Mehr Effizienz durch Configware: auch für Supercomputing

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

Zusammenfassung. Insbesondere in der Praxis eingebetteter Systeme haben sich rekonfigurierbare Plattformen schon vor vielen Jahren durchgesetzt: nicht nur als beim Anwender programmierbare Akzeleratoren, sondern neuerdings auch zur Verminderung des Stromverbrauchs. Bei Supercomputing oder „high performance computing“ (HPC) und den entsprechenden Anwendungsgebieten wurden diese hochgradig potenten Möglichkeiten jedoch weitgehend ignoriert. Seit etwa zwei Jahren ist aber auch hier ein anfänglich 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

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.

Sogar in den Medien wird der FPGA (field-programmable gate array) popularisiert: "Quick Change Artists ... can rewire themselves to process different types of signals." schreibt Business Week [1] Mit einem Weltmarktvolumen von derzeit etwa 6 Mrd.US-Dollars bilden FPGAs den am schnellsten wachsenden Sektor des Halbleitermarktes. Der wichtigste Vorteil gegenüber fest verdrahteter Hardware besteht neben der Energiespar- und der Akzelerator-Funktion darin, daß man selbst komplexe Schaltungs-Entwurfs-Projekte auf FPGAs aus dem Katalog realisieren kann, ohne sehr teures eigenes Anwendungs-spezifisches Silizium zu benötigen. Nach neuerer Prognose soll die Anzahl der beginnenden FPGA-basierten Entwurfsprojekte (design starts) im Jahre 2006 um 13.4% steigen und bis 2010 von 80.000 auf 115.000 wachsen [Dataquest]. Ein eindrucksvoller Indikator für die allgemeine Bedeutung der Anwendung rekonfigurierbarer Plattformen ist die Zahl der Treffer bei der Suchmaschine Google (am 4. Oktober 2005 [2]) auf das Stichwort *FPGA* (4.840.000, also fast 5 Millionen) sowie auf die Textzeile „*Reconfigurable Computing"* (258.000 mal), siehe auch Bild 1 und Bild 2

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. FPGA überall (Oct.05).

Verdoppelung alle 10 Monate. Die Menge der für eingebettete Systeme implementierten Software verdoppelt sich alle 10 Monate [3] [5]. Die Dramatik dieses Wachstums wird deutlich durch einen Vergleich mit der Gordon Moore Kurve, die eine Verdoppelung der Integrationsdichte integrierter Schaltungen nur für alle 18 Monate zeigt. Da eingebettete Systeme ohne FPGAs heute undenkbar sind, ist dieses Wachstum der Motor des kommerziellen Durchbruchs der FPGAs. Gleichzeitig bestimmt dies die am Arbeitsmarkt von unseren jetzigen und künftigen Absolventen verlangten Qualifikationen: eine Herausforderung für Kurrikulums-Entwicklung (siehe Kapitel 6.).

Morphware. Was ist ein FPGA? Statt „weiche Hardware“ (paradox, da weich ≠ hart) bevorzuge ich die von DARPA-geförderten Parallelrechner-Projekten eingeführte Bezeichnung „Morphware“ [6] [7], um den Gegensatz zur fest verdrahteten „Hardware“ zu verdeutlichen. Die Funktion von FPGAs sei am Beispiel der alten Insel-Architektur veranschaulicht (Bild 3**)**. Ein Feld rekonfigurierbarer Logik-Blöcke (CLB steht für „configurable logic block“) ist in ein Geflecht rekonfigurierbarer „Verdrahtung“ eingebettet, bestehend aus „Drahtstückchen“, Schaltboxen und Verbindungsboxen (Bild 3 a). Durch Konfiguration kann jedem CLB eine eigene logische Funktion gewählt

werden. Bild Bild 3. d zeigt das Beispiel eines im Geflecht programmierten „Drahtes“. Verbindungsboxen (Bild Bild 3. e) dienen dem Anschluß an einen CLB, und Schaltboxen (Bild 3 b) dem Zusammenbau des „Drahtes“.

Hidden RAM (hRAM). Der Configware-Kode (Rekonfigurations-Kode; Terminologie [8]: s. Bild 4) wird in einem im Hintergrund des FPGA verteilten RAM-Speicher (hRAM für „hidden RAM“) abgelegt. Die in Bild Bild 3.b und e mit „FF“ bezeichneten Flipflops sind Teile dieses hRAM. Bild 3 c zeigt die Schaltbox-Konfiguration für 010010. Durch Herunterladen von vollständig oder partiell neuem Configware-Kode in den hRAM kann jederzeit die Funktionalität des FPGA geändert werden, auch beim Kunden. Vergleichbar mit dem Hochfahren eines Computers muß auch beim Einschalten von Morphware erst einmal der Configware-Kode geladen werden. Es gibt Bestrebungen hin zu FPGAs mit nicht-flüchtigem hRAM, beispielsweise durch flash-Speicher-Technologie, womit Morphware sofort nach dem Einschalten betriebsfertig wäre.

Grobkörnige Morphware. Da ein CLB einen nur etwa ein Bit breiten Datenpfad darstellt, bezeichnen wir FPGAs als „feinkörnig rekonfigurierbar“. Demgegenüber gibt es auch grobkörnig rekonfigurierbare Morphware-Architekturen: rDPAs (reconfigurable Data Path Arrays) [10] [11] [12] mit Pfadbreiten wie beispielsweise 32 Bit [13] [14]. Beispielsweise der KressArray [15] [16] [17], eine Verallgemeinerung des systolischen Array [18] [19] [20], wird durch einen Architektur „Design Space Xplorer“ unterstützt [21] [22] [23]. Relativ wenig bekannt sind neben dem Systolischen Array fest verdrahtete Varianten [24] von Antimaschinen-Architekturen[1], die zwar nach dem gleichen Paradigma zu modellieren sind, jedoch mangels Rekonfigurierbarkeit nach der Fabrikation eigentlich nicht Morphware genannt werden kann (Bild 4).

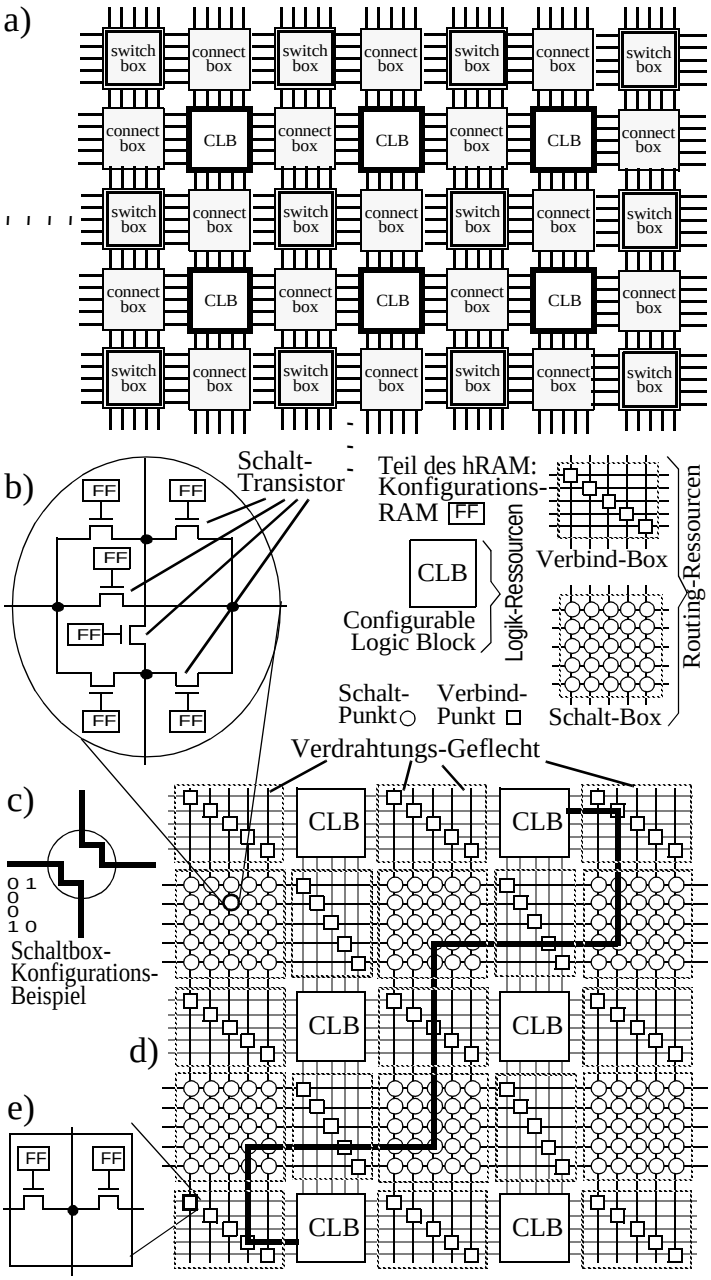

Bild 3. FRGA Insel-Architektur (fine grain morphware):
a) globale Sicht, b) Schaltpunkte einer Schaltbox,
c) FPGA detailliert (nur 1 konfigurierter „Draht“ gezeigt),
d) Verbindungspunkte einer Verbindungs-Box.

