The Pervasiveness of Reconfigurable Computing

The von Neumann Paradigm loosing its Dominance

Reiner Hartenstein, IEEE life fellow, TU Kaiserslautern, http://hartenstein.de

Abstract. Reconfigurable Computing, the second RAM-based machine paradigm offers a drastic reduction of the electric energy budget and speedup factors by several orders of magnitude - compared to using the von Neumann paradigm, now loosing its dominance. Coming along with a growing configware industry the new discipline of configware engineering is developing as the counterpart of Software Engineering. This paper intends to be a wake-up call discussing the impact of this primarily non-instruction-stream-based fundamentally different mind set on the skill requirements of the IT job market, and on CS and related curricula.

1. PREFACE

Currently the dominance of the basic computing paradigm is gradually wearing off with the gworing pervasiveness of Reconfigurable Computing (RC) - bringing profound changes to the practice of both, scientific computing and ubiquitous embedded systems, as well as new promise of disruptive new horizons for affordable very high performance computing. Due to RC the desk-top personal supercomputer is near. To obtain the payoff from RC we need a new understanding of computing and supercomputing. For bridging the translational gap, the software / configware chasm, we need to think outside the box.

Sceptic about the significance of RC, some colleagues from CS pointed toward the rise and fall of hardware / software co-design (HS codesign). Sure, this has been obscured by renaming conference series and changing slogans of the EDA industry: co-design, H/S co-design, CODES, High-Level Synthesis, System Synthesis, ESDA (electronic system design automation), ESL (electronic system-level design). However, the truth is, that hardware / software codesign is a long-lasting success story within the (also undersurface) embedded systems success story [2] [3] - despite troublesome experiences with EDA industry products. A fall happened inside the CS curricula because the currently still dominant CS culture mainly failed to cure the hardware / software chasm. This is a reason, why embedded software is often implemented by hardware people. The embedded systems scene now is running its own curriculum development effort, since typical CS graduates are not really qualified and tend to miss this most important job market[1].

FPGA: ~10 million hits with Google

Inside the embedded Systems scene at first glance the use of reconfigurable devices like FPGAs has looked more like a variety of hardware design, but on a strange platform. Now we have 2 reconfigurable computing scenes (fig. 2). Meanwhile FPGAs are also used everywhere for high performance in scientific computing, where this is really a new computing culture - not at all a variety of hardware design. Instead of HS codesign we have here software / configware co-design (SC co-design), which is really a computing issue. This major new direction of developments in science will determine how academic computing will look in 2015 or even earlier. The instruction-stream-based mind set will loose its monopoly-like dominance and the CPU will quit its central role - to be more an auxiliary clerk, also for software compatibility issues.

This new direction has not yet drawn the attention of the curriculum planners within the embedded systems scene. For computer science this is the opportunity of the century, of decampment for heading toward new horizons. This should be a wake-up call to CS curriculum development. Each of the many different application domains has only a limited view of computing and takes it more as a mere technique than as a science on its own. This fragmentation makes it very difficult to bridge the cultural and practical gaps, since there are so many different actors and departments involved. Only Computer Science can take the full responsibility to merge Reconfigurable Computing into CS curricula for providing Reconfigurable Computing Education from its roots. CS has the right perspective for a transdisciplinary unification in dealing with problems, which are shared across many different application domains. This new direction would also be helpful to reverse the current down trend of CS enrolment.

2. THE PERVASIVENESS OF THE FPGA

The FPGA (field-programmable gate array) is an array of gate level reconfigurable elements (rE) embedded in a reconfigurable interconnect fabrics [6]. Its ***configware*** code (reconfiguration code [7]: fig 8) is stored in a distributed hRAM memory (***hRAM*** for „***hidden RAM***"), hidden in the background of the FPGA circuitry. Comparable to booting a computer the configware has to be loaded after each power-on. FPGAs are with 6 billion

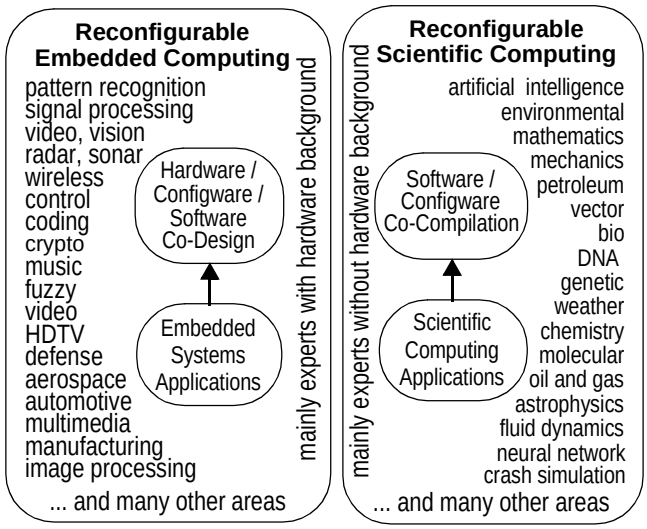

Fig. 2. Two different reconfigurable computing cultures.

1). The amount of software code implemented for embedded systems doubles very 10 months [5] (fig. 3).

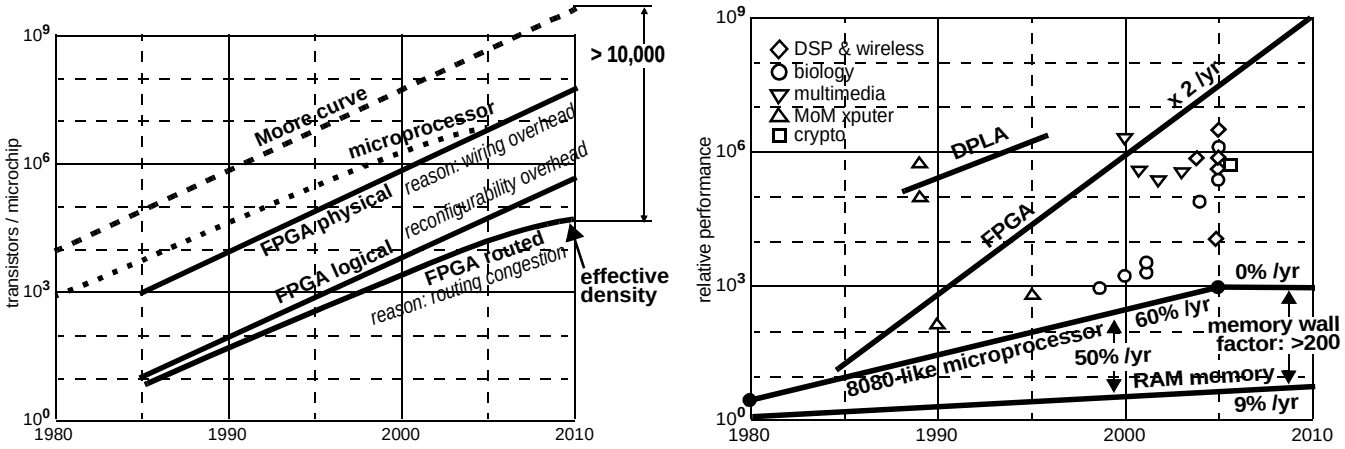

Fig. 1. The Reconfigurable Computing Paradox: a) FPGA integration density, b) FPGA usage speedup factors.

US-Dollars the fastest growing segment of the semiconductor market. Complex projects can be implemented on FPGAs, commodities off the shelf (COTS), without needing very expensive customer-specific silicon. The growth of the number of design starts is predicted from 80.000 in 2006 to 115.000 in 2010 [Dataquest]. Impressive are the hit rates by Google [1]. for ***FPGA*** with almost 10 millions, and of „***Reconfigurable Computing***" with almost 300,000 [8]. The combination of topic area keywords and FPGA [1] illustrates, that FPGAs massively go in a wide variety of application areas.

