Configware für Supercomputing: Aufbruch zum Personal Supercomputer

Reiner Hartenstein, TU Kaiserslautern
http://hartenstein.de

Zusammenfassung. Insbesondere in der Praxis eingebetteter Systeme haben sich rekonfigurierbare Plattformen schon vor Jahren durchgesetzt: nicht nur als programmierbare Akzeleratoren, sondern neuerdings auch zur Stromersparnis. Bei Supercomputing oder „high performance computing“ (HPC) und deren Anwendungsgebieten wurden diese Möglichkeiten jedoch weitgehend ignoriert. Seit etwa zwei Jahren ist aber ein noch zögerlicher Strategie-Wechsel zum Doppel-Paradigma zu beobachten, nämlich durch Eingliederung programmierbarer Akzeleratoren in das Grundmodell der System-Architekturen - die Anti-Maschine als Gegenstück zum von-Neumann-Paradigma. Diese Dualität der Maschinenparadigmen ist geeignet, das Lehrgebäude der Informatik und ihrer Anwendungen in seinen Fundamenten zu erschüttern. Das hier vorliegende Papier gibt einen kurzen Überblick, der in etwa auch für Newcomer lesbar sein soll, und veranschaulicht die Grundlagen dieser neuen Entwicklung.

1. Der Durchbruch des FPGA

Die Medien popularisieren den FPGA (field-programmable gate array), mit 6 Mrd.US-Dollars am schnellsten wachsender Sektor des Halbleitermarktes. Komplexe Projekte sind auf FPGAs aus dem Katalog realisierbar - ohne sehr teures eigenes Anwendungsspezifisches Silizium. Die Zahl der „design starts“ soll im Jahre 2006 um 13.4% steigen und bis 2010 von 80.000 auf 115.000 wachsen [Dataquest]. Eindrucksvoll ist die Zahl der Treffer bei Google (Bild 1, Oktober 2005 [1]) auf ***FPGA*** (4.840.000, fast 5 Millionen) und „***Reconfigurable Computing***" (258.000 mal). FPGAs gehen in alle denkbaren Anwendungsgebiete (Bild 2).

Google (Oktober 2005)	
Schlüsselwort	Trefferzahl
FPGA	4.680.000
Reconfigurable Computing	250.000
Anti Machine	13.300
Configware	4.920

Bild 1. Fast 5 Mio Treffer.

Verdoppelung alle 10 Monate. Die Menge für eingebettete Systeme implementierter Software verdoppelt sich alle 10 Monate [2] (die Gordon Moore Kurve für Mikroelektronik nur alle 18 Monate [4]). Eingebettete Systeme sind ohne FPGAs heute undenkbar. Dies ist der Motor des kommerziellen Durchbruchs der FPGAs und bestimmt die von unseren Absolventen verlangten Qualifikationen: eine Herausforderung für Kurrikulums-Entwicklung (s. Kapitel 6.).

Morphware. Ein FPGA ist ein Feld rekonfigurierbarer Logik-Blöcke (***CLB*** für „configurable logic block“), in ein Geflecht rekonfigurierbarer „Verdrahtung“ eingebettet [8]. Statt „weiche Hardware“ (weich ≠ hart) bevorzuge ich für derartiges die von DARPA-geförderten Projekten [6] eingeführte Bezeichnung ***„Morphware“*** [7], um den Gegensatz zur fest verdrahteten „Hardware“ zu verdeutlichen.

FPGA and ...	Google
... embedded	2.160.000
... wireless	1.420.000
... automotive	629.000
low power	463.000
... medical	385.000
... physics	231.000
... chemical	172.000
... defense	172.000
... bio	93.400
... weather	89.200
... chemistry	80.800
... molecular	77.000
supercomputing	48.400
n body problem	28.200
... oil and gas	14.900

Bild 2. FPGAs überall (Okt.05).

Hidden RAM (hRAM). Der ***Configware***-Kode (Rekonfigurations-Kode [9]: s. Bild 3) wird in einem im Hintergrund des FPGA verteilten RAM-Speicher (***hRAM*** für „hidden RAM“) abgelegt, wobei mit neuem Configware-Kode jederzeit die Funktionalität des FPGA geändert werden kann, auch beim Kunden. Vergleichbar mit dem Hochfahren eines Computers muß auch beim Einschalten von Morphware erst einmal der Configware-Kode geladen werden.

Grobkörnige Morphware. Da ein CLB nur ein Bit breit ist, nennen wir FPGAs „**feinkörnig** rekonfigurierbar“. Jedoch grobkörnig rekonfigurierbare Morphware-Architekturen, rDPAs (reconfigurable Data Path Arrays) [10] [11], haben Pfadbreiten wie z. B. 32 Bit [12] [13]. Z. B. der KressArray [14] [15] [16], eine Verallgemeinerung des systolischen Array [17] [18], wird durch einen Architektur „Design Space Xplorer“ unterstützt [19] [20].

Terminologie. Das Morphware-Grundparadigma ist **nicht** Befehlsstrom-basiert, weshalb hier unbedingt der Terminus „Configware“ verwendet werden soll statt „Software“ (Bild 3). Letzterer muß eindeutig auf Befehlsstrom-basierte Systeme beschränkt bleiben (begründet in Kapitel 4. und durch Bild 5). Vielmehr stehen Datenströme im Vordergrund. Dieses Papier verwendet den Terminus Datenstrom („data stream“) so, wie er um 1980 oder früher für systolische Arrays definiert wurde (Bild 4) [17] [18]. Transistoren sehen alle ähnlich aus. Welcher Schaltvorgang eines Transistors dient der Rekonfiguration und welcher der Operation? Eine grobe Richtschnur kann auch der Schalt-Zeitpunkt seiner Anwendung sein wie es in Bild 6 gezeigt wird.

Plattform		Programm-Quelle	Maschinen-Paradigma
Hardware-Logik		(kein Programm)	(keines)
Morphware	rekonfigurierbare Logik	***Configware***[9]	
	Reconfigurable Computing	***Configware + Flowware***	Anti-Maschine (AM)
Hardware-Prozessor	Datenstrom-basiert	***Flowware***[30]	
	Befehlsstrom-basiert	***Software***	von Neumann (vN)
embedded systems + reconfigurable supercomputing, etc.		***Software + Configware + Flowware***	dual paradigm: vN + AM

Bild 3. Zur Terminologie im heutigen Zeitalter des Doppel-Paradigma (vergleiche mit Bild 5).

2. Morphware-Architekturen und deren Effizienz

MOPS per Milliwatt. Die Flächen-Effizienz und der Energie-bezogene Durchsatz (MOPS per Milliwatt) sind auch von allgemeinem wirtschaftlichen Interesse. So geht etwa ein Viertel des gesamten Stromverbrauchs von Amsterdam in Internet-Server-Farmen. In New York City nehmen Internet-Server-Farmen eine Gebäude-Nutzfläche von insgesamt einem Viertel eines Quadratkilometers ein. Die Flächen-Effizienz bestimmt natürlich auch, ob ein Supercomputing-Zentrum eine große Halle benötigt und eine jährliche Stromrechnung bis zu Millionenhöhe hat. Gegenüber fest verdrahteter Hardware steht der Standard-Mikroprozessor um Größenordnungen zurück (Bild 7). Grobkörnige Morphware erreicht fast die Effizienz fest verdrahteter Hardware. FPGAs sind immerhin um etwa 2 Größenordnungen besser als der Standard-Mikroprozessor. Bild 12 zeigt einige publizierte Faktoren. Dies wird in Kapitel 5 diskutiert.