Terminologie. Das Morphware-Grundparadigma ist nicht Befehlsstrom-orientiert, weshalb der Terminus „Software“ hier fehl am Platze ist (Bild 4). Vielmehr stehen Datenströme im Vordergrund. In diesem Zusammenhang möchte ich noch auf Terminologie-Probleme hinweisen, die sich aus der Zersplitterung in unterschiedliche Kulturen der Anwendungsgebiete ergeben, die untereinander arm an Querverbindungen sind. Je nach Gebrauch durch Spezialisten kann der gleiche Terminus unterschiedliche Bedeutungen haben kann. Beispiele sind Datenstrom (data stream) und Datenfluß (data flow), und andere. Verwirrung stiftet immer wieder der historische Terminus „dataflow machine“ [25], der für ein exotisches, indeterministisches Maschinen-Paradigma aus einem nun verödeten Forschungsgebiet steht. In diesem Papier verwende ich den Terminus Datenstrom („data stream“) nicht in diesem Sinne, sondern vielmehr so, wie dieser Terminus um 1980 oder früher vom Gebiet der systolischen Arrays definiert worden ist (Bild 5) [18] [19] [20]. Auch lehne ich mich nicht anden Begriff „stream-based computing“ oder „stream processing“ an, wo mit „stream“ lediglich salopp eine Folge „einander ähnlicher“ Datenobjekte gemeint ist wie beispielsweise bei Video, wobei jedoch kein Anti-Maschinen-Paradigma berührt wird. Gelegentlich wird „rekonfigurierbar“ mit mikroprogrammierbar“ verwechselt. Letzteres ist jedoch eindeutig (Mikro-)Befehlsstrom-basiert und gehört somit zur Software-Domäne und nicht zum Bereich der Configware. Für die Programmierungs-Quellen für Morphware (und auch für deren festverdrahtete Variante) sollte keinesfalls die Bezeichnung „Software“ verwendet werden, sondern vielmehr der Terminus „Configware“. Der Begriff „Software“ sollte eindeutig auf Befehlsstrom-basierte Systeme oder Subsysteme beschränkt werden, was in Kapitel 4. und durch Bild 6

1). die Anti-Maschine [9], das Gegenstück zum von-Neumann-Paradigma [26] [27] wird behandelt in Kapitel 3.

begründet wird. Transistoren sehen alle gleich aus. Welcher Schaltvorgang eines Transistors dient der Rekonfiguration und welcher der Operation? Eine grobe Richtschnur kann auch der Schalt-Zeitpunkt sein wie es in Bild 7 gezeigt wird. .

2. Morphware-Architekturen und deren Effizienz

Energie-Ersparnis. Dieses Kapitel gibt für Reconfigurable Computing eine Bilanz der technologischen Basis bezüglich Flächen-Effizienz (Flächenverbrauch auf integrierten Schaltungen) und der Durchsatz- und Energie-Effizienz (MIPS per Milliwatt). Diese Parameter sind auch von allgemeinem wirtschaftlichen Interesse angesichts des Flächen- und Energie-Verbrauchs für die Datenverarbeitung. Gravierende Zahlen ergeben sich bei Server-Farmen für das Internet. 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 entscheidet natürlich auch darüber, ob ein Supercomputing-Zentrum eine große Halle voller Baugruppen-Schränke benötigt oder mit nur wenigen Schränken auskommt.

Plattform		Programm-Quelle	Maschinen- Paradigma
Hardware		(nicht programmierbar)	(keines)
morphware	rekonfigurierbare Logik	***Configware***	
	Reconfigurable Computing (Datenstrom-basiert)	***Configware und Flowware***	Anti-Maschine
fest verdrahteter Prozessor	Datenstrom-basiert	***Flowware***	
	Befehlsstrom-basiert	***Software***	von Neumann

Bild 4. Zur Terminologie.

Flächeneffizienz. Bild 8. faßt die Flächen-Effizienz auf der integrierten Schaltung zusammen [28], gemessen als Integrationsdichte in Transistoren pro Chip. Kurve c zeigt, daß die physische Integrationsdichte von FPGAs (c) etwa um 2 Größenordnungen (Faktor 100) hinter der Gordon-Moore-Kurve (a) zurücksteht, da ein Großteil der Fläche für Verdrahtungs-Ressourcen verbraucht wird. Die logische Integrationsdichte (d) gibt die Dichte der Transistoren an, die unmittelbar der Anwendung dienen. Diese liegt abermals um zwei Größenordnungen zurück, da etwa 99 von 100 Transistoren der Rekonfigurierbarkeit dienen (Rekonfigurierbarkeits-Overhead). Bei sehr komplexen FPGAs treten weitere Effizienz-Verluste auf (Kurve e) dadurch, daß beim Routing wegen Verdrahtungs-Stau nicht alle CLBs anschließbar sind.[28].

Relativierung dieser Bilanz. Diese auf den ersten Blick enorm schlecht erscheinende Bilanz bei FPGAs muß allerdings relativiert werden, was allein schon die Anwendungs-Praxis bei Configware beweist, die gegenüber Implementierungen per Software Akzelerationsfaktoren von bis zu mehreren Größenordnungen erreicht, allein schon durchdrastisch höhere Parallelität (Bild 10). Der geringen Flächeneffizienz bei FPGAs sind die vielfachen Facetten der Ineffizienz von Software-Implementierungen gegenüberzustellen, angefangen mit der sehr schlechten logischen Integrationsdichte auch bei modernen Mikroprozessoren, über die nur sehr geringe Parallelität innerhalb eines Mikroprozessors, bis zur gegenüber Configware sehr viel höheren Zahl notwendiger Speicherzugriffe.

Grobkörnige Morphware. Um Größenordnungen besser ist die Flächenbilanz bei grobkörniger Morphware ((b) in Bild 8). An die Stelle sehr vieler kleiner CLBs treten hier wenige großflächige rDPUs (reconfigurable Data Path Units) [15] [16] [13], die ohne weiteres im sehr flächeneffizienten Stil von Vollkundenschaltungen (full-custom) realisiert werden können. Die rDPUs sind in der Regel zu einer rDPA genannten Matrix gruppiert (reconfigurable Data Path Array). Die geringe Zahl von rDPUs ergibt auch einen um Größenordnungen geringeren Bedarf an hRAM und an rekonfigurierbaren Verdrahtungs-Ressourcen. Die Flächeneffizienz eines solchen rDPA liegt deshalb nur relativ geringfügig unter der Gordon-Moore-Kurve. Aus dem gleichen Grund tritt ein Verdrahtungsstau viel seltener und in viel geringerem Umfang auf, weshalb sich hier die logische Flächeneffizienz kaum von der physischen unterscheidet. Grobkörnige Morphware-Lösungen können also recht nahe an die Flächeneffizienz fest verdrahteter Anwendungs-spezifischer Lösungen heran.

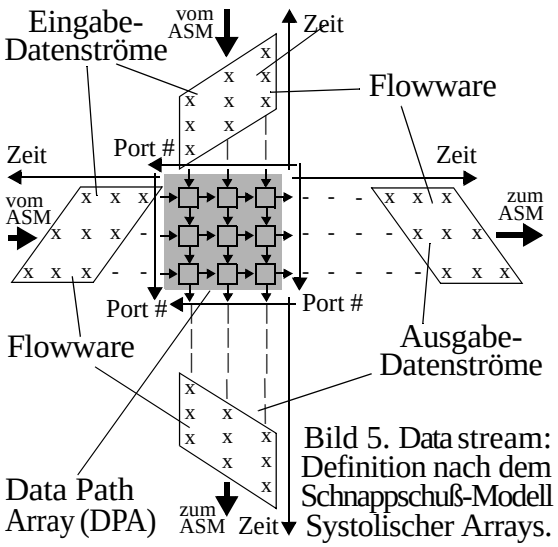

Bild 5. Data stream: Definition nach dem Schnappschuß-Modell Systolischer Arrays.

MOPS per Milliwatt. Mehr denn je ist heute angesichts dramatisch steigender Energiekosten die auf den Energieverbrauch bezogene Durchsatz-Effizienz (in MOPS/mW) von Interesse, zumal große Super-computer-Zentren häufig schon bisher eine jährliche Stromrechnung bis zu Millionenhöhe haben. Bild 9 a gibt eine vergleichende Bilanz von Durchsatz und

Verlustleistung sowie deren Fortentwicklung mit der Einführung neuer Halbleiter-Technologie-Generationen. Gegenüber fest verdrahteter Hardware steht der Standard-Mikroprozessor von anfänglich 2 Größenordnungen bis nunmehr 3 Größenordnungen zurück. Grobkörnige Morphware (mit grobkörniger Rekonfigurierbarkeit) erreicht hierbei fast die Effizienz fest verdrahteter Hardware, was durch die Flächeneffizienz erklärt wird (Bild 8 b). Bei feinkörniger Morphware wie FPGAs steht der Standard-Mikroprozessor von damals um eine Größenordnung bis heute um 2 Größenordnungen zurück. Das Gerücht, daß FPGAs sehr Energie-hungrig seien, ist vor Allem ein Hinweis auf Qualifikations-Probleme bei der Anwendungs-Entwicklung. Es mangelt meist an der algorithmischen „Cleverness" bei der Software-zu-Configware-Migration. Eine neue Taxonomie der Algorithmen und Architekturen wird für die Lehre und für die Praxis benötigt, die bei Software-zu-Configware-Migration u. a. auch den eventuell erhöhten Verdrahtungs-Aufwand berücksichtigt. Mehr hierzu wird in Kapitel 6 diskutiert. Auch eine Reihe publizierter Akzelerations-Faktoren bis zu mehr als 2 Größenordnungen widerlegt solche Gerüchte (Bild 10)

3. Die Geschichte der Grundmodelle hin bis zur Morphware

Der Schwanz wedelt mit dem Hund. Der heutige PC, der selbst die Bedienung seines eigenen Bildschirms nicht mehr schafft, ist ein Beweis dafür, daß Mikroprozessoren wegen ihrer durch hohen Verbrauch an Speicherzyklen gekenn-zeichneten Befehlsstrom-getriebenen sequentiellen Arbeitsweise ohne einen hohen Aufwand an Akzeleratoren nicht mehr auskommen [28]. Aus der Sicht der klassischen Informatik wedelt der Schwanz mit dem Hund. Damit wird de facto das von-Neumann-Maschinenparadigma (vN-Paradigma) des Mainframe-Zeitalters (Bild 11 a) abgelöst durch eine Symbiose aus Wirtsrechner (host) und Akzelerator (Bild 11 b), das zweigleisige Maschinen-Paradigma des PC-Zeitalters, eine Kombination aus dem Befehlsstrom-getriebenem vN-Paradigma und dem Datenstrom-getriebenen Grundmodel (Anti-Maschine) des Akzelerators. Wie auch das Bild 11 b zeigt, ist dabei die schwierige Zusammenarbeit von Programmierern mit Hardware-Experten notwendig, da eine diese Kluft zwischen den Kulturen überbrükkende interdisziplinäre Qualifikation von unseren Kurrikula kaum vermittelt werden.

