Dynamisch rekonfigurierbare Hardware als Basistechnologie für intelligente technische Systeme

Dipl.-Ing. Sebastian Korf
Heinz Nixdorf Institut
Fürstenallee 11, 33102 Paderborn
Tel. 05251/606345, Fax. 05251/606351
E-Mail: korf@hni.uni-paderborn.de

Dipl.-Ing. Gregor Sievers, Dipl.-Ing. Johannes Ax, M.Sc. Dario Cozzi,
Dr.-Ing. Thorsten Jungeblut, Dipl.-Ing. Jens Hagemeyer,
Dr.-Ing. Mario Porrmann, Prof. Dr.-Ing. Ulrich Rückert
Universität Bielefeld, Center of Excellence Cognitive Interaction Technology
Universitätsstraße 21-23, 33615 Bielefeld
Tel.0521/10612048, Fax. 0521/10612348
E-Mail: mporrmann@cit-ec.uni-bielefeld.de

Zusammenfassung

Ein wesentliches Ziel bei der Entwicklung mikroelektronischer Schaltungen ist der effiziente Umgang mit den gegebenen Ressourcen Fläche, Zeit und Energie. Mikroprozessoren weisen durch ihre Programmierbarkeit ein hohes Maß an Flexibilität auf. Im Vergleich zu anwendungsspezifischen Schaltungen (ASICs) bieten sie durch die sequentielle Verarbeitung der Programme jedoch eine eingeschränkte Leistungsfähigkeit. ASICs sind indes leistungsfähiger aber weniger flexibel, da sie nach der Fertigung nicht mehr verändert werden können. Dynamisch rekonfigurierbare Hardware bietet für viele Anwendungen einen guten Kompromiss zwischen Mikroprozessoren und ASICs. Die partielle Rekonfigurierbarkeit heutiger feldprogrammierbarer Bausteine (FPGAs) ermöglicht die Veränderung eines Teils der Logikblöcke und deren Verbindungsstruktur im Betrieb. Hardwarefunktionen lassen sich zur Laufzeit auf das FPGA laden und nach Abschluss der Verarbeitung wieder entfernen, um die Ressourcen zukünftigen Hardwarefunktionen zur Verfügung zu stellen. Für intelligente technische Systeme ist in vielen Fällen eine Kombination der oben beschriebenen Ansätze sinnvoll. Daher stellen wir eine neue Hardwareplattform vor, die rekonfigurierbare Prozessoren und FPGAs in einer Systemumgebung vereint. Mit Hilfe der modularen Prototyping-Umgebung RAPTOR werden die Konzepte sowohl prototypisch umgesetzt als auch in realen technischen Systemen getestet und bewertet.

Schlüsselworte

FPGA, Rekonfiguration, Prozessor, VLIW

1 Einleitung

Intelligente technische Systeme stellen hohe Anforderungen an die Ressourceneffizienz der zu Grunde liegenden Informationsverarbeitungsplattformen. Neben einer hohen Leistungsfähigkeit und geringen Kosten sind Energieeffizienz, Flexibilität und Fehlertoleranz von entscheidender Bedeutung. Klassische, Mikrocontroller-basierte Hardwareplattformen können die geforderten Randbedingungen in vielen Fällen nicht einhalten. Daher präsentieren wir in diesem Artikel dynamisch rekonfigurierbare Architekturkonzepte, die ihre interne Struktur zur Laufzeit an sich ändernde Umgebungsanforderungen anpassen können. Die traditionell statische Aufteilung in Hardware und Software kann durch eine dynamische Partitionierung zur Laufzeit ersetzt werden.

Wir stellen zwei Ansätze für die Realisierung dynamisch rekonfigurierbarer Systeme vor: Architekturen auf Basis Feldprogrammierbarer Gate-Arrays (FPGAs) und rekonfigurierbare eingebettete Prozessoren.

FPGA-basierte Systeme bieten aufgrund ihres feingranular rekonfigurierbaren internen Aufbaus eine hohe Flexibilität, die allerdings mit einem erheblichen Mehraufwand bezüglich der benötigten Hardwareressourcen und einer hohen Entwurfskomplexität einhergeht. Um diese Architekturen einem breiteren Anwenderfeld zugängig zu machen, haben wir eine Entwicklungsumgebung realisiert, die alle notwendigen Entwurfsschritte kapselt und weitestgehend automatisiert. In Kapitel 2 werden diese feingranularen Architekturen und die Entwicklungsumgebung vorgestellt.

Neben FPGA-basierten Architekturen entwickeln wir grobgranulare, Prozessor-basierte Architekturen. Hier zielen wir darauf ab, mit minimalen zusätzlichen Kosten für die Rekonfigurierbarkeit einen möglichst großen Effekt zu erzielen. Der von uns entwickelte CoreVA Parallelprozessor kann zur Entwurfszeit in weiten Bereichen konfiguriert und so an die Anforderungen einer bestimmten Anwendungsklasse angepasst werden. Zur Laufzeit ermöglicht die dynamische Rekonfiguration eine schnelle Adaption des Prozessors an sich ändernde Umgebungsbedingungen. Weitere Details zu den grobgranularen Architekturen werden in Kapitel 3 beschrieben.

Anhand einer Beispielanwendung zeigen wir in Kapitel 4, dass mit den eingesetzten Methoden die Ressourceneffizienz trotz des Mehraufwandes für die dynamische Rekonfiguration verbessert werden kann. Insbesondere werden die Systemarchitektur und die Implementierungsdetails eines Hardwarebeschleunigers für Kryptographie mit elliptischen Kurven beschrieben. Das letzte Kapitel fasst die erzielten Ergebnisse zusammen und gibt ein Ausblick auf zukünftige Arbeiten.

