High Performance Computing: über Szenen und Krisen

(eingeladener Beitrag)

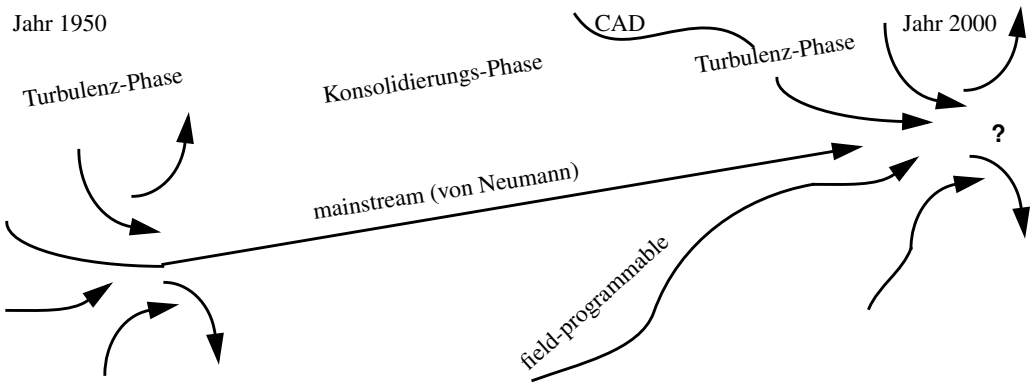
Reiner W. Hartenstein

Universität Kaiserslautern
Erwin-Schrödinger-Straße, D-67663 Kaiserslautern, Germany
Fax: ++49 631 205 2640, email: hartenst@rhrk.uni-kl.de
http://dr_bunsen.informatik.uni-kl.de

Zusammenfassung

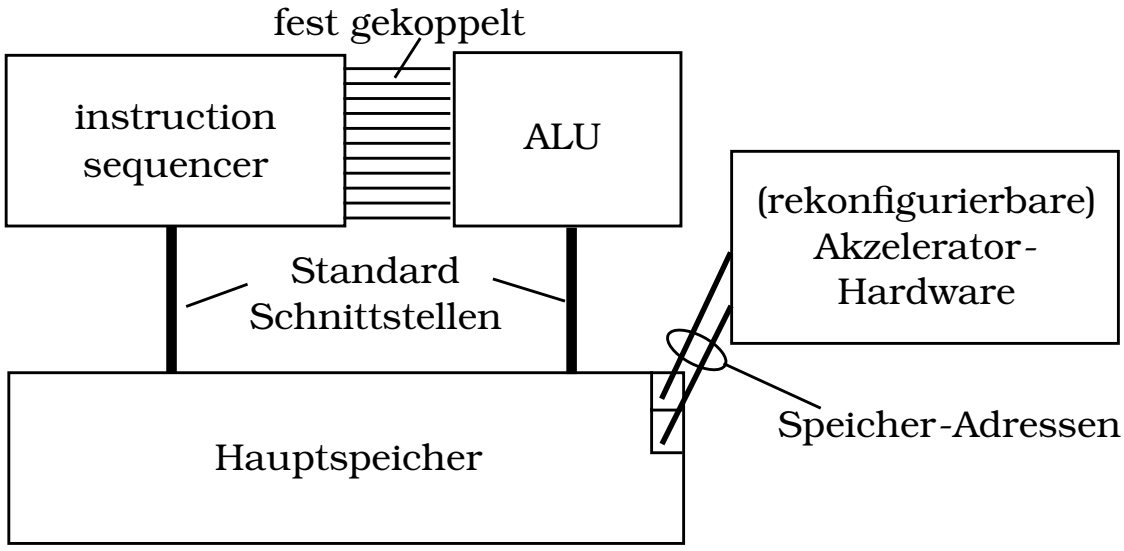
"High Performance Computing" ist ein schillernder Begriff. In einer Vielfalt von Forschungs- und Entwicklungs-Szenen spielt er unter unterschiedlichen Aspekten eine wichtige Rolle. Das Gebiet zersplittert sich immer mehr. Zu den Disziplinen der Supercomputer der "massiv parallelen Computersysteme" und spezifischer Anwendungsgebiete kommen noch Szenen der Embedded Systems, der Unterhaltungselektronik sowie der Desktops, Laptops und Handhelds hinzu. Alle wollen doch das gleiche: nämlich "high performance" und das zu einem angemessenen Preis, oder sogar extrem billig.

Figure 1. Entwicklungsphasen einer Wissenschaft



In der Entwicklung wissenschaftlicher Disziplinen wechseln sich Perioden einer geordneten Mainstream-Entwicklung mit chaotischen Perioden des Umbruchs ab, wo bisher belächelte Abseits-Richtungen die Chance haben, zu einem neuen Mainstream zu werden (Figure 1.). Gerade jetzt kündigen zunehmende Turbulenzen wieder einmal einen Umbruch an. Die Vielfalt der Szenen des "High Performance Computing" wird neu gemischt. Wird das Monopol des von-Neumannschen- Mainstream-Paradigma (computing in time) nun auch bei Mehrzweck-Hardware gebrochen über eine wachsende Anwendung strukturprogrammierbarer Hardware-Plattformen (computing by the yard)? Der Beitrag versucht, mehr Durchblick durch die Turbulenzen und Ausblick auf Tendenzen zu vermitteln, auch aus Sicht der Technischen Informatik.

Figure 2. Erweiterung der ALU durch die Hintertür (bei CCMs)



1. Vorwort

"High Performance Computing" ist ein schillernder Begriff. In einer Vielfalt von Forschungs- und Entwicklungs-Szenen spielt er unter unterschiedlichen Aspekten eine wichtige Rolle. Das Gebiet zersplittert sich in immer mehr. Dem Supercomputing nahestehende Gruppen diskutieren gelegentlich sogar eine Aufspaltung der ganzen Disziplin in "Computer Science" und "Computing Science". Warum bilden "Real Time Systems", "Multimedia-Hardware", "Digitale Signalverarbeitung" und "Custom Computing Machines" und andere so stark von einander isolierte Szenen? Alle wollen doch das gleiche: nämlich "high performance" und das zu einem angemessenen Preis. Die Mannigfaltigkeit der Anwendungen reicht für eine Erklärung der babylonischen Verwirrung wohl nicht aus. Ein neues IEEE Technical Committee genannt "Engineering of Computer-Based Systems (ECBS)", versucht Ordnung in das Chaos zu bringen. Aber was fehlt, sind Szenen-übergreifende allgemeine Modelle, nach denen das TC-ECBS noch sucht.