The Strategic Significance of Reconfigurable Computing. The area of embedded systems is unthinkable without FPGAs [9]. This has been the driving force behind the commercial break-through of FPGAs. The pervasiveness of FPGAs within the embedded systems community is demonstrated by the number of hits by Google [1] [8] upon ***FPGA*** combined with application areas like ***embedded*** (3,280,000), ***wireless*** (1,490,000), ***automotive*** (915,000), ***multimedia*** (731,000), ***signal processing*** (647,000), ***music*** (398,000), ***image processing*** (272,000) and others. About 90% of all software is implemented for embedded systems [5] (Rammig's law in fig 3) dominated by FPGAs usage, where frequently hardware / configware / software partitioning problems have to be solved. The quasi monopoly of the von Neumann mind set in most of our CS curricula prohibits this dichotomic qualification of our graduates, needed for the requirements of the contemporary and future job market. At a summit meeting of US state governors Bill Gates has drastically criticized this Situation in CS education.

Check the pervasiveness [8] **of Reconfigurable Computing by http://fpl.org/pervasiveness.html**

FPGAs in Scientific Computing. The Pervasiveness of FPGAs is not limited to embedded systems, but is also spread over practically all areas of scientific computing, where high performance is required and access to a supercomputing center is not available or not affordable.Some examples are: ***medical*** (710,000), ***physics*** (508,000), ***chemical*** (247,000), ***mathematics*** (171,000), ***fluid dynamics*** (162,000), ***astrophysics*** (158,000), ***bio*** (140,000), ***weather*** (118,000) and other mostly non-embedded scientific applications.

FPGAs and the EDA industry. The pervasiveness of FPGAs also reaches the EDA (Electronic Design Automation) industry, where all major firms spend a substantial effort to offer a variety application development tools and environments for FPGA-based product development. Also FPGA vendors have cooperations with firms in the EDA industry and offer such tools and development environments. Since this is a highly complex market his paper does not go into detail because of a lack of space.

Configware Industry. After switch-on of the supply power the configuration code has to be downloaded to the FPGA's hRAM, which is a kind booting like known from the vN processor. But the source of this code for FPGAs[1] is not software: it definitely does not program instruction streams. The advent of FPGAs provides a second RAM-based fundamental paradigm: the ***Kress-Kung machine*** [11], which, however, is not instruction-stream-based. Instead of organizing the schedule for instruction executions the compilation for FPGAs has to organize the resources by placement and routing.and, based on the result, to implement the data schedules for preparing the data streams moving through these resources (fig 4d). FPGAs or, the Kress-Kung machine, respectively, has ***no „instruction fetch" at run time***. Not to confuse students and customers with the term „software" another term is used for these non-procedural programming sources of RC: the term ***configware***. Not only FPGA vendors offer configware modules to their customers. Also other commercial sources are on the market: a growing configware industry - the little sister of the software industry.

3. THE RECONFIGURABLE COMPUTING PARADOX

Compared to software implementations sensational speed-up factors have been reported for software to configware migrations by using FPGAs. Fig. 1 b shows a few speedup factors picked up from literature, reporting a factor of 7.6 in accelerating radiosity calculations [12], a factor of 10 for FFT (fast Fourier transform), a speedup factor of 35 in traffic simulations [13]. For a commercially available Lanman/NTLM Key Recovery Server [14] a speedup of 50 - 70 is reported. Another cryptology application reports a factor of 1305 [16]. A speedup by a factor of 304 is reported for a R/T spectrum analyzer [18]. In the DSP area [19] for MAC [19] operations a speedup factor of 100 has been reported compared to the fastest DSP on the market (2004) [20]. Already in 1997 versus the fastest DSP a speedup between 7 and 46 has been obtained [21]. In Biology and genetics (also see [22] [22]) a speedup of up to 30 has been shown in protein identification [24], by 133 [25] and up to 500 [26] in ge-

1).in this context the term FPGA does not mean non-FPGA on-chip-modules like processor cores etc., which are usually embedded in modern so-called platform-FPGAs.

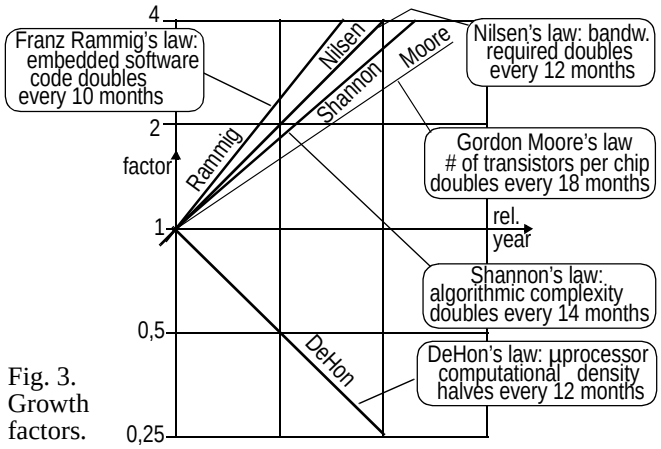

Fig. 3. Growth factors.

nome analysis, as well as 288 with the Smith-Waterman pattern matching algorithm at the National Cancer Institute [28]. In the multimedia area we find factors ranging from 60 to 90 in video rate stereo vision [29] and from 60 to 90 in real-time face detection [30], and of 457 for hyperspectral image compression [31]. In communication technology we find a speedup by 750 for UAV radar electronics [32]. These are just a few examples from a wide range of publications [34] [35] [37] [38] [39] [41] [43] reporting substantial speedups by FPGAs.

The von Neumann paradigm is loosing its dominance.

Alternatives to the FPGA. With the MoM-2, an early reconfigurable computer architecture the following speedup factors have been obtained: 54 for computing a 128 lattice Ising model, >160 for Lee Routing, >300 for an electrical rule check, >300 for a 3by3 2D FIR filter [19], and between 2,300 and 15,000 for grid-based VLSI design rule check [44] [45] [46] [47]. Instead of FPGAs, which have been very small at that time, the MoM-2 used DPLA, a programmable PLA, which has been designed at Kaiserslautern and manufactured via the multi-university VLSI project E.I.S [48]. The DPLA has been especially efficient for computing Boolean expressions. At the time it has been designed, a single DPLA replaced 256 state of the art FPGAs available commercially.

Saving electric energy. A number of publications reports a substantial reduction of the amount of the electricity invoice by up to an order of magnitude though a partial software to configware migration using FPGAs. This is economically highly interesting, which is illustrated, for example by the yearly electricity bill of about 50 million US-Dollars, more than Google's entire equipment cost. How are such power savings by FPGA possible, although the wireless communication industry complaints, that FPGAs are so power-hungry [49]. This is another facet of the Reconfigurable Computing paradox.

FPGA's bad efficiency. The trend line in fig. 1 b, obtained by linear extrapolation, indicates that FPGA performance doubles every year. By the year 2005, 2 decades after the market introduction of the first little FPGA, this yields a lead of a factor of substantially more than 10,000 over the 8080-compatible microprocessor. These unbelievable performance margins contrast against the very bad technological parameters of the FPGA, like area efficiency.integration density, clock frequency, and, compared to hardwired ASICs: power dissipation [44]. The effective integration density (transistors per chip) of FPGAs (fig 1 a) is substantially more than 4 orders of magnitude (more than 10,000) behind the Gordon Moore curve. Three categories of overhead are contributing: wiring overhead, reconfigurability overhead, and routing congestion. Due to wiring overhead, the physical integration density, i. e. the real number of transistors, is down by about 2 orders of magnitude because wiring patterns take most of the chip area. The logical integration density is reduced by another 2 orders of magnitude, since, roughly only 1 of about a hundred transistors directly deserves the application, whereas the other 99 transistors are needed for reconfigurability. A third overhead effect, growing with the size of the FPGA, is routing congestion, because of local excess demands of routing resources not all **rE**s can be connected. FPGAs have more bad parameters. The FPGA clock frequency with around 500 MHz is about almost an order of magnitude lower that of 8080-compatible newest microprocessors with around 3 GHz. Compared to non-reconfigurable ASICs the power dissipation of FPGA-based solutions is substantially higher. All these parameters look very bad. Why are finally the performance and even electricity consumption results so good? This is the Reconfigurable Computing Paradox.

Explaining the paradox. How are these tremendous speedups by up to 4 orders of magnitude and the enormous power savings by about one order of magnitude g possible despite of the really bad technology parameters of FPGAs? The impact of the paradigm shift is the solution of this riddle. It is the paradigm shift, which brings completely different optimization mechanisms, which are so tremendously more effective and capable to override all these bad parameters. It is the main advantage that the machine paradigm is not instruction-stream-based and avoids the von Neumann bottleneck and related problems by three categories of new features: (1) highly parallel distributed memory organization with auto-sequencing memory banks (ASM), and, (2) no caches, and, (3) stalling-proof massive pipelining. Another reason is the fact, that the development of important parameters of von Neumann processors (fig. 3) are slowing down or stopping (like d. g. the growth of the clock frequency), or even negative growth rates, like the computational density (the computational effect per transistor). We should not forget the increase of power dissipation of microprocessors, so that even faster future models would need liquid cooling.