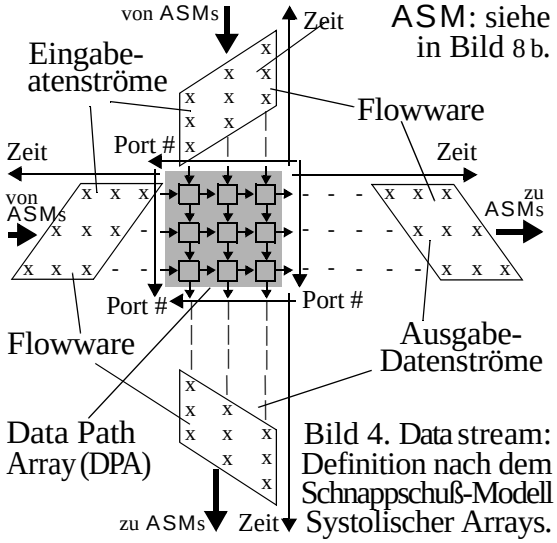


Bild 4. Data stream: Definition nach dem Schnappschuß-Modell Systolischer Arrays.

Flächeneffizienz [22] (Bild 7). Die physische Integrationsdichte (Transistoren pro Chip) von FPGAs liegt etwa 2 Größenordnungen (Faktor 100) hinter der Gordon-Moore-Kurve wegen des Flächenbedarfs der Verdrahtungs-Ressourcen. Die logische Integrationsdichte, die unmittelbar der Anwendung dient, liegt um 4 Größenordnungen hinter der Gordon-Moore-Kurve, da etwa 99 von 100 Transistoren der Rekonfigurierbarkeit dienen (Rekonfigurierbarkeits-Overhead).

Relativierung dieser Bilanz. Dem stehen die vielfachen Facetten der Ineffizienz Befehlsstrom-basierter Implementierungen gegenüber: sehr schlechte logische Integrationsdichte auch bei Mikroprozessoren, geringe Parallelität im Mikroprozessor, massiv höhere Zahl von Speicherzugriffen. So werden bei Software-zu-Configware-Migrationen in der Praxis durch erheblich höhere Parallelität Akzelerationsfaktoren von bis zu mehreren Größenordnungen erzielt (Bild 9).

Grobkörnige Morphware. Um Größenordnungen besser ist die Effizienz bei grobkörniger Morphware (Bild 7). Um Größenordnungen geringerer Bedarf an hRAM und rekonfigurierbaren Verdrahtungs-Ressourcen ergibt sich dadurch, daß im ***rDPA (reconfigurable Data Path Array)*** nur wenige großflächige ***rDPUs (reconfigurable Data Path Units)*** [12] [14] [15] die vielen kleinen CLBs ersetzen, die sehr flächeneffizient im Stil von Vollkundenschaltungen (full-custom) realisiert werden können.

3. Die Geschichte der Grundmodelle hin bis zur Morphware

Der Schwanz wedelt mit dem Hund. Heute kommen Mikroprozessoren wegen Speicherzyklen-hungriger Befehlsstrom-getriebener sequentieller Arbeitsweise ohne Akzeleratoren nicht aus [22]: der Schwanz wedelt mit dem Hund. Das von-Neumann-Maschinenparadigma (vN) im ***Mainframe-Zeitalter*** (Bild 10 a) wurde abgelöst durch eine Symbiose aus Wirtsrechner (host) und Akzelerator (Bild 10 b). Die Zweigleisigkeit des Maschinen-Paradigmas im ***PC-Zeitalter*** aus Befehlsstrom-getriebenem vN-Paradigma und Datenstrom-getriebenen Akzelerator wurde von Anwendungs-Programmierern oft nicht gesehen: die Kluft „the Hardware / Software chasm".

Wandel der Paradigmen. Eingebettete Systeme sind ohne Akzelerator-FPGAs im heutigen ***Morphware-Zeitalter*** parktisch undenkbar. Ein drittes wiederum zweigleisiges Grundmodell dominiert jetzt (Bild 10 d): die Symbiose von Mikroprozessor (Bild 10 a) und ebenfalls programmierbarem Morphware-Akzelerator (Bild 10 c). Dies ist das Basismodell des heutigen Morphware-Zeitalters. Bild 10 e zeigt das Modell der Zukunft mit ***Ko-Compiler*** (Bild 15), der von einer gemeinsamen Quelle automatisch in Software und Configware partitioniert [23]. Solche Co-Compiler sind im akademischen Bereich schon implementiert worden [23] [24]: leicht zu realisierbar, da die Grundlagen schon seit Jahrzehnten bekannt sind [26] (siehe auch Absatz „Persönlicher Supercomputer (PS)„ in Kapitel 5).

Strategische Bedeutung von Morphware. Der allgemeine Durchdringungsgrad der FPGAs in unsere Welt wird demonstriert durch die Google-Trefferzahl (Bild 2) [1] auf ***FPGA*** kombiniert mit Anwendungsgebieten wie ***embedded*** (2.160.000 mal), ***wireless*** (1.420.000 mal), oder ***automotive*** (629.000 mal) etc. Ca. 90% aller Software wird heute für von FPGAs dominierte eingebettete Systeme implementiert [2], wobei stets Hardware / Configware / Software Partitionierungs-Probleme gelöst werden müssen. Das Quasi-Monopol des von-Neumann-Paradigma in unseren Kurrikula verhindert die Qualifizierung unserer Absolventen. Auf einem Gipfeltreffen von US-Gouverneuren hat Bill Gates diese Situation der Informatik-Lehre drastisch kritisiert.

4. Das Doppel-Paradigma als zeitgemäßes Grundmodell

Mit Morphware liegt ein zweites RAM-basiertes Grundparadigma vor: die ***Anti-Maschine*** [27]. Jedoch haben wir bei der Kompilation statt den Ablauf von Befehlsströmen die Plazierung und Verdrahtung (placement and routing) zu realisieren und daran anschließend die Organisation der Datenströme (Bild 5). Morphware bzw. die Anti-Maschine kennt ***keinen „instruction fetch" zur Laufzeit*** (Bild 6), was wegen der Speicherzugriffs-Lücke (Bild 13) erheblich zum Akzelerationsfaktor beitragen kann. Das dem Betrieb vorausgehende Herunterladen von Rekonfigurationskode in das hRAM ist die Vorwegnahme viel komplexerer „instruction fetches".

Die Dualität der Maschinenparadigmen erschüttert das Lehrgebäude der Informatik in den Fundamenten. Doch Paradigmenwechsel sind unbequem und gefährden oft das Revier etablierter Platzhirsche, was Widerstand erzeugt. Wegen abwegiger Terminologie ist diese Dualität der Paradigmen oft schwer zu sehen. Als Gegenmittel gegen die Babylonische Sprach-Verwirrung sei noch einmal an die Gliederung der Plattformen und ihrer Programm-Quellen erinnert, quasi als ein terminologischer Grundriß (Bild 3). Die Anti-Maschine ist das Datenstrom-basierte Gegenstück zur Befehlsstrom-basierten von-Neumann-Maschine. Die Anti-Maschine hat keinen Programmzähler (Bild 8 b). Deren „Prozessor" ist **keine CPU** (Bild 8 a), sondern nur eine ***DPU (Data Path Unit***: ohne Programm-Zähler, Bild 8 b).

Die Technologie der Anti-Maschine hat einen oder meist mehrere ***Datenzähler (AG)*** als Bestandteil von Adress-Generatoren in Datenspeichern ***ASM*** (***Auto-Sequencing Memory***, s. Bild 8 b). ASMs erzeugen oder empfangen ***Flowware***-programmierte Datenströme [30] (Bild 4). Ein AG ist eine Verallgemeinerung der DMA-Schaltung (Direct Memory Access) für Befehls-Zyklen vermeidende Block-Transfer [28] [29]. ***Generische Adress-Generatoren*** (***GAG***) sind weit entwickelt [29] [31] [32], basierend auf verteilten Speicher-Architekturen [34].

Plattform	Energie-bezogener relativer Durchsatz: MOPS/mW	relative logische Flächen-Effizienz: Transistoren/Chip
Gordon-Moore-Kurve	-	4
Standard Mikroprozessor	0	1
Signalprozessor (DSP)	1	2
feinkörnige Morphware (FPGA)	2	0
grobkörnige Morphware (rDPA[a])	4	4
Akzelerator-Hardware	4	4

a). full-custom design

Bild 7. Relativer Effizienz-Vergleich der Plattformen, angegeben in Größenordnungen.