Ein Umbruch scheint sich abzuzeichnen. Allmählich wird die Vielfalt der Szenen neu gemischt. Eine der "Parallel Computing-Szenen" gibt sogar offen zu, sich in einer schweren Krise zu befinden. Das Warten auf den Terabyte-(pro Sekunde-)Bus erscheint nicht mehr unbedingt als Hoffnung auf einen Ausweg aus dieser Krise. Auf dem High-Performance-Computing-Symposium in New Delhi Ende 1995 wurde die feldprogrammierbare Logik per Eröffnungs-Keynote durch einen hochkarätigen Sprecher vom MIT hier offiziell eingeführt [1]. Er spach vom "Computing by the Yard" (statt computing in time). Dies ist insofern ein Meilenstein, als noch kurz zuvor solche Forschungsszenen sich überhaupt nicht für diese neue Technologie- Plattform interessiert haben. Der Beitrag versucht, mehr Durchblick durch die Turbulenzen und Tendenzen zu verschaffen insbesondere aus der Sicht der Technischen Informatik.

2. Einführung

Das komplette *system on a chip* (SOC) ist im Vormarsch. Bei VLSI-Entwurfs-Werkzeugen rutscht der Brennpunkt des allgemeinen Interesses alle 6 Jahre eine Abstraktionsebene höher. Nach Polygonen, Transistoren, der Gatternetz-Ebene (logic synthesis) hat die Register Transfer-Ebene mit high level synthesis und VHDL (mehr in Europa) oder Verilog (mehr in den USA) ihren Höhepunkt überschritten.

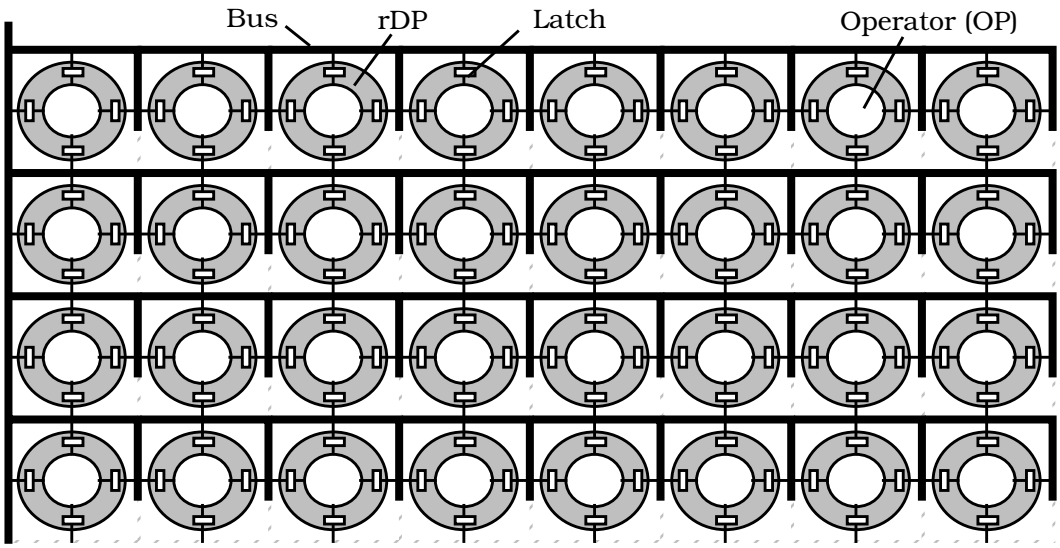
Nun wendet sich das Interesse mehr und mehr der Prozessor-Ebene zu, das vielleicht um das Jahr 2000 seinen Höhepunkt erreichen könnte. Spezielle Prozessoren und Microcontroller haben im Mikrochip-markt bereits einen höheren Weltumsatz als Universalprozessoren. Anwendungs-spezifische Mikroprozessoren sind auf mehreren F&E-Szenen im Vormarsch ([2], [3]): als ASIPs (application-specific instruction set processors [4], - [7]), als FCCMs (FPGA-based custom computing machines [8] - [15]), oder durch Anwendung des Hardware-Software-Co-Design ([16] - [26]). Die radikalere ASIP-Methode kreiert vollständig anwendungsspezifische Befehlssätze, weshalb auch Compiler, Betriebssystem, Simulator und Emulator, mit jedem ASIP neu generiert werden müssen. Bei FCCMs und H/S-Codesign hingegen erfährt ein Universalprozessor nur wenige anwendungsspezifische Erweiterungen seines (Standard-)Befehlssatzes durch externe Akzelerator-Hardware (AH). Der Vorteil ist die Verwendbarkeit kommerziell erhältlicher Software. Der Nachteil der anscheinend flexibleren Methode über FCCMs oder H/S-Codesign besteht darin, daß die Schnittstelle zwischen AH einerseits und Host nebst Systemsoftware andererseits eigentlich Flickwerk ist. Sobald sich der Befehlssatz ändert, wird ein neuer Instruction Sequencer benötigt wegen der festen Kopplung zwischen letzterem und der ALU (Bild 2). Um dies zu vermeiden, muß die AH beispielsweise über reservierte Speicheradressen angeschlossen werden (Figure 2.).

