Network-on-Chip basierende Laufzeitsysteme für dynamisch rekonfigurierbare Hardware

Ronald Hecht, Dirk Timmermann, Stephan Kubisch, Elmar Zeeb

Universität Rostock
Institut für Angewandte Mikroelektronik und Datentechnik
Richard-Wagner-Str. 31, 18119 Rostock-Warnemünde
{ronald.hecht,dirk.timmermann}@etechnik.uni-rostock.de,
{stephan.kubisch,elmar.zeeb}@stud.uni-rostock.de

Abstract: Die Kombination aus Standardprozessoren und rekonfigurierbarer Hardware hat sich in Forschung und kommerziellen Anwendungen mit schnell veränderlichen Parametern und Protokollen als flexible und leistungsfähige Architektur erwiesen. Jedoch werden die Möglichkeiten der dynamischen partiellen Rekonfigurierbarkeit noch nicht ausgenutzt, durch die eine weitere Steigerung der Anpassungsfähigkeit möglich wäre. Die Kontrolle und Steuerung könnte der ohnehin schon vorhandene Prozessor und das darauf laufende Betriebssystem übernehmen. Da aber Umsetzbarkeit und Leistung des Gesamtsystems erheblich von der Schnittstelle zwischen rekonfigurierbarer Hardware und nutzender Anwendung abhängt, sind hier skalierbare und zukunftsweisende Architekturen erforderlich. Im Rahmen dieses Beitrags werden Aspekte zur Erweiterung eines bestehenden Betriebssystems um dynamisch rekonfigurierbare Hardware und dessen Anbindung behandelt. Es wird ein Network-on-Chip basierter Protokollstack vorgestellt und ein Ausblick auf die prototypische Implementierung gegeben.

1 Einleitung

Heutige Betriebssysteme haben die Aufgabe, die Komplexität der Hardware vor dem Benutzer zu verbergen und vorhandene Ressourcen zu verwalten. Das Teilen von Speicher, Prozessorleistung und Peripherie zwischen mehreren Prozessen, Applikationen und Nutzern verbessert die Auslastung und senkt somit auch die Kosten. Fügt man einem solchen System rekonfigurierbare Hardware hinzu, kann die vorhandene FPGA-Fläche ebenfalls als gemeinsam nutzbare Ressource verstanden werden. Dem Betriebssystem können dann neue Aufgaben, wie die FPGA-Konfiguration, die Verwaltung der verfügbaren Fläche und die Kommunikation mit den geladenen Hardwaremodulen zugeordnet werden. Dieser Ansatz wurde erstmals von G. Brebner in [Br96, Br97] verfolgt. Dort wird die „Swappable Logic Unit“ (SLU) als kleinste rekonfigurierbare Einheit mit den Eigenschaften eines Beschleunigers, Speichers oder IO-Geräts definiert. Er zieht den Vergleich zur Verwaltung virtuellen Speichers und zu einem Multiprozessorsystem. Diese Verschränkung der Analogien und die Zweidimensionalität der FPGA-Fläche erschweren es, performante und ska-

lierbare Lösungen für die Verwaltung rekonfigurierbarer Hardware zu finden. So müssen Verdrahtung und Platzierung innerhalb der SLUs zur Designphase stattfinden. Daraus entstehende abstrakte Binärdateien werden dann zur Laufzeit in Bitfiles konvertiert und in das FPGA geladen, so wie kompilierte Programme in einem Standardbetriebssystem ausgeführt werden.

Den vorgestellten Kommunikationsmechanismen, wie registerbasiertem Anlegen der Parameter und Auslesen der Ergebnisse, Message-Passing, ereignisgesteuerter Ein- und Ausgabe und entfernten Prozeduraufrufen, fehlt es jedoch an einer leistungsfähigen zugrunde liegenden Infrastruktur. Der Datenaustausch wurde, wie die partielle Rekonfiguration, über die Konfigurationsanschlüsse des FPGAs realisiert. Die zugrunde liegende FPGA-Architektur (XC6200) besaß noch nicht die Kapazität, komplexe Systems-on-Chip (SoCs) zu integrieren.