Die fest verdrahtete Variante der Anti-Maschine. Von der Technologie her unterscheiden wir zwei Arten von Anti-Maschinen (Bild 3): die zur Morphware zählende programmierbare (rekonfigurierbare) Anti-Maschine, die zwei Arten von Programmierungs-Quellen benötigt (Bild 5 c/d): ***Configware*** zur Struktur-Programmierung, und ***Flowware***, eine Art Datenfahrplan, der passend zur vorausgegangenen Konfiguration die notwendigen Datenströme programmiert. Nach dem gleichen Grundmodell der Anti-Maschine können auch fest verdrahtete Strukturen verwendet werden (beispielsweise [35]), wobei die „Konfiguration“ gleichsam vor der Fabrikation eingefroren und in Hardware gegossen wird (vgl. Bild 6). Da nach der Fabrikation keine Konfiguration mehr möglich ist, wird nur eine Art Programmierungs-Quelle benötigt, nämlich nur noch ***Flowwar****e*.

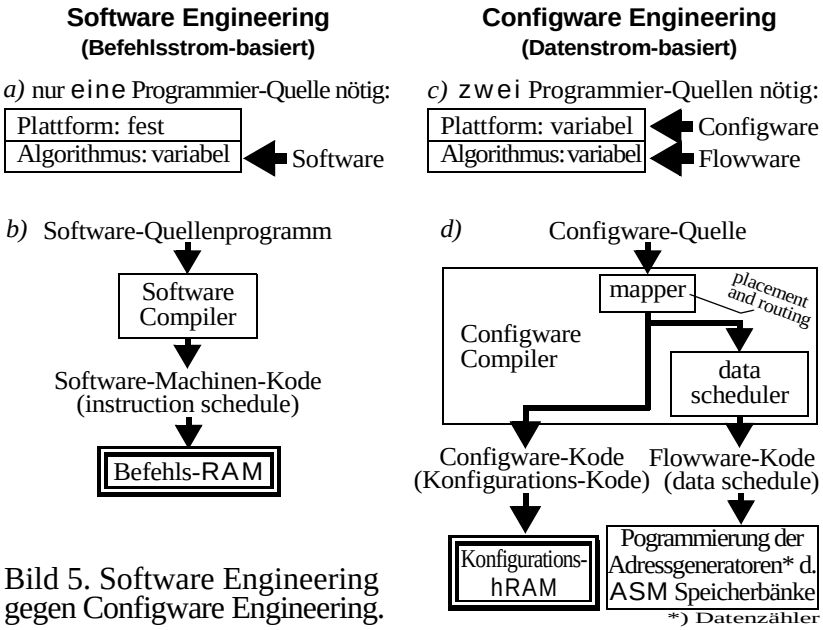
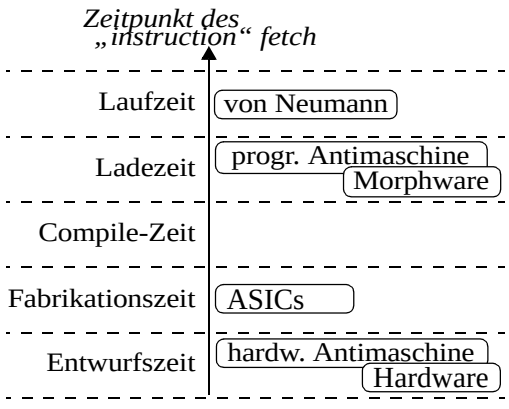


Bild 5. Software Engineering gegen Configware Engineering.

Software Engineering gegen Configware Engineering. Das Doppel-Paradigmen-Modell wird im folgenden durch Gegenüberstellung veranschaulicht. Beim klassischen Software-Prozessor ist nur der Algorithmus variabel, jedoch die Plattform fest verdrahtet, weshalb nur ein einziger Typ von Programmquelle benötigt wird, nämlich Software (Bild 5a), aus welcher Software-Befehlskode generiert und im ***Befehls-RAM*** abgelegt wird, letztlich ein Zeitfahrplan für den ***Software-Prozessor*** (Bild 5b). Beim Anti-Maschinen-Paradigma, ist hingegen nicht nur der Algorithmus, sondern auch die Plattform programmierbar, weshalb zwei Arten von Programmierungs-Quellen benötigt werden: ***Configware*** und ***Flowware*** (Bild 5c). Configware [9] dient der strukturellen Programmierung der Plattform durch ***Placement und Routing*** (Plazierung und „Verdrahtung“) oder einen anderen geeignete Abbildung (beispielsweise durch simulated Annealing [14] [15] [19] [20] [21]) über den „***Mapper***“ (Bild 5d). Flowware [36] dient der Programmierung der von der programmierten Plattform benötigten ***Datenströme*** durch den „***data scheduler***“ gemäß Bild 5d.

Flowware-Sprachen sind einfach zu implementieren [36] [37]. Interessant ist die Gegenüberstellung [8] mit Software-Programmier-Sprachen. Sprach-Elemente für Kontroll-Befehle wie Sprünge und Schleifen können einfach aus Software-Sprachen übernommen werden, jedoch für Datenadressen anstelle von Befehlsadressen. Flowware-Sprachen sind mächtiger und erlauben mehrere Datenzähler und damit parallele Schleifen. Flowware-Sprachen sind einfacher, denn es gibt keine Befehle zur Datenmanipulation, wegen Vorkonfiguration (zur Laufzeit kein "instruction fetch", sondern nur "data fetch" (siehe Bild 6).



ild 6. Zeitpunkt des “Instruction“ Fetch..

Dynamisch rekonfigurierbare Architekturen und deren Umgebungen veranschaulichen den eigenen Charakter des ***Configware Engineering***, denn sie können in schneller Folge zwischen Laufzeit und Konfigurationszeit wechseln. Noch komplexer können Teile von partiell rekonfigurierbaren FPGAs sich in der Laufzeit befinden, während andere in der Konfigurationsphase stehen, wodurch ein FPGA u. a. sich selbst rekonfigurieren kann. Es können mehrere Makros im gleichen FPGA resident sein. Einige können ausgelagert oder geladen werden während gleichzeitig andere Markos sich in der Laufzeit befinden. ***Configware-Betriebssysteme*** dienen hierbei dem Management [38] [39]. Auf dieser Basis kann sogar ***Fehlertoleranz*** durch Selbstreparatur realisiert werden [40] [41]. Dynamische Rekonfigurierbarkeit ist verwirrend für Anfänger und sollte deshalb vielleicht erst im Hauptstudium behandelt werden.

5. Die Einführung von Configware beim Supercomputing

Das Vordringen rekonfigurierbarer Plattformen wie FPGAs in der Supercomputing-Szene wird demonstriert durch die Google-Antwort [1] auf *FPGA*, jeweils in Kombination mit ***"high performance computing"*** (64.400 mal), oder ***supercomputing*** (48,400 mal). Die starke FPGA-Durchdringung (pervasiveness of FPGAs) vieler typischer Supercomputing-Anwendungsgebiete wird sichtbar durch die von Google angegebene Zahl der Treffer auf ***FPGA*** in Kombination mit beispielsweise den Stichworten ***medi-***

cal (385.000 mal), ***physics*** (231.000 mal), ***defense*** (172.000 mal), und ***weather*** (82,900 mal) etc. (Bild 2.). Bei einer Supercomputing-Anwendung im Gebiet „***Öl and Gas***“ erbrachte die Migration auf FPGAs einen Durchsatzverbesserung um den Faktor von 17 [42] [43]. Als Nebenwirkung ergab sich eine enorme Ersparnis an elektrischer Energie, denn mit FPGAs kann man Energie sparen [44]. Der Geophysiker Herb Riley (R. Associates Inc., Houston, Texas) berichtet, daß dies bei 7 US-Cents je kWh mehr als 10,000 US-Dollars in der Stromrechnung per Jahr einspart - per 19-Zoll-Einschub mit 64-Prozessoren. Dies bedeutet eine jährliche Ersparnis von rund einer halben Million US-Dollars mit 50 solchen Einschüben [42]. Doch nun möchte ich mit den traditionellen Problemen befassen und anschließend mit deren Überwindung, wie dies durch Reconfigurable Computing versprochen wird.