3. Revolution gegen von Neumann?

Abseits vom Mainstream des ablaufprogrammierten von-Neumann-Prozessors entwickelte sich lange unbemerkt die field-programmable logic (FPL). Das FPLA beispielsweise ist schon drei Jahrzehnte alt. Doch inzwischen haben FPL-Bausteine wie FPGAs mehrere Milliarden US-Dollars Weltumsatz erreicht. Hardware ist soft geworden und neben der Ablaufprogrammierung wird die Strukturprogrammierung immer wichtiger ([27] - [30]), Zu rekonfigurierbaren Bausteinen kommen nun immer mehr rekonfigurierbare kommerzielle boards hinzu (wie z. B. [31] - [35]). Eine wichtige Anwendung sind embedded reconfigurable accelerators (beispielsweise [36] - [39]). Werden rekonfigurierbare Architekturen (wie beispielsweise auch [40] - [43]), d. h. strukturprogrammierbare Prozessoren irgendwann einmal dem von-Neumann-Paradigma den Rang ablaufen? Oder werden sich diese beiden Welten zu einem neuen Lehrgebäude vereinigen um ein neues Super-Paradigma hervorzubringen? Aber bis dahin ist noch ein weiter Weg. Derzeit sind die zwei Welten des computing in time und des computing in space noch nicht miteinander vermählt. Zu den systolischen Arrays ([44], [45]), dem Produkt ihrer ersten flüchtigen Begegnung, kamen sie eigentlich nur wie die Jungfrau zum Kind.

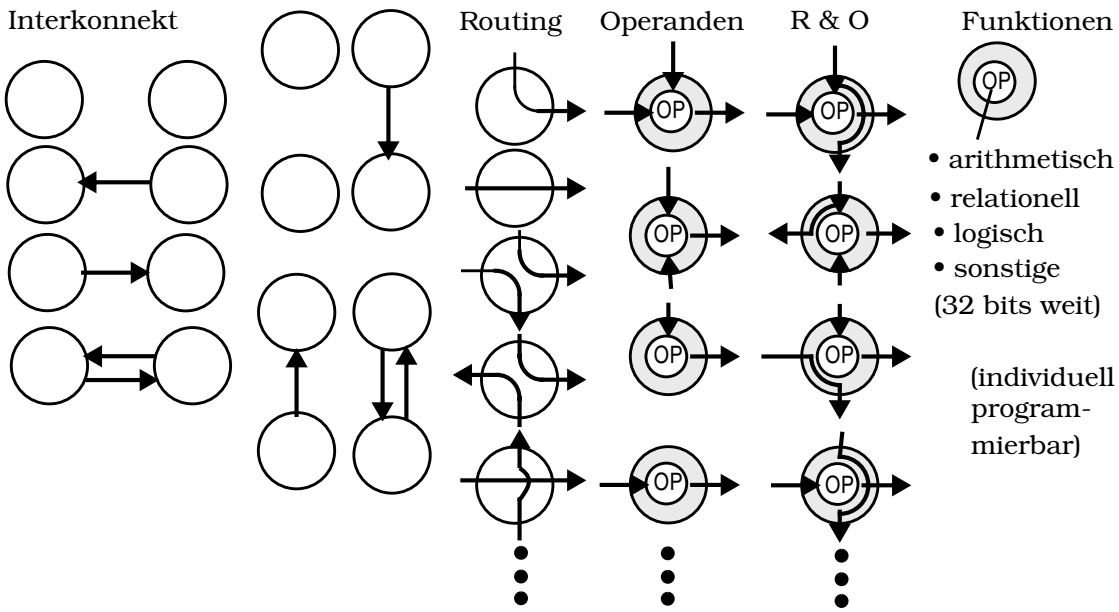
4. Rekonfigurierbare parallele ALUs

Figure 3. : Beispiel eines rDPA (reconfigurable Data Path Array) nach Kress



Apropos Arrays: die rekonfigurierbaren Arrays [46] demonstrieren, daß die Strukturprogrammierung nicht notwendigerweise eine Domäne wirrer Sauerkrautstrukturen ist, wie etwa für schnelles Prototyping von Hardware. Wie einfach so ein Array zu programmieren ist, soll am Beispiel des rDPA (rekonfigurable Data Path Array) nach Kress demonstriert werden [47] [48]. Figure 3. zeigt einen solchen Array von rDPs (rekonfigurierbaren Datenpfaden) und Figure 4. veranschaulicht die strukturellen Programmiermöglichkeiten: des Interkonnekt zwischen den rDPs, des Routing durch einen rDP hindurch, des Routing zum Operator (OP) im Inneren des rDP, sowie die Wahl der Funktion des OP. Als Beispiel einer Anwendung sollen die 8 Gleichungen in Figure 5. dienen. Ein erster Versuch ergebe die Lösung nach Figure 6. , in der 17 Zyklen den einzigen verfügbaren Bus zum Durchsatz-Engpaß machen. Ein Optimierungs-Algorithmus nach der Methode des simulated annealing erzeugt dann nach nur wenigen Sekunden Rechenzeit daraus die Lösung nach Figure 7. , wo der Bus nur noch für einen einzigen Transferzyklus benötigt wird. Alle anderen Verbindungen werden lokal zwischen direkt benachbarten rDPs angelegt. Die Möglichkeit, rDPs auch ausschließlich als Routing-Elemente zu programmieren, d. h. ohne Nutzung des im rDP enthaltenen Operators OP, verschafft mehr Flexibilität zugunsten besserer Optimierungslösungen. Verblüffend ist bei dieser Methode, daß gute Lösungen entstehen ohne den Zwang, diese auch nachvollziehen zu müssen. Auch entfällt die Mühe der Konzipierung komplizierter und womöglich trotzdem wenig effizienter Heuristiken für einen Compiler. In Kaiserslautern wurde ein doppelt partitionierender Compiler entwickelt, der in einem C-Dialekt geschriebene Programme umsetzt in sequentiellen Kode für einen (von-Neumann-)Host und Xputer-Daten-Sequenzierer, sowie Rekunfigurationskode für ein rDPA ([63] - [65])

Figure 4. Möglichkeiten des Programmierung des rDPA nach Figure 3.