Damit sich aber der Einsatz beschleunigender, dynamisch rekonfigurierbarer Hardware und die damit verbundene Verwaltung lohnt, muss sich die Gesamtgeschwindigkeit gegenüber reinen Softwarelösungen erheblich verbessern. Durch den Einsatz tiefer Pipelines und massiver Parallelität werden Hardware- stets den Softwarelösungen überlegen sein. Jedoch darf der Datentransfer zu und von den Hardwaremodulen nicht vernachlässigt werden. Tatsächlich stellt die Hardware/Software- Schnittstelle meist den eigentlichen Flaschenhals dar. Teilen sich mehrere Applikationen die FPGA-Fläche, kommt noch die Zeit für Kontextwechsel hinzu. Insgesamt wird die Gesamtausführungszeit durch drei Faktoren beeinflusst: die Ausführungszeit des Cores, den Datentransfer und die Zeit für die dynamische Konfiguration und deren Steuerung.

Um diesen Schwierigkeiten entgegenzutreten, wird von T. Marescaux in [MBV+02] ein Network-on-Chip (NoC) als Kommunikationsplattform vorgeschlagen. Dies ermöglicht einheitliche Schnittstellen zu den rekonfigurierbaren Modulen, eine skalierbare Netzwerktopologie und konsistente Kommunikationsmechanismen zwischen den Modulen untereinander und dem Betriebssystem. Es wurde auf einem Xilinx Virtex FPGA erfolgreich implementiert und in Zusammenhang mit Linux zu einer prototypischen Laufzeitumgebung umgesetzt.

Im Rahmen dieses Beitrags wird an diese Arbeiten angeknüpft, und es werden die durch die Laufzeitumgebung definierten Anforderungen an das NoC abgeleitet. Kapitel 3 beschreibt einen Protokollstack für zukünftige NoC-basierende rekonfigurierbare Systeme. Abschließend wird die zugrunde liegende Prototypplattform vorgestellt und ein Ausblick auf nachfolgende Forschungen gegeben.

2 Anforderungen an die Kommunikationsinfrastruktur

Mit der Einführung des Xilinx Modular Designflows wurde eine handhabbare und zuverlässige Umgebung für die praktische Realisierung dynamisch rekonfigurierbarer Systeme geschaffen. Jedoch sind an dessen Verwendung eine ganze Reihe von Bedingungen und Einschränkungen geknüpft. Zunächst sind nur vertikale zuvor festgelegte Bereiche dynamisch rekonfigurierbar. Die Kommunikation zwischen diesen Bereichen hat über eine

begrenzte Anzahl, ebenfalls vorher definierter Leitungen, zu erfolgen. Globale Leitungen außer dem globalen Takt sind nicht erlaubt. Somit müssen rekonfigurierbare Hardwaremodule konstante und fest platzierte Schnittstellen haben, um gegeneinander austauschbar zu sein. Zentral gesteuerte SoC-Bussysteme können hier auf Grund der begrenzten Skalierbarkeit und der hohen Anzahl an globalen Signalen nicht eingesetzt werden.

Abgesehen von der physikalischen Verbindung müssen die Kommunikationsmechanismen zwischen den Modulen untereinander und zum Betriebssystem vereinheitlicht sein. Nur so ist es bei ausgelasteter FPGA-Fläche möglich, die Module, die nicht geladen werden können, durch ein Softwareäquivalent zu ersetzen. Die Infrastruktur muss außerdem einen autonomen Datenaustausch zwischen Hardwaremodulen unterstützen.

