Reconfigurable Computing:
urging a revision of basic CS curricula

(keynote)
Reiner Hartenstein, © 2002, University of Kaiserslautern, Germany
http://configware.de reiner@hartenstein.de

Abstract

The paper introduces the antimatter model of computing sciences to model the duality of basic computing paradigms: classical instruction-stream-based computing versus data-stream-based computing preferable for reconfigurable computing. This new model is intended to be understood not only by experts, but also by a wider audience in order to prepare for the overdue drastic revision of fundamentals of CS and CS&E curricula - not only in academia.

The antimatter of computing introduced as a part of this model is intended to be the common general model for reconfigurable computing systems and the structural co-processing part of embedded systems coming with hardware/software co-design and configware/software co-design.

1. Introduction

In 1927 Paul Dirac stated (fig. 1), that an electron (fig. 2 a) cannot be described without having an anti electron as a partner (fig. 2 b). In 1932 this positron has been detected within cosmic radiation, what verified Paul Dirac's prediction of the existence of antimatter. A new exciting research area had been opened up (fig. 1). Finally anti hydrogen has been created (fig. 2 b), the anti atom of the hydrogen atom (fig. 2 a). Meanwhile particle physics claims, that each particle has a corresponding antiparticle. All charges have a similar amount, but with opposite sign. But there is an asymmetry between matter and antimatter, which is no yet well known and is still under investigation.

There are asymmetries between matter and antimatter

Another phenomenon is annihilation, happening when a particle and its antiparticle collide and disappear being converted into energy. However, in our usual all-day, we don't care about antimatter. Also in computer science the overwhelming majority of people does not care about its antimatter, nor has the slightest idea of its existence.

2. Matter of Computing

In Computing the world of matter is the world of instruction streams, controlled by the program counter used as state register. For simplicity I call the basic principles the von Neumann paradigm (vN paradigm), partitioning the machine into CPU datapath, instruction sequencer, RAM, and, I/O. In this world of matter model the instruction stream electron is spinning around the CPU nucleus (fig. 3 a).

1928: Paul Dirac: „there should be an anti electron having positive charge" and "there are asymmetries "(Nobel price 1933)
1932: Carl David Anderson detected this „positron" in cosmic radiation (Nobel price 1936)
1954: new accelerators: cyclotron, like Berkeley's Bevatron
1955 Owen Chamberlain et al. create anti proton on Bevatron
1956: anti neutron created on Bevatron
1965: creation of a deuterium anti nucleus at CERN
1995: hydrogen anti atom created (fig. 2) at CERN – by forcing positron and anti proton to merge by very low energy.

Fig. 1: History of antimatter.

The secret of success of software industry is based on RAM, vN paradigm, compatibility, relocatability, and, scalability.

Figure 8 a summarizes the properties of this vN machine paradigm:. This control-procedural execution mechanism is modelled into the brain of each computing science student - as a common machine paradigm, with, or, without stack mechanism extensions. Due to the simplicity of this machine paradigm zillions of programmers can be educated.

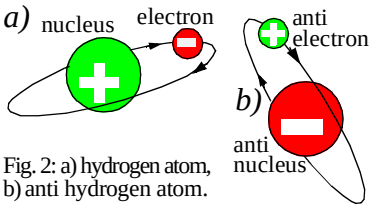

Fig. 2: a) hydrogen atom, b) anti hydrogen atom.

Hardwired vN processors can be fabricated in volume. All personalization (programming) is achieved by downloading the fully relocatable machine code into the immensely scalable RAM. The dominating instruction set became a quasi standard, where compatibility is managed from generation to generation. All three, compatibility, having a machine paradigm, and being RAM-based, are the basis of the tremendous success of the software industry. This is the "normal" matter of computing.

