

Die Welt der rekonfigurierbaren Prozessoren

Teil 2: Aktuelle Produkte und deren Zielmärkte

Neben klassischen sequenziellen Prozessoren gewinnen in der Verarbeitung von Signal-Daten zunehmend programmierbare Logikbausteine an Bedeutung, die ihren Vorteil vor allem aus einem hohen Grad an Parallelität schöpfen. Dazwischen gibt es einige Konzepte, die beide Welten miteinander verknüpfen. Während sich der Teil 1 dieses Artikels [30] allgemein mit den Architekturklassen befasste, geht es nun in Teil 2 um konkrete Implementierungen.

Von Prof. Dr. Christian Siemers

Rekonfigurierbare Prozessoren stellen eine erhebliche Neuerung gegenüber den reinen Von-Neumann-Maschinen dar, so dass sich die Frage stellt, was denn damit gewonnen wird. Drei Eigenschaften spielen dabei eine Rolle, die sie besonders gegenüber den üblichen Prozessoren hervorheben: Erhöhung der Rechenleistung, Echtzeit-Fähigkeit und Verlustleistungsminimierung.

Eine Erhöhung der Verarbeitungsleistung bei gleichzeitig konstant bleibender oder sogar sinkender Verlustleistung erscheint relativ einfach erreichbar zu sein. Bei automatischer Erzeugung der Konfiguration etwa durch einen C/C++-Compiler und Optimierung auf die Bereiche, die durch ein Profiling als laufzeitintensiv identifiziert wurden, wird genau diese Erhöhung des Durchsatzes erreicht. Die cASIP-Klasse zielt in Kombination mit entsprechenden Compilern exakt auf die Verarbeitungsleistung.

Echtzeit-Fähigkeit ist wesentlich schwieriger zu erzielen als eine Steigerung der Rechenleistung, da für letztere eine statistische Verbesserung ausreicht, während die Echtzeit deterministisch garantiert sein muss. Ein Weg dahin besteht in einem „Thread“-Modell, wobei jedem Thread ein bestimmtes Zeitverhalten als Eigenschaft zugeordnet wird. Anhand dieser Eigenschaften können dann die Threads ver-

teilt werden, z.B. auf den PLD-Anteil in μ P/PLD-Kombinationen, falls besondere Echtzeit-Anforderungen zu erfüllen sind. Mit anderen Worten: Es ist

Klasse	Steigerung der Verarbeitungsleistung	Echtzeit-Fähigkeit	Verlustleistungsminimierung
μ P/PLD-Kombination	in Grenzen	ja, aber schwieriges Co-Design	–
cASIP	ja, insbesondere mit Compiler-Unterstützung	–	–
FPFA	ja, insbesondere für datenstromorientierte Algorithmen	ja, aber schwieriges Co-Design	ja, bei hoher Verarbeitungsleistung

Tabelle 2. Architekturklassen und erreichbare Ziele

ein schwieriger Weg, per Co-Design eine Applikation entsprechend zu verteilen und dann auch noch die Echtzeit-Fähigkeit nachzuweisen, aber sowohl die μ P/PLD-Kombination als auch die FPFA-Klasse unterstützen die Implementierung wenigstens, indem sie Partitionierungen und kooperative Designs ermöglichen.

Bleibt noch die Verlustleistungsminimierung mit Hilfe der neuen Architekturen. Hier sticht insbesondere die FPFA-Klasse hervor, die eine drastische Steigerung des Durchsatzes insbesondere bei datenstromorientierten Algorithmen (also „klassische“ digitale Kommunikationstechnik) mit geringer elektrischer Leistungsaufnahme verbindet. Hier werden auch schon Faktoren von 10 bis 100 genannt, sowohl für die Er-

höhung der Verarbeitungsleistung als auch zugleich für die Verlustleistungsminderung [13, 14]. Tabelle 2 fasst die Ziele der Architekturklassen zusammen.

Die Klasse der gemischten Architekturen (μ P/PLD)

Zu dieser Klasse gehören u.a. die AT94k-Familie von Atmel, die IP-Blöcke (Intellectual Property) Nios II (Altera), PicoBlaze (Xilinx) und MicroBlaze (Xilinx) sowie die Hard-Core-Implementierung eines oder zweier PowerPC 405 in einem Xilinx-FPGA. Während die AT94k-Familie eine reine Hardware darstellt und einen 8-bit-Mikrocontroller (AVR) mit einem AT40k-FPGA (Atmel) auf einem Chip verbindet, bieten die beiden „Großen“ im PLD/FPGA-Geschäft, Altera und Xilinx, jeweils zunehmend CPU-IPs für ihre FPGAs an.

Die Produkte sind unterschiedlich, die Software-Tools zur Herstellung der Konfiguration für den Chip und der

Programme für den Prozessorteil stammen jeweils vom Hersteller, aber die Entwicklungssystematiken ähneln sich doch sehr (Bild 7). Nach einer intensiven Systembetrachtung in Schritt 1 ist der Systementwickler aufgefordert, eine den Randbedingungen entsprechende Partitionierung durchzuführen. Genau dieser Schritt ist außerordentlich schwierig und stützt sich üblicherweise auf die individuellen Erfahrungen, denn die späteren Ergebnisse hängen sehr von der frühen Partitionierung ab. Im Anschluss daran sind es zwei Designflüsse, wobei der Hardware-Teil (Bild 7: linker Designfluss) durch Makros erheblich erleichtert werden kann.