Symbiose-Modelle. Aus dem Gebiet der eingebetteten Systeme sind FPGAs als Akzeleratoren im heutigen Morphware-Zeitalter nicht mehr wegzudenken, ja mittlerweise „mainstream" geworden. Ein drittes wiederum zweigleisiges Grundmodell wird erforderlich (Bild 11 c), die Symbiose von Mikroprozessor und ebenfalls programmierbarem Morphware-Akzelerator: das Basismodell des heutigen Morphware-Zeitalters. An die Stelle des Akzelerator-Entwurfes durch Hardware-Experten tritt die strukturelle Programmierung der Akzelerator-Plattform. Da häufig Morphware-Akzeleratoren zusammen mit Hardware-Akzeleratoren im gleichen System vorkommen, könnte dieses Modell präzisiert werden gemäß Bild 11 d, was für den Programmierer allerdings nicht nötig ist wegen der unveränderlich vorgegebenen Hardware. Bild 11 e zeigt aus dieser Sicht das Modell der Zukunft mit Co-Compiler, der von einer gemeinsamen Quelle automatisch in Software und Configware partitioniert [29]. Solche Co-Compiler sind leicht zu realisieren, da die grundlagenden Methdologien hierzu vielfach schon seit Jahrzehnten bekannt sind [31] (siehe auch Absatz „Persönlicher Supercomputer„ in Kapitel 5).

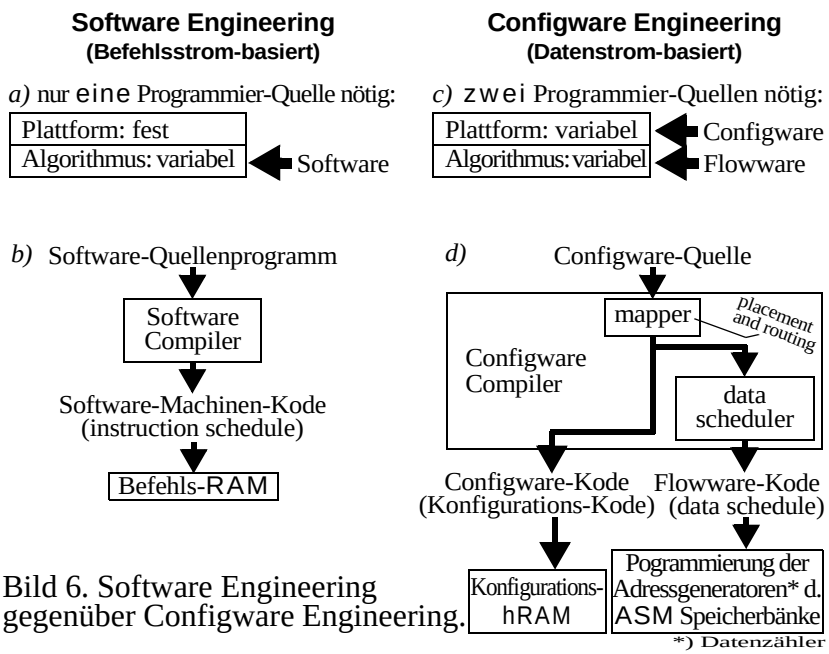

Bild 6. Software Engineering gegenüber Configware Engineering.

Strategische Bedeutung des Morphware-Paradigma. Die Morphware-Akzeleratoren sind heute noch überwiegend FPGAs, die mit ca. 6 Mrd.US-Dollar weltweitem Marktvolumen derzeit das am schnellsten wachsende Markt-Segment der Mikroelektronik darstellen. Der allgemeine Durchdringungsgrad der FPGAs in unsere technische Welt ist hoch beeindruckend. Eindrucksvoll ist dabei die Google-Trefferzahl (Bild 2 Stand vom Oktober 2005) [2] auf *FPGA* selbst (4.680.000 mal), auf *FPGA* kombiniert beispielsweise mit *„embedded"* (2.160.000 mal), mit *wireless* (1.420.000 mal), oder mit *automotive* (629.000 mal). Es ist eindeutig klar, daß eingebettete Systeme aller denkbaren Anwendungsgebiete durch die Mitverwendung von FPGAs dominiert werden. Gegen die 90% aller Software wird heute für eingebettete Systemeimplementiert mit steigender Tendenz [3]. Eingebettete Software ist fast immer von der FPGA-Anwendung berührt, sodaß hier bei der Systementwicklung stets Hardware / Configware / Software Partitionierungs-Probleme gelöst werden müssen. Das Quasi-Monopol des von-Neumann-

Paradigma in unseren Kurrikula verhindert dabei, daß unsere Absolventen für den durch eingebettete Systeme beherrschten Arbeitsmarkt qualifiziert werden. Kürzlich hat Bill Gates in einer Rede auf einem Gipfeltreffen von Gouverneuren der USA die Situation der Lehre in der Informatik mit sehr drastischen Worten kritisiert. Auf moderne Maschinenparadigmen geht das folgende Kapitel 4 ausführlich ein.

4. Das Doppel-Paradigma als zeitgemäßes Grundmodell

Mit Morphware haben wir neben dem Mikroprozessor gemäß von-Neumann-Paradigma ein zweites RAM-basiertes Grundparadigma vorliegen, das ich gern Anti-Maschine nenne [9]. Jedoch haben wir bei der Kompilation statt der zeitlichen Organisation von Befehlsströmen die Plazierung und Verdrahtung (placement and routing) zu realisieren und daran anschließend die Organisation der Datenströme (Bild 6). Morphware bzw. die Anti-Maschine kennt keinerlei „instruction fetch“ mehr während der Laufzeit (Bild 7), was übrigens wegen der Langsamkeit von Speicherzugriffen erheblich zum Akzelerationsfaktor beitragen kann. Dabei ist das dem Betrieb vorausgehende Herunterladen von Rekonfigurationskode in das hRAM die Vorwegnahme sehr viel komplexerer „instruction fetches“.

Dualität der Maschinenparadigmen. Diese Dualität der Maschinenparadigmen müßte das Lehrgebäude der Informatik und ihrer Anwendungen eigentlich in seinen Fundamenten erschüttern. Doch Paradigmenwechsel sind unbequem und gefährden manchmal das angestammte Revier etablierter Platzhirsche, was enorme Trägheitsmomente erzeugt. Wegen Modell-Phobie und abwegiger Terminologie ist diese Dualität der Paradigmen aber oft schwer zu verstehen. 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 4). Unter Anti-Maschine sei hier das Datenstrom-basierte Gegenstück zur Befehlsstrom-basierten von-Neumann-Maschine verstanden. Die Anti-Maschine hat keinen Programmzähler (Bild 12 b), sodaß deren Prozessor keine CPU ist sondern nur eine DPU (Data Path Unit: ohne Programm-Zähler). Im Gegensatz dazu wird beim Terminus „CPU“ stets die interne Existenz eines Befehlsabwicklers mit Programmzähler unterstellt (Bild 12 a). Hoffentlich ist dieses Modell einfach genug

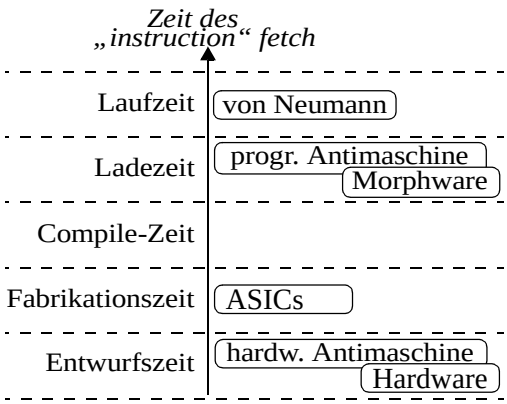

Bild 7. Zeit des “Instruction“ Fetch..

Die Technologie der Anti-Maschine. Die Anti-Maschine hat einen Datenzähler als Bestandteil eines Adress-Generators im Datenspeicher, der deswegen ASM heißt (Auto-Sequencing Memory, s. Bild 12 b). Dies ist eine Verallgemeinerung der DMA-Schaltung (Direct Memory Access) für den Befehls-Zyklen vermeidenden Block-Transfer [32] [33]. Die Aufgabe eines oder mehrerer ASM ist die Erzeugung bzw. Aufnahme Flowware-programmierter Datenströme [34], wie veranschaulicht in Bild 5. Die ASM-Methodologie für generische Adress-Generatoren (GAG) ist weit entwickelt [33] [35] [36] [37], basierend auf verteilten Speicher-Architekturen [38] [39]. In der Morphware-Praxis herrschen parallele Anti-Maschinen vor wie nach Bild 13 b, wobei ein rDPA (Array von rDPUs) mehrere parallele Datenströme aus einem verteilen Speicher bedient. Hierbei ist die nur aus einer einzigen rDPU bestehende Architektur wohl ein seltener Spezialfall (Bild 13 a).