2.1 Problematic matter of computing

But the world of matter in computing has a lot of problems, like, for example, the processor / memory communication bottleneck, i. E. the central von Neumann bottleneck, which is widening from generation to generation and has reached about 2 orders of magnitude. Also the performance of software processor solutions are massively behind comparable dedicated hardwired solutions (fig. 4). For a 0.13 µ technology the difference in MOPS/mW between "intel inside" and the best possible hardwired solution spans up to a factor of 3 orders of magnitude - with growing tendency for future technology generations (fig. 4). There are fundamental flaws in software architectures. First, using time multiplexing with a single piece of logic. Secondly, the overhead associated in moving data back and forth between memory and logic. Third, control itself is overhead. Fourth, pipelining in micro-processors adds another entire level of control overhead for marginal improvements. Chips these days are almost all memory, and the reason is that the architecture is so wrong.

Only about one percent of the power is going into real logic functions and 99 percent is going into caches and other

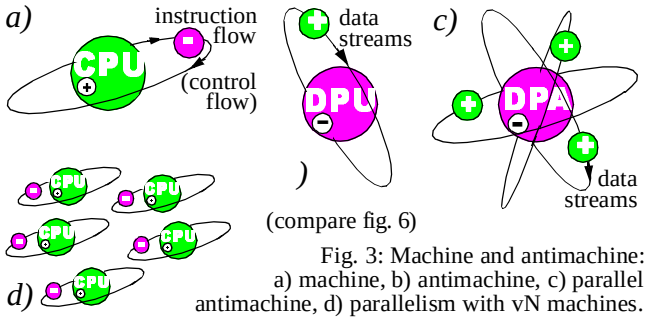

Fig. 3: Machine and antimachine: a) machine, b) antimachine, c) parallel antimachine, d) parallelism with vN machines.

overhead [1]. It is shocking to find the performance difference tom the vN anti world as a factor of 100 to 1,000 [1], even to 10.000. The metric for what is a good solution has been wrong all the time. By hardwired solutions almost 1,000 MOPS per milliwatts or almost 1,000 MOPS per square millimeter can be obtained [1] (fig. 4).

After about a dozen of technology generations the microprocessor is a methusela [3]. Not only in embedded systems the microprocessor more and more takes the role of a handicapped needing a wide variety of prostheses so that the accelerators often occupy more silicon real estate than the processor core itself.

2.2 Matter of Parallel Computing

Parallelism by concurrent computing in the world of matter simultaneously uses several or many machines (CPUs), what I would like to model as a swarm of atoms (fig. 3 d), where each has its own instruction stream electron spinning around.

For many application areas process level parallelism yields only poor speed-up improvement per processor added. Amdahls law explains just one of several reasons of limiting resource utilization. Another dominating problem is the instruction-driven late binding of communication paths, which often leads to massive communication switching overhead at run-time. R&D in the past has largely ignored, that the vN paradigm is not a communication paradigm. Processor architecture has reached a dead end with masses of research projects around speculative execution pipelining cache-related strategies, and others, usually achieving only marginal improvements.

Not only sequential computing, but also parallel computing still computing in time, but concurrently. Locality is not known at machine level, since it is only a physical detail at lower levels - within the responsibility of completely different people coming from the anti world, what is indicated not only by communication barriers.

2.3 The Revision of CS curricula is overdue

In the procedural world of computing in time the typical mind set can be illustrated by the submarine model, a vertical hierarchy of layers: math formula, control/data flow graph, source program, assembler program, and finally relocatable machine code. The hardware is a kind of invisible at lowest level under the surface, like a submarine. This submarine model still pushed by typical software faculties in undergraduate curricula does not support treating hardware (or configware) and software as alternatives. This unbelievable practice ignores the existence of embedded system design, or, cynically uses this term for implementing application software under a frozen-hardware-first philosophy - which definitely is no co-design.