Als Makros bieten sich z.B. Standard-Peripherieelemente wie Schnittstellen, Timer usw. an. Diese können

hinzugefügt werden, wobei insbesondere die Einbindung in die Software interessant ist. Sie erfolgt z.B. über „#include“-Dateien in C, Bereitstellung von Gerätetreibern usw. Wirklich interessant ist der Eigenbau von Peripherieelementen oder auch Co-Prozessoren. Hierdurch lassen sich applikationsspezifisch große Einsparungen an Rechenzeit für den Prozessor erzielen, allerdings ist das manuelle Design sicher von wesentlich höherem Schwierigkeitsgrad.

Einige der am Markt verfügbaren Architekturen sind in *Tabelle 3* zusammengefasst, ohne Anspruch auf Vollständigkeit. Im Gegenteil: Es gibt tatsächlich erheblich mehr Soft-Cores, d.h. Prozessorkerne, die in einer Hardware-Beschreibungssprache als Intellectual Property (IP) vorliegen, als die Tabelle auflistet. Bei Hard-Cores hingegen sieht es anders aus: Hier ist die Zahl sehr begrenzt, und zwar auf die Tabelleneinträge für Atmel und Xilinx. Altera bietet zwar nach wie vor die Kombination zwischen ARM-CPU und Stratix-II-FPGA an, weist auf der Homepage [15] jedoch ausdrücklich darauf hin, dass besser ein NIOS-II-Design zu verwenden sei. Aus diesem Grund wurde auf diesen Eintrag verzichtet.

Tabelle 3 gibt natürlich Details nicht wieder. So sind z.B. die Aussagen über die enthaltenen Prozessorkerne sehr summarisch, beispielsweise liegt der Performance-Unterschied zwischen dem NIOS-IIe (Economy-Version, optimiert auf möglichst wenig Logik-elemente) und der auf Geschwindigkeit getrimmten Fast-Version NIOS-IIf bei einem Faktor 1:7, was allein aus den Taktfrequenzen nicht zu errechnen wäre. Zu NIOS-II existiert übrigens ein Manual zu Erzeugung von

Mehrprozessorsystemen auf einem Chip [16].

Die 8-bit-Versionen AT94k und PicoBlaze sind typische kleine Mikrocontroller, die in Kombination mit dem „restlichen“ FPGA-Anteil (der beim PicoBlaze durchaus 99,7 % betragen kann [17]) allerdings sehr leistungsfähige Systeme ergeben können. Bei diesen Systemen ist Assembler als Entwicklungssprache durchaus noch akzeptabel, obwohl auch für AVR leistungsfähige C-Compiler angeboten werden. Die in der Spalte Entwicklungsumgebung gegebenen Informationen beziehen sich auf typische Sprachen zur Beschreibung der Programme, die Umgebungen selbst sind häufig Eclipse-basiert bzw. ähnlich aufgebaut.

Die 32-bit-Prozessoren sind natürlich von sich aus leistungsstark, selbst die Taktfrequenzen lassen wenig Wünsche offen. Will man übrigens die Soft-Cores verändern, beispielsweise durch Hinzufügen neuer Instruktionen, dann wird es knifflig, und dann hören auch die guten Tutorials der Anbieter auf. Allerdings sind die angebotenen IPs in Grenzen konfigurierbar (beim NIOS II können etwa bestimmte Hardware-Optionen wie Multiplizierer ein- oder ausgeschaltet werden, beim MicroBlaze sind Ausnahmenbehandlung und Cache-Architektur einstellbar), so dass ein wenig Adaptierbarkeit erreicht wird. Dennoch: Will man applikationsspezifische Prozessoren, sollte man einen Blick auf die nächste Klasse werfen.

► Die ASIP/cASIP-Klasse

ARC: Der Herausforderer

ARC [18] setzte 2004 mit seiner ARC-700-Familie einen Meilenstein bei den konfigurierbaren Prozessoren und stieg

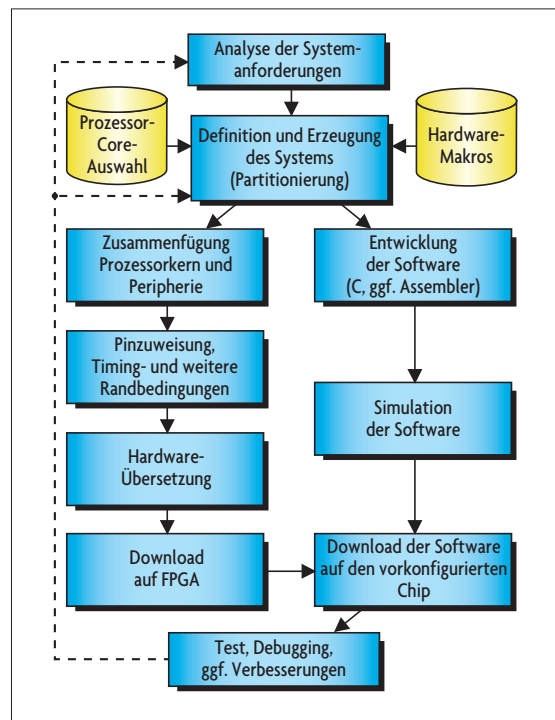


Bild 7. Designfluss für gemischte µP/PLD-Architekturen.

damit zugleich in die Oberliga der synthetisierbaren Mikroprozessorkerne auf [19] – in Konkurrenz zu ARM 9, ARM 11, Mips 24K Pro und Tensilica Xtensa (siehe unten). ARC will dabei besonders die System-on-Chip-Designer ansprechen, also jene Designs, die auf einen quasi vollständigen Systemchip hinauslaufen.