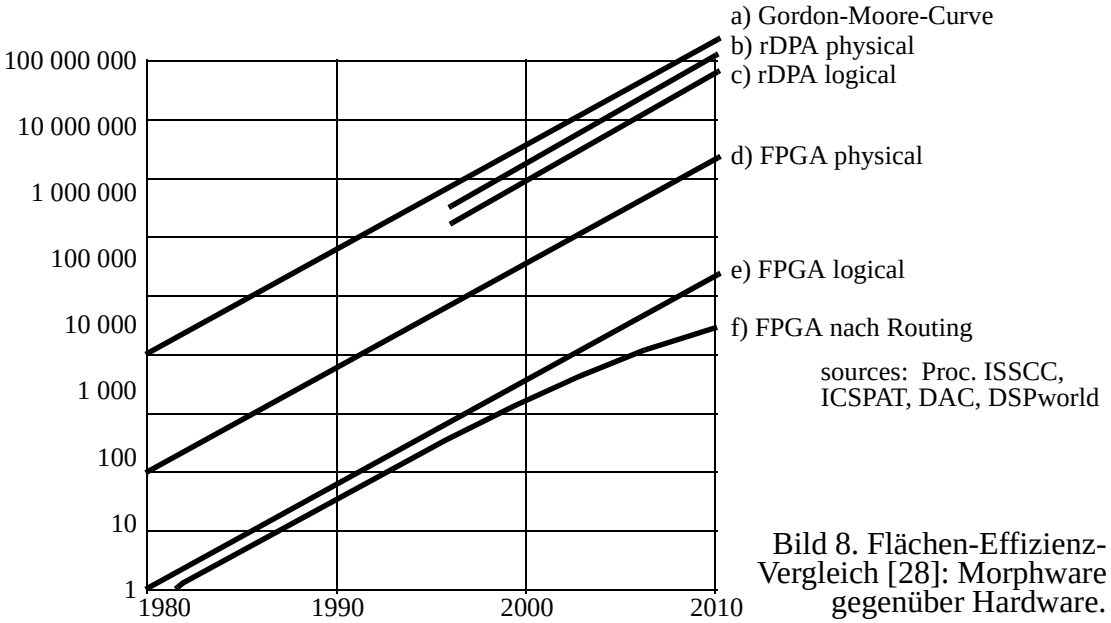

Bild 8. Flächen-Effizienz-Vergleich [28]: Morphware gegenüber Hardware.

Die fest verdrahtete Variante der Anti-Maschine. Von der Technologie her unterscheiden wir zwei Arten von Anti-Maschinen (Bild 4): die zur Morphware zählende programmierbare (rekonfigurierbare) Anti-Maschine, die zwei Arten von Programmierungs-Quellen benötigt: *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 [24]), wobei die „Konfiguration" gleichsam vor der Fabrikation eingefroren und in Hardware gegossen wird. Da nach der Fabrikation keine Konfiguration mehr möglich ist, wird nur eine Art Programmierungs-Quelle benötigt, nämlich nur noch *Flowware*.

Grundlegende Modelle. „Computing" bedeutet die Implementierung von Algorithmen. Die klassische Implementierung per Kompilation aus Software ist Befehlsstrom-basiert und der dabei generierte Maschinenkode ist ein Zeitschritt-Fahrplan für Abruf und Ausführung von Befehlen über der Zeit. Die klassische Form der nebenläufigen Parallelität in einer solchen Umgebung (concurrency) aus dem Mainframe-Zeitalter bedeutet den Wettlauf mehrerer solcher Befehlsströme, also die Parallelität mehrerer quasi voneinander unabhängiger Zeitfolgen. Das Modell des PC-Zeitalters (Bild 11 b) führt zwar zur Ablösung des reinen von-Neumann-Modelles, allerdings jedoch quasi subkutan und ist deshalb für den Programmierer zunächst einmal uninteressant, da die fest verdrahteten Akzeleratoren ja, von Hardwareleuten zuvor entworfen, fertig und unveränderbar vorgegeben sind. Dies ändert sich jedoch mit Anbruch des Configware-Zeitalters. Jetzt liegen zwei verschiedene Arten programmierbarer Plattformen vor: der klassische Befehlsstrom-Prozessor für Software (sequential computing: computing in time), sowie eine rekonfigurierbare, also Struktur-programmierbare Akzelerator-Plattform (Bild 11 c). für Configware zwecks „Reconfigurable Computing" (RC), also computing in space and time. Dessen Anwendungen ergeben sich oft aus der Software-zu-Configware-Migration, also Migration von der Zeit-Domäne nach Raum und Zeit.

Software Engineering gegen Configware Engineering. Es liegt also ein Doppel-Paradigmen-Modell zu Grunde, das im folgenden durch Gegenüberstellung veranschaulicht werden soll. 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 6a). Aus dem als Software vorliegenden Quellenprogramm wird dann Software-Befehlskode generiert, der letztlich ein Zeitfahrplan ist und dann im Befehls-RAM abgelegt und damit bereitgestellt wird zur Abarbeitung durch den Software-Prozessor (Bild 6b). Beim zweiten Paradigma, dem RC-Paradigma (Reconfigurable-Computing-Paradigma), ist hingegen nicht nur der Algorithmus, sondern auch die Plattform programmierbar, weshalb zwei Arten von Programmierungs-Quellen benötigt werden: Configware und Flowware (Bild 6c). Configware dient der strukturellen Programmierung der Plattform durch *Placement und Routing* (Plazierung und „Verdrahtung") oder einen anderen geeignete Abbildung (beispielsweise durch simulated Annealing [15] [16] [21] [22] [23]) über den *„Mapper"* (Bild 6d). Flowware dient der Programmierung der von der programmierten Plattform benötigten Datenströme durch den *„data scheduler"* gemäß Bild 6d.

Flowware/Sprachen. Interessant ist die Gegenüberstellung [40] von Flowware-Sprachen und Software-Programmier-Sprachen. Flowware-Sprachen sind einfach zu implementieren [41] [42]. 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 jedoch mächtiger und erlauben mehrere Datenzähler gleichzeitig und damit parallele Schleifen, im Gegensatz zu Software-Sprachen. Flowware-Sprachen sind einfacher, denn es gibt keine Befehle zur Datenmanipulation, was durch die Configware ja vorkonfiguriert ist. Bekanntlich gibt es beim Betrieb von Morphware zu Laufzeit kein "instruction fetch", sondern nur "data fetch".

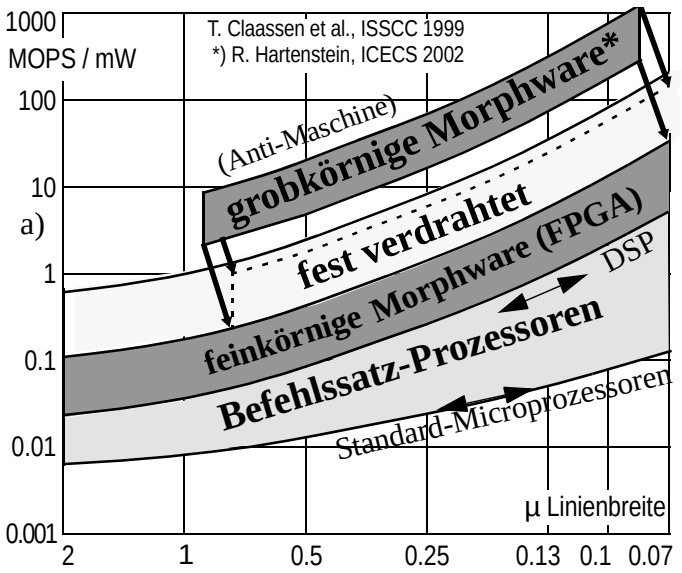

Bild 9. Entwicklung v. Durchsatz-/Energie-Fffizienz.

Dynamisch rekonfigurierbare Architekturen. und deren Umgebungen veranschaulichen den eigenen Charakter des Configware Engineering Solche Architekturen können in schneller Folge zwischen Laufzeit und Konfigurationszeit abwechseln. Noch komplexer können Teile von partiell rekonfigurierbaren FPGAs sich in der Laufzeit befinden, während andere in der Konfigurationsphase stehen. Dadurch kann ein FPGA u. a. sich selbst rekonfigurieren. 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 und der Verwaltung [43] [44]. Auf dieser Basis kann sogar Fehlertoleranz durch Selbstreparatur realisiert werden [45] [46]. Dynamische Rekonfigurierbarkeit ist aber eher verwirrend für Anfänger, diesen sehr schwer zu erklären, und sollte deshalb vielleicht erst im Hauptstudium behandelt werden.

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

		Speicher-zyklen	Nano-sekunden
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**

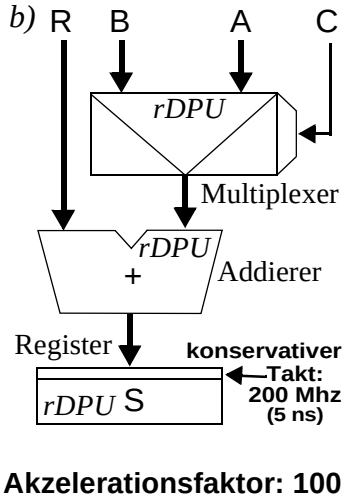

Bild 10. Veranschaulichung der Akzeleration durch Software-zu-Configware-Migration: a) Befehlsstrom-basierte Ausführung auf einem hypothetischen einfachen Prozessor (C = 1), b) Datenstrom-basierte Ausführung in einem rDPA (rekonfigurierbarer Datenpfad-Array).

Geschichte der Grundmodelle. Es wird leider immer noch gern übersehen, daß das klassische Quasi-Monopol des von-Neumann-Modelles, oder wie auch immer wir es nennen wollen, schon seit langer Zeit überrundet ist. Längst herrscht in der Praxis eines der folgenden Symbiose-Modelle. Bild 11 veranschaulicht die Geschichte der Computer-Modelle: Bild 11 a zeigt das ursprüngliche Modell des Mainframe-Zeitalters, Bild 11 b das etwa ab Ende der 70er-Jahre geltende Modell des PC-Zeitalters mit festverdrahteten Akzeleratoren wie beispielsweise Graphik-Karten oder -Chips, Bild 11c das vereinfachte Modell des Configware-Zeitalters, Bild 11d das ab Ende der 90er Jahre und noch heute vorherrschende hybride Modell des Configware-Zeitalters, bei welchem noch rekonfigurierbare Akzeleratoren hinzukommen. Idealerweise sollte hier nun ein Software- / Configware-Co-Compiler den traditionellen Software-Compiler (a) ersetzen. Bild e zeigt das zukünftige Co-Compiler-basierte Modell, das im akademischen Bereich schon voreinem Jahrzehnt implementiert worden war [47] [48] [49].