2 Feingranulare rekonfigurierbare Architekturen

Die Verwendung einer dynamisch rekonfigurierbaren Systemarchitektur auf einem FPGA bietet viele Vorteile gegenüber einer rein statischen Systemarchitektur. Im Vordergrund steht dabei die Erhöhung der Flexibilität, da Hardwarefunktionen zur Laufzeit auf dem FPGA ausgetauscht werden können. Darüber hinaus ist es möglich, die Hardware durch neue Funktionen zu erweitern. Ein Beispiel hierfür ist die Integration neuer Mobilfunkstandards in das System. Durch den Zeitmultiplex-Betrieb des FGPAs ist die Verwendung eines kleineren, kostengünstigeren FPGAs möglich, wodurch die Verlustleistung gesenkt wird. Durch verschiedene Hardwaremodule eines Algorithmus mit unterschiedlichen Konfigurationen, zum Beispiel auf die Ausführungszeit oder den Ressourcenbedarf optimiert, kann die Verlustleistung weiter reduziert werden.

In der Regel sind dynamisch rekonfigurierbare Systeme, wie in Bild 1 dargestellt, in statische und dynamische Systemkomponenten aufgeteilt, die über eine Kommunikationsinfrastruktur verbunden sind. Im Folgenden wird zunächst der Ansatz zur dynamischen Rekonfiguration des FPGA-Herstellers Xilinx und anschließend ein flexiblerer, feingranularer, kachelbasierter Ansatz [KLH+11] vorgestellt.

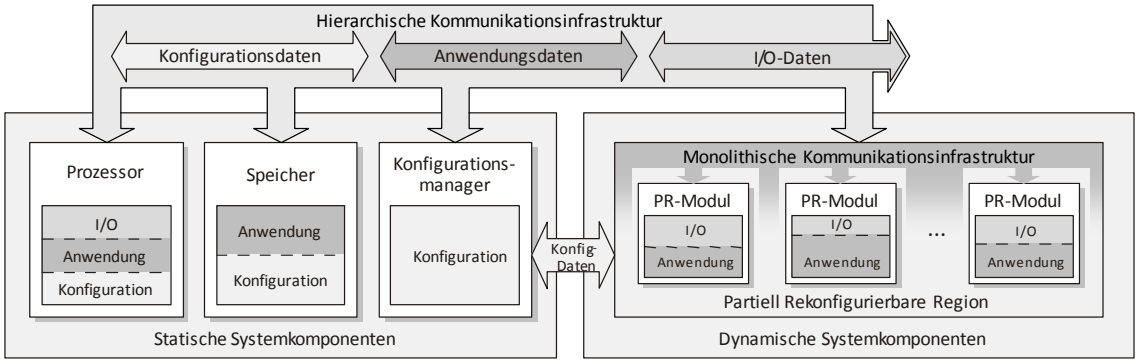

Bild 1: Partitionierung eines dynamisch rekonfigurierbaren Systems in statische und dynamische Systemkomponenten

Der Xilinx-Entwurfsablauf erstellt für jedes partiell rekonfigurierbare Hardware-Modul (PR-Modul) eine dedizierte Kommunikationsinfrastruktur (siehe Bild 2 a). Dadurch ist jedes PR-Modul auf eine feste Position fixiert. Um ein Modul an verschiedene Positionen platzieren zu können, wird jeweils ein separater Bitstrom benötigt. Dieser Ansatz limitiert den Einsatz der dynamischen Rekonfiguration auf eine geringe Anzahl von Modulen, da sonst eine Vielzahl von unterschiedlichen Bitströmen erstellt und bereitgestellt werden muss.

Ein wesentlich flexibleres System wird mit Hilfe eines kachelbasierten Ansatzes erreicht. Für den Entwurf eines feingranularen, kachelbasierten, dynamisch rekonfigurierbaren Systems wurde der Integrierte Design Flow für Rekonfigurierbare Architekturen (INDRA) [HKK+07] entwickelt. Dieser vereint kommerzielle und selbstentwickelte Werkzeuge, welche spezielle Entwurfsaufgaben ausführen. Zu diesen gehören die Systempartitionierung, das Floorplanning, die automatische Generierung der Konfigurati-

onsdaten (Bitströme) für die statischen und rekonfigurierbaren Komponenten des Systems und die Erstellung einer homogenen Kommunikationsinfrastruktur.

Ein FPGA besteht aus unterschiedlichen Basisblöcken, wie z.B. konfigurierbare Logikblöcke (CLBs) oder eingebetteten Speicher (BRAM). Der partiell rekonfigurierbare Bereich (PR-Bereich) wird, wie in Bild 2 b dargestellt, in Kacheln (Tiles) unterteilt. Es existieren verschiedene Typen von Tiles, die sich in der Zusammensetzung der Basisblöcke unterscheiden.