Alle Möglichkeiten, das ARC-basierte SoC zu gestalten, beziehen sich auf die Designzeit, zur Laufzeit sind derzeit keine Konfigurationen vorgesehen. ARC unterscheidet dabei Konfigurierbarkeit und Erweiterbarkeit: Konfigurierbarkeit bedeutet die Auswahl aus einem Pool von Basiskomponenten, Erweiterbarkeit das Einfügen von neuen Bestandteilen, ggf. auch vom Benutzer definiert. Diese Grenze ist natürlich fließend und abhängig davon, was als

Core/PLD	Anbieter	CPU-Typ	Datenbreite	Max. CPU-Taktfrequenz; Architektur	PLD-Typ	Entwicklungsumgebung CPU/PLD
NIOS II	Altera	NIOS II (Soft-Core in 3 Versionen: economy, standard, fast)	32	200/165/195 MHz; RISC (1/5/6 Pipeline-Stufen)	Stratix, Stratix II, Cyclone, Cyclone II	C, Assembler/VHDL
AT94k	Atmel	AVR (Hard-Core)	8	25 MHz; RISC (2 Pipeline-Stufen)	AT40k	(C,) Assembler/VHDL
MicroBlaze	Xilinx	MicroBlaze (Soft-Core)	32	90 bis 200 MHz; RISC (3 Pipeline-Stufen)	Spartan 3, Virtex II Pro, Virtex 4	C, Assembler/VHDL
PicoBlaze	Xilinx	PicoBlaze (Soft-Core)	8	88 bis 200 MHz; RISC (2 Zyklen/Instruktion)	CoolRunner, Spartan 1/2/3, Virtex xx	Assembler/VHDL
PowerPC	Xilinx/IBM	PPC405 (Hard-Core)	32	450 MHz; RISC (5 Pipeline-Stufen)	Virtex II Pro, Virtex 4	C, Assembler/VHDL

Tabelle 3. Übersicht zu Architekturen der µP/PLD-Klasse

Tabelle 4.
Übersicht zur Konfigurierbarkeit/Erweiterbarkeit von ARC600/700 (ohne Anspruch auf Vollständigkeit)

	ARC600	ARC700
Basis:		
Anzahl der Pipeline-Stufen	5	7
Max. Taktfrequenz (0,13-µm-Prozess)	290 MHz	533 MHz
Verlustleistung	0,04 mW/MHz	0,15 bis 0,16 mW/MHz
Präzise Ausnahmenbehandlung	nein	ja
Konfigurierbare Komponenten:		
Memory Management Unit (MMU)	nein	ja, Page-Größe 8 Kbyte
Translation Look-aside Buffer (TLB)	nein	ja, 256 Einträge, 2-Wege-assoziativ
Cache / Closely Coupled Memory (CCM)	ARC605/610: CCM ARC625: Cache/CCM	ARC710: CCM ARC725/750: Cache/CCM
Cache-Konfiguration	Instruktion und/oder Daten, 2 bis 32 Kbyte, 1- bis 4-Wege-assoziativ, 16 bis 128 byte Cache-Zeilenlänge	Instruktion und/oder Daten, 2 bis 32 Kbyte, 1- bis 4-Wege-assoziativ, 16 bis 128 byte Cache-Zeilenlänge
DSP-Instruktionen	optional	ja, wenige zusätzlich optional
DSP-Speicher X,Y	optional (außer ARC605)	optional
Erweiterungen:		
Gleitkomma-Einheit	optional	optional

Basiskomponentensatz bezeichnet wird und was nicht.

Die Komponenten liegen bei ARC als synthesefähige Verilog-Beschreibungen vor, deren Zusammenfügung mit Hilfe des Tools „ARChitect“ erfolgt. *Tabelle 4* fasst zusammen, was in den einzelnen Familien konfigurierbar ist

sowie welche Erweiterungen angeboten werden bzw. denkbar sind.

Die ARC600-Familie zielt auf geringe Verlustleistung, ARC700 auf große Systeme mit Betriebssystemen. Letzteres ist aus der Integration der MMU mit dem (großen) Translation Look-aside Buffer klar ersichtlich, auch die

präzise Ausnahmenbehandlung – die vollständige Ausführung aller Instruktionen vor dem Fehler, die Unterdrückung der fehlerhaften Instruktion vor einem Schreiben der Ergebnisse, Übermittlung der Fehlerquelle und Neustart der Ausführungspipeline an der unterbrochenen Stelle – fällt hier hinein.