Aus Sicht des Betriebssystems sind für die Verwaltung der FPGA-Fläche, für das Scheduling und das automatisierte Laden und Unterbrechen von Modulen unterstützende Mechanismen nötig. Somit sollte allein der Wunsch, mit einem noch nicht geladenen Modul Daten austauschen zu wollen, die partielle Konfiguration auslösen. Möchten zwei Hardwaremodule miteinander kommunizieren, wovon eines noch nicht geladen ist, kann auch hier die Konfiguration automatisiert werden. Für die Realisierung von Timesharing muss es dem Betriebssystem auch möglich sein, Module zu unterbrechen, durch andere auszutauschen und sie später wieder fortzusetzen.

Da verschiedene Anwendungen auch unterschiedlichste Anforderungen an die Qualitäts-Parameter der Kommunikation haben, müssen eine ganze Reihe von Service-Klassen unterstützt werden. Für Multimedia- und Krypto-Anwendungen sind hoher Durchsatz, dagegen in Echtzeitsystemen und bei der Steuerung von Peripherie geringe Latenz bei niedrigem Durchsatz erforderlich.

Zusammenfassend lässt sich sagen, dass die durch die dynamische partielle Rekonfiguration gestellten physikalischen Bedingungen den größten Einfluss auf die Wahl einer geeigneten Kommunikationsinfrastruktur haben. Die festen Schnittstellen und deren begrenzte Anzahl von Leitungen, drängen den Vergleich zu Computernetzwerken auf. Hier ist das Hinzufügen und Entfernen von Rechnern ohne Schwierigkeiten möglich. Höhere Protokollschichten stellen Dienste zur einfachen und sicheren Übertragung von Daten zur Verfügung. Spezielle Steuerprotokolle reagieren auf dynamische Netzwerkveränderungen. Im folgenden Kapitel wird unter Anlehnung an das ISO/OSI-Schichtenmodell auch für dynamisch rekonfigurierbare Systeme eine Netzwerk-basierte Infrastruktur abgeleitet, die alle genannten Bedingungen und Anforderungen erfüllen wird.

3 Ein Protokollstack für Networks-on-Chip in dynamisch rekonfigurierbaren Systemen

Wie in [MBV+02] vorgeschlagen wurde, stellt die Verwendung eines Network-on-Chip eine geeignete und zukunftsweisende Infrastruktur zur Verbindung von dynamisch rekonfigurierbaren Bereichen (Tiles) mit dem steuernden Betriebssystem dar. Jeder Bereich hat eine feste Schnittstelle zum Netzwerk. Somit lassen sich IP-Cores beliebig in die rekonfigurierbaren Bereiche laden. Da die Hardware/Software-Schnittstelle erheblichen Einfluss

auf die Gesamtleistung des dynamisch rekonfigurierbaren Systems hat, macht es Sinn, den Prozessor mit dem darauf laufenden Betriebssystem in das NoC zu integrieren (vgl. Abbildung 1). Die FPGAs der Xilinx Virtex-II Pro-Reihe enthalten schon PowerPC-Kerne, wodurch dieser Ansatz auch praktisch realisierbar ist.

Die durch ein Network-on-Chip verbundenen Elemente werden als Ressourcen (R) bezeichnet [SH03b]. Jede Ressource ist ein Rechen- oder Speicherelement. Dynamisch rekonfigurierbare Bereiche (Re) werden gleichwertig behandelt. Durch lokalen Speicher (M) und Caches (C) kann der Verkehr auf dem NoC minimiert werden. Die Schnittstelle zu den Switches (S) wird durch ein Resource-Network-Interface (RNI) hergestellt, welches sich wiederum in zwei Teile untergliedern lässt (vgl. Abbildung 2). Das Resource-Independent-Network-Interface (RINI) enthält Funktionen, die für jede Ressource gleich sind. Bei einer partiellen Rekonfiguration muss dieser Teil nicht verändert werden. Das Resource-Dependent-Network-Interface (RDNI) ist eine ressourcenspezifische Schnittstelle und muss für jedes rekonfigurierbare Modul implementiert werden und damit auch rekonfigurierbar sein.