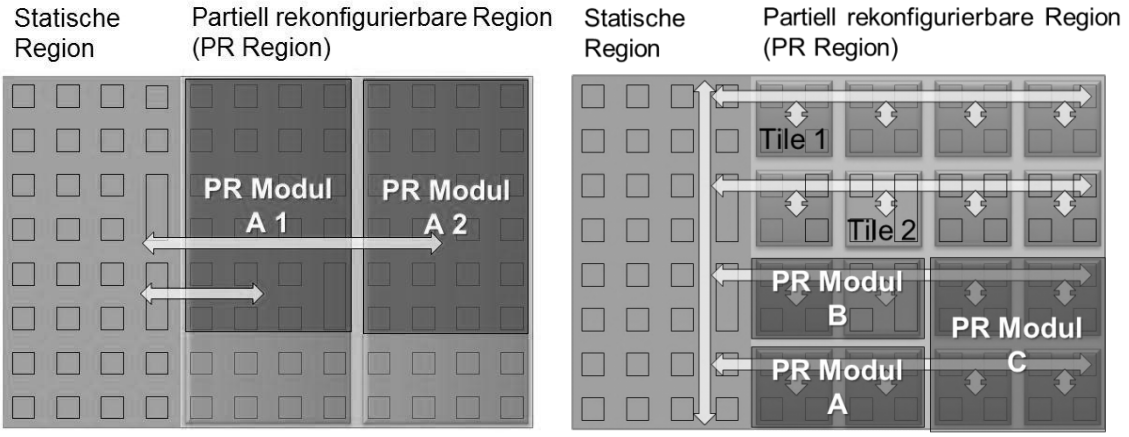

Bild 2: a) Dynamisch rekonfigurierb. System auf Basis des Xilinx Entwurfsablaufs b) Feingranulares, kachelbasiertes dynamisch rekonfigurierb. System

Die Höhe eines Tiles entspricht dabei dem kleinsten rekonfigurierbaren Bereich auf einem Xilinx FPGA – dem sogenannten *configuration frame*. Die Breite eines *configuration frames* variiert je nach Basisblock-Typ und FPGA-Familie. Ein PR-Modul belegt keine einzelne PR-Region, sondern die minimale Anzahl an benötigten Tiles, so dass mit diesem Ansatz eine sehr hohe Ressourceneffizienz erreicht werden kann. Die Flexibilität bei der Platzierung der PR-Module wird allerdings durch einen Mehraufwand beim Entwurf der Kommunikationsinfrastruktur erkauft.

In einem ersten Schritt wird die Anwendung in statische und dynamische Komponenten partitioniert. Die Partitionierung hängt dabei stark von dem ausgewählten FPGA bzw. der FPGA-Familie ab. Die Granularität der PR-Region ist durch die Länge eines *configuration frame* bestimmt und daher FPGA-Familien-spezifisch. Die Höhe variiert bei aktuellen Xilinx FPGAs zwischen 16 und 40 CLBs.

Die statischen und dynamischen System-Komponenten werden durch eine HDL (engl.: Hardware Description Language) oder eine Netzliste beschrieben. Anschließend erfolgt die Erstellung der Kommunikationsinfrastruktur. Um die Anzahl der möglichen Positionen der Hardwaremodule beibehalten zu können, sollte die Kommunikationsinfrastruktur keine weitere Heterogenität in das System einbringen. Kommerzielle FPGA-Entwurfswerkzeuge unterstützen den Entwurf von regelmäßigen Strukturen (insbesondere bei Platzierung und Verdrahtung) nicht, sodass hierfür ein neuer Entwurfsablauf und eine neue Werkzeugkette entwickelt wurden. Ausgehend von einer abstrakten

Hardwarebeschreibung ermöglicht der Design Flow für Homogene Hard Makros (DHHarMa) [KCK+11] die automatische Erzeugung eines homogenen Designs. Somit können komplexe Kommunikationsinfrastrukturen für dynamisch rekonfigurierbare Systeme erstellt werden. Die Kommunikationsinfrastruktur wird als eingebettetes Hard-Makro dem statischen Bereich hinzugefügt. Ein Hard-Makro ist ein vorplatziertes und bereits geroutetes Design, welches ohne weiteren Platzierungs- und Routing-Prozess in ein Hardware-Design integriert werden kann.

3 Grobgranulare rekonfigurierbare Architekturen

Der CoreVA Prozessor basiert auf einer konfigurierbaren VLIW (Very Long Instruction Word)-Architektur mit 32-Bit Adressraum. Das VLIW-Prinzip erlaubt feingranulare Parallelität bei vergleichsweise geringen Hardwarekosten. Im Gegensatz zu RISC Prozessoren, die Instruktionen sequentiell verarbeiten, werden bei dem CoreVA Prozessor Instruktionsgruppen von mehreren Operationen pro Takt auf parallele Ausführungseinheiten verteilt. Die Parallelität der Architektur und die Anzahl der Funktionseinheiten, dedizierter Multiplizierer/Dividierer oder Load/Store-Einheiten sind zur Entwurfszeit frei konfigurierbar. Die Ablaufplanung und Verteilung von Instruktionen auf die verfügbaren Verarbeitungseinheiten wird vom Compiler bestimmt [KLS+04]. Dies bedeutet einen geringeren Hardwareaufwand als bei superskalaren Prozessoren, bei denen die Zuteilung auf die verschiedenen Funktionseinheiten in Hardware realisiert ist.

Die Architektur besitzt dedizierte Daten- und Instruktionscaches. Als lokaler Speicher kommt eine Registerbank mit 31 32-Bit-Einträgen zum Einsatz. Im SIMD (Single Instruction, Multiple Data) Modus können in jeder Funktionseinheit zwei 16-Bit Berechnungen durchgeführt werden. Hierdurch erhöht sich die Parallelität der Architektur.

Bild 3 a zeigt die sechsstufige Pipeline des Prozessors. In der Standardkonfiguration verfügt der CoreVA über vier parallelen Ausführungseinheiten und jeweils zwei Multiplizier-, Dividier- sowie Load/Store-Einheiten. Um Datenkonflikte innerhalb der Pipeline-Struktur zu vermeiden, sind Bypässe zwischen den einzelnen Pipelinestufen des CoreVA Prozessors vorhanden. Eine vollständige Implementierung aller möglichen Bypass-Pfade vergrößert den Prozessor jedoch und begrenzt die maximale Taktfrequenz des Prozessors. Aus diesem Grund kann zur Entwurfszeit die für den jeweiligen Einsatzzweck optimale Bypass-Konfiguration gewählt werden.