Die Einfügung von nutzerdefinierten Erweiterungen erfolgt im Übrigen per SystemC- bzw. Verilog-Beschreibung unter Einbindung des ARChitect. So sind z.B. Co-Prozessor-Instruktionen, die nebenläufig zu allen Bearbeitungen in der Pipeline stattfinden und auch eine unterschiedliche Anzahl von Phasen haben können, sowie zusammengefasste bzw. kombinierte Instruktionen wie bei DSPs möglich.

Tensilica Xtensa

Die Grundidee zur Xtensa-Architektur [20] entstand 1997: Es sollte ein kompletter Designfluss entstehen, der das Design eines rekonfigurierbaren Prozessors mit einer Steigerung der Rechenleistung und Einsparungen bei Verlustleistung und Chipfläche ermöglicht. Hierzu wurde letztendlich eine Basisarchitektur erfunden, die dann in Verbindung mit den Design-Tools zur Designzeit so konfiguriert werden kann, dass eine applikationsspezifische Hardware entsteht. *Bild 8* zeigt den detaillierten Designfluss.

Zunächst war „Handarbeit“ angesagt, um die neuen Instruktionen zu identifizieren und zu beschreiben. Die dafür eigens entwickelte Sprache heißt „Tensilica Instruction Extension“ (TIE) Language. Dieses Verfahren hat den großen Vorteil, dass das Produkt Tool-unterstützt korrekt ist: Aus der High-Level-Beschreibung werden automatisch die entsprechenden Beschreibungen für die Hardware und die Compiler-/Assembler-Konfiguration erzeugt. Dieses Verfahren ist auch schon beim LISATek-System (siehe Teil 1 [30], *Bild 3*) zum Einsatz gekommen.

Seit Mitte 2004 wird nun auch ein automatisiertes Verfahren zur Herstellung der spezifischen Instruktionen angeboten, XPRES (Xtensa PProcessor Extension Synthesis) genannt [21]. *Bild 9* zeigt den entsprechenden XPRES-Designfluss. Der große Vorteil von XPRES

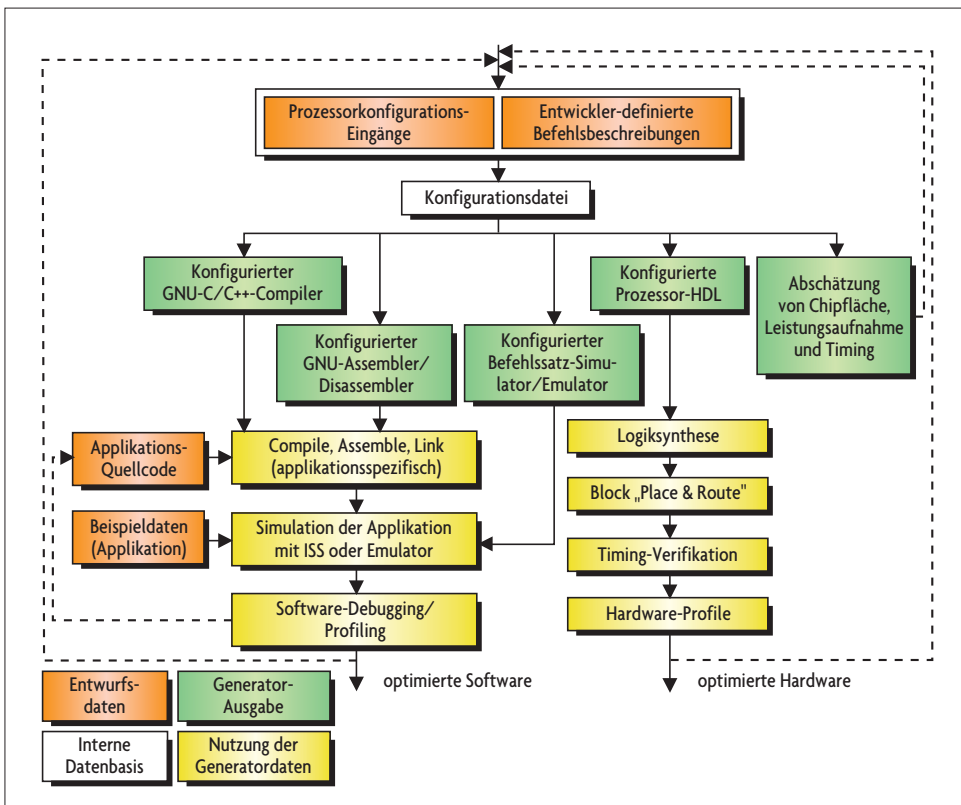


Bild 8. Detaillierter Designfluss für die Xtensa-Architektur von Tensilica.

ist, dass hieraus automatisch neue Instruktionen erzeugt werden: Der Designer sieht sofort, welchen Effekt er erzielt – ggf. sogar in (geschätzter) Form von Gatterzahlen (Siliziumfläche) und Verlustleistung.