5. Auf der Suche nach neuen Paradigmen

Figure 5. Ein Anwendungsproblem als Beispiel für die rDPA-Strukturprogrammierung

```
y10 := a0 * (b0 + 2 * c0);          y11 := a1 * (y10 + 2 * c1);
y20 := 5 * d0 + e0 + (f0 + b0):     y21 := 5 * y20 + e1 + (f1 + y10):
y30 := g0 * (h0 + 2 * e0);          y31 ;= y30 * (y40 + 2 * e1);
y40 := (5 * d0 + e0) * f0;          y41 := (5 * y20 + e1) * f1;
```

Ausgereifte Disziplinen haben in der Regel ein Ordnungsschema, das für die Navigation in den Weiten des Design-Universums als eine Art Koordinatensystem eine zumindest grobe Orientierung unterstützt. Vielleicht spannt ein allgemeines Modell für die Grundstrukturen von Objekten dieser Disziplin zu einem solchen Koordinatensystem die Dimensionen auf. Für die klassische Disziplin des sequentiellen Computing steht im von-Neumann-Paradigma ein hervorragendes Basismodell zur Verfügung, das sich wie ein roter Faden durch das gesamte Lehrgebäude hindurchzieht. Doch wie sieht es in neueren Disziplinen der Informatik oder der Technischen Informatik aus, die jetzt diese Turbulenzen verursachen? An welchem Paradigma orientieren sich Hardware/Software Co-Design, Custom Computing Machines, Parallel Computing, High Performance Computing, oder wie auch immer solche nicht-klassischen Disziplinen heißen mögen?

Figure 6. Erster Lösungsansatz für das rDPA-Routing-und-Placement zum Problem nach Figure 5.

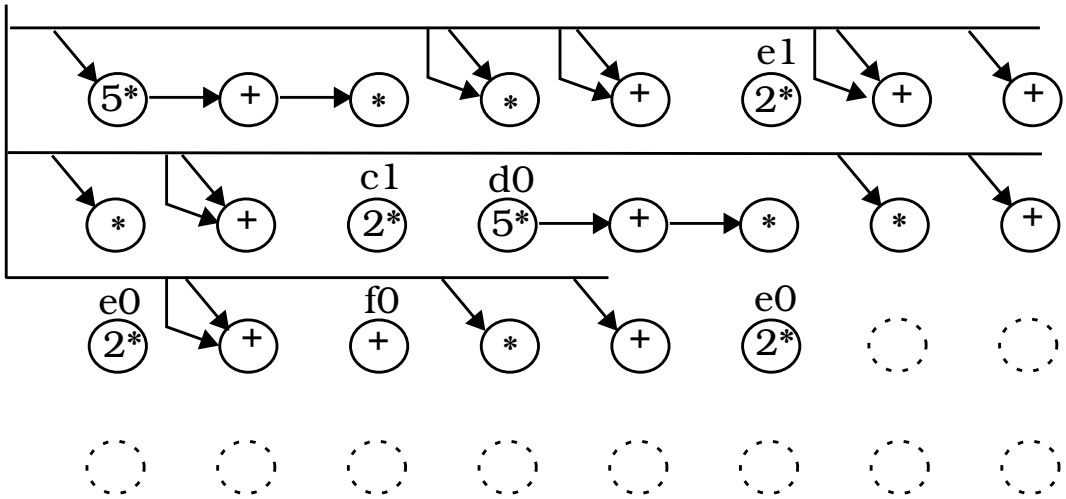
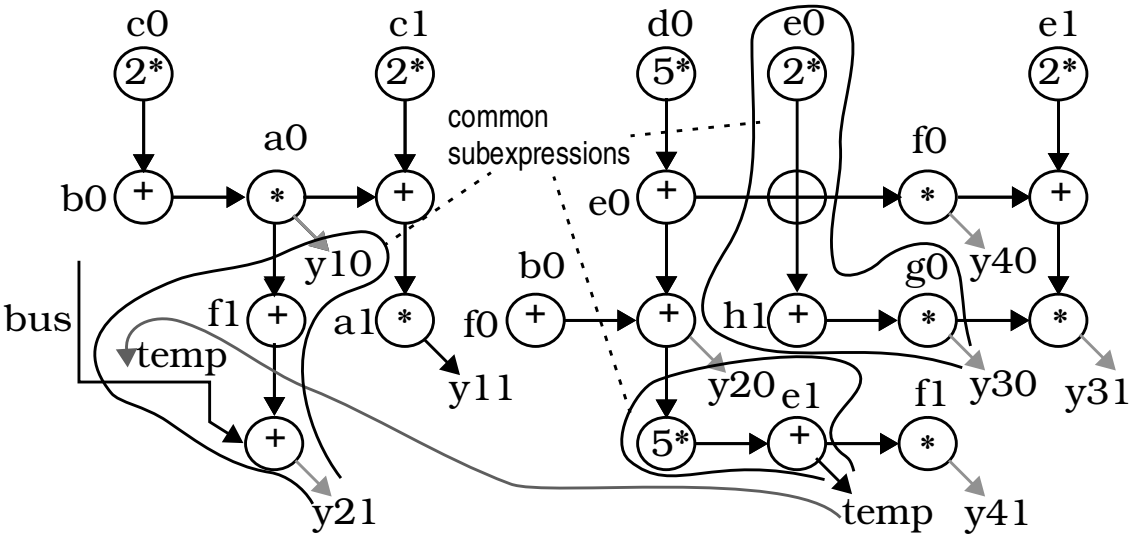


Figure 7. Optimiertes rDPA-Routing-und-Placement-Lösung der Gleichungen in Figure 5.