5. Die Einführung von Configware beim Supercomputing

Das Vordringen rekonfigurierbarer Plattformen wie FPGAs in der Supercomputing-Szene wird demonstriert durch die Google-Antwort [2] auf *FPGA*, jeweils in Kombination mit *"high performance computing"* (64.400 mal), oder *supercomputing* (48,400 mal). Die starke FPGA-Durchdringung vieler typischer Supercomputing-Anwendungsgebiete wird sichtbar durch die von Google angegebene Zahl der Treffer auf *FPGA* in Kombination mit beispielsweise den Stichworten *medical* (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

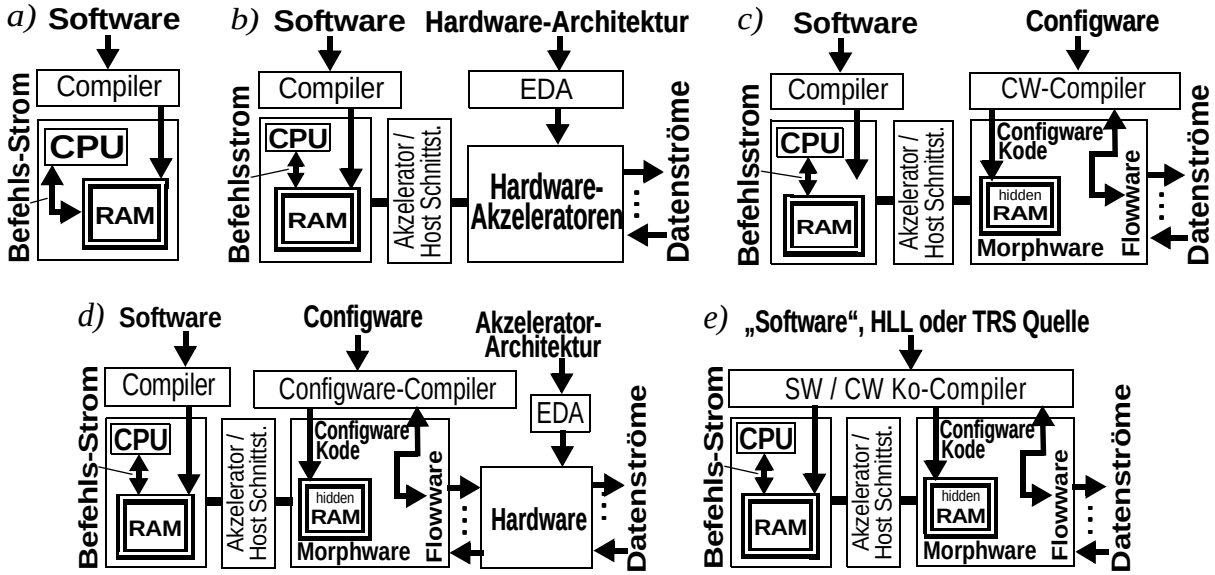

Bild 11. Geschichte der Maschinen-Paradigmen: a) Mainframe-Zeitalter (von-Neumann-artig), b) PC-Zeitalter, c) gegenwärtiges Morphware-Zeitalter, vereinfacht, d) realistisch, e) zukünftig.

FPGAs einen Durchsatzverbesserung um den Faktor von 17 [50] [51]. Als Nebenwirkung ergab sich eine enorme Ersparnis an elektrischer Energie, denn mit FPGAs kann man Energie sparen [52]. 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 [50]. Doch nun möchte ich mit den traditionellen Problemen befassen und anschließend mit deren Überwindung, wie dies durch Reconfigurable Computing versprochen wird.

Der Kommunikationsaufwand steigt überproportional. Die stark unterschiedlichen Kulturen beim Supercomputing und bei eingebetteten Systemen verursachen erhebliche Trägheitsmomente, die den breiten Durchbruch des Reconfigurable Computing bei Supercomputern derzeit noch verzögern. Wesentliche Unterschiede sind die typischen Engpässe auf dem Wege zu hoher Rechenleistung, vor Allem bestimmt durch die langsame Speicher-Zugriffszeit [53] (Bild 14). Bus-Systeme und andere Schalt-Systeme tendieren zu hohem Verwaltungsaufwand [54]. Beim Supercomputing ist deshalb der Transport von Daten während der Laufzeit das vorherrschende Problem, wohingegen an kostengünstigen CPU-Ressourcen dank der Gordon-Moore-Kurve heute ein Überfluß herrscht, der schon lange an das Schlaraffenland erinnert. Typisch ist die Denkweise in kommunizierenden nebenläufigen Prozessen und das Gewicht der von MPI (Message Passing Interface). Je nach Anwendung und Architektur können können massiv parallele nebenläufige Systeme erheblich behindert werden durch Kommunikations-Stau zur Laufzeit. Je größer die Anzahl der CPUs ist, desto schneller steigt der Kommunikationsaufwand in einer Parallelrechnerumgebung (egal welchen Typs) überproportional an. Der Zusammenhang ist nicht-linear.

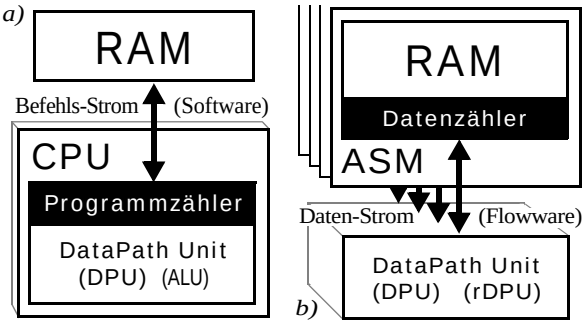

Bild 12. Grund-Paradigmen: a) von Neumann Maschine, b) sehr einfache Anti-Maschine.

MPI (Message Passing Interface). Für Leser aus der Embedded Systems Szene 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. Das Message-Passing Modell bietet dem Programmierer die Grundlage für eine effiziente und portable Implementierung einer umfangreichen Klasse paralleler Algorithmen. Gegenwärtig definiert der MPI-Standard (MPI-1) die Komponenten. Alle Funktionen stehen in einer FORTRAN- und einer C-Version in Form einer Bibliothek zur Verfügung. MPI benutzt eine Reihe interner Datenstrukturen, die Objekte wie Prozeßgruppen, Kommunikatoren und Datentypen repräsentieren. Die Größe und die Gestalt dieser Datenstrukturen bleibt dem Nutzer verborgen. Message Passing Interface (MPI) ist ein auf dem *Communicating Sequential Processes* Modell (*CSP* [55], elektronisch: [56]) nach Tony Hoare [57] basierendes Protokoll, das quasi-parallel Berechnungen auf verteilten, heterogenen, lose-gekoppelten Computersystemen ermöglicht. 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 sowohl auf PC-Clustern (Nachrichtenaustausch z.B. über TCP), als auch auf dedizierten Parallelrechnern ausführbar (mit Kommunikation beispeilsweise über den gemeinsamen Hauptspeicher).

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 insbesondere angesichts des Überfluß an kostengünstigen CPU-Ressourcen dank der Gordon-Moore-Kurve, woran schnelle Schaltmatrizen und GIGAbit Ethernet tendenziell nichts ändern. Der Kommunikationsaufwand steigt überproportional. Jedoch die Datenstrom-dominierte Denke des Reconfigurable Computing verfolgt den umgekehrten Ansatz: der Hocker wird bewegt und nicht das Klavier. Deshalb ist in der Szene der eingebetteten Systeme MPI unbekannt und der Prozeßbegriff allenfalls nur ein Rand-Aspekt, da es sich bei diesem alternativen Paradigma nicht um nebenläufige Prozesse handelt. Nicht die Daten werden mit verwaltungsintensiven Kommunikations-Resoursen nach komplexen Strategien umhertransportiert, sondern die Lokalität der Operation wird an die richtige Stelle im Datenstrom plaziert. Datenströme ergeben sich hier direkt aus dem Anwendungsproblem als per ASM (lesend) generiertem Strom der Operanden und (schreibend) der Ergebnisdaten (Bild 12 und 13). Die Kommunikationswege der DPUs untereinander werden als Pipe-Netzwerk zur Compile-Zeit optimiert, vor der Laufzeit geschaltet und bleiben während der Laufzeit unverändert. Wie das Beispiel in Bild 10 b veranschaulicht erfolgt zur Laufzeit die Kommunikation zwischen den rDPUs ohne Zwischenspeicherung[1] und ganz

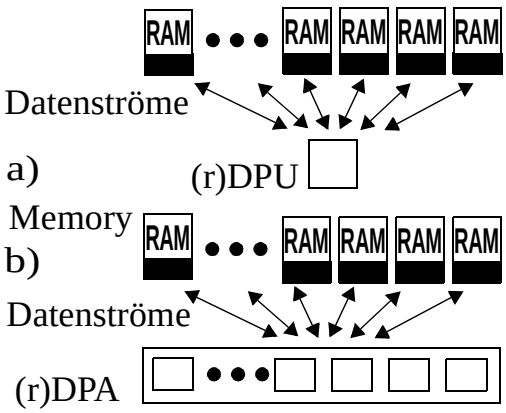

Bild 13. Zur Anti-Maschine: a) einfache Anti-Maschine mit verteilter Speicherrchitektur, c) parallele Anti-Maschine.

ohne Verwaltungsaufwand. Da die Implementierung einer Anwendung meist deutlich weniger Prozessoren bzw. rDPUs hat als Datenobjekte, ist also viel weniger zu bewegen, und dies auch noch zur weniger kostbaren Compile-Zeit.Außerdem werden - im Gegensatz zur klassischen parallelität - gemäß Antimaschinen-Paradigma zur Laufzeit auch keine Befehle umhertransportiert: ein weiterer Akzelerations-Aspekt. Nicht: der richtige Befehl zum richtigen Zeitpunkt in der richtigen CPU, sondern das richtige Datenobjekt zum richtigen Zeitpunkt am richtigen DPA oder rDPA Array-Port ist hier die Devise.