Kein Wunder, dass Xtensa 2004 bei sämtlichen Benchmarks in dieser Prozessorklasse (und darüber hinaus) die Wettbewerber geschlagen hat [22]. Bild 10 zeigt die Xtensa-V-Architektur und welche Komponenten konfigurierbar sind. Die neuere LX-Architektur zeichnet sich durch zahlreiche Verbesserungen aus, so eine optionale zweite Load/Store-Pipeline, konfigurierbare Peripherie, das neue FLIX-Format (Flexible Length Instruction eXtension) für die designerdefinierten Teile und einen verbesserten DSP-Teil. Dies alles hat seinen Preis: So lag die Lizenz für eine Single-Prozessor-Implementierung des Xtensa LX in 2004 bei 550 000 Dollar, das XPRES-System für eine („Floating“) Lizenz bei zusätzlichen 100 000 Dollar.

Stretch S5000

Die Firma Stretch, Inc. [23], geht einen Schritt weiter als Tensilica und ARC: Ihre S5-Architektur ist im Betrieb rekonfigurierbar, weil die applikations-spezifischen Ausführungseinheiten SRAM-basiert konfigurierbar sind. Das hat mehrere Konsequenzen:

Zunächst hat Stretch, deren Architektur und Bausteine 2004 angekündigt wurden [24] und mittlerweile lieferbar sind, die Tensilica-Xtensa-V-Architektur als Grundlage gewählt – sicher eine gute Wahl, denn dies erspart viel Entwicklungsarbeit und liefert zugleich eine bekannte Basis. Die beiden darin vorhandenen designerdefinierten Blöcke – einer für den spezifischen Registersatz, einer für die Ausführung (Bild 10) – wurden durch feste Hardware-Blöcke ersetzt, von denen der Ausführungsblock SRAM-basiert konfigurierbar ist (Bild 11).

Dies bedeutet, dass Stretch konfigurierbare Prozessoren bzw. Mikrocontroller „von der Stange“ liefern kann, die auf einer bestimmten Konfiguration der Xtensa-Architektur basieren und dennoch konfigurierbar sind. Der Stückpreis beträgt ca. 35 Dollar in Stückzahlen [24]. Für den Anwender ergibt sich der große Vorteil, dass gleich

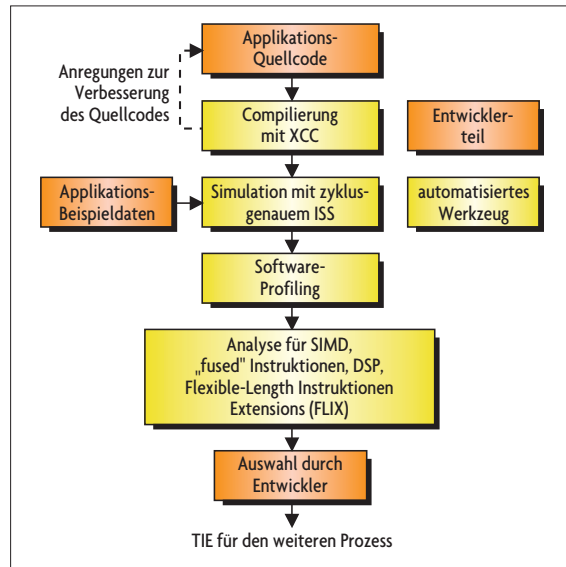


Bild 9. Beim XPRES-Designfluss werden Befehlsatzerweiterungen automatisch anhand des Quellcodes einer Applikation erzeugt.

mit der realen Hardware entwickelt werden kann.

Die Entwicklung selbst unterscheidet sich kaum von der für „normale“ Mikrocontroller: Das Programm wird in C/C++ entwickelt und übersetzt. Einziger Unterschied (Bild 5 in [30]) ist das Profiling zur Ermittlung der Laufzeitdaten und ggf. die Auswahl von verschiedenen Konfigurationsmöglichkeiten durch den Designer. Dieses Verfahren wird durch den Stretch-C-Compiler (SCC) durchgeführt, der

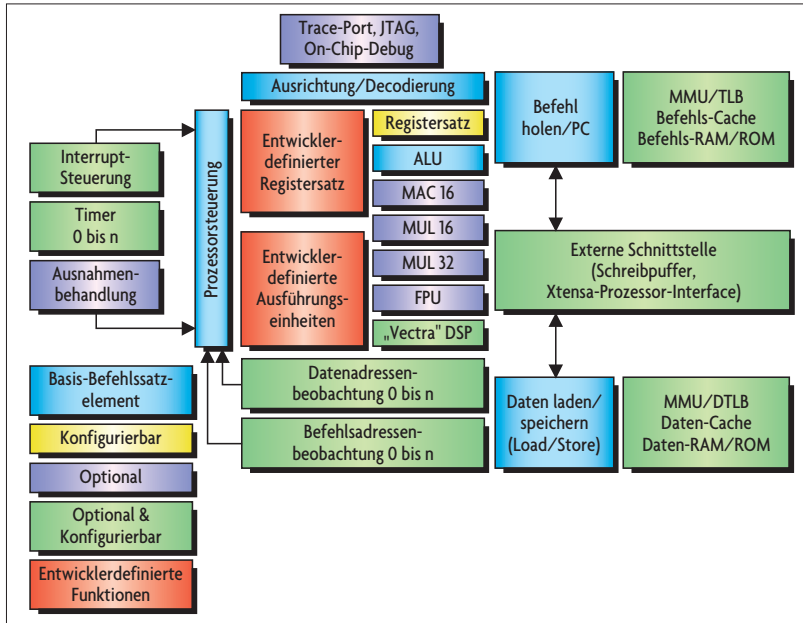


Bild 10. Die Xtensa-V-Architektur bietet dem Entwickler optionale und konfigurierbare Erweiterungen wie in einem Baukastensystem.

