiterten Bridges wird diese Alternative noch attraktiver, da bei Nutzung von mehreren separaten Bridges das Potenzial des Kommunikationsnetzes mit vielen parallelen Leitungen zu einer schnellen Rekonfiguration beitragen kann und diese erheblich beschleunigen würde.

3 Zusammenfassung

In diesem Papier wurden Analyseergebnisse neuer multigranularer, dynamisch rekonfigurierbarer Array-Architekturen und daraus resultierende Vorschläge für Architekturansätze vorgestellt. Weitere Arbeiten des Projektes sind die Realisierung des vorgestellten Konzepts in einem Demonstrator.

Das Ziel ist eine neuartige Architektur, die in VHDL spezifiziert vorliegen soll. Dieses Modell soll benutzt werden, um die Verifikation der Anwendbarkeit und der verbesserten Eigenschaften des Arrays durchzuführen. Dies schließt die Analysen und Optimierungen bezüglich der Fläche und Verlustleistung ein. Zu Synthesezwecken und Layou-

terstellung soll die UMC 0.13 µm Standardzellentechnologie eingesetzt werden, um aussagekräftige Ergebnisse zu erreichen und die Vergleichbarkeit mit kommerziellen Lösungen zu ermöglichen.

Literaturverzeichnis

[Kr01] R. Kress: **A fast reconfigurable ALU for Xputers**; Ph. D. dissertation, Kaiserslautern University, 1996

[Be01] Juergen Becker, Thilo Pionteck, Manfred Glesner: **DReAM: A Dynamically Reconfigurable Architecture for Future Mobile Communication Applications**; in: 10th International Conference on Field Programmable Logic and Applications, Villach, Österreich, 2000.

[Be02] Juergen Becker, Thilo Pionteck, Manfred Glesner: **Dynamisch rekonfigurierbare Hardwarearchitekturen für flexible System-on-Chip Lösungen in der Mobilkommunikation**; in: Proceedings of Dresdner Arbeitstagung "Schaltungs- und Systementwurf" (DASS´2000), Dresden, Germany, May 16-17, 2000

[XP01] PACT XPP Technologies Corporation: http://www.pactcorp.com

[XP02] **The XPP Communication System**, PACT XPP Technologies Corporation, Technical Report 15, 2000.

[XP03] V. Baumgarte, F. Mayr, A. Nückel, M. Vorbach, M. Weinhardt: PACT XPP Technologies - **A Self-Reconfigurable Data Processing Architecture**; The 1st Int´l. Conference of Engineering of Reconfigurable Systems and Algorithms (ERSA´01), Las Vegas, NV, June 2001.

[XI01] **Xilinx, Inc**; http://www.xilinx.com

[AL01] Altera Corp.: http://www.altera.com

[TR01] Triscend Inc.: http://www.triscend.com

[TR02] Triscend A7 Configurable System-on-Chip Platform - Data Sheet http://www.triscend.com/products/ dsa7csoc_summary.pdf

[AT01] Atmel Corp.: http://www.atmel.com

[QU01] **QuickSilver Technology, Inc.**; http://www.quicksilvertech.com

[Go01] S. Copen Goldstein, H. Schmit, M. Moe, M. Budiu, S. Cadambi, R. R. Taylor, R. Laufer **"PipeRench: a Coprocessor for Streaming Multimedia Acceleration"** in ISCA 1999. http://www.ece.cmu.edu/research/piperench/

[Ba01] N. Bagherzadeh, F. J. Kurdahi, H. Singh, G. Lu, M. Lee: **"Design and Implementation of the MorphoSys Reconfigurable Computing Processor"**; J. of VLSI and Signal Processing-Systems for Signal, Image and Video Technology, 3/ 2000

[Zh01] Hui Zhang, Vandana Prabhu, Varghese George, Marlene Wan, Martin Benes, Arthur Abnous, **"A 1V Heterogeneous Reconfigurable Processor IC for Baseband Wireless Applications"**, Proc. of ISSCC2000