Proven by a multimedia application example. It has been demonstrated [92], that all algorithms needed for a world HDTV set (frame rate conversions, noise and artefacts removal,

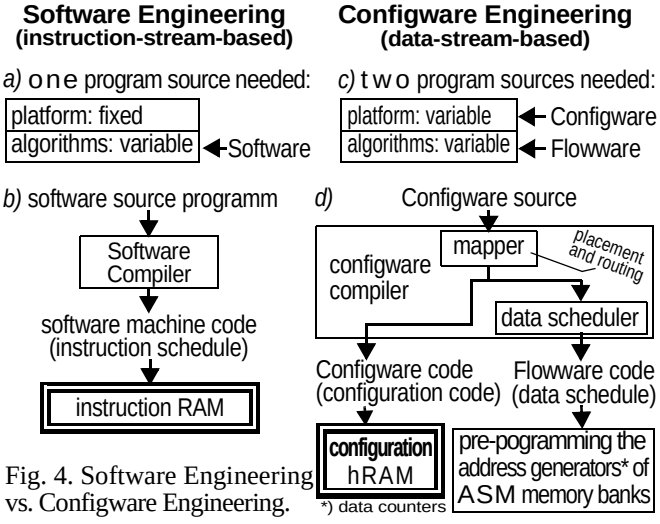

Fig. 4. Software Engineering vs. Configware Engineering.

contrast improvements, image data format standards conversion, adaptation to a wide variety of screen sizes, media server nerwork functions and many more algorithms) can be successfully operated with 8 memory banks on board of a coarse grain array (rDP) chip needing a clock frequency of only 250 MHz.

Minimizing the number of memory cycles. The most important means of speedup is the enormous reduction of the number of memory cycles needed with reconfigurable solutions- because of the growing processor communication bandwidth gap sometimes called memory wall [100]. After switching on the supply power, the downloading of configware code into the hRAM, which reminds to booting, is a kind of super „instruction fetch" **before** run time. FPGAs or, the Kress-Kung machine, respectively, has ***no „instruction fetch" at run time***. By avoiding the memory wall this massively contributes to the speedup obtained by software to configware migration (fig 1 b).

Algorithmic cleverness. RC is also the case for highly effective algorithm transformations. For instance, in genome analysis a datapath width of 64 bits is an overkill, causing an immense wasting of resources, because here 2 bit path width is an optimum. The space in this paper is too short to mention all occasions to draw an enormous payoff from fine grain parallelism. Often the many enormous speedups having been published are the result of algorithmic cleverness of the implementer. By the way: this kind of algorithmic cleverness is usually not taught at academic CS and CE departments.

4. WHAT MEANS RECONFIGURABLE COMPUTING

It may be called the second paradox of Reconfigurable Computing, that despite of its enormous pervasiveness, most professionals inside computer Science and related areas do not really understand its issues. To support configware engineering projects often a hardware expert is hired who may be good implementer, but is not a good translator. From a traditional CS perspective most people do not understand the key issues of this paradigm shift, or, do not even recognize at all, that RC is paradigm shift.A good approach of explanation is to compare the mid set of the Software area vs. the one of the configware field. An dominant obstacle for understanding is also the lack of a common accepted terminology, which massively causes confusion.

Software Engineering vs. Configware Engineering. In total we have 3 different kinds of programming sources (fig 4). The dual-paradigm model can be illustrated by contrasting via Nick Tredennick's model of computer history (fig 4a vs. fig 4c). With the classical software processor only the algorithm is variable, whereas the resources are fixed (hardwired), so that only one type of program source is needed: software (fig 4a), from which the compiler generates software machine code to be downloaded into the ***instruction RAM*** - the instruction schedule for the ***software processor*** (fig 4b). For the Kress-Kung machine paradigm, however, not only the algorithm, but also the resources are programmable, so that we need two different kinds of programming sources (fig 6): ***Configware*** and ***Flowware*** (fig 4c).

1) Configware [7] deserves structural programming of the resources by the „***mapper***" using ***placement and routing*** or similar mapping methods (for instance by simulated annealing [50] [51] [52] [53] [54]) (fig 4d).

2) Flowware [55] deserves programming of the ***data streams*** by the „***data scheduler***" (fig 4d), which generates the flowware code needed for downloading into the generic address generators (GAG) within the ASM auto-sequencing memory banks (fig 4d)

These two different fundamental machine principles, von Neumann software machine vs. the Kress-Kung machine, the configware machine, are contrasted by fig.7.

Flowware Languages are easy to implement [55] [56]. A comparison with software programming languages is interesting [6]. Flowware language primitives for control instructions like jumps and loops can be simply adopted from classical software languages, however, for being used for manipulation of data addresses instead of instruction addresses. Flowware languages are more powerful than software languages and permitting parallel loops by using several data counters used simultaneously, such flowware language primitives are more powerful than these software primitives. Not handling instruction streams, flowware languages are much more simple (because at run time there is only "data fetch", however, no "instruction fetch".

Terminology. Since the basic paradigm is **not** instruction-stream-based, necessarily the term „Configware" should be used for program sources, instead of the term „Software", which would be confusing (fig 8). The term „software" must be unconditionally restricted to traditional sources of classical instruction-stream-based computing (which is reasoned in fig 4). In fact this paradigm relies on data streams, however, not on instruction streams.

Equivocalities of the term „data stream". In computing and related areas there is a babylonian confusion around the term „stream", „stream-based" or „data stream". There is an urgent need to establish a standards committee to work on terminology. For the area of reconfigurable computing the best suitable definition of „data stream" has been established around the year 1980 by the systolic array scene [57] [58], where data streams enter and leave a datapath array being a pipe network (illustrated by fig 9). In fact, there a set of data streams is a data schedule specifying, which data item has to enter or leave which port of the array at which point of time.

The tail is wagging the dog. Because if their memory-cycle-hungry instruction-stream-driven sequential mode of operation microprocessors usually need much more powerful accelerators [44]: the tail is wagging the dog. The instruction-stream-based-only fundamental mind set (vN-only paradigm) as a common model often is still a kind of monopoly inside the qualification background of CS graduates. The real model practiced now is not the von Neumann paradigm (vN) handed down from the ***mainframe age.*** In fact, during the ***PC age*** it has been replaced by a symbiosis of the vN host and the non-vN (i. e. non-instruction-stream-based) accelerators. Meanwhile we have arrived at the (kind of post-PC) ***morphware age*** with a third basic model, where the accelerator has become programmable (reconfigurable). Useful for application development are ***Co-Compilers*** (fig 12), automatically partitioning from the programming source into software and configware [60]. The methodology is known from academic co-compilers [60] [61], easy to implement since most of their fundamentals have been published decades ago [63]. There is a number of trend indications pointing toward an auxiliary clerk role of the CPU for running old software and taking care of compatibility issues. „FPGA main processor vs. FPGA co-processor" asks the CEO of Nallatech [65]: Is it time for vN to retire? The RAMP project, for instance proposes to run the operating system on FPGAs [66]. In fact, in some embedded systems, the CPU has this role already

#		FPGA	rDPA
1	terminology	field-programmable gate array	reconfigurable datapath array
2	reconfiguration granularity	fine-grained	coarse-grained
3	data path width	~ 1 bit	e.g. ~ 32 bits
4	physical level of basic reconfigurable units (rU)	gate level	RT level
5	typical rU examples	LUT (look-up table): determines the logic function of the rU (and, or, not, etc. or flip-flop)	ALU-like, floating point, special functions, etc.
6	configuration time	milliseconds	microseconds
7	clock cycle time	~ 0.5 GHz	~1 - 3 GHz
8	typical effective integration density compared to Gordon Moore curve	reduced by a factor of more than 10,000 (fig. 1a)	reduced only by a few percent (fig. 10)

Fig. 5. fine-grained vs. coarse-grained reconfigurability.

now. But often the awareness is missing.