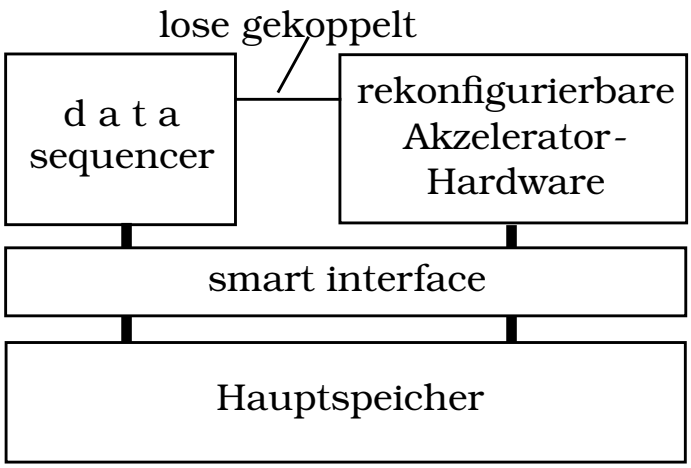


Die Krise des Parallel Computing wird derzeit in den Podiumsdiskussionen ihrer Konferenzen diskutiert. Je mehr der Markt schrumpft, desto höher und weiter wirbelt der Staub durch die Turbulenzen der sich aufblähenden internationalen akademischen Forschung. Die Zahl der Tagungen wächst immer weiter und ist kaum noch abschätzbar. Nachdem man das Ziel aus den Augen verloren hat, vervielfacht man die Anstrengungen. Noch orientiert sich die Szene primär am von-Neumann-Paradigma. Aber vielleicht ist gerade dies die Ursache der Krise. Dieses Modell ist ein Computing-Paradigma, jedoch kein Kommunikations-Paradigma. Man möchte meinen, dies ist das falsche Paradigma, sodaß es kaum eine Navigationshilfe für den Weg durch die Fülle der Architekturen sein kann. Es ist eine Abstraktionsebene zu tief angesetzt. Ist dieses falsche Paradigma schuld daran, daß es hier kaum Übersichts-Papiere gibt und die Fülle der Konferenzen nur recht diffuse Zielrichtungen aufweist und die Programme meist einem chaotischen Gemischtwarenladen gleichen? Oder liegt die Ursache für das Durcheinander lediglich in der Tagungsinflation, kombiniert mit der Knappheit der Reiseetats, was zu einer Strategie verleitet, die in jedem nicht abgelehnten Beitrag einen zahlenden Tagungsteilnehmer sieht?

Nicht nur die Szene des Parallel Computing, sondern auch die der ASAPs (Application-specific Array Processors), bzw. der Systolischen Rechenfelder, die der Custom Computing Machines und die des Hardware/Software Co-Design befassen sich mit “High Performance Computing“. Das Gebiet der ASAPs als wissenschaftliches Gebiet macht einen sehr ordentlichen Eindruck. Es hat in sich Mainstream-Charakter. Das von-Neumann-Paradigma kommt darin allerdings nicht vor (da vom Sequencer allenfalls nur noch der Taktgeber übriggeblieben ist). Vielleicht hat dieses geradezu klassisch aussehende Lehrgebäude gerade deshalb eine so gute Systematik.

Bei Hardware/Software Co-Design wird der höhere Durchsatz durch Software/Hardware-Migration er-reicht. Gegenstand der Migration sind ist meist besonders oft durchlaufene und deshalb zum Engpaß gewordene Schleifen aus der Software, die in einen mächtigen Hardware-Operator als Akzelerator umgesetzt werden. Bei Hardware/Software Co-Design, insbesondere beim Entwurf von Embedded Systems angewandt, ist der sogenannte Hardware/Software Trade-off das wichtigste Ziel. Dies heißt: soviel Durchsatz wie nötig, und dies so billig wie möglich. Die Vertreter des Hardware/Software Co-Design beklagen das Fehlen eines allgemeinen Modelles [17], sodaß dieses Gebiet durch die Fülle der Architekturen zerrissen wird. Das Gebiet der FPGA-basierten Custom-Computing Machines ist eine Schwester des Hardware/Software Co-Design, denn auch hier wird Akzelerator-Hardware an einen von-Neumann-Host angehängt. Ein feinerer Unterschied besteht darin, daß erstere Disziplin sich mehr auf die rekonfigurierbare Hardware-Plattform des Akzelerators konzentriert, während bei letzterer das Streben nach automatisierter Partitionierung mehr im Vordergrund steht. Aber ein brauchbares Paradigma fehlt beiden.

Figure 8. Prozedural datengetriebener Rechner



6. Ein neues Computing-Paradigma beseitigt den Bremsklotz

Das von-Neumann-Paradigma eignet sich für keine der beiden Disziplinen so recht. Die wichtigste Ursache dafür ist die feste Kopplung zwischen ALU und “instruction sequencer“ (Befehls-Abwickler) beim Host. Nicht nur wegen des kompakten Befehlskodes benötigt man jeweils einen neuen Befehls-Abwickler, sobald die ALU verändert wird. Auch das Anhängen rekonfigurierbarer Akzelerator-Hardware an den Host kommt einer Änderung der ALU gleich. Deshalb verbindet man solche Akzeleratoren über den Hintereingang mit dem Host: der Akzelerator wird über zweckentfremdete Adressen des Hauptspeichers angesprochen (Figure 2.). Der Aufruf einer Akzelerator-Funktion erfolgt also im Modus “transport-triggered“ ([49] - [54]) anstatt per “instruction fetch“. Eine solche Anbindung ist eher Flickwerk, denn die Systematik eines allgemeinen Modelles.

Wenn ein rekonfigurierbarer Akzelerator verwendet wird, etwa mit FPGAs, so kann dieser jederzeit neu strukturprogrammiert werden. Dabei entsteht jedesmal ein anderer Zusatz zur ALU, und somit eigentlich jeweils eine neue ALU. Gibt es hier keinen Ausweg? Ist zu vermeiden, daß bei einer auch nur teilweisen Rekonfiguration der ALU die Prozessor-Architektur vollständig auseinanderfällt? Wie wäre es, wenn man auf den Befehls-Abwickler verzichten würde, da er doch soviele Probleme macht? Aber ohne Sequencer hat man keinen Prozessor mehr. Ohne Sequencer geht es also nicht. Ein Ausweg ist der "Data Sequencer" (Daten-Abwickler) [55] [56]. Ein solcher ist nur lose mit der ALU gekoppelt über nur wenige Entscheidungs-Bits (Figure 8.). Wegen des Fortfalls des klassischen "instruction fetch" bleibt nur noch eine Operator-Auslösung im Modus "transport-triggered". Bei einem Akzelerator macht dies Sinn, da man für eine bestimmte Anwendung immer wieder den gleichen Verbundoperator aufruft. Der "instruction fetch" ist also aus der Laufzeit in die Kompilationszeit des Akzelerators vorverlegt worden.