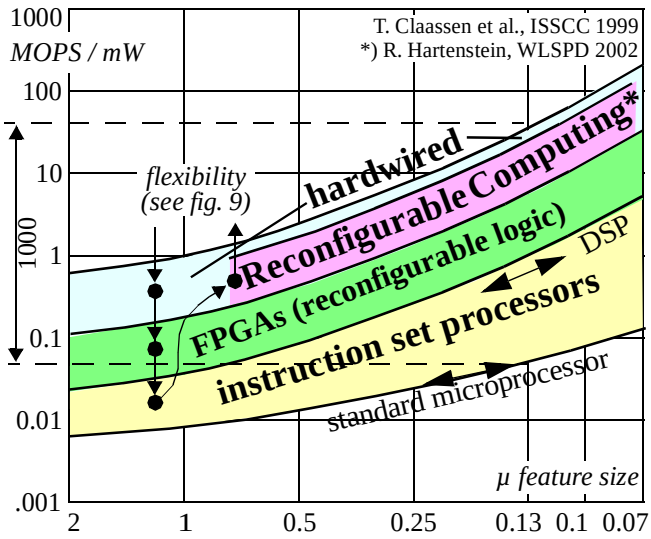

Fig. 4: Energy efficiency vs. flexibility including Reconfigurable computing.

The rapidly increasing number and attendance of conferences (the three most important ones, FCCM, FPGA, and FPL ([4], the eldest and largest), and workshops (like WLSPD) on reconfigurable logic (RL) and reconfigurable computing (RC) as well as the adoption of this topic area by congresses like DAC, ASP-DAC, DATE, ISCAS, SPIE, and many others indicate, that reconfigurable platforms are heading for mainstream ([5] [6] [7] et al.). PL is the fastest growing segment of the semiconductor market [Dataquest]. Mostly driven by telecom applications growing 20% annually, PLD revenue will jump to $7.04 billion in 2004 [IC Insights]. For our software faculties it is time to stop ignoring all this.

About 20 billion of microprocessors are used in embedded systems and 90% of all existing software is running on them [8]. Probably 90% of all programmers will write software for embedded applications by 2010. Based on a model not permitting hardware and software as alternatives, our basic CS curricula ignore this important actual development The revision of the fundamentals of CS curricula is urgently needed.

3. Antimatter of Computing

Meanwhile a world of antimatter of computing has emerged, a world of data streams, controlled by data sequencers - in contrast to the instruction streams of the world of matter (fig. 13). Here the data counter is the antiparticle of the program counter, part of the antimachine, the Xputer machine paradigm (X paradigm), where programming means data scheduling, instead of instruction scheduling as needed for vN. We simply model this data-stream-centric X paradigm by an anti atom (figure 3 b), where the data flow positron (data counter as state register) is spinning around the negative DPU nucleus (data path unit). In contrast to the "von Neumann" machine, which does not support reconfigurable datapaths, its antimachine, the X paradigm, supports both, hardwired and reconfigurable datapaths.

ACRI
Alliant
American Supercomputer
Ametek
Applied Dynamics
Astronautics
BBN
CDC
Convex
Cray Computer
Cray Research
Culler-Harris
Culler Scientific
Cydrome
Dana/Ardent/ Stellar/Stardent
DAPP
Denelcor
Elexsi
ETA Systems
Evans and Sutherland Computer
Floating Point Systems
Galaxy YH-1
Goodyear Aerospace
MPP
Gould NPL
Guiltech
ICL
Intel Scientific Computers
International Parallel Machines
Kendall Square Research
Key Computer Laboratories
MasPar
Meiko
Multiflow
Myrias
Numerix
Prisma
Tera
Thinking Machines
Saxpy
Scientific Computer Systems (SCS)
Soviet Supercomputers
Supertek
Supercomputer Systems
Suprenum
Vitesse Electronics

Fig. 5: Gordon Bell's "Dead Supercomputer Society" [keynote at ISCA 2000].