The Dichotomy of Machine Paradigms is rocking the foundation walls of Computer Science. Because of the lack of a common terminology this duality of paradigms is difficult to understand for people with a traditional CS background. A taxonomy of platform categories and their programming sources, quasi of a terminology floor plan, should help to catch the key issues (fig 8). The Kress-Kung machine is the data-stream-based counterpart of the instruction-stream-based von Neumann paradigm. The Kress-Kung machine does not have a program counter (fig 7 b), and, its processing unit ***is not a CPU*** (fig 7 a). Instead, it is only a ***DPU (Data Path Unit)***: without an instruction sequencer (fig 7 b).

The enabling technology of the Kress-Kung machine has one or mostly several ***data counters*** as part of the ***Generic Adress Generators*** (***GAG***) [67] [69] [70] within data memory banks called ***ASM*** (***Auto-Sequencing Memory***, see fig 7 b). ASMs send and/or receive data streams having been programmed from ***Flowware*** sources [68] (fig 9). An ASM is the generalization of the DMA circuit (Direct Memory Access) [71] [72] for executing block transfers without needing to be controlled by instruction streams inside. ASMs, based on the use of distributed memory architectures [76] are very powerful architectural resources, supporting the optimization of the data storage schemes for minimizing the number of memory cycles [70]. The MoM Kress-Kung machine based on such generic address generators has been published in 1990 [73] [74]. The use of data counters replacing the program counter has first been published in 1987 [75].

Hardwired versions of the Kress-Kung machine. We may distinguish 2 classes of Kress-Kung machines (fig 8): programmable ones (morphware: reconfigurable Kress-Kung machine, needing 2 types of programming sources (see next paragraph and fig. 4 c/d): ***Configware*** for structural programming, and ***Flowware***, for data scheduling. However, also hardwired Kress-Kung machines can be implemented for instance, (the BEE project [77]), where the configuration is been frozen and cast into hardware before fabrication. The lack of reconfigurability after fabrication by not using FPGAs of such hardwired Kress-Kung machines substantially improves the computational density (fig 1 a) for much higher speedup factors and might make sense for special purpose or domain-specific applications. Since after fabrication a reconfiguration is impossible, only one programming source is needed: ***Flowware***.

Dynamically reconfigurable architectures and their environment illustrate the specific flavor of ***Configware Engineering*** being able to rapidly shift back and force between run time mode of operation and configuration mode. Even several separate macros can be resident in the same FPGA. Even more complex is the situation when within partially reconfigurable FPGAs some modules are in run time mode, whereas at the same time other modules are in the configuration phase, so that a FPGA could reconfigure itself. Some macros can be removed at the same time, when other macros are active by being in the run time mode. ***Configware operating systems*** are managing such scenarios [78] [79]. On such a basis even ***fault tolerance*** by self-repair can be implemented [80] [81]. The electronics within the Cibola satellite [82] scheduled to be launched by October 2006 uses such fault tolerance mechanisms to cope with fault introduced by cosmic radiation [84]. Dynamic reconfigurability is confusing for beginners and should be introduced not earlier than art graduate courses.

New educational approaches needed. Although configware engineering is a discipline of its own, fundamentally different from software engineering, and, a configware industry is already existing and growing, it is too often ignored by our curricula. Modern FPGAs as COTS (commodities off the shelf) have all 3 paradigms on board of the same VLSI chip: hardwired accelerators, microprocessors (and memory banks), and FPGAs, and we need both, software and configware, to program the same chip. To cope with the clash of cultures we need interdisciplinary curricula merging all these different backgrounds in a systematic way. We need innovative lectures and lab courses supporting the integration of reconfigurable computing into progressive curricula.

5. RECONFIGURABLE SUPERCOMPUTING

The penetration of reconfigurable platforms like FPGAs in the supercomputing community is demonstrated by the number of hits of Google [8] in reply to ***FPGA***, in combination with ***"high performance computing"*** (81,200), or ***supercomputing*** (65,500). The pervasiveness of FPGAs for many typical supercomputing application areas is shown by the number of Google hits on ***FPGA*** in combination with, for instance, the keywords ***medical*** (710,000 times), ***physics*** (508,000), ***defense*** (287,000), and ***weather*** (118,000) etc. [1]. In the supercomputing application on „***oil and gas***" for the migration onto FPGAs a speedup factor of 17 has been reported [87] [88], together with an enormous reduction of the electricity bill s a side effect, since with FPGAs you can save energy [89]. A geophysicist reports, that with 7 US-Cents pro kWh more than 10,000 US-Dollars per year on the electricity bill could be saved - for each 19 inch module with 64 processors: a yearly saving of half a million US-dollars with 50 such modules[87]. By the way, did you know, that with a 50 million US-dollars Google's yearly electricity bill exceeds the cost of all its equipment?

Platform FPGAs. So far this paper has pointed toward RC using FPGAs. Originally simple FPGAs have been general purpose devices, since flipflops and gates (LUTs) are general purpose elements. But for completeness it should be mentioned, that modern FPGAs, which are often called

#	program source	is compiled into
1	Software	instruction schedule
2	Flowware	data schedule
3	Configware	placement and routing (into a pipe network)

Fig. 6. Terminology: sources by compilation targets.

Platform		Program Source	Machine Paradigm
hardwired logic		(no „program“)	(none)
Morphware® [85][86]	reconfigurable Logic	***Configware***[7]	
	Reconfigurable Computing	***Configware + Flowware***	Kress-Kung machine (AM)
hardwired processor	Data-stream-based	***Flowware***[68]	
	Instruction-stream-based	***Software***	von Neumann (vN)
embedded systems + reconfigurable supercomputing, etc.		***Software + Configware + Flowware***	dual paradigm: vN + AM

Fig. 8. Contemporary Terminologie fo the dual paradigm computing age (compare fig 4 and 7).

platform FPGAs, include other modules embedded in their reconfigurable interconnect fabrics, like, for instance, one or several microprocessors cores (Power PC, ARM, or others) multiple memory banks, multipliers, floating point units, etc., and fast I/O interfaces. Such platform FPGAs are not fully general purpose. The particular collection of on-chip extras usually targets a particular user market segment and makes the platform more or less domain-specific.

rDPA (reconfigurable Data Path Array). Distinguishing reconfigurable devices has several dimensions. A second dimension makes us compare fine-grained vs. coarse-grained reconfigurability (fig. 5). FPGAs with **rE**s of ~1 bit datapath width are ***fine-grained*** reconfigurable. However, coarse-grained reconfigurable architectures (fig 10) [44], ***rDPAs (reconfigurable Data Path Array)*** [90] [91], have path widths like e. g. 32 bits [92] [97]. like, for instance, the KressArray [50] [51] [98], a generalization of the systolic array [57] [58], which is supported by an architecture „Design Space Xplorer“ [52] [53]. For coarse-grained morphware the hRAM and reconfigurable interconnect fabrics requirements are lower by 4 orders of magnitude since rDPAs [19] have only a few ***rDPUs (reconfigurable Data Path Units)*** [50] [51] [92] (typically around a hundred, maybe, 64 or 256, for instance). For the same reason and because of more compact configuration codes the reconfiguration time is lower by orders of magnitude (line 6 in fig 5). The table in fig. 5 summarizes several differences between fine grain and coarse grain. The integration density of rDPAs almost reaches the area efficiency of the Gordon-Moore curve, which is 4 orders of magnitude better than that of FPGAs.

Payoff by rDPA use vs. FPGA:
- **conform to the CS mind set**
- **compute density: x 10,000**
- **ease of co-compilation**

Conforming to the mind set of CS. Other advantages are the ease of compilation for rDPAs, and configuration times by 2 - 3 orders of magnitudes faster. Because rDPAs and their rDPUs are objects of a higher abstraction level, where we find ALUs, registers, and memory banks, these resources come very close to the mind set of CS experts, where the abstraction level of FPGAs better fit to the background of hardware people. Another advantage over FPGAs is the ease of co-compilation. This is a very important advantage, because the state of the art in application development tools for FPGAs [99] is insufficient. The slogan system level design „The business press doesn't understand EDA, because EDA is too esoteric." [93] could be generalized into: „The CS community doesn't understand EDA, because EDA is too esoteric“. ESL [19] offered the enticing hope of specifying a system in an implementation-neutral language, pushing a button, and out would emerge the full, hardware [/ configware] / software co-design. But the dream remained elusive [94] The CS community should take the responsibility instead of waiting for the EDA industry to follow. With the personal supercomputer (chapter 6), based on an on-chip rDPA [19] accelerator, and supported by software / configware co-compilers as demonstrated would de-couple scientific high performance computing from the problems of the EDA industry.