Zur Optimierung der Ressourceneffizienz eines Prozessorsystems ist neben dem Prozessorkern auch eine umfassende Betrachtung der weiteren Systemkomponenten erforderlich. Insbesondere die Art der Speicheranbindung hat einen großen Einfluss auf die Ausführungszeiten und den Ressourcenbedarf. Zur Entwurfszeit kann die Anzahl der Speicherschnittstellen der CPU festgelegt sowie Größe und Anzahl der Speicherzeilen des Caches konfiguriert werden. Über einen 32-Bit breiten Systembus können die Caches des Prozessors auf einen Speicher (DDR2 SDRAM) zugreifen. Zudem wird diese Schnittstelle zur Initialisierung und zum Debuggen des Prozessors verwendet.

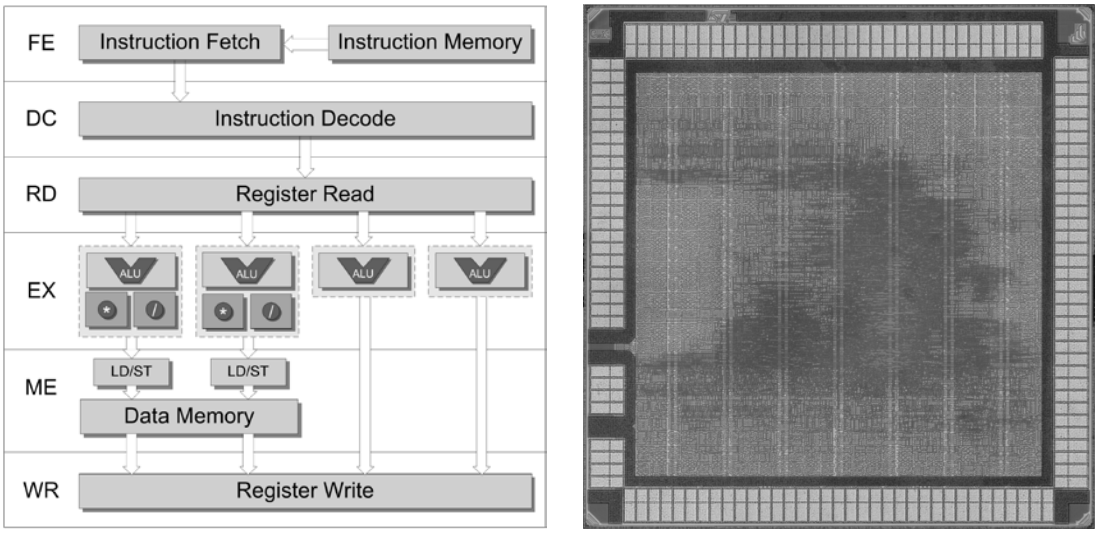

Bild 3: a) Pipelinestruktur des CoreVA VLIW Prozessors
b) Chipfoto einer CoreVA ASIC Realisierung

Neben der Konfiguration zur Entwurfszeit ist der CoreVA Prozessor auch in weiten Teilen zur Laufzeit rekonfigurierbar. Die CoreVA-Architektur ermöglicht es, einzelne Bypässe dynamisch zu deaktivieren. Hierdurch kann, abhängig von der jeweiligen Anwendung, die Konfiguration mit der höchsten Ressourceneffizienz gewählt werden [JHP+11]. Zusätzlich kann der Cache im Betrieb rekonfiguriert werden. So ist es beispielsweise möglich, die Allokationsart des Caches zu ändern oder den Cache als Scratchpad-Speicher zu verwenden [JSP+10].

Bisher wurden zwei anwendungsspezifische integrierte Schaltungen (ASICs) der CoreVA-Architektur in einem 65nm Prozess von STMicroelectronics gefertigt. Bild 3 b zeigt ein Chipfoto des 4-fach parallelen CoreVA-VLIW, der bei einer Taktfrequenz von 400 MHz eine Leistungsaufnahme von ca. 100 mW besitzt. Der in [LJB+] vorgestellte adaptive Subschwell Prozessor CoreVA-ULP (Ultra Low Power, sehr geringe Verlustleistung) basiert auf einer skalaren Ausprägung des CoreVAs. Eine speziell entwickelte Standardzellenbibliothek ermöglicht einen Betrieb bei Spannungen von 200mV bis 1,2V. Ein Power-Management-Modul dient zur adaptiven Steuerung von Taktfrequenz und Versorgungsspannung.

Um die Ressourceneffizienz des Prozessors weiter zu steigern, können anwendungsspezifische Hardwarebeschleuniger an den Prozessor gekoppelt werden. Hardwarebeschleuniger sind digitale Schaltungen, die für eine spezielle Aufgabe optimiert sind. Sie bieten gegenüber reinen Softwarelösungen einen Performanzgewinn von bis zu mehreren Größenordnungen.

Beim CoreVA Prozessor werden Beschleuniger über eine Memory Mapped IO (MMIO)-Schnittstelle in den Adressraum der CPU eingebunden. Die Auswahl der Hardwarebeschleuniger muss zur Entwurfszeit erfolgen. Zudem kann aus Kostengrün-

den die Anzahl von Hardware-Beschleunigern auf einem ASIC aufgrund des Flächenbedarfs beschränkt sein.

Die Latenz eines Zugriffs auf einen Hardwarebeschleuniger ist im besten Fall gleich der Latenz eines Zugriffs auf den Cache bei einem Treffer, d.h. ein Taktzyklus bei einem Schreib- und zwei Taktzyklen bei einem Lesezugriff. Die zusätzliche Addressdekoderlogik verursacht einen Anstieg des Flächenbedarfs um ca. 1 % und der durchschnittlichen dynamischen Leistungsaufnahme um ca. 0,8 %. Der Prozessor kann über die Beschleuniger-Schnittstelle um Komponenten zur Kommunikation mit der Außenwelt erweitert werden. Beispiele sind hier Ethernet oder eine serielle Schnittstelle (UART).