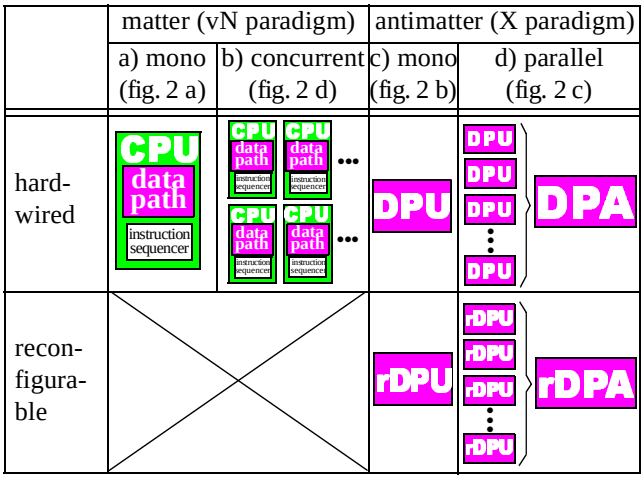

Fig. 6: Asymmetries: machine vs. antimachine (compare fig. 2).

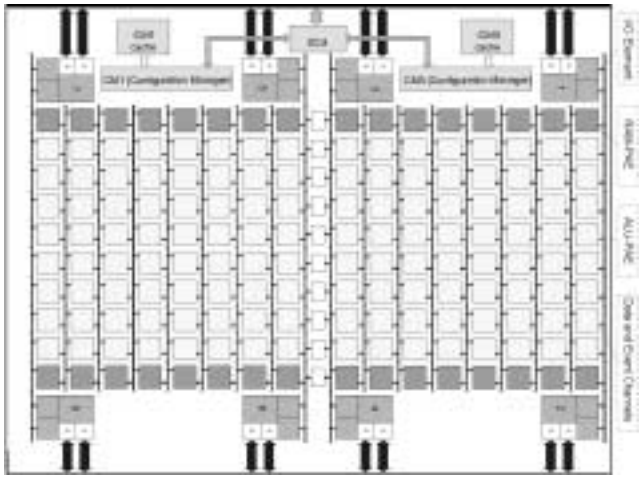

Fig. 7: 32 bit coarse grain rDPA example (PACT AG, Munich).

3.1 Coarse grain reconfigurable systems

Since the antimatter model is a system level concept this paper does not cover FPGAs in detail, but mainly deals with using coarse grain rDPUs or rDPAs (reconfigurable computing). The area efficiency of rDPAs is up to four orders of magnitude better than that of FPGAs [3] [9]. Also energy efficiency is drastically better than with software solutions on microprocessors (see fig. 4).

3.2 Data-stream-based Computing

Classical parallel processing relying on concurrent processes is not the way to go. Generally classical parallel processing has not been successful (see fig. 5). Its fundamental architecture relies on uniprocessors, where all switching of communication paths causes run time overhead. Arrays or other ensembles of CPUs are too difficult to program, and often the run-time overhead is very high, except for a few special application areas favored by Amdahl's law. Problems also stem from the fact, that the CPU operation is control-flow-based. We urgently need the alternative anti paradigm which is not instruction-flow-based.

"von Neuman" is no more the most flexible computing device.

Tailored multiple data streams are pumped from outside through DPUs or the pipe network of data-stream-based DPAs. The KressArray is an early stream-based rDPA [3] [10] [11]. There's no CPU. There's nothing "central". It's fully distributed, with lots of same or different DPUs containing adders, registers, multipliers -- just what's needed for direct mapping of the algorithm onto the architecture. As soon as an application is mapped onto the rDPA, the data streams needed are obtained by using a scheduling algorithm (fig. 12).

The data stream concept has been introduced along with systolic arrays (fig. 13). We divide data stream-based blocks into *systolic arrays* (homogenous DPAs with only linear uniform pipes), *super systolic arrays* (free form pure pipe networks), and, *semi systolic* or *mixed-paradigm* (any kind of mixed or heuristic (r)DPAs). Note, we are not talking about the indeterministic "dataflow machines", a dying exotic research area. This term has been annexed decades ago by these people for a very special concept. So we prefer the term "data streams", instead of "data flow".