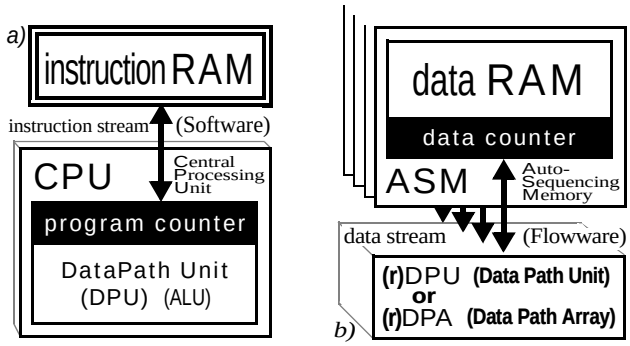

Fig. 7. Fundamental computing machine principles: a) von Neumann, b) Kress-Kung machine (reconfigurable or fixed).

The communication complexity in classical supercomputing is growing disproportionately because of bottlenecks which are typical for classical supercomputing, mainly determined by the memory communication gap [100]. Bus systems and other switching subsystems tend toward a high memory-cycle-hungry overhead [101]: implications of the von Neumann bottleneck Data transport at run time is a dominant problem, whereas there is no lack of affordable CPU ressources. A typical mind set is shown by the standard of MPI (Message Passing Interface), based on Tony Hoare's model of ***Communicating Sequential Processes*** model (***CSP*** [102] [103] [104]) for implementing the exchange of messages between processors in shared memory architectures. The throughput scalability of a particular application often substantially misses the peak performance, which the platform seems to offer [105]. Amdahls law explains just one of several reasons of inefficiency [106]. However, Kress-Kung machines do not have a von Neumann bottleneck. Fig. 13 summarizes the most important sources of speed-up by memory cycle saving from software to configware migration

Moving the stool - not the grand piano. Due to the instruction-stream-based mind set with classical supercomputing the data are moved between CPUs and memory by buses and/or switch boxes at run time. This memory-cycle-hungry method reminds to moving the grand piano to the pianist's stool (fig. 11). However, the data-stream-based mind set of Reconfigurable Computing follows the inverse approach: moving the stool and not the grand piano. Primarily the data are not transferred from memory directly to/from all rDPUs [19]. But the locality of an operation is placed to the right place, where the data stream within a pipe network comes by anyway. These routes through the pipe networks are optimized and decided at compile time, are configured before run time, and remain unchanged all the time during run time. Intermediate results are not stored in memory, but always

piped through from one rDPU to its neighbor. The only memory cycles are needed for transfers between the rDPA's [19] external ports and the ASMs connected to it (fig 7 b and 9). Since most applications have drastically less rDPUs than data items, there is much less to be moved. Avoiding instruction fetch at run time is another speedup aspect. MPI is not needed in such an environment.

Implementation of Flowware. In a pure bred Reconfigurable Computing system the only form of communication via memory modules is ba data streams (fig 9) between the rDPA(s) and the distributed ASM data memory banks, which store only initial operands and final results, however, no intermediate results nor other messages. For optimum parallelism the number of ASM banks [76] should fit to the number of rDPA ports. Over compilation from flowware sources the data streams are implemented (fig 4) by programming the **Generic Adress Generators (GAG)** [67] [69] [70] [107] within the ASMs (fig 7 b). ***Storage schemes*** are not restricted to vectors and matrixes, since GAGs support a wide variety of analytical transformations for a rich supply of storage schemes with minimum memory cycle requirements. GAGs usually do not need memory cycles for run time address computation.

Networks of personal supercomputers (NOPS) are near to reach unbelievable very high performance horizons.

Within this context a ***two-dimensional address space*** opens up an unexpected wealth of efficient and easily GAG-transformable storage schemes [67] [76]. An useful side effect of a 2-D addressing is an excellent versatile visualization support.

Reconfigurable Supercomputing has been commercialized by Cray offering a 19 inch module XD1 with 6 Xilinx FPGAs Virtex-4 [109] and sgi (Silicon Graphics) offers one with its RASC-Technology [110] promising a speedup by 100 for mission-critical applications [112]. In cooperation with Nallatech [87] [88] [114] SGI intends to create a complete ecosystem around RASC to improve acceptance by markets [110] like bio informatics, computational chemistry, medicine, media, oil and gas, most areas working with FFT algorithms, defense etc. The systems of both vendors are still mainly concurrent and the reconfigurable parts are restricted to lower levels of the process hierarchy within a special library being relatively obscure to the user. A Co-Compiler is still missing.

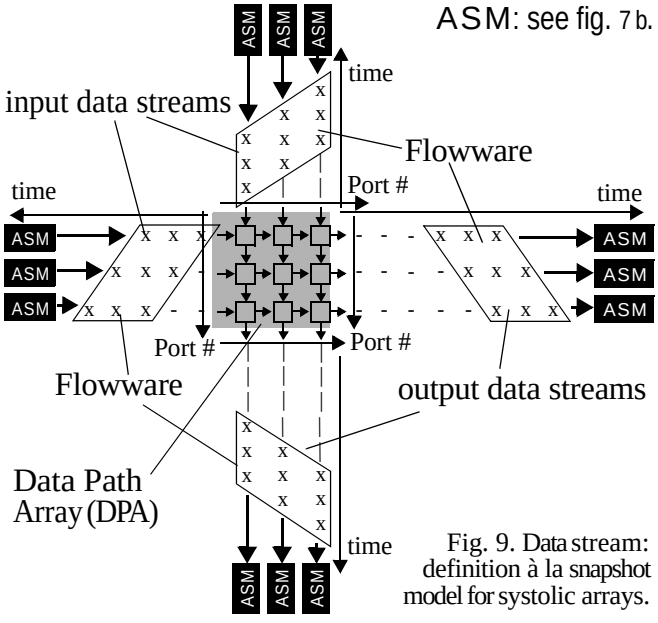

Fig. 9. Data stream: definition à la snapshot model for systolic arrays.

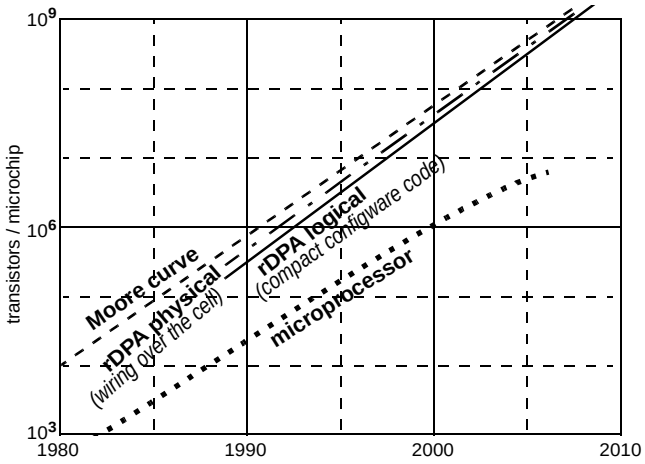

Fig. 10. rDPA (KressArray) integration density.

Crooked Labelling. The difference between Parallel Computing and Reconfigurable Computing is often blurred by projects labelled "reconfigurable", which however, in fact are based on classical concurrency on a single chip where several sequential processes run on several CPUs [115]. This is not Reconfigurable Computing. This is just classical parallel computing. To avoid confusion: switching multiplexers or addressing registers at run time is not „reconfiguration". Note: Reconfigurable Computing never has an instruction fetch at run time.

Reconfigurable Computing and computational biology make CS more fascinating, not only for students.

6. THE PERSONAL SUPERCOMPUTER

The personal supercomputer is near. Because of the ease of co-compilation, with the computing density better than that of FPGAs by 4 orders of magnitude, its much better communication bandwidth and other advantages (compare fig. 5), the rDPA [19] is the case for the desktop personal supercomputer.