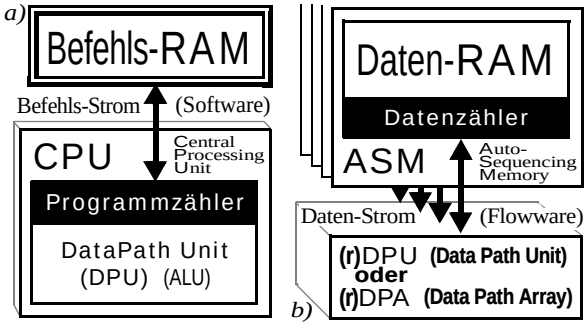


Bild 8. Grund-Paradigmen: a) von Neumann Maschine, b) Anti-Maschine (rekonfigurierbar oder fest).

Kommunikationsaufwand bei klassischem Supercomputing. Der Kommunikationsaufwand steigt überproportional wegen typischer Engpässe, vor Allem bestimmt durch die langsame Speicher-Zugriffszeit [45] (Bild 13). Bus-Systeme und andere Schalt-Systeme tendieren zu hohem Speicherzyklen-hungrigem Verwaltungsaufwand [46]. Der Transport von Daten während der Laufzeit ist ein vorherrschendes Problem, wohingegen an kostengünstigen CPU-Ressourcen heute ein Überfluß herrscht. Typisch ist die Denkweise in kommunizierenden nebenläufigen Prozessen und das Gewicht von MPI (Message Passing Interface). Je größer die Anzahl der CPUs ist, desto schneller kann der Kommunikationsaufwand überproportional steigen. Die Skalierbarkeit des Durchsatzes einer Anwendung verfehlt oft die Spitzenwerte (peak performance), welche die Plattform anzubieten scheint [47]. Amdahls Gesetz erklärt nur eine unter mehreren Ursachen von Ineffizienz [48]. ***Antimaschinen haben jedoch keinen von-Neumann-Engpaß***.

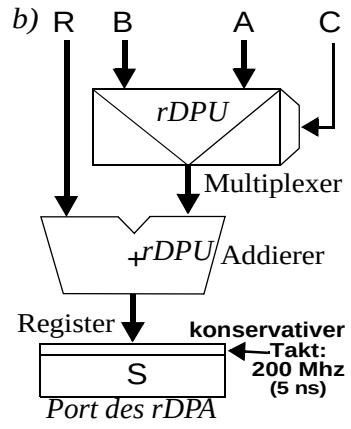
MPI (Message Passing Interface). Für Leser von außerhalb der Supercomputing-Szenen sei dies wie folgt skizziert. MPI ist Standard zur Realisierung des Distributed Memory Programmiermodells durch den Austausch von Nachrichten (insbesondere Daten) zwischen mehreren Prozessen auf Shared Memory Architekturen. Alle Funktionen stehen in einer FORTRAN- oder einer C-Version in Form einer Bibliothek zur Verfügung mit internen Datenstrukturen, die dem Nutzer weitgehend verborgen bleiben. MPI basiert auf dem ***Communicating Sequential Processes*** Modell (***CSP*** [49] [50]) nach Tony Hoare [51] für quasi-parallele Berechnungen auf verteilten, heterogenen, lose-gekoppelten Computersystemen. Die Programmiersprache Occam ist eine praktische Implementierung von CSP. JCSP ist die Verbindung von CSP und Occam-Konzepten in einer Java-API. Parallele MPI-Programme sind somit auf PC-Clustern (Nachrichtenaustausch z.B. über TCP), und auf dedizierten Parallelrechnern ausführbar (mit Kommunikation beispeilsweise über den gemeinsamen Hauptspeicher). MPI ist den wenigsten FPGA-Anwendern bekannt.

Der Hocker wird bewegt und nicht das Klavier. Der Datentransport gleicht bei der Befehlsstrom-basierten Denke des klassischen Supercomputing oft dem Bewegen des Klaviers oder des Konzert-Flügels hin zum Hocker des Klavierspielers. Jedoch die Datenstrom-dominierte Denke des Reconfigurable Computing verfolgt den umgekehrten Ansatz: der Hocker wird bewegt und nicht das Klavier. Deshalb ist hier MPI unbekannt und der nebenläufige Prozeßbegriff ein Rand-Aspekt. Nicht in erster Linie die Daten werden mit Kommunikations-Ressourcen umhertransportiert, sondern die Lokalität der Operation wird an die richtige Stelle in den Datenströmen plaziert, die sich direkt ergeben als per ASM (lesend) generierte Ströme der Operanden und (schreibend) der Ergebnisdaten (Bild 8 b und 4). Die Kommunikationswege zwischen den DPUs als Pipe-Netzwerk - ohne Verwaltungsaufwand und meist ohne Zwischenspeicherung[1] (Bild 9 b) - werden zur Compile-Zeit optimiert, **vor** der Laufzeit geschaltet und bleiben während der Laufzeit unverändert. Da es meist deutlich weniger Prozessoren bzw. rDPUs hat als Datenobjekte gibt, ist also viel weniger zu bewegen, und dies auch noch zur weniger kostbaren Compile-Zeit. Außerdem werden zur Laufzeit keine Befehle umhertransportiert: ein weiterer Akzelerations-Aspekt.

Die Implementierung von Flowware. In einem reinrassigen Reconfigurable Computing System besteht die einzige Form der Kommunikation über Speicher nämlich in den Datenströmen (Bild 4) zwischen den verteilten ASM-Speichern und den DPAs bzw. rDPAs. In den (Daten-)Speichern sind dabei nur noch Anfangs-Operanden und Endergebnisse der Anwendung abzuspeichern, jedoch keine Zwischenergebnisse noch sonstige Signale. Die Anzahl der Bänke des verteilten Speichers [34] wird zwecks maximaler Parallelität möglichst der Anzahl der Ports der Morphware angepaßt. Die Datenströme werden durch Compilation von Flowware implementiert (Bild 5) über die Programmierung der Adreßgeneratoren in den ASMs des verteilten Speichers (Bild 8 b). ***Speicherbelegungs-Schemata*** sind nicht auf Vektoren oder Matrizen beschränkt. Durch **generische Adreßgeneratoren (GAG)** [29] [31] [32] [33] kann auf einen rei-

S := R + (if C then A else B endif);
a)

		Speicherzyklen	Nanosekunden
if C then read A endif	Befehl holen	1	100
	Befehl dekodieren		
	Operand holen	1	100
	Befehl ausführen		
if not C then read B endif	Befehl holen	1	100
	Befehl dekodieren		
addieren	Befehl holen	1	100
	Befehl dekodieren		
	Befehl ausführen		
Speichern	Resultat speichern	1	100
insgesamt		**5**	**500**



Akzelerationsfaktor: 100

Bild 9. Veranschaulichung der Akzeleration: a) Befehlsstrom-basierte Ausf. auf hypothetischem einfachen Prozessor (C = 1), b) Datenstrom-basierte Ausführung in einem rDPA (rekonfigurierbarer Datenpfad-Array).