3.3 There are Asymmetries

There are asymmetries between matter and antimatter in computing. The CPU of the vN paradigm always has only a single instruction stream spinning around (fig. 3 a). (Multiple threads are resource multiplexing but no parallelism.) The anti machine may have one data stream spinning (fig. 3 b) around a DPU or rDPU (fig. 6 c), or several data streams spinning (fig. 3 c) around a (r)DPA (a (r)DPU array, fig. 6 d). The vN paradigm does not support reconfigurable datapaths (fig. 6 a and b) and requires multiple machines for parallelism (fig. 6 b and fig. 3 d).

Note also the different partitioning scheme. The vN paradigm requires the sequencer to be part of the CPU, whereas the (r)DPU or (r)DPA of the anti machine paradigm has no sequencers on board, so that pipelines or sophisticated pipe networks can be formed. The antimatter model locates data sequencers in the surrounding memory architecture, or, does provide no sequencers at all, if signal processing or other streaming media applications are modelled.

On the antimatter side the "instructions" (here called "configuration code") and the data schedule are not fetched at run time, but downloaded already at loading time, or at design time, if DPU and DPA are not reconfigurable, There are no instruction streams read anymore at run time. The data schedule is downloaded (fig. 12) into the data sequencer [12] [13], especially designed for data-stream-based anti machines (Xputer [14]). Several Xputer architecture examples have been implemented [14] [15] [16]. The table in fig. 8 gives a survey over the asymmetry between the worlds of matter and antimatter in informatics.

3.4 Configware Industry

Software does not run on antimachines. What we need is the anti part of software which is called configware (yet some people call it "soft IP cores") and anti compilers generating it.

Traditional procedural computing systems are run by software code downloaded into its RAM, what is the basis of basis of the extreme flexibility. This is the secret of success of the software industry: it is RAM-based (figure 11). But also using reconfigurable platforms (FPGAs, rDPAs or anti machines) are RAM-based: structural code (configuration code) is downloaded into the ("hidden") RAM of FPGAs.

Configware industry is emerging as the anti industry of software industry. Being RAM-based configware industry is taking off to repeat the success story known from software. Like software, also configware may be downloaded over the internet, or even via wireless channels. FPGA, rDPA or antimachine functionality can be defined or upgraded later at the customer's site - in contrast to hardware it replaces: configware use means a change of the business model - providing shorter time to market and product longevity. Often system-level integrated products without reconfigurability will not be competitive.

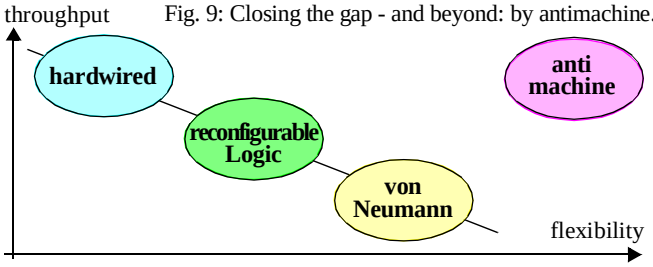

Fig. 9: Closing the gap - and beyond: by antimachine.

3.5 The Antimachine

Why does the antimachine paradigm make sense? Why is it desirable? The von Neumann machine is no more the most flexible computing device (fig. 9). For reconfigurable datapaths von Neumann is the wrong machine paradigm (fig. 8) because its instruction set changes whenever a different configuration code is fetched, so that always a different instruction sequencer would be needed. In using reconfigurable DPUs or DPAs von Neumann architectures fall apart.

The antimachine does not have a von Neumann bottleneck.

For reconfigurable datapaths only the antimachine makes sense. Instructions from the old world of normal matter are usable only for procedural programming (programming in the time domain), whereas antimachine configuration means structural programming by rearranging the internal structure of the datapaths (programming in space). Such a common antimachine paradigm [14] [15] [16], architecture examples [17], and related implementation tools [10] [11] [18] [19] have been published up to more than a decade ago.