Field-Programmable Function Arrays

Die dritte Klasse der rekonfigurierbaren Prozessoren wurde mit FPFA (Field-Programmable Function Arrays) bezeichnet. Darunter versteht man ein (zweidimensionales) Array von ALUs, bei dem sowohl die Funktionen der ALUs als auch die Verbindungen untereinander konfigurierbar sind. Dieser Ansatz eignet sich grundsätzlich als Stand-alone-Prozessor, wird aber gerne als Coprozessor genutzt, weil bestimmte Aufgaben – vornehmlich datenflussorientierte Algorithmen – sich sehr gut auf diese Arrays abbilden lassen, andere hingegen weniger. Von mikrocontrollerähnlichen Strukturen mit integrierten Peripherielementen wird allerdings abgesehen, Peripherielemente wie Kommunikationsschnittstellen etc.

zusammen mit der „Fabric“-Struktur auf diese Übersetzung optimiert wurde.

Der Nachteil dieses Ansatzes ist natürlich leicht ersichtlich: Man tauscht die große Flexibilität gegen Einbußen bei der Rechenleistung. Während die Stretch-Bausteine – derzeit werden S5500, S5610 und S5620 angeboten, die Unterschiede liegen im Bereich der Peripherie – im 130-nm-Herstellungsprozess mit 300 MHz betrieben werden können, ist die Taktfrequenz bei der programmierbaren Struktur auf 100 MHz beschränkt. Im Fall einer anwendungsspezifischen Hardware-Entwicklung ließe sich eine höhere Taktfrequenz realisieren.

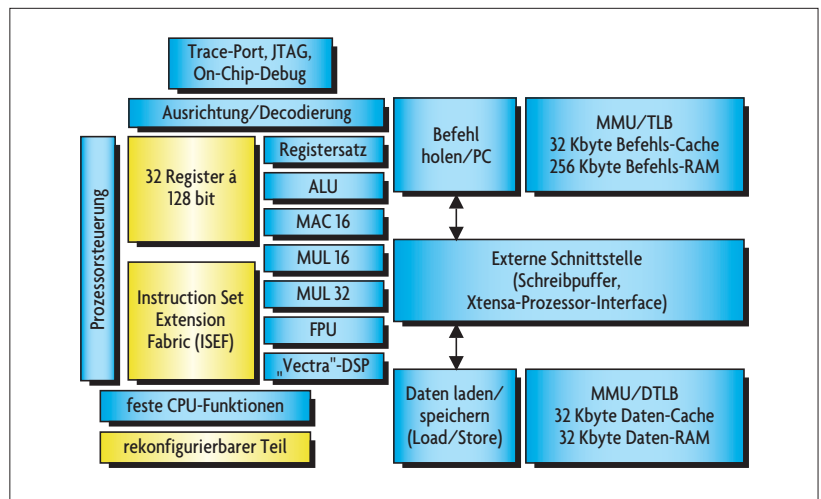


Bild 11. Die Stretch-S5-Architektur basiert auf Tensilicas Xtensa V, verfügt zusätzlich aber über rekonfigurierbare Befehlssatzerweiterungen (128-bit-Registersatz und ISEF).

werden gerne an Mikroprozessoren abgetreten. Soweit sind die Gemeinsamkeiten beschrieben. Die Unterschiede sind in der ALU-Größe, den Verbindungstopologien und der Gesamtstruktur der Architekturen zu finden.

Elixent D-Fabrix

Elixent [25] bietet eine als D-Fabrix (aktuell angekündigt ist die Version 2.0) bezeichnete Architektur an, die

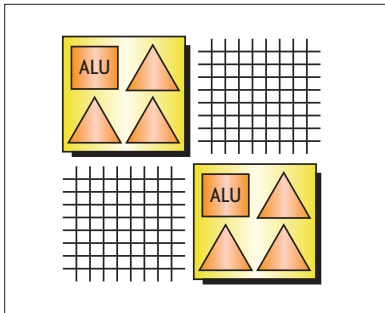


Bild 12. Die D-Fabrix (hier ein „Tile“) von Elixent besteht aus einem Array von ALUs und Verdrahtungsmatrizen.

u.a. von Toshiba als Beschleuniger von Multimedia-Applikationen genutzt wird. Diese Architektur besteht aus so genannten Tiles (Kacheln), die jeweils zwei ALUs, Register zum Speichern und eine „Switchbox“ enthalten, siehe auch *Bild 12*. Von diesen Tiles sind dann 64 bis 1024 in einem Baustein integriert.

Die Verarbeitungsbreite einer ALU beträgt 4 bit. Größere Verarbeitungsbreiten werden natürlich durch Zusammenschalten von mehreren ALUs erreicht, auf Kosten der Verarbeitungszeit (durch Kopplung der Zwischen-

werte, z.B. Überträge bei Addition). Elixent hat damit den Kompromiss zwischen Ausnutzung (Flächeneffizienz) und Geschwindigkeit mehr in Richtung der Effizienz gelegt.