1). falls in diesem Beispiel Register S der Ausgangs-Port des rDPA ist

chen Vorrat generischer Speicherbelegungs-Schemata eine große Vielfalt analytischer ***Transformationen*** angewandt werden zur Abbildung von Speicher-Schemata in die Generierung der benötigten Datenfolgen - meist ohne Verbrauch von Speicherzyklen für die Adreßrechnung. Ein ***zweidimensionaler Adreßraum*** eröffnet für diesen Kontext ungeahnte Möglichkeiten effizienter und leicht GAG-transformierbarer Speicherbelegungs-Schemata [29] [34] Ein 2-dimensionaler Adreßraum hat darüber hinaus noch weitere Vorteile, wie beispielsweise bei der Visualisierung von Prozessen und der Speicherbelegungs-Schemata.

Skalierbarkeit. Auch beim Reconfigurable Computing gibt es Skalierbarkeitsprobleme, die allerdings nicht zur Laufzeit auftreten, sondern bei der Compilation. Hier ist es allerdings ein Problem über der Fläche oder über dem Raum, der für die Verdrahtungs-Ressourcen benötigt wird. Bild 14 a veranschaulicht dies an einem FPGA-Beispiel. Hier helfen u. a. Optimierungs-Methoden nach dem Beispiel des strukturierten VLSI-Entwurf der Mead-&-Conway-Ära [52] mit dem Ziel, miteinander kommunizierende Einheiten direkt aneinanderzustekken, sodaß keine externen Verdrahtungs-Ressourcen benötigt werden (Bild 14 b). Die Skalierbarkeit hängt hier stark von der Anwendung ab. So haben beispielsweise übliche Signalverarbeitungs-Algorithmen keine Skalierungsprobleme. Aber der Viterbi-Algorithmus für die fehlerkorrigierende Dekodierung hat hier einen progressiv steigenden Bedarf an Verdratungs-Ressourcen. Für Interkonnekt-intensive grobkörnige Morphware-Architekturen wie beispielsweise aus der KressArray-Familie lassen sich die erforderlichen Pipe-Netzwerke effizient optimieren [15] [19] [20].

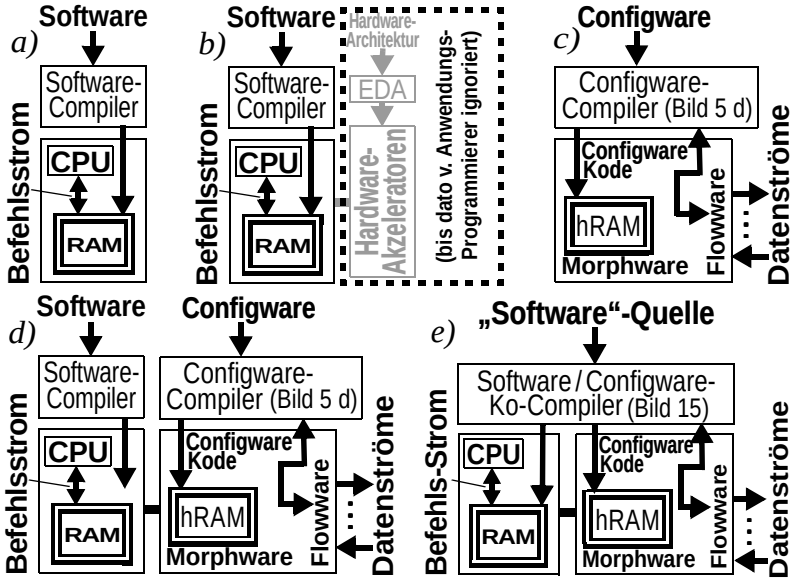


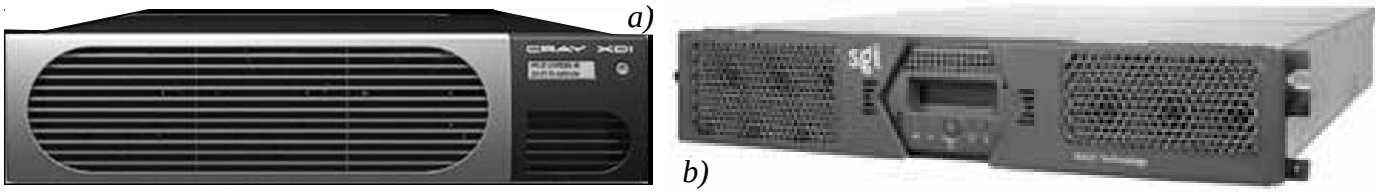
Bild 10. Die Geschichte der Maschinen-Paradigmen:
a) Mainframe-Zeitalter, b) PC-Zeitalter, c) Anti-Maschine,
d) Doppel-Paradigma des Morphware-Zeitalter (jetzt), e) künftig.

Reconfigurable Computing bei Cray und sgi. Reconfigurable Computing wurde inzwischen kommerzialisiert (Bild 11). Cray Inc. in Seattle bietet einen Einschub CRAY XD1 an (Bild 11 a), mit 6 FPGAs vom Typ Xilinx Virtex-4 (XC4VLX160-10 oder XC4VSX55-10) [53]. Auch sgi (Silicon Graphics) bietet einen Einschub an (Bild 11 b) mit einer sogenannten RASC-Technologie [54] ("Rekonfigurierbares Anwendungs-Spezifisches Computing"), die missionskritische Applikationen 100-fach beschleunigen soll [56]. SGI beabsichtigt in Kooperation der Fa.Nallatech [42] [43] [58] rund um RASC ein komplettes „Ökosystem" zu schaffen, das die Annahme der Technologie fördern soll. Die RASC-Technologie von sgi zielt auf Kernanwendungen in mehreren Märkten wie beispielsweise [54]: Bioinformatik, Chemieinformatik, Medizin, Medien, Öl & Gas, in fast allen Anwendungen, die mit FFT-Algorithmen (Fast Fourier Transform) arbeiten, Verteidigung und Nachrichtendienste (Echtzeit-Datenanalysen mit Routinen für Signalverarbeitung, Erkennen von Objektgrenzen, Mustererkennung). Das System ist bei beiden Anbietern aber immer noch nebenläufig und die rekonfigurierbaren Anteile sind in einer unteren Ebene durch eine dem Anwender relativ undurchsichtige spezielle Bibliothek eingebracht worden. Ein Co-Compiler fehlt.

Etikettenschwindel. Schon frühzeitig wurde im deutschsprachigen Raum die Signalverarbeitung als Nachrichtentechnik bezeichnet, obwohl das Signal seinen Nachrichten-Inhalt garnicht kennt. Die Attraktivität der Informatik führte schon früh zur „Entstehung" vieler Bindestrich-Informatiken und beispielsweise zur Umbenennung der Fachbereiche für Elektrotechnik in solche für Elektrotechnik und Informationstechnik. Ein ähnliches Phänomen erleben wir jetzt mit dem Begriff „Reconfigurable Computing". So wird beispielsweise die Unterscheidung zwischen Parallel Computing und Reconfigurable Computing verwischt durch Projekte mit dem Etikett "reconfigurable", die aber in Wirklichkeit klassisches nebenläufiges Parallelrechnen auf einen einzigen Chip übertragen. Ein Parallelrechner ist aber ein Computer, in dem mehrere Prozesse gleichzeitig auf mehreren CPUs ablaufen [66]. Dies ist nicht Reconfigurable Computing.

Reconfigurable Computing statt Cache-Speicher. Bei klassischen Mikroprozessoren nimmt die CPU nur einen winzigen Teil der Chip-Fläche ein. Der weitaus größte Flächenanteil wird für schnelle Cache-Speicher verschwendet - zwecks Umgehung der Speicherzugriffs-Lücke (Bild 13). Guten Nutzen bringt eine Anwendung, wenn die CPU viele, sehr oft wiederholte Befehls-Schleifen von schnellen Cache-Speichern lesen kann wobei jeder Befehl nur ein einziges mal aus dem langsamen Hauptspeicher in den schnellen cache geholt und von dort oft gelesen wird. Der Akzelerationsfaktor hält sich dabei in Grenzen und hängt stark von der Art der Anwendung ab. Erfahrungswerte hierüber sind kaum veröffentlicht worden. Bei der Antimaschine hingegen sind caches weder nützlich, da bei Daten-Schleifen sich die Werte im Allgemeinen nicht wiederholen, noch nötig, da ja mächtigere andere Mechanismen zur Minimierung von Hauptspeicherzyklen zur Verfügung stehen. Wesentlich bessere Akzelerationsfaktoren sind von einem rDPA als Ko-Prozessor zu erwarten, anstelle dann nicht benötigter Cache-Speicher.