4 Partielle Dynamische Rekonfiguration von Hardwareerweiterungen

Die in Kapitel 3 vorgestellte eng gekoppelte Integration von Beschleunigern weist eine geringe Latenz auf. Jeder Beschleuniger führt jedoch zu einem anhaltend höheren Ressourcen- und Flächenbedarf auf dem Chip, wobei viele Beschleuniger nicht dauerhaft von der Anwendung benötigt werden. Beispiele sind hier die Verwendung verschiedener Verschlüsselungsalgorithmen, Filter bei der Bildverarbeitung [NSS+11], Beschleuniger für Mobilfunkstandards [JLP+11] sowie regelungstechnische Anwendungen [PHP+09].

4.1 Systemarchitektur

Um die Vorteile der vorgestellten fein- und grobgranularen Rekonfigurations-Ansätze zu kombinieren, wurde das CoreVA-System um einen FPGA erweitert, auf den Beschleuniger ausgelagert werden können. Die in Kapitel 2 vorgestellten Entwurfsabläufe INDRA und DHHarMa bieten ein hoch flexibles System zur dynamischen Rekonfiguration von Hardwarebeschleunigern. Zur Kopplung von FPGA und Prozessor-ASIC wird der Systembus verwendet (siehe Bild 4).

Zur prototypischen Implementierung der Systemarchitektur setzen wir unser modulares, FPGA-basiertes Rapid-Prototyping-System RAPTOR [PHP+10] ein. Es stehen verschiedene Kommunikations- und Datenverarbeitungsmodule (Daughter-board, DB) zur Verfügung. Das DB-CoreVA stellt eine Testumgebung für die beiden bisher gefertigten ASIC Prototypen der CoreVA Architektur bereit. Der dynamisch rekonfigurierbare FPGA, ein Virtex-4 FX100, ist auf dem DB-V4 Modul integriert. Zusätzlich stellt dieses Modul bis zu 4 GB DDR2-SDRAM zur Verfügung. Durch das DB-ETG können bis zu vier Gigabit Ethernet Anschlüsse in das System integriert werden.

4.2 Prototypische Umsetzung und Bewertung

Auf dem DB-V4 sind die statischen Komponenten – ein Speicher-Controller, ein Rekonfigurations-Controller und eine MMIO-Schnittstelle zu den Hardwarebeschleunigern – direkt an den Systembus des CoreVA-Prozessors angeschlossen. Durch die notwendige externe Kommunikation über den Systembus erhöht sich die Latenz für Lesezugriffe auf dynamisch rekonfigurierbare Beschleuniger von 2 auf 7 Taktzyklen. Schreibzugriffe werden über einen FIFO mit 16 Einträgen entkoppelt, um Wartezyklen der CPU zu vermeiden.

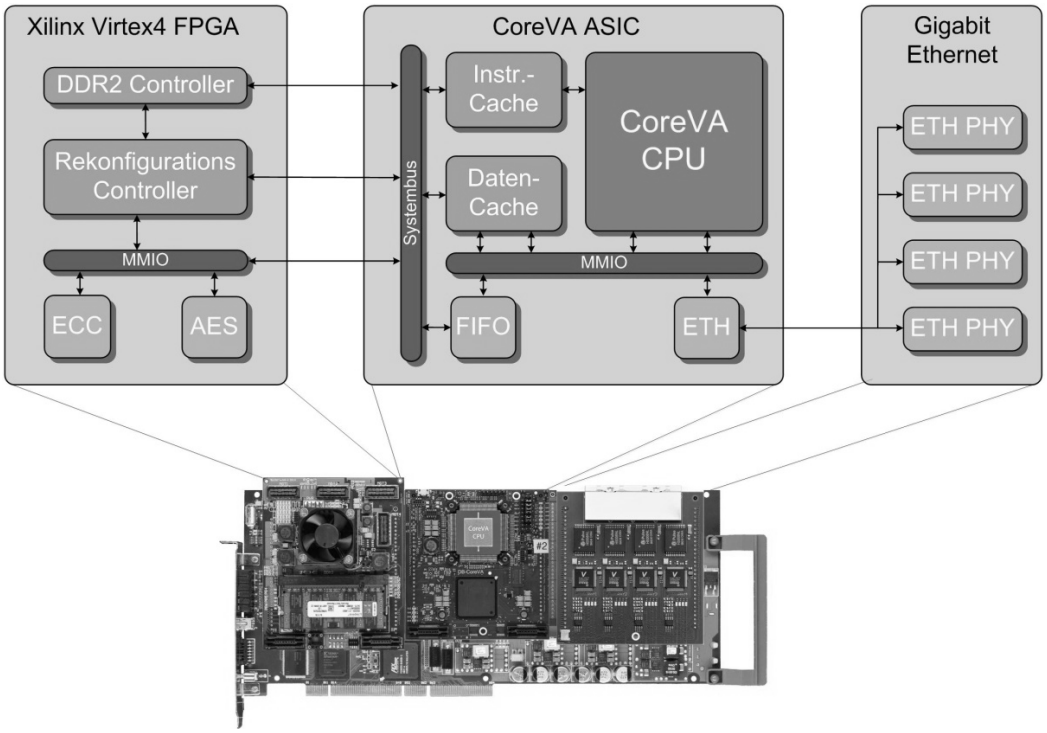

Bild 4: CoreVA CPU mit rekonfigurierbaren Hardwarebeschleunigern auf Basis des Rapid-Prototyping-Systems RAPTOR