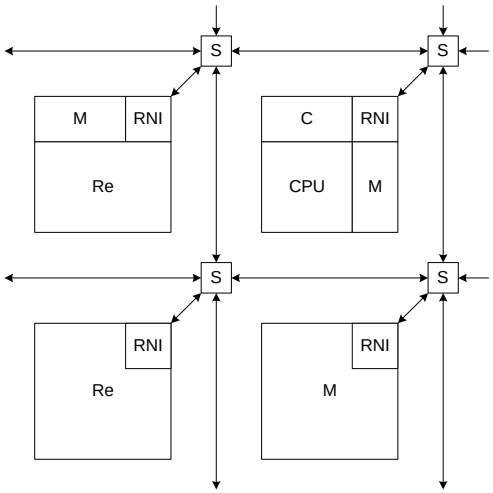


Abbildung 1: Ausschnitt eines Network-on-Chip

In [MMB+03] werden drei Verkehrsarten für dynamisch rekonfigurierbare Systeme definiert. Zum Einen müssen die FPGA Konfigurationsdaten (Bitfiles) zu den rekonfigurierbaren Bereichen transportiert werden. Dies muss jedoch nicht über ein NoC erfolgen, da hierfür schon ein Rekonfigurationsnetzwerk existiert, welches Bestandteil aller FPGA Familien und über externe (SelectMAP) oder interne (ICAP) Konfigurationsanschlüsse ansprechbar ist. Zum Anderen müssen Steuer- und Statusinformationen für die Kontrolle des NoCs und letztendlich die Anwendungsdaten übertragen werden. Es wird vorgeschlagen, die beiden letztgenannten Verkehrsarten über getrennte NoCs zu transportieren. Da aber jedes weitere Network-on-Chip durch die Verbindungsleitungen die rekonfigurierbare Chipfläche weiter einschränkt, ist dies keine elegante Lösung. Werden jedoch Prioritäten in

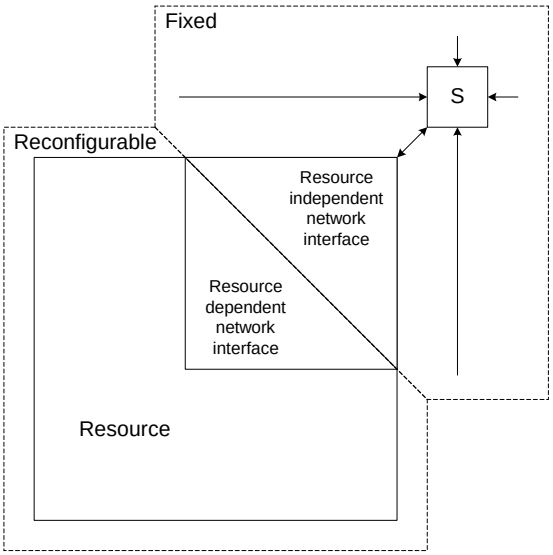


Abbildung 2: Resource-Network-Interface

den Switches ausgewertet, können alle Verkehrsklassen über ein gemeinsames Netzwerk transportiert werden.

Der in dem vorliegenden Beitrag verfolgte Ansatz beruht auf der Definition eines geeigneten Protokollstacks mit Unterstützung verschiedener Verkehrsarten und Qualitätsklassen. Dafür werden aufeinander aufsetzende Abstraktionsebenen unter Anlehnung an das ISO/OSI-Schichtenmodell eingeführt, wobei davon ausgegangen wird, dass das Betriebssystem eine globale Sicht auf das Netzwerk hat und dessen Konfiguration übernimmt. Diese Modularisierung erlaubt eine flexiblere Umsetzung und Integration in bestehende SoC-Betriebssysteme und reduziert die Anzahl der Leitungen für das NoC. Eine Erhöhung des Abstraktionsniveaus ist natürlich gleichbedeutend mit größerem Overhead, hat sich aber auch in bestehenden Computernetzwerken bewährt.