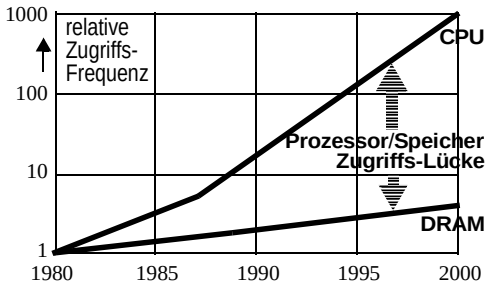

Bild 14. Speicher-Zugriffs-Lücke [53].

Die Implementierung von Flowware. In einem reinrassigen Reconfigurable Computing System besteht die einzige Form der Kommunikation über Speicher nämlich in den Datenströmen (Bild 5) 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 [38] [39] wird zwecks maximaler Parallelität der Anzahl der Ports der Morphware angepaßt. Die Datenströme werden durch Compilation von Flowware implementiert (Bild 6) über die Programmierung der Adreßgeneratoren in den ASMs (Bild 12 b). des verteilten Speichers (Bild 13 b). Speicherbelegungs-Schemata sind nicht auf Vektoren oder Matrizen beschränkt. Durch *generische Adreßgeneratoren (GAG)* [33] [35] [36] [37] kann 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 [33] [38] [39] Ein 2-dimensionaler Adreßraum hat darüber hinaus noch weitere Vorteile, wie beispielsweise bei der Visualisierung von Prozessen und der Speicherbelegungs-Schemata.

Skalierbarkeit. Je nach Anwendung ist beim Supercomputing die Skalierbarkeit erheblich eingeschränkt. Hier ist es vor Allem ein Problem über der Zeit, die zur Ausführung der Kommunikations-Software benötigt wird. Aber 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 17 a veranschaulicht dies an einem FPGA-Beispiel. Hier helfen u. a. Optimierungs-Methoden nach dem Beispiel des strukturierten VLSI-Entwurf der Mead-&-Conway-Ära [58] mit dem Ziel, miteinander kommunizierende Einheiten direkt aneinanderzustecken, sodaß keine externen Verdrahtungs-Ressourcen benötigt werden (Bild 17 b). Die Skalierbarkeit hängt hier stark von der Anwendung ab. So haben beispielsweise übliche Signalverarbeitungs-Algorithmen keine Skalierungsprobleme. Aber beispielsweise der Viterbi-Algorithmus für die fehlerkorrigierende Dekodierung hat hier einen progressiv steigenden Bedarf an Verdratungs-Ressourcen. Für grobkörnige Morphware lassen sich die erforderlichen Pipe-Netzwerke gut durch Mapping-Algorithmen wie beispielsweise mit Simulated Annealing optimieren [16] [21] [22].

Reconfigurable Computing bei Cray und sgi. Reconfigurable Computing wurde inzwischen kommerzialisiert. Cray sgi Bild 16 Cray Inc. in Seattle bietet einen Einschub CRAY XD1 an (Bild 16 a), mit 6 FPGAs vom Typ Xilinx Virtex-4 (XC4VLX160-10 oder XC4VSX55-10) [59]. Auch sgi (Silicon Graphics) bietet einen Einschub an (Bild 16 b) mit einer sogenannten RASC-Technologie [60] ("Rekonfigurierbares Anwendungs-Spezifisches Computing"), die missionskritische Applikationen 100-fach beschleunigen soll [62]. SGI beabsichtigt in Kooperation der Fa.Nallatech [50] [51] [64] 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 [60]: Bioinformatik/Chemieinformatik (Vergleichs- und Kontrast-Routinen beim Durchsuchen von Molekül- oder DNA-Datenbanken), Medizinische Bildgebende Verfahren: (detaillierte Bildverarbeitung und Rendering-Prozesse), Medien, Broadcaster, Bildverarbeitung, Wassermarkierung, Bewegung erfassen, Daten-Konversion, Öl & Gas (Zeitanalyse beim Ölfluss und fast alle Anwendungen, die mit FFT-Algorithmen (Fast Fourier Transform) arbeiten), Verteidigung, Nachrichtendienste (Echtzeit-Datenanalysen mit Routinen für Signalverarbeitung, Erkennen von Objektgrenzen, Mustererkennung).

Ein Meilenstein für Multi-Paradigm Computing. Originalton SGI: „SGI präsentiert den FPGA-interessierten HPC-Benutzern eine Technologie, welche die Industriestandard-Computing-Systeme in puncto Leistungsfähigkeit und Flexibilität einen großen Schritt voran bringt. Ähnlich wie vor 20 Jahren bei der Revolution der Supercomputer und dem Erscheinen der Cluster vor einem Jahrzehnt bietet jetzt auch RASC die Chance, die Anwendungsleistung um Größenordnungen nach oben zu katapultieren. Jedoch anders als bisher: Bei Anschaffung der SGI-RASC-Technologie bleibt das Investment in die bestehende HPC-Umgebung geschützt. Durch ein Erweitern der Plattform wird es möglich, den Durchbruch zu erzielen, auf einer einzigen rekonfigurierbaren Maschine eine Vielfalt von Applikationen mit extrem beschleunigter Anwendungsleistung zu fahren.“ Cray sagt: „Neue FPGA-Technologie erlaubt Anwendern, die Hardware zu ändern“ und sgi nennt: „Rekonfigurierbares Anwendungs-Spezifisches Computing“. Kein Meilenstein, denn demnach wurde von Cray und sgi nur die halbe Meile zurückgelegt. „Hard-

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

Plattform	Jahr	Anwendung	Faktor
Mikroprozessor nach FPGA [66]	2004	diverse MAC-basierte Anwendungen	x 100
Signalprozessor nach FPGA [67]	2004	diverse DSP Anwendungen	x 7 bis x 46
PACT Xtreme 4*4 Array [68]	2003	16-Punkt FIR-Filter	x 16 MOPS/mW
MoM Antimaschine mit DPLA[a] gegen VAX 11/750 [69] [70] [71] [72] [73]	1989 1990	**Design Rule Check** (gleicher Algorithmus)	15.000
		Design Rule Check (untersch. Algorithmen)	2.300
		Electrical Rules Check	> 300
		Lee Routing	> 160
		3x3 2D-FIR-Filter	> 300
gegen SPARC 10/51 [74]	1995	Ising model 128 lattice, 1000 iterations	53,8

a). DPLA: programmierbarer PLA, gefertigt 1983 im Rahmen des E.I.S.-Projekt [75]

Bild 15. Software-zu-Configware Migration: einige Beispiele für Akzelerationsfaktoren.

ware zu ändern“? Sollte heißen: Morphware-Programmierung. „Rekonfigurierbares Anwendungs-Spezifisches Computing“? Dies ist eine Tautologie, denn Rekonfiguration heißt bereits Programmierung für die Anwendung. Die Terminologie zeigt noch immer die Kluft zwischen den Kulturen. Das System ist bei beiden Anbietern immer noch nebenläufig und die rekonfigurierbaren Anteile sind in einer unteren Ebene durch eine spezielle Bibliothek auf eine für den Anwender undurchsichtige Weise eingebracht worden. Ein Co-Compiler fehlt.

Reconfigurable Computing versus Parallel Processing. Eine nützliches Werk über wichtige Aspekte der Parallelverarbeitung ist „The Sourcebook for Parallel Computing“ [65], mit ausführlichen Einführungen über Parallelrechner, Hyper-Threading, Nebenläufigkeit, symmetrische und asymmetrische Multiprozessorsysteme, Dual-Core parallele Anwendungen, Software-Technologien, und Algorithmen. Klassische Parallelität mit nebenläufigen Prozessen (concurrent computing) hat eine Anzahl schwieriger Probleme als Nachteile gegenüber der Parallelität von Antimaschinen, die keinen von-Neumann-Engpaß haben [24] [76] [77] [78]. Bei klassischer Parallelverarbeitung kann die Skalierbarkeit des Durchsatzes einer Anwendung oft the nicht die Spitzenwerte erreichen (peak performance), welche die Plattform anzubieten scheint [65]. Bis heute verbleibt die Zuständigkeit zu Erreichung und Optimierung der Vision skalierbarer Parallelität in der Hand des Anwenders. Amdahls Gesetz erklärt nur eine unter mehreren Ursachen von Ineffizienz [79]. Standard-Mikroprozessor-Chips vom vN-Typ bestehen fast nur aus Cache-Speichern, eine völlig falsche Architektur [24]. Die Metrik zum Anpeilen des Ziels war von Anfang an falsch [24].

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 einige Projekte mit dem Etikett “reconfigurable”, die aber in Wirklichkeit klassisches Parallelrechnen auf einen einzigen Chip übertragen. Ein Parallelrechner ist ein Computer, in dem mehrere Prozesse gleichzeitig auf mehreren CPUs ablaufen [80]. Massiv-parallele Computer besitzen dabei einige zehn bis einige tausend CPUs, die alle gleichzeitig die gleichen Operationen durchführen - natürlich nebenläufig. Multithreading bietet einen anderen Zugang zur einer Art Nebenläufigkeit. Oft sind mehrere Rechner zu einem Cluster zusammengeschlossen, wobei oft jeder einzelne Rechner in einem solchen Cluster selbst noch mehrere Prozessoren hat. Siehe auch. All dies ist nicht Reconfigurable Computing.