Der in das System integrierte DDR2-SDRAM-Speicher wird als Hauptspeicher des CoreVA und für die (partiellen) Bitströme der einzelnen Hardwarebeschleuniger verwendet. Große Datenpakete können über eine direkte, schnelle Punkt-zu-Punkt-Verbindung zwischen Beschleunigern und Speicher ausgetauscht werden. Der Rekonfigurations-Controller kann ebenfalls über diese schnelle Schnittstelle mit dem Speicher-Controller kommunizieren. Die Systemarchitektur stellt sicher, dass die eingesetzten FPGAs immer mit der maximal möglichen Datenrate rekonfiguriert werden können. Eine Rekonfiguration kann über zwei verschiedene Schnittstellen durchgeführt werden. Der nur FPGA-intern verfügbare Konfigurationsanschluss (ICAP, Internal Configuration Access Port) bietet eine maximale Konfigurationsrate von 400 MB/s, der intern und extern verfügbare SelectMAP Konfigurationsanschluss von maximal 50 MB/s. Die Rekonfigurationszeit hängt hauptsächlich von der Größe des Bitstroms eines PR-Moduls ab. Diese steigt linear mit der Anzahl der verwendeten Ressourcen auf dem FPGA. Daher bestimmt die Anzahl der benötigten *configuration frames* die Rekonfigurationszeit.

In Bild 5 a wird die Rekonfigurationszeit für verschiedene Bitströme über die ICAP-Schnittstelle dargestellt. Der Rekonfigurations-Controller wird von dem auf der CoreVA CPU laufenden Programm gesteuert. Durch einen Zugriff über den Systembus teilt die CoreVA CPU dem Controller mit, dass ein bestimmter Beschleuniger benötigt wird. Dieser initialisiert den Transport des entsprechenden Bitstroms aus dem Speicher und führt die Rekonfiguration durch. Durch Modifikationen am Bitstrom durch den Rekonfigurations-Controller wird eine Umplatzierung von PR-Modulen zur Laufzeit ermöglicht, wodurch das System sehr flexibel auf Änderungen der Umgebung zur Laufzeit reagieren kann. So ist es möglich, die dynamische Fläche zur Laufzeit zu defragmentieren, um zum Beispiel Platz für zukünftig benötigte Hardwarebeschleuniger zu schaffen.

PR-Modul	**Bitstrom (in kByte)**	**Rekonfigurations-zeit (ms)**
Min. Modul (1 Tile)	202	0,50
ECC-Modul (6 Tiles)	1212	3,03
Max. Modul (16 Tiles)	3232	8,08

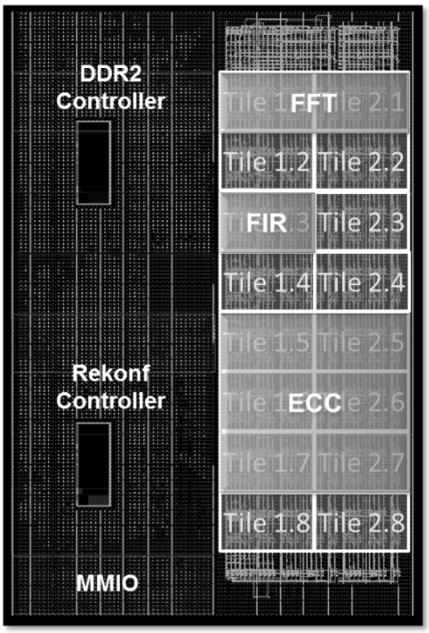

Bild 5: a) Bitstromgröße und die benötigte Rekonfigurationszeit (ICAP)
b) Prototypische Umsetzung des dynamisch rekonfigurierbaren Systems

Anhand eines Hardwarebeschleunigers für die Kryptographie mit elliptischen Kurven (ECC) haben wir eine Bewertung der neu eingeführten Systemarchitektur durchgeführt. Die Kryptographie mit elliptischen Kurven kann zur Verschlüsselung und zur digitalen Signatur bei der Übertragung von Daten verwendet werden. Durch seine hohe Komplexität kann dieser Algorithmus bei vergleichbarer Sicherheit einen kleineren Schlüssel als herkömmliche Verschlüsselungsalgorithmen verwenden. Multiplikation und Quadrierung basierend auf endlichen Körpern beanspruchen den größten Anteil an der Ausführungszeit des Algorithmus. Aus diesem Grund wurde eine Hardwareerweiterung entwickelt, die diese Operationen beschleunigt [JPD+10].

Bild 6 zeigt die Taktzyklen für einen eng gekoppelten Hardwarebeschleuniger (Hwacc ASIC), für den dynamisch rekonfigurierbaren Beschleuniger (Hwacc FPGA) sowie für eine reine Softwareimplementierung des Algorithmus. Der Flächenbedarf der ASIC-Implementierung der CoreVA CPU steigt durch bei Verwendung des eng gekoppelten

Beschleunigers um ca. 30% an (65nm Standardzellen-Bibliothek), gegenüber einer reinen Softwareimplementierung ist eine Leistungssteigerung um den Faktor 17 möglich. Der dynamisch rekonfigurierbare Beschleuniger auf dem FPGA erzielt eine Beschleunigung um den Faktor 11, wobei der Beschleuniger nicht dauerhaft auf dem FPGA vorhanden sein muss. In Bild 5 b ist die Umsetzung des dynamisch rekonfigurierbaren Systems auf dem Virtex-4 FX100 dargestellt. Auf der linken Seite sind die statischen Systemkomponenten, auf der rechten Seite die dynamisch rekonfigurierbare Fläche mit 16 Tiles angeordnet.