Der Preis hierfür ist das Programmier-Interface, denn die Datenbreite ist in Software-Sprachen eine eher untergeordnete Größe, nur Hardware-Beschreibungssprachen können bitgenau codieren. So ist es erklärlich, dass die angebotenen Beschreibungssprachen VHDL/Verilog, Matlab und Handel-C [26] heißen.

Handel-C, von der britischen Firma Celoxica entwickelt und vertrieben, ist eine C-ähnliche Sprache, die allerdings sehr viele Erweiterungen und Abwandlungen gegenüber C aufweist. Insbesondere ist es hier möglich, die Datenbreite der Variablen bitgenau zu bestimmen, was Handel-C zur Hardware-Beschreibungssprache auch für FPGAs und natürlich PPFAs prädestiniert. Ein echter C-Compiler wäre allerdings Voraussetzung, wenn die Elixent-DFA1000-Familie zur Beschleunigung von bestehenden Applikationen durch einfache Neucompilierung eingesetzt werden soll.

NEC Dynamically Reconfigurable Processor (DRP)

NEC Electronics [27] setzt auf eine Eigenentwicklung, den Dynamically Reconfigurable Processor (DRP). Dieser besteht wiederum aus einem Array von ALU/Register-Blöcken, in diesem Fall mit 8 bit Datenverarbeitungsbreite (*Bild 13*). Ein Array umfasst dann etwa 64 solcher „Processing Elements“, am

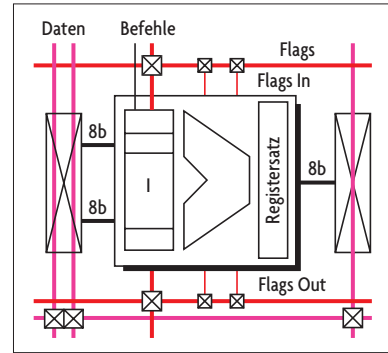


Bild 13. „Processing Element“ (PE) des DRP (Dynamically Reconfigurable Processor) von NEC. Typischerweise umfasst ein Array 64 dieser PEs.

Rand befindet sich jeweils ein Anschluss für Speicher. Die Betriebsfrequenz dieser Chips liegt bei etwa 200 bis 300 MHz.

Die wichtige Nachricht ist jedoch, dass diese Architektur einen C-Compiler hat, der automatisch die Applikation teilt und die Konfigurationen aus dem Quellcode heraus berechnet. Damit stellt NEC dieses Array auf die Software-Stufe der Mikroprozessoren, was der Akzeptanz sehr förderlich sein sollte.

PACT XPP

Der „Klassiker“ unter den PPFAs ist wohl nach wie vor die „eXtreme Processing Platform“ (XPP) der Firma PACT [14, 28]. Die Mitglieder dieser Familie sind hierarchisch aufgebaut und bestehen aus einem Array von einem bis vier „Processing Array Clustern“ (PAC) mit jeweils 48 bis 64 Processing Array Elements (PAE). Jedes PAE enthält eine ALU und ein Register,

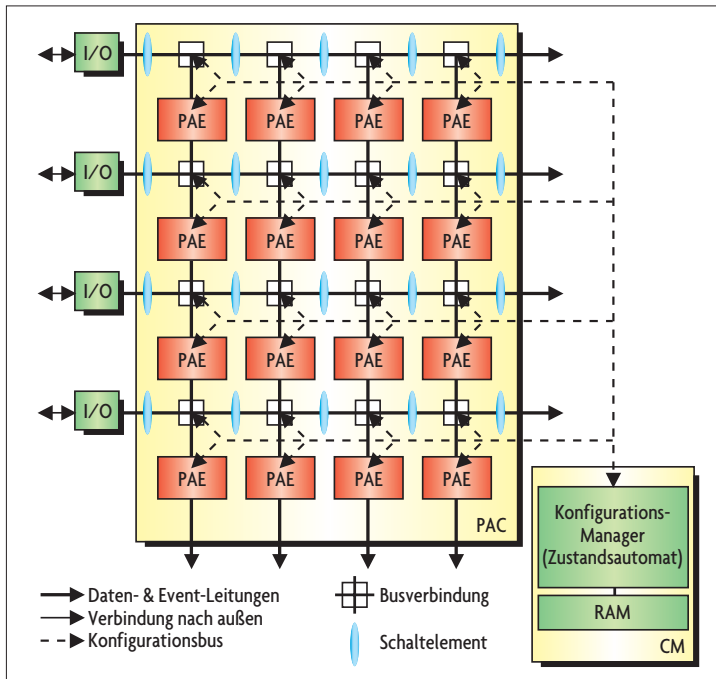


Bild 14. „Processing Array Cluster“ der XPP-Architektur.

alternativ auch lokales RAM oder ähnliche spezielle Bestandteile. Bild 14 zeigt ein aus Darstellungsgründen reduziertes Processing Array Cluster mit 16 PAEs.