3.1 Bitübertragungsschicht

Die Verbindungen zwischen benachbarten Switches und den zugehörigen Resource-Network-Interfaces werden durch die Bitübertragungsschicht definiert. Hierbei spielen die physikalischen und geometrischen Eigenschaften des Netzwerks, die Verteilung des Taktes, Daten- und Steuerleitungen die wesentliche Rolle. Die Hauptaufgabe dieser Schicht ist die Übertragung von Bits oder Wörtern. Die in [Sh03a] vorgestellte Nostrum-Gitter-Architektur definiert Datenbusbreiten von 256 Bit. Durch die Beschränkung der durch den Xilinx Modular Designflow vorgegebenen Routing-Ressourcen zur Verbindung benachbarter Bereiche sind jedoch nur 16 oder 32 Bit realisierbar. Zukünftige FPGA-Familien

könnten aber schon eine vollständige NoC-Implementierung mit festverdrahteten Switches enthalten. Die zugehörigen rekonfigurierbaren Bereiche müssten separat konfigurierbar und integrierte Prozessoren mit dem NoC verbunden sein. Eine solche Hardware-Plattform wäre eine ideale Basis für dynamisch rekonfigurierbare Systeme.

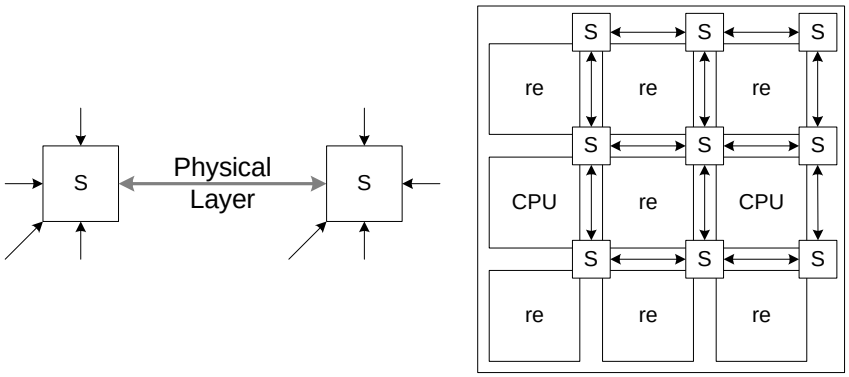


Abbildung 3: Bitübertragungsschicht und zukünftige FPGAs mit intergriertem NoC

3.2 Sicherungsschicht

Die Sicherungsschicht stellt eine verlässliche Verbindung zwischen zwei Switches her, realisiert Flusskontrolle, Fehlererkennung und Fehlerkorrektur. Sie ermöglicht das Übertragen von Paketen beliebiger, aber begrenzter Länge. Der Protokollheader enthält die physikalische Quell- und Zieladresse, ein Prioritätsfeld und definiert den Typ der transportierten Daten (vgl. Abbildung 4). Die physikalischen Adressen sind statisch vergeben und bezeichnen die geographische Lage der entsprechenden Ressource auf dem Chip. Davon ausgehend können die Switches recht einfach auf Basis von xy-Wormhole-Routing implementiert werden können. Die Route ist dabei deterministisch und wird allein durch die physikalische Adresse bestimmt. Eine relative Vergabe der Adressen wie in [MBV+02] ist auch möglich, aber aufwendiger zu implementieren und aus Betriebssystemsicht schwerer zu verwalten. Das Prioritätsfeld gestattet die Unterstützung verschiedener Verkehrsklassen in den Switches, welche mit Hilfe von virtuellen Kanälen implementiert werden. Somit lassen sich die Kontroll- und Steuerfunktionen des Betriebssystems von den Anwendungsdaten entkoppeln.