Figure 9. Gegenüberstellung der Computing-Paradigmen

Computer (von-Neumann- Maschine)	Xputer (d-Paradigma)	Datenfluß-Maschine
prozedurale Sequenzierung (deterministisch)		Arbiter-Sequenzierung (indeterministisch)
Kontrollfluß-getrieben	Daten-getrieben	

Durch Austausch des Befehls-Abwicklers gegen einen Daten-Abwickler erhalten wir einen Nicht-von Neumann-Prozessor, der nicht Kontrollfluß-getrieben ist, sondern Datenfluß-getrieben [59]. Ein solcher datenprozeduraler Prozessor (d-Prozessor, auch Xputer[1] genannt) arbeitet voll deterministisch und ist deshalb nicht zu verwechseln mit einem sogenannten Datenflußrechner [57], dessen Ausführungsreihenfolge nicht vorhersagbar ist, da sie durch einen Arbiter bestimmt wird (vgl. Figure 9.). Anstelle von Programm-Sprüngen und Programm-Schleifen führt ein d-Prozessor Datensprünge und Datenschleifen aus, d. h. Sprungziele sind nicht Befehle, sondern Datenobjekte [58]. Wie das von-Neumann-Paradigma die Universalität der von-Neumann-Prozessoren garantiert, so gibt es auch ein d-Paradigma, das die Universalität von d-Prozessoren sichert ([59] - [62]).

Ein d-Prozessor kann zwar auch im Stand-alone-Betrieb arbeiten. Er kann jedoch aus kommerzieller Sicht kein genereller Ersatz des von-Neumann-Prozessors sein, allein schon deswegen, weil die im Handel erhältliche System- und Anwendungs-Software darauf nicht läuft. Doch bietet sich der d-Prozessor als Koprozessor für die Anwendung als flexibler oder gar universeller Akzelerator zu einem von-Neumann-Host an ([63] - [65]). Mehrere verschiedene Mechanismen, die hier nur am Rande gestreift werden, geben dem d-Prozessor für viele Algorithmen aus Signalverarbeitung, Bildverarbeitung, Multimedia-Anwendungen und anderen Anwendungsgebieten ein enormes Akzelerations-Potential zu günstigen Hardwarekosten ([37] - [39]). Zu diesen Akzelerations-Mechanismen gehören Parallelität auf Datenpfad-Ebene sowie die Vermeidung bestimmer Arten von Overhead oder dessen Laufzeit-zu-Kompilationszeit-Migration. So werden beispielsweise viele Kommunikationspfade zur Ladezeit geschaltet statt während der Laufzeit. Die Wichtigkeit dieser Migration wird durch die meist unbewältigte Kommunikations-Umschalt-Explosion "klassischer" massiv paralleler Rechnersysteme unterstrichen. Auch immer wieder gemessene Speedup-Faktoren (im Extremfall mehr als drei Größenordnungen) zeigen, daß d-Prozessoren eine vielversprechende universelle Plattform sind ([12], [38], [55], [60] - [68]).

Doch ein Paradigmenwechsel ist mit Umlernen und auch sonst sehr viel Mühe verbunden. Lohnt sich dies überhaupt? Einige Vorzüge des d-Paradigmas seien genannt. Wichtig ist: der Datenabwickler eines d-Prozessors ist universell, während der Befehlsabwickler des von-Neumann-Prozessors ALU-spezi-

1. nicht von Neumann: nicht zu verwechseln mit dem Transputer, der ein von-Neumann-Prozessor ist

fisch ist. Bei Änderung der ALU braucht der von-Neumann-Prozessor einen neuen Sequencer. Ein d-Prozessor kann jedoch für eine große Mannigfaltigkeit von ALUs und anderen Datenpfad-Systemen, wie beispielsweise sogar ein ganzes Netzwerk aus vielen ALUs (beispielsweise [47], [48]) immer wieder den gleichen Sequencer verwenden. Das d-Paradigma ist also ideal für die Anwendung rekonfigurierbarer Datenpfade, sogar auch für die Anbindung der Strukturprogrammierung an einen von-Neumann-Host.

7. Zusammenfassung

Soft gewordene Hardware bringt die Parallelität der Raum-Dimension, das Computing by the Yard [1] in die Aufmerksamkeit der bisher meist nur von-Neumann-orientierten, also überwiegend prozedural denkenden High-Performance- und Parallel-Computing-Szenen. Dies ist ein Meilenstein. Dies zeigt einen möglichen Weg zur dichotomischen Computing Science (Computing über der Zeit und im Raum). Die Systolic-Array-Szene zeigt Abbildungen, die beide Welten miteinander verbinden. Doch das noch fast monopolhaft dominante von-Neumann-Paradigma ist ein Paradigma des Computing über der Zeit, jedoch nicht des Computing im Raum. Schon zeigt die Konferenzen-Inflation des High-Performance-Computing die Massenbewegung eines generellen Aufbruchs an — weg vom von-Neumann-Paradigma? Eine neue Religion oder nur eine neue Sekte? Doch noch ist die Marschrichtung nicht bekannt. Aber es wachsen die Zweifel am Sinn krampfhaften Festhaltens am von-Neumann-Paradigma als vermeintlich allein seligmachend.

Literaturhinweise

[1] A. Agarwal: Hot Machines; Proc. Int'l Conf on High Performance Computing; Dec. 27-30, 1995, New Delhi, India

[2] A. Postula, D. Abramson, P. Logothethis: Synthesis for Prototyping of Application-specific Processors; Proc. 3rd Asia Pacific Conf. on Hardware Description Languages (APCHDL'96), Bangalore. India, Jan. 1996

[3] R. Hartenstein, J. Becker, R. Kress: Application Specific Design Methodologies: General Model vs. Tinker Toy Approach; (auf diesem Workshop)

[4] A. Postula, D. Abramson, P. Logothethis: Synthesis for Prototyping of Application Specific Processors; Proc. 3rd Asia Pacific Conference on Hardware Description Languages (APCHDL´96), Bangalore, India, Jan. 1996

[5] Ing-Jer Huang, A. Despain: Synthesis of Application Specific Instruction Sets; IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 14, No. 6, June 1995

[6] H. Akaboshi, H. Yasuura: COACH: A Computer Aided Design Tool for Computer Architects; IEICE Trans. Fundamentals, Vol. E76-A, No. 10, Oct. 1993

[7] J. Sato, M. Imai, T. Hakata, A. Y. Alomary, N. Hikichi: An Integrated Design Environment for Application Specific Integrated Processor; Proc. IEEE Int'l Conf. on Computer Design: ICCD 1991, pp. 414-417, Oct. 1991