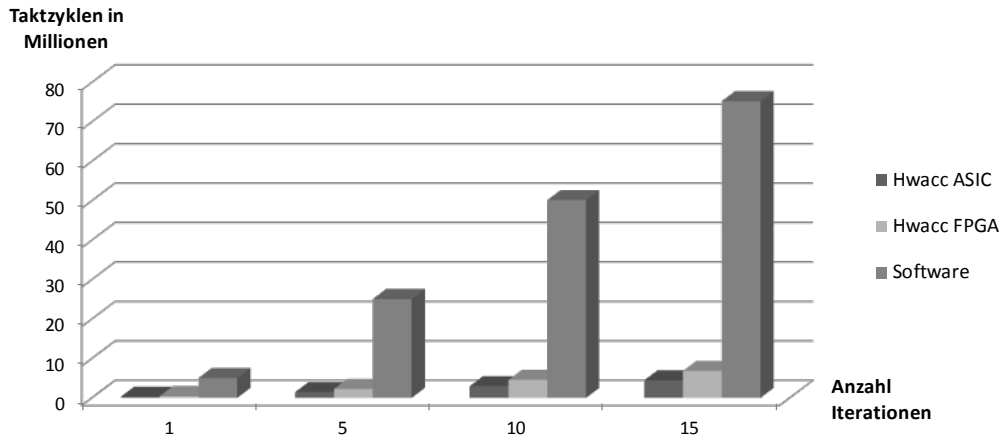

Bild 6 : Taktzyklen-Berechnung ECC Algorithmus

5 Resümee und Ausblick

In dieser Arbeit wurden feingranular rekonfigurierbare Hardwarebeschleuniger an den CoreVA Prozessor angebunden. Hierzu wurden die Werkzeuge des INDRA und des DHHarMa Entwurfsablauf verwendet um ein hoch flexibles dynamisch rekonfigurierbares System zu implementieren. Als Anwendungsbeispiel wurde ein Kryptographiebeschleuniger so angepasst, dass er in den dynamisch rekonfigurierbaren Bereich eines Virtex-4 FPGAs platziert werden kann. Im Vergleich zur reinen Softwareimplementierung kann durch diesen Hardwarebeschleuniger eine Steigerung der Ausführungszeit um den Faktor 11 erzielt werden. Hierdurch konnte gezeigt werden, dass durch Verwendung unseres Ansatzes die Ressourceneffizienz des Gesamtsystems gesteigert werden kann. In weiterführenden Arbeiten analysieren wir die Verwendung von Beschleunigern für die Bildverarbeitung und regelungstechnischen Anwendungen.

Danksagung

Dieser Beitrag ist im Sonderforschungsbereich 614 „Selbstoptimierende Systeme des Maschinenbaus“ der Universität Paderborn entstanden und wurde auf seine Veranlassung unter Verwendung der ihm von der Deutschen Forschungsgemeinschaft zur Verfügung gestellten Mittel veröffentlicht.

Literatur

[HKK+07] HAGEMEYER, J.; KETTELHOIT, B.; KOESTER, M.; PORRMANN, M.: A Design Methodology for Communication Infrastructures on Partially Reconfigurable FPGAs. In: Proceedings of Field Programmable Logic and Applications (FPL), 2007.

[JLP+11] JUNGEBLUT, T.; LIß, C.; PORRMANN, M.; RÜCKERT, U.: Design-space Exploration for Flexible WLAN Hardware. In Zorba, N.; Skianis, C.; Verikoukis, C. (Editors) Cross Layer Designs in WLAN Systems, ISBN: 978-1848768109, Troubador Publishing, Leicester, UK, pp. 521-564, November 2011.

[JHP+11] JUNGEBLUT, T.; HÜBENER, B.; PORRMANN, M., RÜCKERT, U.: A Systematic Approach for Optimized Bypass Configurations for Application-specific Embedded Processors. In: ACM Transactions on Embedded Computing Systems (TECS), 2011. Unpublished Article.

[JPD+10] JUNGEBLUT, T.; PUTTMANN, C.; DREESEN, R.; PORRMANN, M.; THIES, M.; RÜCKERT, U.; KASTENS, U.: Resource Efficiency of Hardware Extensions of a 4-issue VLIW Processor for Elliptic Curve Cryptography. Advances in Radio Science, Volume 8, 2010, pp.295-305.

[JSP+10] JUNGEBLUT, T.; SIEVERS, G.; PORRMANN, M.; RÜCKERT, U.: Design Space Exploration for Memory Subsystems of VLIW Architectures. In: Proceedings of the fifth IEEE International Conference on Networking, Architecture, and Storage (NAS2010), pp. 377-385, Macau, China, July 15-17, 2010.

[KCK+11] KORF, S.; AND COZZI, D.; KOESTER, M.; HAGEMEYER, J.; PORRMANN, M.; RÜCKERT, U.; SANTAMBROGIO, M. D.: Automatic HDL-Based Generation of Homogeneous Hard Macros for FPGAs. In: Proceedings of International Symposium on Field-Programmable Custom Computing Machines, 2011.

[KLH+11] KOESTER, M.; LUK, W.; HAGEMEYER, J.; PORRMANN, M.; RÜCKERT, U.: Design Optimizations for Tiled Partially Reconfigurable Systems. In: Very Large Scale Integration (VLSI) Systems, IEEE Journal of , vol.19, no.6, pp. 1048-1061, 2011.

[KLS+04] KASTENS, U.; LE, D. K.; SLOWIK, A.; THIES, M.: Feedback Driven Instruction-set Extension. In: Proceedings of ACM SIGPLAN/SIGBED 2004, Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'04), Washington, D.C., USA, June 2004.