Diese Definition der Sicherungsschicht unterscheidet sich zu der in [Sh03a], welche den Transport von Worten mit 256 Bit von einem Switch zu einem benachbarten vorgibt. Aufgrund der niedrigen möglichen Wortbreite bei einer FPGA-Implementierung für dynamische Rekonfiguration ist der Transport von Paketen aus mehreren Worten zu bevorzugen. Durch die statische Vergabe der Adressen muss auch die Routing-Funktionalität aus der Vermittlungsschicht in die Sicherungsschicht verlegt werden.

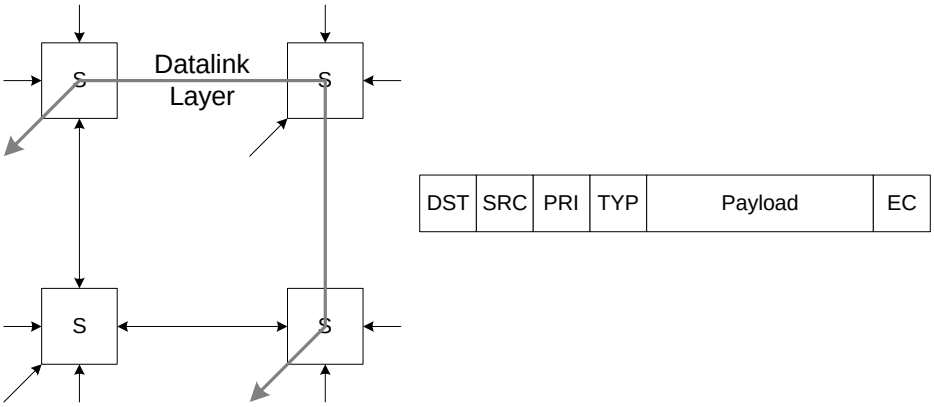


Abbildung 4: Sicherungsschicht und deren Paketaufbau

3.3 Vermittlungsschicht

Die Aufgabe der Vermittlungsschicht ist der Transport von Paketen zwischen den Ressourcen. Da bei einer dynamischen Rekonfiguration die Hardwaremodule in beliebige Tiles geladen werden sollen, ist hier eine logische Adressierung sinnvoll. Logische Adressen setzen sich aus einer modulspezifischen und einer prozessspezifischen Adresse zusammen. Die Zuordnung der logischen zur physikalischen Adresse wird einmal vor Absenden eines Pakets durch eine Adressauflösungstabelle im RINI ermittelt. Diese Tabellen werden durch das Betriebssystem zur Laufzeit mit Hilfe von Adressauflösungspaketen konfiguriert. Es können weiterhin Fehler- und Statuspakete generiert werden, die für die Verwaltung der rekonfigurierbaren Bereiche vom Betriebssystem verarbeitet werden. Sie geben Auskunft über unbekannte logische Adressen, statistische Informationen und Füllstände der FIFOs in den RNIs. Steuer-, Fehler- und Statusfunktionen werden zu einem NOC-Management-Protokoll zusammengefasst.

Zur Unterscheidung von Daten- und Managementpaketen wird das Typfeld der Sicherungsschicht verwendet. Der Protokollheader der Datenpakete enthält die logische Quell- und Zieladresse, einen Quality-of-Service Identifier und ein Protokollfeld zur Angabe des darüberliegenden Protokolls der Transportschicht. Managementpakete besitzen einen Opcode, welcher die darin enthaltenen Daten spezifiziert. Pakete zur Adressauflösung enthalten beispielsweise eine physikalische und die zugehörige logische Adresse.