Clearing out the microprocessor chip. Within 8080-like microprocessors the CPU takes only a small percentage of the chip area. Most of the area is lost for caches to cope with the memory wall. Caches are useful for highly frequented instruction loops where, however, the acceleration factors are limited and strongly depend on the type of application. But for the Kress-Kung machine caches are useless, since in data loops the values usually do not repeat. With this paradigm other much more powerful mechanisms are available to minimize memory cycles. This is the case for a rDPA as a co-processor, replacing the caches and other stuff not needed any more on the microprocessor chip.

The Personal Supercomputer. PCs with a powerful programmable accelerator on board of the processor chip and using a co-compiler (fig 12) are the key of the personal supercomputer (PS). Forerunners of such a PS have been published years ago, for instance, related to n-body simulation [43]. Astrophysicists have complained, that even the most powerful available supercomputer enabled the simulation of star clusters only up to a size of 100. GRAPE, a PC extension board, however supported sizes up to about 1000 [116] [118], but without changing the algorithm, for example the goal of Prof. Rainer Spurzem of the more than 300 years old Astronomisches Rechen-Institut (University of Heidelberg) [117], together with Prof. Reinhard Männer (University of Mannheim): a reconfigurable

accelerator AHA-GRAPE [119] supporting much more than just n-body simulation. However, by orders of magnitude more powerful is the use of coarse-grained morphware (fig 10). In fact, the PS is near: not only for the desktop of individual users, but also as a component, not only for ***networks of reconfigurable computers*** (***NORCs*** [120] [121]), but particularly for ***networks of personal supercomputers (NOPS),*** new supercomputing centers reaching hitherto unbelievable very high performance horizons, as well as for ***grids of personal supercomputers (GOPS).***

The Technology is available. Coarse-grained reconfigurable datapath arrays, rDPAs, with up to more than a hundred DPUs on a single microchip are available already to-day [92]. Since caches do not make sense for data, such a rDPA could easily be placed onto the intel-compatible processor chip: the PS chip would be ready. Also software / configware co-compiler (fig 12) for the PS have been implemented in academia [60] [61] [122] [123], for instance accepting C language sources for the coarse-grained reconfigurable KressArray [50] [51] [98] [125]. Implementing such a co-compiler is no problem and parts of the methodology are decades old [126]. The personal supercomputer (PS) is near. Only an investor is missing for a commercial co-compiler [127].

7. CURRICULUM RECOMMENDATIONS

The Productivity Crisis. Rapidly growing complexity and pervasiveness of RC-based multi-paradigm devices leads to a productivity crisis of major proportions. On the other hand RC is an efficient approach to cope with the accelerating VLSI design crisis. While the economic importance of RC and its FPGAs is widely acknowledged, but the strategic dimension of RC has not been appreciated until recently, academia has failed to pay sufficient attention to the education of a community of high-quality system designers and configware programmers using such platforms. This has motivated a recent but ever growing interest in the question of educating specialists in this domain and this has also been recognized as a particularly difficult problem.

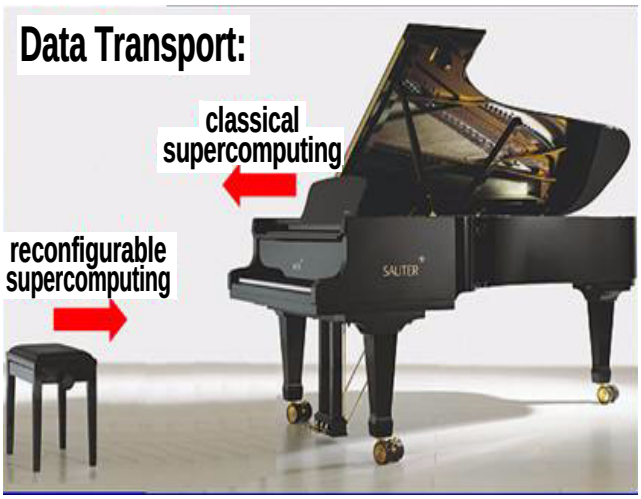

Fig. 11. Not software, but configware gives you wings.

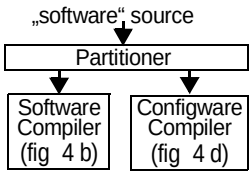

Fig. 12. Software / Configware-Co-Compiler

Fragmentation. Each of these application domains has only a limited view of computing and takes it more as a mere technique than as a science on its own. Consequences are, that it makes it very difficult to bridge the cultural and practical gaps. Given this fragmentation, it can be rather hard to investigate, since there are so many different actors and departments involved. Including and programming reconfigurable platforms in the design of embedded systems as well as embedded real-time systems and all other application areas requires more skills at least from computer sciences. Currently it requires to involve experts from different backgrounds, with dissenting points of view, not only for test and verification of such designs, if at all possible, being very expensive and delaying significantly the introduction of products.

Consortia. The well-known American ACM/AIS/IEEE Computing Curricula Commission [130] as well as other consortia deal with curriculum innovation. The European ARTIST2 [131] [132] [133] consortium works on embedded system education. ARTES is swedish strategic initiative on Real-Time-Systems [135]. Career Space is an industrial consortium in information and communication technology (ICT) [136]: for bridging the current qualification gap, threatening European prosperity because ICT graduates need solid foundations and skills on both sides: engineering sciences and computer science, stressing a wide perspective.

Education missing the job market. Rapidly growing complexity and pervasiveness of RC-based multi-paradigm devices has caused a severe design productivity crisis. On the other side Reconfigurable Computing provides an efficient methodology to solve this crisis. Most new application areas of FPGAs have only a truncated perspective „Computing" and deal with FPGA usage mainly by means of a bag of tricks, rather than as a fundamental scientific prerequisites urgently needed, so that it is very difficult to bridge the gap between cultures and practices. Many different activists and departments and departments are involved in this harmful fragmentation.

Algorithmic Cleverness is missing. To-day experts with different backgrounds and diverging points of view are needed, not only for test and verification of modern designs, experts with different backgrounds and diverging points of view needed, if possible at all, which is expensive and substantially delays the product introduction. Although the economic necessity of RC and FPGAs has been widely recognized, the academic domain mainly missed the education of a sufficiently large share of highly qualified system designers and configware developers. Configware engineering and the programming of morphware requires much more computer science skills, rather than tricks from the culture of a particular application domain. A typical problem is the lack of algorithmic cleverness needed for software to configware migration. A new taxonomy of algorithms and architectures is needed, which extends the notion of algorithm beyond the time domain.

The harmful Monopoly of the von Neumann Paradigm. Our growing configware industry is still mainly ignored by our curricula - mainly, but not only, by our CS curricula. Commodity of the shelf (COTS) FPGAs have all two paradigms together and with several memory banks on board of the same chip. To master the collision of cultures we need transdisci-

feature	von Neumann		reconfigurable computing	
	efficiency	memorycycles	efficiency	memorycycles
PU to PU communication	via memory: cycle-hungry	massive	piped through	no
PU to memory communication	sequential (von Neumann bottleneck)	massive	parallelism by distributed ASM memory	few
execution trigger	instruction fetch needed	massive	no instruction: data-transport-triggereds	no
address computation	memory-cycle-hungry done by CPU	yes	GAG-internal address computation	no
fine-grained parallelism	memory-cycle-hungry done by CPU	massive	occasion f. algorithmic cleverness	few

Fig. 13. Software to Configware migration: speedup sources

plinary curricula which systematically connect or even merge the different backgrounds. We need innovative courses and lab courses for integration of Reconfigurable Computing in advanced curricula. We need to combat trends toward specialization as a main goal of education. We need interdisciplinary curricula which are methodologically interwoven with Computer Science curricula to achieve a unification of fundamentals. We need new computer science curricula, which avoid the monopoly of the von Neumann model mind set and fully accept the dual-paradigm model for teaching the foundations never later than as early as possible at freshmen level.

Unified foundations needed. Meanwhile it has become evident, that many fundamental problems are directly going across many application domains. We need to counter the current trend, where specialization is the target of education systems. We need to go toward interdisciplinary CS-related curricula for unifying the foundations of the discipline since it has become evident that fundamental problems are shared across several different application domains. We need a transdisciplinary approach toward hardware/configware/software co-design, not only in practice, but even more urgently for curricula in Electrical Engineering, Computer Engineering, Computer Science, and Information Technology.