[8] D. Buell, K. Pocek: IEEE Workshop on FPGAs for Custom Computing Machines (FCCM'93) Napa, CA, April 1993

[9] D. Buell, K. Pocek: IEEE Workshop on FPGAs for Custom Computing Machines (FCCM'94), Napa, CA, April 1994

[10] P. Athanas, K. Pocek: IEEE Workshop on FPGAs for Custom Computing Machines (FCCM'95), Napa, CA, April 1995

[11] J. Arnold, K. Pocek: IEEE Workshop on FPGAs for Custom Computing Machines (FCCM'96), Napa, CA, April 1996

[12] R. Hartenstein et al.: Custom Computing Machines (invited opening keynote); DMM'95 - Int'l Symp. on Design Methodologies in Microelectronics, Smolenice Castle, Slovakia, September 1995

[13] R. Hartenstein: Custom Computing Machines; aktuelles Schlagwort; GI Informatik-Spektrum 18: p. 228-229, Springer-Verlag, Juni, 1995)

[14] N. N.: Brigham Young University Reconfigurable Logic Lab Bibliography; e-mail: wirthlin@fpga.ee.byu.edu, 1995

[15] S. A. Guccione: List of FPGA-based Computing Machines; guccione@ccwf.cc.utexas.edu, last updated: June 2, 1995

[16] K. Buchenrieder: Hardware/Software Co-Design in der Industrie; IT Press (i. Vorb. f. 1996)

[17] R. Gupta: Hardware/Software Co-Design; Proc. 9th Int'l Conf. on VLSI Design, Bangalore. India, Jan. 3-6, 1996

[18] R. Gupta: Co-Synthesis of Hardware and Software for Digital Embedded Systems; Kluwer, 1995

[19] R. Hartenstein: Hardware/Software Co-Design; aktuelles Schlagwort; GI Informatik-Spektrum 18: p. 286-287, Springer-Verlag, Oktober, 1995)
[20] Proc. 1st Int'l Workshop on Hardware/Software Co-Design CODES/CASHE '92, Estes Park, Colorado, USA, 1992
[21] Proc. 2nd Int'l Workshop on Hardware/Software Co-Design CODES/CASHE '93, Innsbruck, Austria, 1993
[22] Proc. 3rd Int'l Workshop on Hardware/Software Co-Design CODES/CASHE '94, Grenoble, USA, Sept. 1994
[23] Proc. 4th Int'l Workshop on Hardware/Software Co-Design CODES/CASHE '96, Pittsburgh, USA, March 1996B. K. Fawcett: FPGAs as Configurable Computing Elements; Int'l Worksh. on Reconfigurable Architectures, @ ISPS'95 - 9th Int'l Parallel Processing Symposium, Santa Barbara, 24. - 29. April 1995
[24] R. Gupta: Cosynthesis of Hardware and Software for Digital Embedded Systems; Kluwer 1995
[25] R. Gupta, G. de Micheli: Hardware/Software Co-Synthesis for Digital Systems; IEEE D&T of Computers, Sept. 1993
[26] K. Buchenrieder: Hardware/Software Co-Design; IT Press, Chicago 1995
[27] H. Grünbacher, R. Hartenstein (Editors.): Field-Programmable Gate Arrays: Architectures and Tools for Rapid Prototyping; Second International Workshop on Field-Programmable Logic and Applications, Vienna, Austria, Aug./Sept. 1992; Lecture Notes in Computer Science 705, Springer-Verlag, 1992
[28] R. Hartenstein, M. Servít (Editors.): Field-Programmable Logic: Architectures, Synthesis and Applications; Fourth International Workshop on Field-Programmable Logic and Applications, Prague, Czech Republic, Sept. 1994; Lecture Notes in Computer Science 849, Springer-Verlag, 1994
[29] W. Moore, W. Luk (Editors.): Field-Programmable Logic and Applications; Fifth International Workshop, Oxford, United Kingdom, Aug./Sept. 1995; Lecture Notes in Computer Science 975, Springer-Verlag, 1995
[30] M. Glesner, R. Hartenstein (Editors.): Field-Programmable Logic and Applications; Sixth International Workshop, Darmstadt, Germany, Sept. 1996; Lecture Notes in Computer Science, Springer-Verlag, 1996
[31] N. N.: WILDFIRE Custom Configurable Computer WAC4010/16; Document # 11502-0000, Rev. C, Annapolis Micro Systems, Inc., April 1995
[32] P. Chan: A Field-Programmable Prototyping Board: XC4000 BORG User's Guide; UCSRC-CRL-94-18, April 1994
[33] S. Casselman, J. Schewel, M. Thornburg: H.O.T. (Hardware Object Technology) Programming Tutorial; Release 1, Virtual Computer Corporation, January 1995
[34] H. Chow, S. Casselman, H. Alunuweiri: Implementation of a Parallel VLSI Linear Convolution Architecture Using the EVC1; in: [10]
[35] N.N.: EVC-1 Info 1.1; Virtual Computer Corporation, 1994
[36] A. Koch, U. Golze (Technical University Braunschweig, Germany): A Universal Co-Processor for Workstations; in: W. R. Moore, W. Luk (eds.): More FPGAs; Abbington EE&CS Books, Oxford, UK 1993 (selection from Proc. Int'l Symposium on field-programmable Logic and Applications, Oxford, UK, Sept. 1993)
[37] J. Becker, R. Hartenstein, R. Kress, H. Reinig: High-Performance Computing Using a Reconfigurable Accelerator; Proc. Workshop on High Performance Computing, Montreal, Canada, July 1995
[38] R. Hartenstein, J. Becker, R. Kress, H. Reinig: High-Performance Computing Using a Reconfigurable Accelerator; CPE Journal, Special Issue of Concurrency: Practice and Experience, John Wiley & Sons Ltd., 1996 (eingeladener Nachdruck von [37])
[39] R. Hartenstein, J. Becker, R. Kress: An Embedded Accelerator for Real Time Image Processing; 8th EUROMICRO Workshop on Real Time Systems, L'Aquila, Italy, June 1996
[40] S. Guccione, M. Gonzales: Classification and Performance of Reconfigurable Architectures; FPL´95 - Int'l Symposium on field-programmable Logic and Applications, Oxford, UK, 29-31 August 1995
[41] B. K. Fawcett: FPGAs as Configurable Computing Elements; Int'l Workshop on Reconfigurable Architectures @ IPPS´95 - 9th Int'l Parallel Processing Symposium, Santa Barbara, CA, 24-29 April 1995
[42] A. Koch, U. Golze (Technical University Braunschweig, Germany): A Universal Co-Processor for Workstations; in: W. R. Moore, W. Luk (eds.): More FPGAs; Abbington EE&CS Books, Oxford, UK 1993 (selection from Proc. Int'l Symposium on field-programmable Logic and Applications, Oxford, UK, Sept. 1993)
[43] R. Gupta, G. De Micheli: Hardware-Software Cosynthesis for Digital Systems; IEEE Design & Test, Sept. 1993
[44] S. Y. Kung: VLSI Array Processors; Prentice-Hall, 1988
[45] N. Petkov: Systolische Algorithmen und Arrays; Akademie-Verlag, Berlin 1989
[46] P. Treleaven, M. Pacheco, M. Vellasco: VLSI Architectures for Neural Networks, IEEE Micro, Vol. 9, Nr. 6
[47] R. Hartenstein, R. Kress: A Datapath Synthesis System for the Reconfigurable Datapath Architecture; Asia and South Pacific Design Aut. Conf., ASP-DAC'95, Nippon Convention Center, Makuhari, Chiba, Japan, Aug. 29 - Sept. 1, 1995
[48] R. Kress: A Fast Reconfigurable ALU for Xputers; Ph.D. Thesis, University of Kaiserslautern, 1996
[49] G. J. Lipovski: On a Stack Organization for Microcomputers; in [50]
[50] R. Hartenstein, R. Zaks: Microarchitecture of Computer Systems; North Holland 1975
[51] H. Corporaal, H. Mulder: MOVE: A framework for high-performance processor design; Proc. Supercomputing ´91, Albuquerque, IEEE Computer Society Press, November 1991
[52] H. Corporaal, P, van der Arend: MOVE32INT, a Sea of Gates realization of a high performance Transport Triggered Architecture; Microprocessing and Microprogramming vol. 38, pp. 53-60, North-Holland, 1993
[53] H. Corporaal: Evaluating Transport Triggered Architectures for scalar applications; Transport Triggered Architecture; Microprocessing and Microprogramming vol. 38, pp. 45-52, North-Holland, 1993