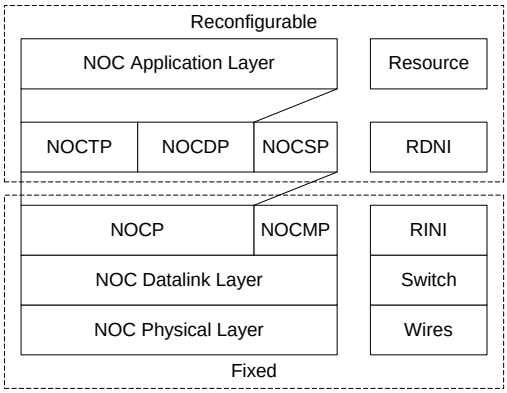
3.4 Transportschicht

Die Transportschicht stellt eine sichere Verbindung zwischen zwei Ressourcen her. Sie kann verbindungslos und verbindungsorientiert sein. Über verbindungslose Dienste können Nachrichten ausgetauscht werden. Verbindungsorientierte Protokolle beinhalten den

Auf- und Abbau von virtuellen Punkt-zu-Punkt-Verbindungen. In dieser Schicht wird auch die Segmentierung von Nachrichten in Pakete und deren Wiederzusammensetzung vorgenommen. Laufende Forschungen untersuchen, welche Transportprotokolle für den Einsatz in einem dynamisch rekonfigurierbaren System sinnvoll sind. Verbindungslose Protokolle können für entfernte Prozeduraufraufe und distributed shared memory verwendet werden. Verbindungsorientierte Protokolle sind für die sichere Übertragung von großen Datenmengen zweckmäßig. Bestandteil beider Protokolle ist auch ein Quell- und Zielport, um mehreren Applikationen den Zugriff auf einzelne Hardwaremodule zu ermöglichen. Für die Steuerung rekonfigurierbarer Module, wie deren Start und Unterbrechung, ist auch ein einfaches und schnelles Signalprotokoll nötig. Da jede Anwendung und deren unterstützende Hardware eigene Kommunikationsmechanismen definiert, macht es Sinn, die Transportschicht in dem Resource-Dependent-Network-Interface zu implementieren.

3.5 Anwendungsschicht

So wie im TCP/IP-Referenzmodell und im Schichtenmodell für NoCs in [Sh03a] werden die Sitzungs- und die Darstellungsschicht nicht gesondert betrachtet, sondern in die Anwendungsschicht integriert. Hier sind anwendungsspezifische Funktionen und Protokolle implementiert, die dem Entwickler völlig frei stehen. Bei der Integration eines Hardwarebeschleunigers für Kryptosysteme, sollten an dieser Stelle Funktionen wie Schlüsselinitialisierung, Auswahl des Algorithmus und die Ver- und Entschlüsselung von Datenströmen implementiert werden. Die Kommunikation mit rekonfigurierbaren Peripherie-Controllern würde deren Steuerung und die Bereitstellung einer zuverlässigen bidirektionalen Verbindung beinhalten.



NOCP: NoC Protocol; **NOCMP:** NoC Management Protocol; **NOCTP:** NoC Transport Protocol; **NOCDP:** NoC Datagram Protocol; **NOCSP:** NoC Signaling Protocol; **RINI:** Resource Independent Network Interface; **RDNI:** Resource Dependent Network Interface

Abbildung 5: NoC Protokoll Stack

Der in Abbildung 5 dargestellte Protokollstack vereint alle diskutierten Schichten und zeigt die zugehörigen Baugruppen des NoCs. Auch hier ist zu erkennen, dass sowohl die Transportschicht, als auch die Anwendungsschicht rekonfigurierbar und somit auch vom Entwickler zu definieren sind. Allerdings können hier wiederverwendbare Komponenten einer bestehenden Modulbibliothek entnommen werden.

4 FPGA Prototyp

Für die Evaluierung des Betriebssystems und des Network-on-Chip wird das in Abbildung 6 dargestellte Avnet Virtex-II Pro (XC2VP20) Entwicklungsboard eingesetzt. Auf einem der FPGA-internen PowerPCs läuft das Betriebssystem Linux, welches zur Zeit um die Funktionen zur dynamischen Rekonfiguration der übrigen FPGA-Fläche und der Kommunikation mit dem NoC erweitert wird. Das Schreiben der Konfigurationsdaten erfolgt über die interne FPGA-Konfigurationsschnittstelle ICAP, wodurch die maximale Geschwindigkeit von 66MByte/s erreicht werden kann. Durch die PCI-Schnittstelle wird der Einsatz des Boards auch als Erweiterung in einem Standard-PC ermöglicht. Hier kann ein außerhalb des FPGAs befindliches Betriebssystem die dynamische Rekonfiguration übernehmen. Die Übertragung der partiellen Bitfiles und die Kommunikation mit dem NoC geschieht dann über den PCI-Bus.