[LJB+13] LUTKEMEIER, S.; JUNGEBLUT, T.; BERGE, H. K. O.; AUNET, S.; PORRMANN, M.; RÜCKERT, U.: A 65 nm 32 b Subthreshold Processor With 9T Multi-Vt SRAM and Adaptive Supply Voltage Control. In: IEEE Journal of Solid-State Circuits, vol.PP, no.99, pp.1-12, doi: 10.1109/JSSC.2012.2220671.

[NSS+11] NAVA, F; SCIUTO, D.; SANTAMBROGIO, M. D.; HERBRECHTSMEIER, S.; PORRMANN, M.; WITKOWSKI, U.; RUECKERT, U.: Applying dynamic reconfiguration in the mobile robotics domain: a case study on computer vision algorithms. In: ACM Transactions on Reconfigurable Technology and Systems (TRETS), Volume 4, Issue 3, pp. 29:1–29:22, August 2011.

[PHP+09] PAIZ, C.; HAGEMEYER, J.; POHL, C.; PORRMANN, M.; RÜCKERT, U.; SCHULZ, B.; PETERS, W.; BÖCKER, J.: FPGA-Based Realization of Self-Optimizing Drive-Controllers. In: Proceedings of the 35th Annual Conference of the IEEE Industrial Electronics Society (IECON 2009), pp. 2868–2873, Porto, Portugal, November 3-5, 2009.

[PHP+10] PORRMANN, M.; HAGEMEYER, J.; POHL, C.; ROMOTH, J.; STRUGHOLTZ, M.: RAPTOR – A Scalable Platform for Rapid Prototyping and FPGA-based Cluster Computing. In: Parallel Computing: From Multicores and GPU's to Petascale, Advances in Parallel Computing, Volume 19, pp. 592–599, ISBN: 978-1-60750-529-7, IOS press, 2010.

[HSB06] HUEBNER, M, SCHUCK, C., BECKER, J.; Elementary Block Based 2-dimensional Dynamic and Partial Reconfiguration for Virtex-II FPGAs, In: Proceedings of Int. Parallel and Distributed Processing Symposium. (IPDPS), 2006.

Autoren

Dipl.-Ing. Sebastian Korf ist seit 2012 wissenschaftlicher Angestellter im Fachgebiet Schaltungstechnik im Heinz Nixdorf Institut (HNI) der Universität Paderborn. Von 2004 bis 2011 studierte er Ingenieurinformatik an der Universität Paderborn.

Dipl.-Ing. Gregor Sievers ist wissenschaftlicher Angestellter in die Arbeitsgruppe Kognitronik und Sensorik am Exzellenzcluster Cognitive Interaction Technology (CITEC) der Universität Bielefeld. Zuvor war er ab 2009 Stipendiat im Graduiertenkolleg „Automatische Konfigurierung in offenen Systemen“ der Universität Paderborn. Von 2004 bis 2009 studierte er Ingenieurinformatik an der Universität Paderborn.

Dipl.-Ing. Johannes Ax ist seit 2011 Stipendiat der Graduiertenschule des CITECs und arbeitet in der Arbeitsgruppe Kognitronik und Sensorik am CITEC, Universität Bielefeld. Von 2005 bis 2011 studierte er Ingenieurinformatik an der Universität Paderborn.

M.Sc. Dario Cozzi ist seit 2011 Stipendiat der Graduiertenschule Cognitive Interaction Technology und arbeitet in der Arbeitsgruppe Kognitronik und Sensorik am CITEC, Universität Bielefeld. Von 2004 bis 2011 studierte er Ingenieurinformatik am Politecnico di Milano.

Dr.-Ing. Thorsten Jungeblut ist akademischer Rat in der Arbeitsgruppe Kognitronik und Sensorik am CITEC, Universität Bielefeld. Im Jahr 2011 promovierte er mit seiner Arbeit „Entwurfsraumexploration ressourceneffizienter VLIW-Prozessoren“. Zuvor war er ab 2005 wissenschaftlicher Mitarbeiter in der Fachgruppe Schaltungstechnik am HNI, Universität Paderborn.

Dipl.-Ing. Jens Hagemeyer ist seit 2012 Wissenschaftlicher Angestellter in die Arbeitsgruppe Kognitronik und Sensorik am CITEC, Universität Bielefeld. Zuvor war er ab 2006 Wissenschaftlicher Angestellter im Fachgebiet Schaltungstechnik, HNI, Universität Paderborn. Von 2000 bis 2006 studierte er Ingenieurinformatik an der Universität Paderborn.

Dr.-Ing. Mario Porrmann ist Akademischer Direktor in der Arbeitsgruppe Kognitronik und Sensorik am CITEC, Universität Bielefeld. Im Jahr 2001 promovierte er mit seiner Arbeit zur „Leistungsbewertung eingebetteter Neurocomputersysteme“ am HNI, Universität Paderborn. Von 2001 bis 2009 war er Akademischer Oberrat und von 2010 bis 2011 Vertretungsprofessor am Fachgebiet Schaltungstechnik, Universität Paderborn.

Prof. Dr.-Ing. Ulrich Rückert leitet seit 2009 die Arbeitsgruppe Kognitronik und Sensorik am CITEC, Universität Bielefeld. Er promovierte 1989 am Lehrstuhl für Bauelemente der Elektrotechnik der Universität Dortmund. Nach weiteren vier Jahren als Oberingenieur in der Fakultät für Elektrotechnik der Universität Dortmund folgte er 1993 einem Ruf auf eine C3-Professur für CAD-Methoden der Mikroelektronik an die TU Hamburg-Harburg. Von 1995 bis 2009 war er Professor für Schaltungstechnik am Heinz Nixdorf Institut der Universität Paderborn.