Persönlicher Supercomputer (PS). PCs mit einem leistungsfähigen universellen programmierbaren Akzelerator an Bord werden über die Nutzung eines Co-Compilers (Bild 15) der Schlüssel zum persönlichen Supercomputer (PS). Vorstufen zum persönlichen Su-



ild 11. Supercomputer mit FPGAs als Akzeleratoren: a) Cray XD1: Akzelerations-Faktor von mehr als 100 bei Genome-Sequencing, b) sgi (Silicon Graphics) RASC™: nennt Akzeleration „um Größenordnungen".

Plattform	Jahr	Anwendung	Faktor
Mikroprozessor nach FPGA [59]	2004	diverse MAC-basierte Anwendungen	x 100
Signalprozessor nach FPGA [60]	2004	diverse DSP Anwendungen	x 7 bis x 46
PACT Xtreme 4*4 Array [61]	2003	16-Punkt FIR-Filter	x 16 MOPS/mW
MoM Antimaschine mit DPLA[a] gegen VAX 11/750 [22] [62] [63]	1989	Design Rule Check (gleicher Algorithmus)	15.000
		Design Rule Check (unterschiedliche Algorithmen)	2.300
gegen SPARC 10/51 [64]	1995	Ising model 128 lattice, 1000 iterations	53,8

a). DPLA: programmierbarer PLA, entworfen an der TU Kaiserslautern, Chip gefertigt 1983 im Rahmen des E.I.S.-Projekt [65]

Bild 12. Software-zu-Configware Migration: einige Beispiele für Akzelerationsfaktoren.

percomputer wurden vor Jahren veröffentlicht, beispielsweise im Zusammenhang mit der n-Körper-Simulation [67]. Astrophysiker hatten sich beklagt, daß der teuerste verfügbare Supercomputer nur die Simulation von Sternenhaufen bis etwa zur Größe 100 ermöglicht. GRAPE, nur eine Erweiterungskarte für den PC erlaubte Größen bis etwa 1000 [68], aber nicht die Änderung des Algorithmus. Daran arbeitet beispielsweise Prof. Rainer Spurzem vom mehr als 300 Jahre alten Astronomischen Recheninstitut (seit 1945 an der Universität Heidelberg) in Zusammenarbeit mit Prof. Reinhard Männer (Universität Mannheim): eine rekonfigurierbare Akzelerator-Karte AHA-GRAPE [69] für den PC, die weit mehr können soll als nur n-Körper-Simulation. Teils um Größenordnungen mächtiger ist aber die Anwendung grobkörniger Morphware (Bild 12).

Die Technologie für den PS ist verfügbar. Grobkörnig rekonfigurierbare Arrays, RDPAs, mit Hundert oder mehr DPUs auf einem einzigen Mikrochip gibt es schon heute [12], da Cache-Speicher hier sinnlos sind. Auch Software / Configware Co-Compiler (Bild 15) für den PS sind zwar m. W. kommerziell noch nicht verfügbar, wurden aber im akademischen Bereich schon implementiert [23] [24] [70] [71], z. B. in der Programmiersprache C geschriebene Quellen akzeptierend für den grobkörnig rekonfigurierbaren KressArray [14] [15] [16] [73]. Die Implementierung solcher Co-Compiler ist unproblematisch und die Methodologie ist teilweise schon einige Jahrzehnte alt [74]. Der Weg zum Personal Supercomputer (PS) ist also nicht weit. Es fehlt nur noch ein Investor für Ko-Compiler [75].

6. Kurrikulums-Empfehlungen

Konsortia. Neben der sehr bekannten amerikanischen ACM/AIS/IEEE Computing Curricula Kommission [77] gibt es noch weitere Konsortia, die sich mit Kurrikulums-Innovation befassen. ARTIST2 [78] [79] [80] ist ein Europäisches Konsortium über Ausbildung in Eingebetteten Systemen. ARTES ist eine Schwedische strategische Initiative in Real-Time-Systemen [82]. Career Space ist ein Konsortium eines Dutzend großer Firmen der Informations- and Kommunikations-Technologie (ICT) [83]: zur Überbrückung der gegenwärtigen Qualifikations-Lücke, die Europas Prosperität bedrohe. ICT Absolventen benötigen demnach solide Grundlagen und Fähigkeiten von beiden Seiten: Ingenieurwissenschaften und Informatik, mit Betonung einer breiten Perspektive.

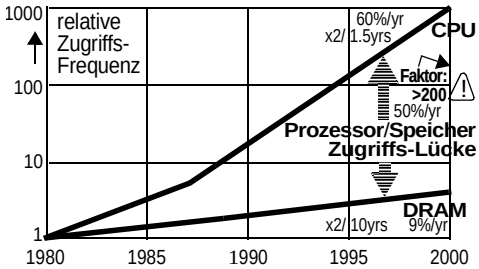


Bild 13. Speicher-Zugriffs-Lücke [45].

Reconfigurable Computing Education. Obwohl die Zielgebiete dieser Konsortien die Haupt-Anwendungsgebiete von FPGAs sind, kommen letztere in diesen Empfehlungen so gut wie garnicht vor. Google ergibt beispielsweise bei „real-time computing“ (ARTES-Konsortium) nur 256.000 Treffer, wohingegen „FPGA“ fast 5 Millionen mal gefunden wird (Bild 1). Deshalb erstaunt die Einäugigkeit dieser Empfehlungen durch Ausklammern des Reconfigurable Computing oder der FPGAs. Eine as Gebiet des Supercomputing mit einschließende Initiative gegen diese Einäugigkeit ist eine neue Workshop-Reihe: The 1st International Workshop on Reconfigurable Computing Education (RE education 2006) [84] am 1. März 2006 in Karlsruhe, am Rande des IEEE Computer Society Annual Symposium on VLSI (ISVLSI) am 2. - 3. März 2005 in Karlsruhe, das erstmalig in Europa statt in den USA stattfindet [85]. Die Gründung einer GI / ITG Fachgruppe „Reconfigurable Computing“ würde ich gern unterstützen.

Akademische Lehre am Arbeitsmarkt vorbei. Rasch wachsende Komplexität und Verbreitung der „RC-based multi-paradigm devices“ führt zu einer Produktivitäts-Krise. Andererseits ist Reconfigurable Computing eine effiziente Methode zur Lösung der galoppierenden Mikroelektronik-Entwurfs-Krise. Jedes der vielen Anwendungsgebiete von FPGAs hat nur einen eingeschränkten Blickwinkel auf „Computing“ und sieht dies mehr als Trickkiste an, denn als notwendige Grundlagen-Wissenschaft, sodaß es sehr schwierig ist, die kulturelle Kluft und Lücken der Praxis zu überbrücken. In Fragmentierung sind sehr viele verschiedene Aktivisten und Abteilungen involviert.