Persönlicher Supercomputer. PCs mit einem leistungsfähigen universellen programmierbaren Akzelerator an Bord sind über die Nutzung eines Co-Compilers sind der Schlüssel zum persönlichen Supercomputer (PS). Vorstufen zum persönlichen Supercomputer wurden vor Jahren veröffentlicht im Zusammenhang mit der n-Körper-Simulation [81] [82] [83] [84] [85]. Astrophysiker hatten sich beklagt, daß der teuerste verfügbare Supercomputer nur die Simulation von Sternenhaufen bis etwa zur Größe 100 ermöglicht [86]. GRAPE, nur eine Erweiterungskarte für den PC erlaubte Größen bis etwa 1000 [87] [88], 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 [89] für den PC [89], die weit mehr können soll als nur n-Körper-Simulation. Diese Projekte sind FPGA-basiert, d. h. beruhen auf feinkörniger Rekonfigurierbarkeit. Teils um Größenordnungen mächtiger ist die Anwendung grobkörniger Morphware (Bild 15). Auch Software / Configware Co-Compiler für den PS sind zwar meines Wissens kommerziell noch nicht verfügbar, wurden aber im akademischen Bereich schon implementiert [47] [48] [49] [90] [91] [92], beispielsweise in der Programmiersprache C geschriebene Quellen akzeptierend für den grobkörnig rekonfigurierbaren KressArray [15] [16] [17] [94]. Die Implementierung solcher Co-Compiler ist unproblematisch. Gute Erfahrungen liegen vor und die Methodologie ist teilweise schon einige Jahrzehnte alt [95] [96] [97] [98] [99]. Der Weg zum PS ist also nicht weit. Es fehlt nur noch ein Investor.

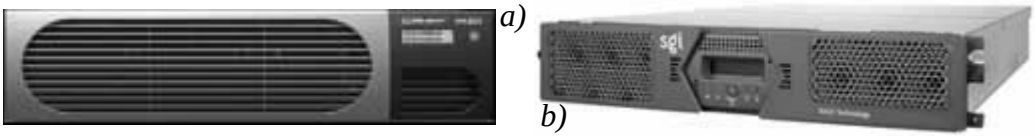

Bild 16. Supercomputer mit FPGA-basierten rekonfigurierbaren Akzeleratoren:
a) Cray XD1: Akzelerations-Faktor von mehr als 100 bei Genome-Sequencing,
b) sgi (Silicon Graphics) RASC™: nennt Akzeleration „um Größenordnungen".

6. Kritik an den Kurrikulums-Empfehlungen

Konsortia. Neben der sehr bekannten amerikanischen ACM/AIS/IEEE Computing Curricula Kommission [100] gibt es noch weitere Konsortia, die sich mit Kurrikulums-Innovation befassen. ARTIST2 [101] [102] [103] ist ein Europäisches Konsortium mit dem Ziel der Verbesserung de Ausbildung in Eingebetteten Systemen. ARTES ist eine Schwedische strategische Initiative in Real-Time-Systemen, mit dem Ziel die nationale Kompetenz zu stärken [105]. Hauptziel ist die Fortentwicklung der Ausbildung im Hauptstudium unter akademisch/industrieller Kooperation. Career Space ist ein Konsortium größerer Firmen der Informations- and Kommunikations-Technologie (ICT) [106]: BT, Cisco, IBM Europa, Intel, Microsoft Europa, Nokia, Nortel Networks, Philips Semiconductors, Siemens, Telefonica and Thales, 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.

Reconfigurable Computing Education. Obwohl die Zielgebiete dieser Konsortien die Haupt-Anwendungsgebiete von FPGAs sind, kommen letztere in diesen Empfehlungen kaum oder 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 unverständliche Einäugigkeit mit der in diesen Empfehlungen das viel mehr Gewicht habende Gebiet des Reconfigurable Computing oder der FPGAs und ihrer Anwendungen weitgehend ausgeklammert wird. Eine Initiative gegen diese Einäugigkeit ist eine neue Workshop-Reihe: The 1st International Workshop on Reconfigurable Computing Education (RE education 2006) [107], der auch das Gebiet des Supercomputing mit einschließen soll. *RC education 2006* findet statt 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 [108]. Es sollte auch eine von der GI und der ITG getragene Fachgruppe „Reconfigurable Computing" gegründet werden, was ich jederzeit gern unterstützen würde.

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 eigene Wissenschaft. sodaß es sehr schwierig ist, die kulturelle Kluft und Lücken der Praxis zu überbrücken. Wegen dieser Fragmentierung kann dessen Untersuchung sehr problematisch sein, zumal so viele verschiedene Aktivisten und Abteilungen involviert sind. Die Programmierung von Morphware bei der Entwicklung von eingebetteten Systemen und all den Anwendungen erfordert viel mehr fähigkeiten, zumindest aus der Informatik. Heute werden Experten mit unterschiedlichem Background und divergierenden Blickwinkeln benötigt, nicht nur für Test und Verifikation solcher Entwürfe, wenn überhaupt möglich, was sehr teuer ist und die Produkteinführung erheblich verzögert. Während die wirtschaftliche Notwendigkeit von Reconfigurable Computing and FPGAs weitgehend erkannt wurde, verfehlte der akademische Bereich den heutigen Arbeitsmarkt durch sehr mangelhafte Aufmerksamkeit über die Ausbildung eines genügend großen Anteils hochqualifizierter auch für Reconfigurable Computing kompetenter Systemenwickler und Configware-Programmierer.

Schädliches Monopol des von-Neumann-Paradigma. Von unseren Kurrikula wird immer noch ignoriert, daß es neben der Software-Industrie inzwischen eine wachsende Configware-Industrie gibt. Moderne FPGAs aus dem Lagerregal haben alle drei 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 Informatins-Technologie.

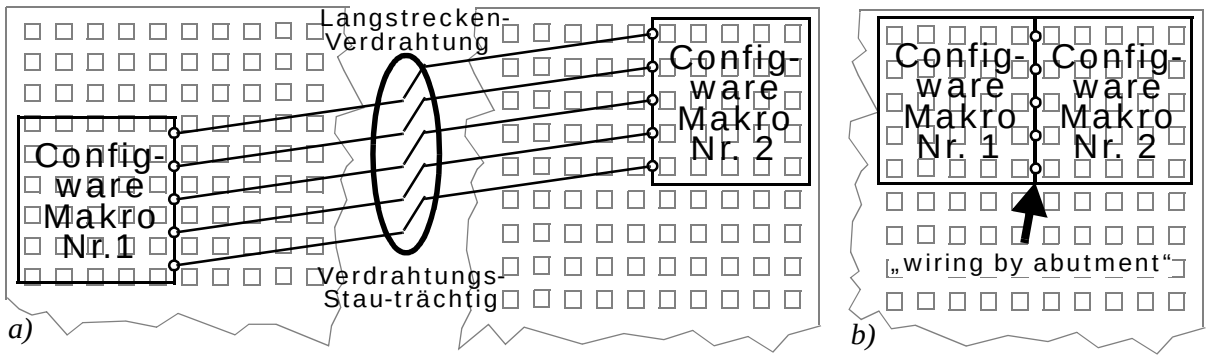

Bild 17. Skalierbarkeits-Problem beim FPGA a) Beispiel, b) Strukturierter Configware-Entwurf.

7. Schlußfolgerungen

FPGAs und grobkörnig rekonfigurierbare Plattformen des Reconfigurable Computing bieten neben erheblicher Stromersparnis gegenüber dem Befehlsstrom-basierten von-Neumann-Paradigma Akzelerationsfaktoren um bis zu mehreren Größenordnungen. Deren Programmierung ist ebenfalls RAM-basiert, was in der Praxis zu einer Doppel-Paradigma-Methodologie geführt hat durch die Entstehung von *Configware Engineering* als Gegenstück zum Software Engineering. Das neue 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 wegen der tiefen Kluft zwischen den Kulturen. Das akademische Bildungswesen ist gefordert, mit neuen Kurrikula über eine integrierende Doppel-Paradigma-Denkweise diese Kluft zu überwinden. Aus interdisziplinär muß intra-disziplinär werden.

8. Literatur

[1] N. N.: Mighty Morphing Power Processors; Business Week, June 2005
[2] R. Hartenstein: The Pervasiveness of Reconigurable Computing; http://hartenstein.de/pervasiveness.html
[3] F. Rammig (interview): Visions from the IT Engine Room; IFIP TC 10 - Computer Systems Technology, URL: [4]
[4] http://www.ifip.or.at/secretariat/tc_visions/tc10_visions.htm
[5] N. N., Department of Trade and Industry (DTI), London, UK, 2001
[6] http://morphware.net
[7] http://www.morphware.org/
[8] http://configware.org
[9] http://antimachine.org
[10] http://en.wikipedia.org/wiki/RDPA
[11] R. Hartenstein (invited embedded tutorial): Coarse Grain Reconfigurable Architecture; Asia and South Pacific Design Automation Conference 2001 (ASP-DAC 2001), 30. Januar - 2w. Februar 2001, Yokohama, Japan
[12] 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
[13] http://pactcorp.com
[14] 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
[15] http://kressarray.de
[16] R. Kress et al.: A Datapath Synthesis System (DPSS) for the Reconfigurable Datapath Architecture; ASP-DAC 1995
[17] http://en.wikipedia.org/wiki/KressArray
[18] N. Petkov: Systolic Parallel Processing; North-Holland; 1992
[19] M. Foster, H. Kung: Design of Special-Purpose VLSI Chips: Example and Opinions. ISCA 1980
[20] H. T. Kung: Why Systolic Architectures? IEEE Computer 15(1): 37-46 (1982)
[21] U. Nageldinger et al.: Generation of Design Suggestions for Coarse-Grain Reconfigurable Architectures; FPL 2000
[22] U. Nageldinger: Coarse-grained Reconfigurable Architectures Design Space Exploration; Dissertation, 2001, URL: [23]
[23] http://xputers.informatik.uni-kl.de/papers/publications/NageldingerDiss.html
[24] C. Chang et al: The Biggascale Emulation Engine (Bee); summer retreat 2001, UC Berkeley
[25] D. Gajski et al.: A second opinion on dataflow machines; Computer, Febr. 1982
[26] A. Burks, H. Goldstein, J. von Neumann: Preliminary discussion of the logical design of an electronic computing instrument; US Army Ordnance Department Report 1946.

[27] H. Goldstein, J. von Neumann, A. Burks: Report on the mathematical and logical aspects of an electronic computing instrument; Princeton Institute of Advanced Study, 1947.