Reconfigurable Computing Education. Although the target areas of all these consortia are the main application domains of reconfigurable resources, FPGAs are hardly mentioned in their recommendations. Our answer to this one-eyed viewpoint is our Reconfigurable Computing Education initiative also including all areas of supercomputing by founding a new workshop series: The 1st International Workshop on Reconfigurable Computing Education (RE education 2006) [137] on March 1, 2006, at Karlsruhe, in conjunction with the IEEE Computer Society Annual Symposium on VLSI (ISVLSI) on March 2 - 3, 2006 [138]. I would support founding an IEEE Computer Society task force, as well as of a GI / ITG Fachgruppe on Reconfigurable Computing.

8. CONCLUSIONS

Compared to the instruction-stream-based von Neumann paradigm, FPGAs and coarse-grained reconfigurable platforms for Reconfigurable Computing offer - in addition to drastic savings on the electric energy budget - speedup factors by several orders of magnitude. Their programming is RAM-based too, which in practice leads to a dual paradigm methodology by using ***Configware Engineering as the counterpart of Software Engineering***. The 2nd RAM-based paradigm avoids most of the often serious communication bottlenecks coming along with concurrent instruction streams. The personal supercomputer is near, not only for the desktop, but also for a new road map to large scale supercomputing of up to now unthinkable highest performance dimensions. Our academic education system should accept this fascinating challenge, especially with new curricula in CS and CE for providing an integrating dual paradigm mind set to bridge the gap and to cure severe qualification deficiencies of our graduates. We need a unification in dealing with problems, which are shared across many different application domains. Interdisciplinary must become transdisciplinary.

9. LITERATURE

[1] R. Hartenstein: Why we need Reconfigurable Computing Education; 1st International Workshop on Reconfigurable Compting Education, March 1, 2006, Karlsruhe, Germany

[2] S. Hang: Embedded Systems – Der (verdeckte) Siegeszug einer Schlüsseltechnologie; Deutsche Bank Research, Jan. 2001, [3], URL: [4]

[3] S. Hang ([2] title translated into English): Embedded Systems - The (subsurface) Triumph of a Key Technology;

[4] http://xputers.informatik.uni-kl.de/VerdeckterSiegeszug.pdf

[5] F. Rammig (interview): Visions from the IT Engine Room; IFIP TC 10 - Computer Systems Technology, URL: [10]

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

[7] http://configware.org

[8] R. Hartenstein: The Pervasiveness of Reconfigurable Computing; http://hartenstein.de/pervasiveness.html

[9] S Hauck: The Roles of FPGAs in Reprogrammable Systems; Proceedings of the IEEE, 86/4, April, 1998.

[10] http://www.ifip.or.at/secretariat/tc_visions/tc10_visions.htm

[11] http://antimachine.org

[12] A. A. Gaffar and W. Luk: Accelerating Radiosity Calculations; FCCM 2002

[13] M. Gokhale et al.: Acceleration of Traffic Simulation on Reconfigurable Hardware; 2004 MAPLD International Conference, Sept.r 8-10, 2004, Washington, D.C., USA

[14] F. Dittrich: World's Fastest Lanman/NTLM Key Recovery Server Shipped; Picocomputing, 2006 URL: [15]

[15] http://www.picocomputing.com/press/KeyRecoveryServer.pdf

[16] K. Gaj, T. El-Ghazawi: Cryptographic Applications; RSSI Reconfigurable Systems Summer Institute, July 11-13, 2005, Urbana-Champaign, IL, USA URL: [17]

[17] http://www.ncsa.uiuc.edu/Conferences/RSSI/presentations.html

[18] J. Hammes, D. Poznanovic: Application Development on the SRC Computers, Inc. Systems; RSSI Reconfigurable Systems Summer Institute, July 11-13, 2005, Urbana-Champaign, IL,USA

[19] ***ASM*** = Auto-Sequencing Memory; ***DSP*** = Digital Signal Processing; ***EDA*** = Electronics Design Automation; ***ESL*** = Electronic System-Level Design; ***FIR*** = Finite Impulse Response; ***FPGA*** = Field-Programmable Gate-Array; ***MAC*** = Muliply and Accumulate; ***PU*** = Processing Unit ***rDPA*** = reconfigurable Data Path Array; ***rDPU*** = reconfigurable Data PathUnit; ***rE*** = reconfigurable Element

[20] W. Roelandts (Keynote-Adresse): FPGAs and the Era of Field Programmability; International Conference on Field Programmable Logic and Applications (FPL), Aug. 29 - Sept 1, 2004, Antwerp, Belgium,