Algorithmische Cleverness fehlt. Heute werden Experten mit unterschiedlichem Background und divergierenden Blickwinkeln

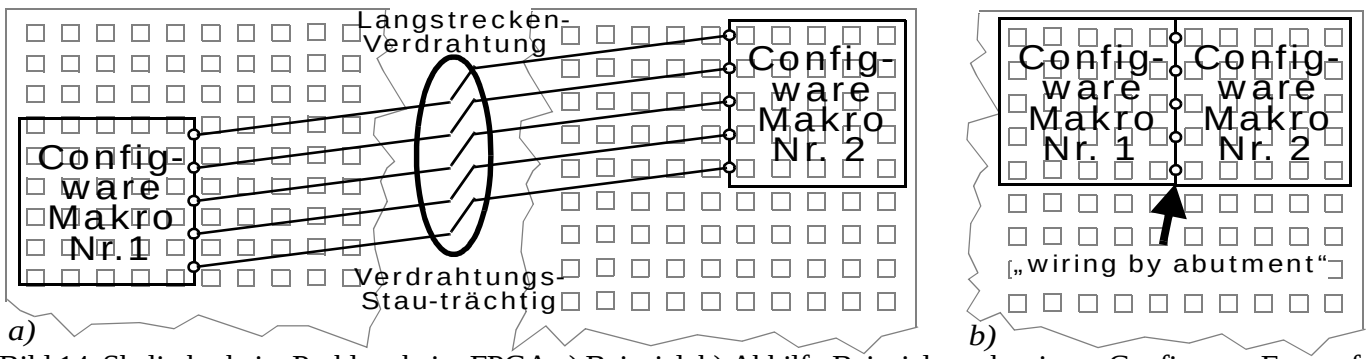


Bild 14. Skalierbarkeits-Problem beim FPGA a) Beispiel, b) Abhilfe-Beispiel: strukturierter Configware-Entwurf.

benötigt, nicht nur für Test und Verifikation moderner Entwürfe, wenn überhaupt möglich, was sehr teuer ist und die Produkteinführung erheblich verzögert. Während die wirtschaftliche Notwendigkeit von Reconfigurable Computing and FPGAs erkannt wurde, verfehlte der akademische Bereich die Ausbildung eines genügend großen Anteils hochqualifizierter kompetenter Systemenwickler und Configware-Programmierer. Die Programmierung von Morphware erfordert viel mehr Informatik-Fähigkeiten. Es mangelt oft an der algorithmischen „Cleverness“ zur Software-zu-Configware-Migration. Eine neue Taxonomie der Algorithmen und Architekturen wird benötigt, die den Algorithmus-Begriff von der Zeit in den Raum erweitert.

Schädliches Monopol des von-Neumann-Paradigma. Von unseren Kurrikula wird immer noch die wachsende Configware-Industrie ignoriert. Moderne FPGAs aus dem Lagerregal haben alle zwei Paradigmen zusammen mit mehreren Speicherbänken auf dem gleichen Mikrochip. Um den Zusammenprall der Kulturen zu meistern, brauchen wir interdisziplinäre Kurrikula, die all die verschiedenen Backgrounds auf eine systematische Weise miteinander verknüpfen, ja, sogar verschmelzen. Wir brauchen innovative Kurse und Praktika zwecks Integration von Reconfigurable Computing in fortschrittliche Kurrikula. Wir müssen dem Trend gegensteuern, der eine Spezialisierung als Hauptziel der Ausbildung sieht. Wir brauchen interdisziplinäre, methodisch mit der Informatik verflochtene Kurrikula zur Vereinheitlichung der Grundlagen. Ebenso brauchen wir neue Informatik-Kurrikula, die endlich das Monopol des von-Neumann-Modelles aufgeben und vom ersten Semester an das Doppel-Paradigma als Grundlage anerkennen. Es ist evident geworden, daß viele Grundlagenprobleme quer über viele Anwendungsgebiete gehen. Wir brauchen endlich eine interdisziplinäre Vorgehensweise in die Richtung zum Hardware-Configware-Software Co-Design, nicht nur für die Praxis, sondern auch für Kurrikula der Elektrotechnik, Informatik, und Informations-Technologie.

7. Schlußfolgerungen

FPGAs und grobkörnig rekonfigurierbare Plattformen des Reconfigurable Computing bieten neben erheblicher Stromersparnis gegenüber dem Befehlsstrom-basierten von-Neumann-Paradigma Akzelerationsfaktoren um mehrere Größenordnungen. Deren Programmierung ist ebenfalls RAM-basiert, was in der Praxis zu einer Doppel-Paradigma-Methodologie führt durch die Entstehung von ***Configware Engineering als Gegenstück zum Software Engineering***. Das neue zweite Paradigma kennt die meisten der oft gravierenden Kommunikationsengpässe nebenläufiger Systeme nicht. Deshalb findet die schon ein Jahrzehnt bei eingebetteten Systemen vorherrschende Methodologie seit etwa 2 Jahren auch Eingang beim Supercomputing, wenngleich noch zögerlich durch die ***tiefe Kluft zwischen den Kulturen***. Das akademische Bildungswesen ist gefordert, insbesondere in Informatik und E-Technik mit neuen Kurrikula über eine integrierende Doppel-Paradigma-Denkweise diese Kluft und schwere Qualifikations-Mängel unserer Absolventen zu überwinden. Aus interdisziplinär muß intra-disziplinär werden.

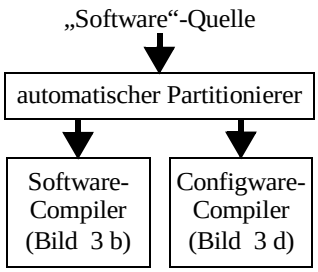


Bild 15. Software / Configware-Ko-Compiler

8. Literatur

[1] R. Hartenstein: The Pervasiveness of Reconfigurable Computing; http://hartenstein.de/pervasiveness.html
[2] F. Rammig (interview): Visions from the IT Engine Room; IFIP TC 10 - Computer Systems Technology, URL: [3]
[3] http://www.ifip.or.at/secretariat/tc_visions/tc10_visions.htm
[4] S. Hang: Embedded Systems – Der (verdeckte) Siegeszug einer Schlüsseltechnologie; Deutsche Bank Research, Januar 2001, URL: [5]
[5] http://xputers.informatik.uni-kl.de/VerdeckterSiegeszug.pdf
[6] http://www.morphware.org/
[7] http://morphware.net
[8] R. Hartenstein: Morphware and Configware; (invited chapter) in: A. Zomaya (editor): Handbook of Innovative Computing Paradigms; Springer Verlag, New York, 2006
[9] http://configware.org
[10] http://en.wikipedia.org/wiki/RDPA
[11] R. Hartenstein (invited embedded tutorial): A Decade of Reconfigurable Computing: A Visionary Retrospective; Design, Automation and Test in Europe, Conference & Exhibition (DATE 2001), 13.-16. März 2001, München
[12] http://pactcorp.com
[13] J. Becker, M. Vorbach: An Industrial/Academic Configurable System-on-Chip Project (CSoC): Coarse.grain XPP/Leon-based Architecture Integration; Design, Automation and Test in Europe, Conference & Exhibition (DATE 2003), 3.-7. März 2003, München
[14] http://kressarray.de
[15] R. Kress et al.: A Datapath Synthesis System (DPSS) for the Reconfigurable Datapath Architecture; ASP-DAC 1995
[16] http://en.wikipedia.org/wiki/KressArray
[17] N. Petkov: Systolic Parallel Processing; North-Holland; 1992
[18] H. T. Kung: Why Systolic Architectures? IEEE Computer 15(1): 37-46 (1982)
[19] U. Nageldinger et al.: Generation of Design Suggestions for Coarse-Grain Reconfigurable Architectures; International Conference on Field Programmable Logic and Applications (FPL), 28. - 30. August 2000, Villach, Österreich
[20] U. Nageldinger: Coarse-grained Reconfigurable Architectures Design Space Exploration; Dissertation, 2001, URL: [21]
[21] http://xputers.informatik.uni-kl.de/papers/publications/NageldingerDiss.html
[22] R. Hartenstein (invited paper): The Microprocessor is no more General Purpose; Proc. IEEE International Symposium on Innovative Systems (ISIS), 9. - 11. Oktober 1997, Austin, Texas, USA
[23] J. Becker et al.: Parallelization in Co-Compilation for Configurable Accelerators; Asian-Pacific Design Automation Conference (ASP-DAC), 10. - 13. Februar 1998, Yokohama, Japan
[24] J. Becker: A Partitioning Compiler for Computers with Xputer-based Accelerators, Dissertation, TU Kaiserslautern 1997, URL: [25]
[25] http://xputers.informatik.uni-kl.de/papers/publications/BeckerDiss.pdf
[26] K. Schmidt: A Program Partitioning, Restructuring, and Mapping Method for Xputers, Dissertation, TU Kaiserslautern 1994
[27] http://antimachine.org
[28] http://de.wikipedia.org/wiki/Direct_Memory_Access