4. Reconfigurable Computing

Reconfigurable Computing (RC) [5] [6] is the reconfigurable form of data-stream-based parallel computing, where (coarse-grained) DPUs or DPAs and the interconnect resources are reconfigurable, so that the pipe network can be configured into a rDPA. Figure 7 shows a 32 bit commercial example [20]. An academic example is the KressArray [10], based on the most straight-forward method of data-stream-based computing. Several architectures are briefly outlined in [5]. A drastically higher integration density can be obtained (see fig. 4), than with FPGAs. Energy efficiency practically reaches that of hardwired solutions

	(a) matter	(b) antimatter
machine paradigm	von Neumann (vN)	Xputer
programming	procedural	structural (re)configuration (super “instruction” fetch)
		data scheduling
“instruction” fetch	at run time	at loading time
program execution at run time	instruction schedule	data schedule
operation spin	instruction flow	data stream(s)
nucleus resources	CPU	(r)DPU, or, (r)DPA (nothing central - no CPU)
	hardwired	hardwired or reconfigurable
parallelism	multiple machines	single machine
RA support	no	yes
state register	program counter	data counter(s)
state register located	within CPU	outside (s)DPU or (r)DPA (in data stream wrappers)

Fig. 8: Asymmetry between matter and antimatter.

language category	Languages	Anti Languages
both deterministic	procedural seqencing: traceable, checkpointable	
operation sequence driven by:	read next instruction goto (instruction address) jump (to instruction address) instruction loop, nesting no parallel loops, escapes, instruction stream branching	read next data item goto (data address) jump (to data address) data loop, nesting, parallel loops, escapes data stream branching
state register	program counter	data counter(s)
address computing	massive mem. cycle overhead	overhead avoided
Instruction fetch	memory cycle overhead	overhead avoided
parallel mem. access	interleaving only	no restrictions

Fig. 10: Programming languages versus programming anti languages.

(figure 4) - about two orders o magnitude better than with software on a microprocessor (figure 4).

Data-stream-based (r)DPA use is more efficient than normal matter concurrency. Classical parallel processing relying on concurrent processes is not the way to go, since its fundamental architecture relies on a uniprocessor, where internal instruction execution pipelining brings only marginal improvements. Its alternative, (locally) distributed computing, i. e. DPU array computing, however, means parallelism by an application-specific pipe network in terms of multiple DPUs, not multiple CPUs.

The secret of success of software industry:
- RAM-based,
- machine paradigm,
- µP compatibility,
- scalable RAM,
- relocatable code

For the secret of success of configware industry is still missing:
- FPGA compatibility,
- fully scalable FPGA,
- relocatable configuration code

rDPUs and rDPAs meet requirements much better than current FPGAs.

Fig. 11: The secret of success.

Data-stream-based RC (with rDPU arrays: rDPAs) is urgently needed, since in application areas like multimedia, wireless telecommunication, data communication and others, the throughput requirements are growing faster than Moore's law. The current state of the art in FPGAs does not yet provide sufficient performance nor area efficiency. For flexibility and low power the only viable solution is one with rDPAs like offered by providers like PACT [20].

4.1 Data-Stream-based Memory Organization

Caches do not improve throughput of antimachines, since data-stream-based computing mainly relies on hardwarized loops, but no instruction streams. Along with power dissipations the processor / memory communication bandwidth gap may be dramatic in extremely data-intensive RA applications.

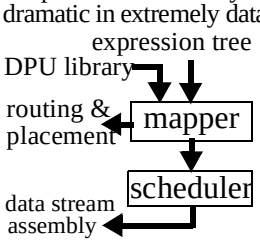

Fig. 12: Data-stream-based compilation principles.