[21] J. Rabaey: Reconfigurable Processing: The Solution to

Low-Power Programmable DSP, Proc. ICASSP 1997
[22] Y. Gu, et al.: FPGA Acceleration of Molecular Dynamics Computations; FCCM 2004 URL: [23]
[23] http://www.bu.edu/caadlab/FCCM05.pdf
[24] A. Alex, J.Rose et al.: Hardware Accelerated Novel Protein Identification; FPL 2004
[25] N. N. Nallatech, press release, 2005
[26] H. Singpiel,C. Jacobi: Exploring the benefits of FPGA-processor technology for genome analysis at Acconovis; ISC 2003, June 2003, Heidelberg, Germany URL: [27]
[27] http://www.hoise.com/vmw/03/articles/vmw/LV-PL-06-03-9.html
[28] N. N. (Starbridge): Smith-Waterman pattern matching; National Cancer Institute, 2004
[29] A. Darabiha: Video-Rate Stereo Vision on Reconfigurable Hardware; Master Thesis , Univ. of Toronto, 2003
[30] R. McCready: Real-Time Face Detection on a Configurable hardware Platform; Master thesis, U. Toronto
[31] T. Fry, S. Hauck: Hyperspectral Image Compression on Reconfigurable Platforms; IFCCM 2002
[32] sP. Buxa, D. Caliga: Reconfigurable Processing Design Suits UAV Radar Apps; COTS Journal, Oct. 2005 URL: [33]
[33] http://www.srccomp.com/ReconfigurableProcessing_UAVs_COTS-Journal_Oct05.pdf
[34] http://helios.informatik.uni-kl.de/RCeducation/
[35] R. Porter: Evolution on FPGAs for Feature Extraction; Ph.D. thesis; Queensland Un. of Technology Brisbane, Australia, 2001 : [36]
[36] http://www.pooka.lanl.gov/content/pooka/green/Publications_files/imageFPGA.pdf
[37] E. Chitalwala: Starbridge Solutions to Supercomputing Problems; RSSI Reconfigurable Systems Summer Institute, July 11-13, 2005, Urbana-Champaign, IL,USA
[38] S. D. Haynes, P. Y. K. Cheung, W. Luk, J. Stone: SONIC - A Plug-In Architecture for Video Processing; FPL 99
[39] M. Kuulusa: DSP Processor Based Wireless System Design; Tampere Univ. of Technology, Publ. No. 296; URL: [40]
[40] http://edu.cs.tut.fi/kuulusa296.pdf
[41] B. C. Schäfer, S. F. Quigley, A. H. C. Chan: Implementation Of The Discrete Element Method Using Reconfigurable Computing (FPGAs); 15th ASCE Engineering Mechanics Conf., June 2-5, 2002, New York, NY, URL: [42]
[42] http://www.civil.columbia.edu/em2002/proceedings/papers/126.pdf
[43] G. Lienhart: Beschleunigung Hydrodynamischer N-Körper-Simulationen mit Rekonfigurierbaren Rechensystemen; Joint 33rd Speedup and 19th PARS Workshop; Basel, Switzerland, March 19 - 21, 2003
[44] R. Hartenstein (invited): The Microprocessor is no more General Purpose; Proc. IEEE International Symposium on Innovative Systems (ISIS), Oct.9-11 1997, Austin, Texas
[45] http://xputers.informatik.uni-kl.de/faq-pages/fqa.html
[46] W. Nebel et al.: PISA, a CAD Package and Special Hardware for Pixel-oriented Layout Analysis; ICCAD 1984
[47] J. Becker et al.: High-Performance Computing Using a Reconfigurable Accelerator; CPE Journal, Special Issue of Concurrency: Practice and Experience, Wiley, 1996
[48] http://xputers.informatik.uni-kl.de/staff/hartenstein/eishistory.html
[49] Rick Kornfeld (personal communication)
[50] http://kressarray.de
[51] R. Kress et al.: A Datapath Synthesis System (DPSS) for the Reconfigurable Datapath Architecture; ASP-DAC 1995
[52] U. Nageldinger et al.: Generation of Design Suggestions for Coarse-Grain Reconfigurable Architectures; International Conference on Field Programmable Logic and Applications (FPL), 28. - 30. August 2000, Villach, Österreich
[53] U. Nageldinger: Coarse-grained Reconfigurable Architectures Design Space Exploration; Dissertation, 2001, URL: [54]
[54] http://xputers.informatik.uni-kl.de/papers/publications/NageldingerDiss.html
[55] http://flowware.net
[56] A. Ast, et al.: Data-procedural Languages for FPL-based Machines; International Conference on Field Programmable Logic and Applications (FPL), 7. - 9. September 1994, Prag, Tschechien
[57] N. Petkov: Systolic Parallel Processing; North-Holland; 1992
[58] H. T. Kung: Why Systolic Architectures? IEEE Computer 15(1): 37-46 (1982)
[59] http://www.organic-computing.de/SPP
[60] J. Becker et al.: Parallelization in Co-Compilation for Configurable Accelerators; Asian-Pacific Design Automation Conference (ASP-DAC), 10. - 13. Februar 1998, Yokohama, Japan
[61] J. Becker: A Partitioning Compiler for Computers with Xputer-based Accelerators, Diss., TU Kaiserslautern 1997, URL: [62]
[62] http://xputers.informatik.uni-kl.de/papers/publications/BeckerDiss.pdf
[63] K. Schmidt: A Program Partitioning, Restructuring, and Mapping Method for Xputers, Diss., Kaiserslautern 1994 URL: [64]
[64] http://www.shaker.de/Online-Gesamtkatalog/Details.asp?ID=930601&CC=41546&ISBN=3-8265-0495-X&Reihe=15&IDSRC=4&LastAction=Search
[65] A. Cantle: Is it time for von Neumann and Harard to Retire?; RSSI Reconfigurable Systems Summer Institute, July 11-13, 2005, Urbana-Champaign, IL,USA
[66] http://bwrc.eecs.berkeley.edu/Research/RAMP/
[67] 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, Croatia
[68] http://flowware.net
[69] M. Herz, et al.: A Novel Sequencer Hardware for Application Specific Computing; Proc. ASAP‘97
[70] M. Herz: High Performance Memory Communication Architectures for Coarse-grained Reconfigurable Computing Systems; Diss. Kaiserslautern, 2001, URL: [107]
[71] http://de.wikipedia.org/wiki/Direct_Memory_Access
[72] S. Heath: Embedded Systems Design, Elsevier, 2003
[73] A. G. Hirschbiel et al.: A Novel Paradigm of Parallel Computation and its Use to Impleent Simple High Performance Hardware; InfoJapan'90 - Int'l Conf. memorating 30th Anniversary, Computer Society of Japan; Tokyo, Japan, 1990
[74] Invited reprint of [73] in Future Generation Computer Systems 7 91/92, p. 181-198, North Holland
[75] W. Nebel et al: Functional Design Verification by Register Transfer Net Extraction from Integrated Circuit Layout Data; IEEE COMPEURO, Hamburg, 1987
[76] F. Catthoor et al.: Data Access and Storage Management for Embedded Programmable Processors; Kluwer, 2002
[77] C. Chang et al: The Biggascale Emulation Engine (Bee); summer retreat 2001, UC Berkeley
[78] H. Simmler et al.: Multitasking on FPGA Coprocessors; International Conference on Field Programmable Logic and Applications (FPL), 28. - 30. August 2000, Villach, Austria
[79] H. Walder, M. Platzner: Reconfigurable Hardware Operating Systems: From Design Concepts to Realizations; ERSA 2003
[80] P. Zipf: A Fault Tolerance Technique for Field-Programmable Logic Arrays; Dissertation, Univ. Siegen, 2002
[81] M. Abramovici, C, Stroud: Improved BIST-Based Diagnosis of FPGA Logic Blocks; IEEE Int'l Test Conf., Atlantic City, Oct. 2000
[82] N. Ambrosiano: Los Alamos and Surrey Satellite contract for Cibola flight experiment platform; Los Alamos National Lab, Los Alamos, NM, March 10, 2004 URL: [83]
[83] http://www.lanl.gov/news/releases/archive/04-015.shtml
[84] M. Gokhale et al.: Dynamic Reconfiguration for Management of Radiation-Induced Faults in FPGAs; Proc. IPDPS 2004
[85] http://www.morphware.org/
[86] http://morphware.net
[87] 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

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

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

[90] http://en.wikipedia.org/wiki/RDPA

[91] 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

[92] http://pactcorp.com

[93] M. Santarini: EDA industry needs a reality check; EDN, 1/19/2006 - URL: [96]

[94] S. Leef: A practical view of ESL design; EE Times, 11/12/2004 - URL: [95]

[95] http://www.eetimes.com/showArticle.jhtml?articleID=52601453

[96] http://www.edn.com/article/CA6298273.html

[97] J. Becker, M. Vorbach: An Industrial/Academic Configurable System-on-Chip Project (CSoC): Coarse.grain XPP/Leon-based Architecture Integration; DATE, Mrch 3.-7, 2003, Munich

[98] http://en.wikipedia.org/wiki/KressArray

[99] K. Compton, S. Hauck: Reconfigurable Computing: A Survey of Systems and Software; ACM Computing Surveys 34/2, June 2002

[100]87. D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R.Thomas, K. Yelick: A Case for Intelligent RAM; IEEE Micro, März / April 1997.

[101]G. Koch et al.: The Universal Bus Considered Harmful; Proc. 1st EUROMICRO Symposium on the Microarchitecture of Computing Systems; June 1975, Nice, France

[102]C. A. R. Hoare: Communicating Sequential Processes, Prentice-Hall, 1985 - URL: [103]

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

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

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

[106]G. Amdahl: Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities; AFIPS 1967

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

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

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

[110]N. N.: SGI ergänzt HPC um RASC - für ´Rekonfigurierbares Anwendungs-Spezifisches Computing'; Munich Sept .2005, - URL: [111]

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

[112]Dietmar Müller: SGI setzt auf rekonfigurierbares Rechnen; ZDNet, 30. September 2005 - URL: [113]

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

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

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

[116]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

[117]http://www.ari.uni-heidelberg.de/

[118]GRAPE Toyoda, S., Mihagawa, H., Kitamura, K., Amisake, T., Hashimoto, E., Ikeda, H., Kusumi, A., and Miyakawa, N. Development of MD engine: High-speed accelerator with parallel processor design for molecular dynamics simulations. J. Comp. Chem. 20, 2 (1999)

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

[120]T. El-Ghazawi, D. Buell, K. Gaj, A. George (tutorial): Reconfigurable High-Performance Computing Seminar; 2005 MAPLD International Conference, September 7-9, 2005, Washington, D.C.

[121]T. El-Ghazawi, D. Buell, K. Gaj, B. Pointer: Reconfigurable Supercomputing; SC|05 Supercomputing, Nov 12-18, 2005, Seattle

[122]K. Schmidt et al.: Automatic Parallelism Exploitation for FPL-based Accelerators; HICSS 1998, Big Island, Hawaii,

[123]J. Becker et al.: A Partitioning Programming Environment for a Novel Parallel Architecture; Proc. IPDPS '96 - URL: [124]

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

[125]R. Kress et al.: An Embedded Accelerator for Real-Time Image Processing; 8th Euromicro Workshop on Real-Time Systems; Juni 1996, L'Aquila, Italien

[126]K. Kennedy, R. Allen: Optimizing Compilers for Modern Architectures: A Dependence-based Approach; Academic Press, 2002

[127]N. N.: Cray and Celoxica Make Reconfigurable Computing Easier to Program; Cray, Inc., Sept 6, 2005, URL: [129]

[128]MAC = Muliply and Accumulate; DSP = Digital Signal Processing; FIR = Finite Impulse Response

[129]http://investors.cray.com/phoenix.zhtml?c=98390&p=irol-newsArticle_print&ID=752700&highlight=

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

[131]N.N.: W2.All.Y1Guidelines f. Graduate Curriculum on Embedded Softw. and Systems; ARTIST Consortium 2003; URL: [132]

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

[133]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 - URL: [134]

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

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

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

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

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