[29] M. Herz et al. (invited paper): Memory Organization for Data-Stream-based Reconfigurable Computing; 9th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 15.-18. September 2002, Dubrovnik, Kroatien
[30] http://flowware.net
[31] M. Herz, et al.: A Novel Sequencer Hardware for Application Specific Computing; Proc. ASAP'97
[32] M. Herz: High Performance Memory Communication Architectures for Coarse-grained Reconfigurable Computing Systems; Ph. D. dissertation, Kaiserslautern, 2001, URL: [33]
[33] http://xputers.informatik.uni-kl.de/papers/publications/HerzDiss.html
[34] F. Catthoor et al.: Data Access and Storage Management for Embedded Programmable Processors; Kluwer, 2002
[35] C. Chang et al: The Biggascale Emulation Engine (Bee); summer retreat 2001, UC Berkeley
[36] http://flowware.net
[37] A. Ast, et al.: Data-procedural Languages for FPL-based Machines; International Conference on Field Programmable Logic and Applications (FPL), 7. - 9. September 1994, Prag, Tschechien
[38] H. Simmler et al.: Multitasking on FPGA Coprocessors; International Conference on Field Programmable Logic and Applications (FPL), 28. - 30. August 2000, Villach, Österreich
[39] H. Walder, M. Platzner: Reconfigurable Hardware Operating Systems: From Design Concepts to Realizations; Proc. ERSA 2003
[40] P. Zipf: A Fault Tolerance Technique for Field-Programmable Logic Arrays; Dissertation, Univ. Siegen, 2002
[41] M. Abramovici, C, Stroud: Improved BIST-Based Diagnosis of FPGA Logic Blocks; Proc. IEEE Int'l Test Conf., Atlantic City, USA, Oct. 2000
[42] N. N.: R. Associates Joins Nallatech's Growing Channel Partner Program; Companies are Using FPGAs to Reduce Costs and Increase Performance in Seismic Processing for Oil and Gas Exploration; FPGA and Structured ASIC Journal BusinessWire, August 08, 2005
[43] http://www.fpgajournal.com/news_2005/08/20050808_03.htm
[44] V. George, J. Rabaey: Low-Energy FPGAs: Architecture and Design; Kluwer, 2001
[45] 87. D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R.Thomas, K. Yelick: A Case for Intelligent RAM; IEEE Micro, März / April 1997.
[46] G. Koch et al.: The Universal Bus Considered Harmful; Proc. 1st EUROMICRO Symposium on the Microarchitecture of Computing Systems; Juni 1975, Nizza, Frankreich
[47] J. Dongarra et al. (editors): The Sourcebook of Parallel Computing; Morgan Kaufmann 2002
[48] G. Amdahl: Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities; Proc. AFIPS 1967
[49] C. A. R. Hoare: Communicating Sequential Processes, Prentice-Hall, 1985 - URL: [50]
[50] http://www.usingcsp.com/cspbook.pdf
[51] http://de.wikipedia.org/wiki/Tony_Hoare
[52] R. Hartenstein: Fundamentals of Structured Hardware Design - A Design Language Approach at Register Level; North Holland, Amsterdam/New York 1977
[53] http://www.cray.com/products/xd1/
[54] N. N.: SGI ergänzt HPC um RASC - für ´Rekonfigurierbares Anwendungs-Spezifisches Computing'; München, 30.09.2005, - URL: [55]
[55] http://www.pressebox.de/pressemeldungen/silicon-graphics-gmbh/boxid-42342.html
[56] Dietmar Müller: SGI setzt auf rekonfigurierbares Rechnen; ZDNet, 30. September 2005 - URL: [57]
[57] http://www.zdnet.de/news/business/0,39023142,39137009,00.htm
[58] http://www.nallatech.com/
[59] W. Roelandts (Keynote-Adresse): FPGAs and the Era of Field Programmability; International Conference on Field Programmable Logic and Applications (FPL), 29. August - 1. September 2004, Antwerpen, Belgien,
[60] J. Rabaey: Reconfigurable Processing: The Solution to Low-Power Programmable DSP, Proc. ICASSP 1997
[61] V. Baumgarten, et al.: PACT XPP - A Self-Reconfigurable Data Processing Architecture; ERSA 2001
[62] http://xputers.informatik.uni-kl.de/faq-pages/fqa.html
[63] W. Nebel et al.: PISA, a CAD Package and Special Hardware for Pixel-oriented Layout Analysis; ICCAD 1984
[64] J. Becker et al.: High-Performance Computing Using a Reconfigurable Accelerator; CPE Journal, Special Issue of Concurrency: Practice and Experience, John Wiley & Sons Ltd., 1996
[65] http://xputers.informatik.uni-kl.de/staff/hartenstein/eishistory.html
[66] http://de.wikipedia.org/wiki/Mehrprozessorsystem
[67] G. Lienhart: Beschleunigung Hydrodynamischer N-Körper-Simulationen mit Rekonfigurierbaren Rechensystemen; Joint 33rd Speedup and 19th PARS Workshop; Basel, Switzerland, March 19 - 21, 2003
[68] T. Narumi, R. Susukita, T. Ebisuzaki, G. McNiven, B. Elmegreen: Molecular dynamics machine: Special-purpose computer for molecular dynamics simulations; Molecular Simulation, vol. 21, pp. 401-415, 1999
[69] R. Männer, R. Spurzem et al.: AHA-GRAPE: Adaptive Hydrodynamic Architecture - GRAvity PipE; International Conference on Field Programmable Logic and Applications (FPL), 30. August - 1. September 1999, Glasgow, Schottland,
[70] K. Schmidt et al.: Automatic Parallelism Exploitation for FPL-based Accelerators; Proc. HICSS 1998, Big Island, Hawaii, USA
[71] J. Becker et al.: A Partitioning Programming Environment for a Novel Parallel Architecture; Proc. IPDPS '96 - URL: [72]
[72] http://ipdps.cc.gatech.edu/1996/PAPERS/S13/HARTEN/HARTEN.PDF
[73] R. R. Kress et al.: An Embedded Accelerator for Real-Time Image Processing; 8th Euromicro Workshop on Real-Time Systems; Juni 1996, L'Aquila, Italien
[74] K. Kennedy, R. Allen: Optimizing Compilers for Modern Architectures: A Dependence-based Approach; Academic Press, 2002
[75] N. N.: Cray and Celoxica Make Reconfigurable Computing Easier to Program; Cray, Inc., 6. September 2005, URL: [76]
[76] http://investors.cray.com/phoenix.zhtml?c=98390&p=irol-newsArticle_print&ID=752700&highlight=
[77] N.N.: Computing Curricula 2004; Joint Task Force for Computing Curricula 2004, 22. November 2004, etc.
[78] N.N.: W2.All.Y1Guidelines for a Graduate Curriculum on Embedded Software and Systems; ARTIST Consortium 2003; URL: [79]
[79] http://www.artist-embedded.org/Education/Education.pdf
[80] P. Caspi et al.: Guidelines for a Graduate Curriculum on Embedded Software and Systems; Workshop on Embedded Systems Education (WESE 2005), September 22nd, 2005, Jersey City, New Jersey, USA
[81] http://www-verimag.imag.fr/~caspi/WESE/wese-cfp.html
[82] http://www.artes.uu.se/events/
[83] http://www.career-space.com/cdguide/serv6.htm
[84] http://helios.informatik.uni-kl.de/RCeducation/
[85] http://isvlsi06.itiv.uni-karlsruhe.de/