The demand has stimulated the development of “embedded memory architecture optimization” methods (characterized by terms like data transfer and storage exploration: DTSE) using data transfer and storage optimization and data reuse transformation [12], loop transformations, smart memory interface, also with 2-D memory organization, generic address generators [13] [21], as well as a mapping method, where both, data sequencer DPUs and application rDPUs can be mapped onto the same rDPA [5] [12]. These new architectures efficiently utilize multiple rDPA ports by multiple memory banks.

1946: machine paradigm introduced (von Neumann et al.)
1980: data streams introduced (systolic: Kung, Leiserson)
1990: anti machine paradigm introduced (Hartenstein et al.)
1992: Paddi reconfigurable datapath system (Rabaey)
1995: super systolic rDPA introduced (Rainer Kress)

Fig. 13: History of basic computing paradigms.

4.2 Compilers for the Antimachine

Multiple data streams being piped through a rDPU array are the key to massive parallelization. By turning away from linear projections DPSS (Data Path Synthesis System [10]) provided a generalization of the systolic array also to support inhomogenous irregular arrays (fig. 12), removing the restriction to applications with only regular data dependencies. After (r)DPA synthesis a scheduling algorithm prepares the data streams (fig. 12).

The KressArray design space Xplorer [11] [17] [18] supports a generic KressArray family covering any path width and a wide variety of inter-rDPU interconnect resources [11]. The Xplorer also supports mapping memory communication resources (data sequencers etc.) together with the application onto the same rDPA [12] [13].

4.3 Anti Programming Languages

Also experimental high level anti programming languages for that new paradigm have been implemented [17]. Instead of a "control flow" sublanguage a "data stream" sublanguage like *MoPL* [22] recursively defines *data goto*, *data jumps*, *data loops*, *nested data loops*, and *parallel data loops*. The alternative languages are practically the same (except loop parallelism, compare lfig. 10) and anti languages are very easy to learn.

4.4 Configware/Software Co-compilation

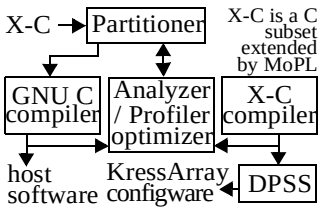

Fig. 14: CoDe-X co-compiler.

Newer commercial platforms bring matter and antimatter together on a single chip, by combining a core processor (ARM, or MIPS), embedded memory and reconfigurable accelerators. Annihilation can be avoided by *co-compilation* including automatic software /configware partitioning. CoDe-X (fig. 14) is the first such co-compilation environment [19], which partitions mainly by identifying parallelizable loops [19] [23] with code down-loadable to the rDPA. MoM (Map-oriented Machine) is an early anti architecture for data-stream-based computing [17] [24].

4.5 The Giga FPGA

It can be predicted, that within some more years FPGAs with 100 million gates or more will be available commercially: the giga FPGA, onto which up to about a hundred soft processor cores can be mapped, leaving sufficient area to other on-board resources like RAM, register files, peripheral circuits and miscellaneous others. This way coarse grain reconfigurable systems like those from PACT Corp. [20] can be mapped onto a (fine grain) FPGA.

4.6 New FPGA architectures needed

A major problem of configware industry in competing with software industry is the fact, that no FPGA architecture is available which is fully scalable and which supports fully relocatable configuration code (fig. 11). The consequence is the need for excessive re-compilation and re-debugging as soon as another FPGA type is used. It is an unanswered question, wether such a FPGA architecture is physically feasible. But it seems to be feasible with an innovative FPGA in connection with a new configware tool tailored to solve this problem. Without solving this problem it is very difficult for configware industry to repeat the success story of the software industry.

5. Conclusions

Fundamentals od current CS education rely on a model not permitting structural and procedural implementations as alternatives. The paper has introduced an antimatter model of data-stream-based computing and has surveyed previous work on its implementation. The paper has shown its computation dichotomy capabilities, its dramatic advantages over the traditional model and urgently advocates its inclusion in basic CS curricula - not only in academia.