[54] H. Corporaal: Transport Triggered Architectures; Ph. D. thesis, Tech. University of Delft, Holland, 1995
[55] R. Hartenstein, A. Hirschbiel, M. Weber: MoM - a partly custom-designed architecture compared to standard hardware; Proc. COMP EURO, Hamburg, Germany, 1989; IEEE Press 1989
[56] A. Hirschbiel: A Novel Processor Architecture Based on Auto Data Sequencing and Low Level Parallelism; Ph.D. Thesis, University of Kaiserslautern, 1991
[57] Theo Ungerer: Datenflußrechner; Teubner, 1993
[58] A. Ast, J. Becker, R. W. Hartenstein, R. Kress, H. Reinig, K. Schmidt: Data-procedural Languages for FPL-based Machines; 4th Int'l Workshop on Field Programmable Logic and Appl.ications, FPL'94, Prague, Sept. 7-10, 1994, Lecture Notes in Computer Science, Springer, 1994
[59] R. Hartenstein, A. Hirschbiel, K. Schmidt, M. Weber: A novel Paradigm of Parallel Computation and its Use to implement Simple High-Performance Hardware; Future Generation Computing Systems 7 (1991/92), (eingeladener Nachdruck von [60])
[60] R. Hartenstein, A. Hirschbiel, M.Weber: A Novel Paradigm of Parallel Computation and its Use to Implement Simple High Performance Hardware; InfoJapan'90- International Conference memorating the 30th Anniversary of the Computer Society of Japan, Tokyo, Japan, 1990
[61] R. W. Hartenstein, M. Riedmüller, K. Schmidt, M. Weber: A Novel Asic Design Approach Based on a New Machine Paradigm; Special Issue of IEEE Journal of Solid State Circuits on ESSCIRC´90, July 1991
[62] R. Hartenstein, M. Riedmüller, K. Schmidt, M. Weber: A Novel ASIC Design Approach based on a New Machine Paradigm; IEEE Journal of Solid State Circuits, July 1991 (eingeladener Nachdruck von [61])
[63] R. Hartenstein, J. Becker, R. Kress: Two-Level Hardware/Software Partitioning Using CoDe-X; Int'l IEEE Symp. on Engineering of Computer Based Systems (ECBS), Friedrichshafen, Germany, March 1996
[64] K. Schmidt: A Program Partitioning, Restructuring, and Mapping Method for Xputers; Ph.D. Thesis, Kaiserslautern 1994
[65] Reiner W. Hartenstein, Jürgen Becker, Michael Herz, Rainer Kress, Ulrich Nageldinger: A Parallelizing Programming Environment for Embedded Xputer-based Accelerators; High Performance Computing Symposium '96, Ottawa, Canada, June 1996
[66] R. W. Hartenstein, R. Kress, H. Reinig: A Dynamically Reconfigurable Wavefront Array Architecture for Evaluation of Expressions; Proc. Int'l Conference on Application-Specific Array Processors, ASAP'94, San Francisco, IEEE Computer Society Press, Los Alamitos, CA, Aug. 1994
[67] Reiner W. Hartenstein, Jürgen Becker, Rainer Kress, Helmut Reinig, Karin Schmidt: A Reconfigurable Machine for Applications in Image and Video Compression; Conference on Compression Technologies and Standards for Image and Video Compression, Amsterdam, The Netherlands, March 1995
[68] Jürgen Becker, Reiner W. Hartenstein, Rainer Kress, Helmut Reinig: A Reconfigurable Parallel Architecture to Accelerate Scientific Computation; Proc. Int'l Conf. on High Performance Computing, New Delhi, India, December, 1995