Informatik Spektrum 18: 228-229 (1995) © Springer-Verlag 1995

Das aktuelle Schlagwort

*Begriffe aus technischer, praktischer und theoretischer Informatik und deren Anwendungen**

Custom Computing Machines

Reiner Hartenstein

Im Laufe dreier Jahrzehnte hat sich die anwender-programmierbare Logik (FPL - field-programmable logic [1][2]) von der Kuriosität zu einer kommerziell wichtigen Technologie-Plattform entwickelt, deren Weltumsatz kurz vor der Milliarden-US-Dollar-Schwelle steht, bei hohen Wachstumsraten. Man wird sich mehr und mehr bewußt, daß hier die Strukturprogrammierung als ein alternatives Basis-Paradigma der klassischen prozeduralen Programmierung gegenübertritt. In beiden Fällen entsteht nach der Fabrikation quasi eine Hardware „ohne Eigenschaften", die erst durch in ein RAM geladenen Programmier-Code individualisiert (engl.: personalized) wird – sequentieller Code beim Mikroprozessor und Rekonfigurations-Code beim FPL-Baustein.

Eine wichtige FPL-Anwendung ist die beim Einsatz von Mikroprozessoren seit Ende der 80er Jahre zunächst in verschiedenen Forschungsprojekten realisierte Anwendungs-spezifische Erweiterung des Befehlssatzes zwecks Durchsatzerhöhung durch Beseitigung von Engpässen. Oft erweisen sich innere Schleifen in der Software als solche Durchsatzengpässe. Dabei werden in eine an den Prozessor angekoppelte FPL-Hardware wenige, beispielsweise zwei oder drei, besonders mächtige Operatoren hineinkompiliert, die im Mikroprozessor quasi wie zusätzliche Befehle aufgerufen werden können. Solche „weichen", d. h. an die Durchsatzprofile von Anwendungsgebieten anpaßbaren Prozessor-Anordnungen wurden als ASIPs (Application specific Instruction Processors [3]) bekannt. Oder im Namen einer internationalen Konferenz-Reihe [4] [5] auch als „Custom Computers" oder „Custom Computing Machines" (CCM) bekannt. Eine wichtige Anwendung für CCMs ist der Hardware-/Software Ko-Design.

Bei der CCM-Entwicklung kann man viererlei Ansätze unterscheiden. Bei der rein externen Anpassung bleibt die Struktur des Mikroprozessors, etwa eines Universal-Mikroprozessors, völlig unberührt. Solche Zusatzhardware, auch Akzelerator genannt, wird dabei über eine schon vorhandene Schnittstelle angeschlossen. Dies kann beispielsweise über die Speicher-Schnittstelle geschehen, wobei ein externer Zusatzoperator über eine für diesen reservierte Speicher-Adresse anstelle einer Speicherzelle aufrufbar ist. Mit diesem Ansatz wurden bei einigen Anwendungen, wie Kryptographie, Datenkompression, string matching und Lösung von Gleichungssystemen, Akzelerationsfaktoren bis zu mehr als zwei Größenordnungen erzielt [6]. Eine zweite Möglichkeit ist die Verwendung eines mikroprogrammierbaren Prozessors ([3] u. a.), was gegenüber ersterem Ansatz vielleicht etwas mehr Flexibilität verspricht. Die dritte Möglichkeit ist der anwendungsspezifische Entwurf eines ganzen Prozessors – also eines ASIP, wie etwa mit Pipeline-Struktur [7].