[28] R. Hartenstein (invited paper): The Microprocessor is no more General Purpose; Proc. IEEE International Symposium on Innovative Systems (ISIS), Austin, Texas, 1997

[29] J. Becker: A Partitioning Compiler for Computers with Xputer-based Accelerators, Ph. D. Dissertation 1997, URL: [30]

[30] http://xputers.informatik.uni-kl.de/papers/publications/BeckerDiss.pdf

[31] K. Schmidt: A Program Partitioning, Restructuring, and Mapping Method for Xputers, Dissertation, TU Kaiserslautern 1994

[32] http://de.wikipedia.org/wiki/Direct_Memory_Access

[33] 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

[34] http://flowware.net

[35] M. Herz, et al.: A Novel Sequencer Hardware for Application Specific Computing; Proc. ASAP‘97

[36] M. Herz: High Performance Memory Communication Architectures for Coarse-grained Reconfigurable Computing Systems; Ph. D. thesis, Kaiserslautern, 2001 -- downloadable from: [37]

[37] http://xputers.informatik.uni-kl.de/papers/publications/HerzDiss.html

[38] F. Catthoor et al.: Data Access and Storage Management for Embedded Programmable Processors; Kluwer, 2002

[39] F. Catthoor et al.: Custom Memory Management Methodology Exploration of Memory Organization for Embedded Multimedia Systems Design; Kluwer, 1998

[40] R. Hartenstein: Morphware and Configware; invited chapter in: (ed.: A. Zomaya) Handbook of Innovative Computing Paradigms; Springer Verlag, New York, 2006

[41] http://flowware.net

[42] A. Ast, et al.: Data-procedural Languages for FPL-based Machines; FPL’94

[43] H. Simmler et al.: Multitasking on FPGA Coprocessors; Proc. FPL 2000

[44] H. Walder, M. Platzner: Reconfigurable Hardware Operating Systems: From Design Concepts to Realizations; Proc. ERSA 2003

[45] P. Zipf: A Fault Tolerance Technique for Field-Programmable Logic Arrays; Dissertation, Univ. Siegen, 2002

[46] M. Abramovici, C, Stroud: Improved BIST-Based Diagnosis of FPGA Logic Blocks; Proc. IEEE Int'l Test Conf. 2000

[47] J. Becker et al.: Parallelization in Co-Compilation for Configurable Accelerators; Proc. ASP-DAC´98

[48] J. Becker: A Partitioning Compiler for Computers with Xputer-based Accelerators, Ph. D. Dissertation, TU Kaiserslautern 1997 - downloadable from [49]

[49] http://xputers.informatik.uni-kl.de/papers/publications/BeckerDiss.pdf

[50] 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

[51] http://www.fpgajournal.com/news_2005/08/20050808_03.htm

[52] V. George, J. Rabaey: Low-Energy FPGAs: Architecture and Design; Kluwer, 2001

[53] 87. D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R.Thomas, K. Yelick: A Case for Intelligent RAM; IEEE Micro, Mar. / Apr. 1997.

[54] G. Koch et al.: The Universal Bus Considered Harmful; Proc. 1st EUROMICRO Symposium on the microarchitecture of computing systems; Nice, France, 1975; North Holland, 1975

[55] C. A. R. Hoare: Communicating Sequential Processes, Prentice-Hall, 1985 - URL: [56]

[56] http://www.usingcsp.com/cspbook.pdf

[57] http://de.wikipedia.org/wiki/Tony_Hoare

[58] R. Hartenstein: Fundamentals of Structured Hardware Design - A Design Language Approach at Register Level; North Holland, Amsterdam/New York 1977

[59] http://www.cray.com/products/xd1/

[60] N. N.: SGI ergänzt HPC um RASC - für ´Rekonfigurierbares Anwendungs-Spezifisches Computing’; München, 30.09.2005, - URL: [61]

[61] http://www.pressebox.de/pressemeldungen/silicon-graphics-gmbh/boxid-42342.html

[62] Dietmar Müller: SGI setzt auf rekonfigurierbares Rechnen; ZDNet, 30. September 2005 - URL: [63]

[63] http://www.zdnet.de/news/business/0,39023142,39137009,00.htm

[64] http://www.nallatech.com/

[65] J. Dongarra et al. (editors): The Sourcebook of Parallel Computing; Morgan Kaufmann 2002

[66] 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,

[67] J. Rabaey: Reconfigurable Processing: The Solution to Low-Power Programmable DSP, Proc. ICASSP 1997

[68] V. Baumgarten, et al.: PACT XPP - A Self-Reconfigurable Data Processing Architecture; ERSA 2001

[69] http://xputers.informatik.uni-kl.de/faq-pages/fqa.html

[70] W. Nebel et al.: PISA, a CAD Package and Special Hardware for Pixel-oriented Layout Analysis; ICCAD 1984

[71] R. Kress et al.: A Reconfigurable Accelerator for 32-Bit Arithmetic; International Parallel Processing Symposium, Santa Barbara, USA, April 1995

[72] M.Weber et al.: Automatic Synthesis of Cheap Hardware Accelerators for Signal Processing and Image Preprocessing; 12. DAGM-Symposium Mustererkennung, Oberkochen-Aalen, 1990

[73] M.Weber et al.: MoM - a partly custom-design architecture compared to standard hardware; IEEE Comp Euro 89, Hamburg, Germany, 1989, IEEE Press 1989, p 5/7-9, 1989

[74] 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

[75] http://xputers.informatik.uni-kl.de/staff/hartenstein/eishistory.html

[76] M. Weber et al.: MOM - Map Oriented Machine; in: E. Chiricozzi, A. D'Amico (editors): Parallel Processing and Applications, North-Holland, 1988

[77] Arvind et al.: A critique of Multiprocessing the von Neumann Style; Proc. ISCA 1983

[78] G. Bell (keynote): All the Chips Outside: The Architecture Challenge; Proc. ISCA 2000

[79] G. Amdahl: Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities; AFIPS Conference Proceedings, 1967 (30).

[80] http://de.wikipedia.org/wiki/Mehrprozessorsystem

[81] G. Lienhart: Beschleunigung Hydrodynamischer N-Körper-Simulationen mit Rekonfigurierbaren Rechensystemen; Joint 33rd Speedup and 19th PARS Workshop; Basel, Switzerland, March 19 - 21, 2003

[82] N. Ebisuzaki et al.; 1997 Astrophysical Journal, 480, pp. 432,

[83] T. Narumi, R. Susukita, H. Furusawa, T. Ebisuzaki: 46 Tflops Special-purpose Computer for Molecular Dynamics Simulations: WINE-2; Proc. 5th Int'l Conf. on Signal Processing, pp. 575-582, Beijing, 2000.

[84] T. Narumi, R. Susukita, T. Koishi, K. Yasuoka, H. Furusawa, A. Kawai, T. Ebisuzaki: 1.34 Tflops Molecular Dynamics Simulation for NaCl with a Special-Purpose Computer: MDM; SC2000, Dallas, 2000

[85] T. Narumi, A. Kawai, T. Koishi: An 8.61 Tflop/s Molecular Dynamics Simulation for NaCl with a Special-Purpose Computer: MDM; SC2001, Denver, 2001

[86] R. Hartenstein: Data-Stream-based Computing and Morphware; Joint 33rd Speedup and 19th PARS Workshop (Speedup / PARS 2003), Basel, Switzerland, March 19 - 21, 2003

[87] 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

[88] T. Narumi: Special-purpose computer for molecular dynamics simulations; Ph D diss., University of Tokyo, 1998

[89] R. Männer, R. Spurzem et al.: AHA-GRAPE: Adaptive Hydrodynamic Architecture - GRAvity PipE; Proc. FPL 1999

[90] J. Becker, K. Schmidt: Automatic Parallelism Exploitation for FPL-based Accelerators; Hawaii Int'l. Conf. on System Sciences (HICSS'98), Big Island, Hawaii,1998

[91] K. Schmidt et. al.: A Novel ASIC Design Approach Based on a New Machine Paradigm; J. SSC 1991 - invited reprint from Proc. ESSCIRC 1990

[92] R. Hartenstein, J. Becker, M. Herz, R. Kress, U. Nageldinger: A Partitioning Programming Environment for a Novel Parallel Architecture; Proc. 10th International Parallel Processing Symposium (IPDPS '96) - URL: [93]

[93] http://ipdps.cc.gatech.edu/1996/PAPERS/S13/HARTEN/HARTEN.PDF

[94] 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

[95] L. Lamport: The Parallel Execution of Do-Loops; C. ACM 17,2, Febr. 1974

[96] D. Loveman: Program Improvement by Source-to-Source Transformation; J. ACM 24,1, Jan. 1977

[97] W. Abu-Sufah, D. Kuck, D. Lawrie: On the Performance Enhancement of Paging Systems Through Program Analysis and Transformations; IEEE-Trans. C-30(5), May 1981

[98] U. Banerjee: Speed-up of Ordinary Programs; Ph.D. Thesis, University of Illinois at Urbana-Champaign, DCS Report No. UIUCDCS-R-79-989, Oct. 1979.

[99] J. Allen, K. Kennedy: Automatic Loop Interchange'; Proc. ACM SIGPLAN'84, Symp. on Compiler Construction, Montreal,Canada, SIGPLAN Notices 19, 6, June 1984

[100]N.N.: Computing Curricula 2004; Joint Task Force for Computing Curricula 2004, 22 November 2004, etc.

[101]N.N.: W2.All.Y1Guidelines for a Graduate Curriculum on Embedded Software and Systems; ARTIST Consortium, May 12, 2003; URL: [102]

[102]http://www.artist-embedded.org/Education/Education.pdf

[103]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

[104]http://www-verimag.imag.fr/~caspi/WESE/wese-cfp.html

[105]http://www.artes.uu.se/events/

[106]http://www.career-space.com/cdguide/serv6.htm

[107]http://helios.informatik.uni-kl.de/RCeducation/

[108]http://isvlsi06.itiv.uni-karlsruhe.de/