6. Literature

1. C. Chang, K. Kuusilinna, R. Broderson, G. Wright; The Biggascale Emulation Engine; FPGA 2002, 10th Int'l Symp. on Field-programmable Gate Arrays; Monterey, CA, Feb 24 - 26, 2002
2. M. Weber et al.: MoM - a partly custom-design architecture compared to standard hardware; IEEE CompEuro 1989
3. R. Hartenstein: The Microprocessor is no more General Purpose (invited paper), Proc. ISIS'97, Austin, Texas, USA, Oct. 8-10, 1997.
4. http://www.fpl.uni-kl.de/fpl/
5. R. Hartenstein (embedded tutorial): A Decade of Research on Reconfigurable Architectures - a Visionary Retrospective; DATE 2001, Munich, March 2001
6. R. Hartenstein (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; ASP-DAC'01, Yokohama, Japan, Jan 30 - Feb. 2, 2001
7. J. Becker: Configurable Systems-on-Chip: Commercial and Academic Approaches; ICECS 2002, Dubrovnik, Croatia, Sept 15-18, 2002
8. F. J. Rammig: Entwurf eingebetteter verteilter Systeme; Anniversary Conference 10 Jahre EAS und DaSS 2002; Dresden, Germany, March 21-22, 2002
9. A. DeHon: Reconfigurable Architectures for General Purpose Computing; rep. no. AITR 1586, MIT AI Lab, 1996
10. R. Kress et al.: A Datapath Synthesis System for the Reconfigurable Datapath Architecture; ASP-DAC'95, Chiba, Japan, Aug. 29 - Sept. 1, 1995
11. U. Nageldinger et al.: KressArray Xplorer: A New CAD Environment to Optimize Reconfigurable Datapath Array Architectures; ASP-DAC, Yokohama, Japan, Jan. 25-28, 2000.
12. M. Herz, R. Hartenstein, M. Miranda, E. Brockmeyer, F. Catthoor: Memory Addressing Organization for Stream-based Reconfigurable Computing; ICECS 2002, Dubrovnik, Croatia, Sept 15-18, 2002
13. M. Herz: High Performance Memory Communication Architectures for Coarse-Grained Reconfigurable Computing Architectures; Ph. D. Dissertation, Universitaet Kaiserslautern, January 2001
14. R. Hartenstein et. al.: A Novel ASIC Design Approach Based on a New Machine Paradigm; IEEE J.SSC, Volume 26, No. 7, July 1991.
15. A. Hirschbiel et al.: A Novel Paradigm of Parallel Computation and its Use to Implement Simple High Performance Hardware; InfoJapan'90, 30th Anniversary of the Computer Society of Japan, Tokyo, Japan, 1990.
16. A. Hirschbiel et al.: A Novel Paradigm of Parallel Computation and its Use to Implement Simple High-Performance-HW; Future Generation Computer Systems 7, 91/92, North Holland - invited reprint of [15]
17. U. Nageldinger et al.: Generation of Design Suggestions for Coarse-Grain Reconfigurable Architectures; Proc. FPL 2000
18. U. Nageldinger: Design-Space Exploration for Coarse Grained Reconfigurable Architectures; Ph. D. Diss., Univ. Kaiserslautern, June 2001
19. J. Becker: A Partitioning Compiler for Computers with Xputer-based Accelerators; Ph. D. dissertation, Kaiserslautern University, 1997.
20. http://pactcorp.com
21. K. Schmidt et al.: A Novel ASIC Design Approach based on a New Machine Paradigm; Proc. ESSCIRC`90, Grenoble, France
22. A. Ast, et al.: Data-procedural Languages for FPL-based Machines; FPL'94
23. K. Schmidt: A Program Partitioning, Restructuring, and Mapping Method for Xputers; Ph.D. Thesis, University of Kaiserslautern 1994
24. M. Weber et al.: MoM - Map-oriented Machine; in: E. Chiricozzi, A. D'Amico: Parallel Processing and Applications, North-Holland, 1988