Das Daten-Routing in diesem Cluster ist horizontal über die gesamte Cluster-Breite ausgeführt, und zwar in segmentierbaren (und damit besser nutzbaren) Leitungen. Vertikales Routing führt durch die PAEs (in beiden Richtungen). Neben den Datenleitungen existieren auch Event-Leitungen, die Rechnungen triggern bzw. unterbinden können. Der „Configuration Manager“ (CM) steuert die dynamischen Rekonfigurationen und basiert auf einem Mikroprozessor.

Auch hier sei wieder der Blick auf das Software-Interface gerichtet. Die intrinsische Programmiersprache heißt NML (Native Mapping Language) und der Name ist Programm: In diesem Assembler muss die Position der Rechnung im Array angegeben werden. Ein C-Compiler existiert, aber die Ergebnisse eines handoptimierten NML-Programms sind offenbar immer noch wesentlich besser. Hier liegt sicher noch Verbesserungspotential.

► Network-on-Chip

Nach Abschluss der Betrachtungen der drei Klassen im Bereich der rekonfigu-

rierbaren Architekturen sei noch ein Blick auf eine Spezialität gerichtet: On-Chip-Netzwerke. Wie in Teil 1 dieses Artikels [30] bereits dargestellt wurde, ist dies ein integraler Bestandteil künftiger SoCs, insbesondere bei der GALS-Architektur (Globally Asynchronous, Locally Synchronous) für große Chips.

Hier bietet die französische Firma Arteris [29] entsprechende Intellectual-Property-Blöcke an. Nun sind diese IPs kaum in die Designs wie Xtensa oder ARC integrierbar, denn deren Produkte sind sozusagen nahezu fertig und werden konfiguriert oder es wird etwas mit entsprechenden Tools dieser Firmen hinzugefügt, so dass die Einfügung von vornherein gesichert ist, aber die Dinge können sich schließlich ändern. So hat z.B. ein typisches Xtensa-LX-Design mehrere hundert Taktomänen, also Bereiche, die mit einem (abschaltbaren) Takt unabhängig von den anderen versorgt werden. Gut zu wissen, dass es bereits Firmen wie Arteris mit entsprechendem IP gibt.

► Echtzeit-Fähigkeit nur in Ansätzen

Der Markt der konfigurierbaren Prozessoren erscheint außerordentlich dynamisch. Die Entwicklungsbestrebungen gehen nicht nur in Richtung höherer Rechenleistung, sondern auch in Richtung der Optimierung von Siliziumfläche und Verlustleistung. Auf der Strecke bleiben leider momentan tragfähige Konzepte in Bezug auf die Echtzeit-Fähigkeit.

Am meisten tut sich auf dem Gebiet der ASIPs bzw. cASIPs, wahrscheinlich deshalb, weil diese den besten Migrationsweg aus Richtung der Mikroprozessoren bieten. Im einfachsten Fall tut es auch ein neuer Compiler – fast unmerklich für die Entwicklung. gs

Literatur

- [13] Hartenstein, R.W.; Hirschbiel, A.G.; Riedmüller, M.; Schmidt, K.; Weber, M.: A Novel ASIC Design Approach Based on a New Machine Paradigm. IEEE Journal of Solid-State-Circuits, Vol. 26, No. 7 (1991).
- [14] Pact: www.pactcorp.com
- [15] Altera: www.altera.com
- [16] NIOS-II-Prozessor: www.altera.com/literature/lit-nio2.jsp
- [17] Xilinx: www.xilinx.com
- [18] ARC: www.arc.com
- [19] Halfhill, T.: ARC700 Secrets Revealed. Microprocessor Report 6/21/04-01.
- [20] Tensilica: www.tensilica.com
- [21] Halfhill, T.: Tensilica's Automaton Arrives. Microprocessor Report 7/12/04-01.
- [22] Halfhill, T.: Tensilica Tackles Bottlenecks. Microprocessor Report 6/1/04-01.
- [23] Stretch: www.stretchinc.com
- [24] Wilson, R.: Two stretch configurable-CPU arena with new entries. EETimes 2004, April 26th.
- [25] Elixent: www.elixent.com
- [26] Celoxica: www.celoxica.com
- [27] NEC Electronics: www.necel.com
- [28] Siemers, C.: Rechenfabrik: Ansätze für extrem parallele Prozessoren. c't 2001, H. 15, S. 170 bis 179.
- [29] Arteris: www.arteris.com
- [30] Siemers, C.: Die Welt der rekonfigurierbaren Prozessoren; Teil 1: Lösungen auf dem Weg zum konfigurierbaren System-on-Chip. Elektronik 2005, H. 21, S. 42ff.



Prof. Dr. Christian Siemers

studierte als „Nordlicht“ Mathematik und Physik in Kiel. Nach einigen Jahren in der Industrie bei Siemens in München und Dräger in Lübeck lehrt er seit zehn Jahren an Fachhochschulen im Gebiet der Technischen Informatik. Zunächst an der FH Stralsund, später an der FH Westküste in Heide/Holstein tätig, ist er seit 2002 an der FH Nordhausen (Thüringen) für Prozesstechnik berufen. Seine Forschungsinteressen liegen in den Gebieten der programmierbaren Architekturen, der Systemmodellierung und der eingebetteten Systeme.

► E-Mail: christian.siemers@computer.org