Abbildung 6: Avnet Virtex-II Pro Entwicklungsboard

5 Zusammenfassung und Ausblick

Die in diesem Beitrag vorgestellte Infrastruktur zur Kommunikation der Elemente eines rekonfigurierbaren System-on-Chip stellt nicht nur eine praktisch realisierbare Lösung dar, sondern definiert auch skalierbare und leistungsfähige Schnittstellen zwischen zukünftigen Hard- und Softwarekomponenten. Unter Anlehnung an das TCP/IP-Referenzmodell wurde ein Protokollstack abgeleitet, der die Aufgaben und Anforderungen an ein dynamisch

rekonfigurierbares System konkreten Abstraktionsschichten zuordnet.

Natürlich bringt die Implementierung einer solchen Architektur auch Nachteile mit sich. Zum Einen entsteht ein durch die Protokollheader verursachter Kommunikationsoverhead. Andererseits steigt die verwendete Chipfläche zur Implementierung der Protokolle an. Auch die Einteilung der FPGA-Fläche in feste gleich große Bereiche verschwendet viele Ressourcen. Werden jedoch die unteren Schichten des Protokollstacks fester Bestandteil zukünftiger FPGAs, sind erhebliche Leistungssteigerungen und Kostensenkungen möglich. Dass dies kein Wunschtraum ist, zeigen die aktuellen Entwicklungen der Firma Xilinx auf dem Gebiet der nachrichtenbasierten Intermodulkommunikation.

Alle bisherigen Forschungen auf dem Gebiet dynamisch rekonfigurierbarer Systeme haben gezeigt, dass dessen Umsetzung eine sehr komplexe und schwierige Aufgabe ist. Nur durch die schrittweise Steigerung des Abstraktionsniveaus wird diese auch zu bewältigen sein.

Literatur

[Br96] Brebner, G.: A virtual hardware operating system for the Xilinx XC6200. In: *LNCS*. volume 1142 of *6th International Workshop on Field Programmable Logic and Applications*. S. 315–333. Springer. 1996.

[Br97] Brebner, G.: The Swappable Logic Unit: A Paradigm for Virtual Hardware. In: *Proc. of the 5th IEEE Symposium on FPGA-Based Custom Computing Machines*. S. 77–86. Napa Valley, California. 1997.

[MBV+02] Marescaux, T., Bartic, A., Verkest, D., Vernalde, S., und Lauwereins, R.: Interconnection Networks Enable Fine-Grain Dynamic Multi-Tasking on FPGAs. In: *LNCS*. volume 2438 of *Proc. 12th Int. Conf. on Field-Programmable Logic and Applications*. S. 795–805. Springer. 2002.

[MMB+03] Marescaux, T., Mignolet, J.-Y., Bartic, A., Moffat, W., Verkest, D., Vernalde, S., und Lauwereins, R.: Networks on Chip as Hardware Components of an OS for Reconfigurable Systems. In: *Proc. 13th Int. Conf. on Field-Programmable Logic and Applications*. S. 595–605. 2003.

[Sh03a] Shashi, K.: On Packet Switched Networks for On-chip Communication. In: *Networks on Chip*. S. 85–106. Kluwer Academic Publishers. 2003.

[SH03b] Soininen, J.-P. und Heusala, H.: A Design Methodology for NoC-based Systems. In: *Networks on Chip*. S. 19–38. Kluwer Academic Publishers. 2003.