Die beiden ersten Ansätze haben den Nachteil der durch die von-Neumann-Prinzipien bedingten sehr engen Kopplung zwischen Befehlsabwickler bzw. Mikrobefehlsabwickler und ALU, was innerhalb der Prozessor-Architektur einen festen Vorrat an Operator-Ressourcen voraussetzt. Deshalb können ohne eine komplette Auflösung der Prozessor-Architektur zusätzliche Operatoren praktisch nur quasi wie trojanische Pferde über hierfür eigentlich nicht vorgesehene Schnittstellen eingeführt werden. Die komplette Entwicklung einer speziellen ASIP-Architektur ist hingegen sehr aufwendig. Alle drei Ansätze slnd praktisch nur über den Einsatz von Hardware-Experten realisierbar, welche die Ergebnisse Ihrer Synthese-Bemühungen quasi als *hardware patches* von außen an die periphere Prozessor-Hardware dranflicken.

Eine viel höhere Flexibilität in der FPL-Anwendung ist hier durch einen vierten Ansatz über ein neues Maschinenparadigma erreichbar, das anstelle eines Befehlsabwicklers einen „*data sequencer*" verwendet [8]. Dieses Paradigma verlangt nur eine sehr lose Kopplung zwischen „Sequencer" und ALU. Deshalb ist der Ressourcen-Vorrat weitgehend austauschbar, etwa durch eine FPL-basierte rekonfigurierbare ALU (rALU [8]), ohne daß dabei die Prozessor-Architektur verlorengeht und neu definiert werden muß. Über neuartige Compiler [9] wird die Anwendung dieses Daten-prozeduralen Maschinen-Paradigmas dem Programmierer auch ohne Hilfe von Hardware-Experten zugänglich. Dies wird dadurch möglich, daß hier an die Stelle von *hardware patches* die Integration von prozeduraler Programmierung und Strukturprogrammierung tritt – unter einer gemeinsamen Benutzeroberfläche, wie sie Programmierern vertraut ist. Der Paradigmenwechsel vom prozeduralen in den strukturellen Bereich kann dabei durch Verwendung eines *cross compiler* (wie beispielsweise in [9]) gegenüber dem Anwender verborgen werden, sofern dies als Vorbeugung gegen Akzeptanzprobleme zweckmäßig erscheint.

Literatur

1. H. Grünbacher, R. Hartenstein: Field-Programmable Gate Arrays: Architectures and Tools for Rapid Prototyping; Springer-Verlag 1992

Reiner Hartenstein, Universität Kaiserslautern,
D-67663 Kaiserslautern

*Vorschläge an: Prof. Dr. Frank Puppe, Institut für Informatik, Universität Würzburg, Am Hubland, D-97074 Würzburg, und Dr. Dieter Steinbauer, GEZ, Freimersdorfer Weg 6, D-50829 Köln

2. R. Hartenstein, M. Servit: Field-Programmable Logic: Architectures, Synthesis and Applications; Springer-Verlag 1994
3. P. Marwedel: Tree-based mapping of Algorithms to Predefied Structures; Proc. ICCD 1993, IEEE Computer Society Press 1993
4. D. A. Buell, K. E. Pocek: Proc. IEEE Workshop on FPGAs for Custom Computing Machines, April 4-7, 1993, Napa, CA, USA; IEEE Computer Society Press 1993
5. D. A. Buell, K. E. Pocek: Proc. IEEE Workshop on FPGAs for Custom Computing Machines, April 10-13, 1994, Napa, CA, USA; IEEE Computer Society Press 1994
6. P. Bertin, D. Roncin, J. Vuillemin: Introduction to Programmable Active Memories; in (Eds.) J. McCanny et al.: Systolic Array Processors; Prentice-Hall 1989
7. I. Huang, A. Despain: High-Level Synthesis of Pipelined Instruction Set Processors and Back End Compilers; Proc. DAC 1992
8. R. Hartenstein et al.: A novel paradigm of parallel computation and its use to implement simple high performance hardware; Future Generation Computer Systems 7 (1991/92)
9. R. Hartenstein, K. Schmidt: Parallelizing Compilation for a Novel Data-parallel Architecture; in (eds.) J. P. Gray, F. Naghdy: Parallel Computing - Technologyand Practice; Omsha Trokyo 1994

Eingegangen am 14.12.1994, in überarbeiteter Form am 06.02.1995