1. Basics of Reconfigurable Computing

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

Already decades ago the instruction-stream-based von Neumann (vN) paradigm [1] [2] lost its dominant role as the basic common model of computing systems. After many technology generations the microprocessor CPU, together with its software still the monopoly-like main focus point of CS education, often cannot meet even usual performance requirements. In most PCs it cannot even drive its own display: it needs a graphics accelerator. The vN paradigm as a common model has been replaced by a symbiosis of CPU and hardwired accelerators (figure 1), not only for embedded systems. Most MIPS equivalents have been migrated from software to accelerators: software to ASIC migration [3], and more and more software to configware migration to be run on FPGAs (Field-Programmable Gate Arrays). The microprocessor has become the tail wagging the dog [4]. The basic model of most accelerators is data-stream-based (figure 1, however, it is not a „dataflow machine“). It is not instruction-stream-based. For detailed explanations on this duality of paradigms see section 1.2.

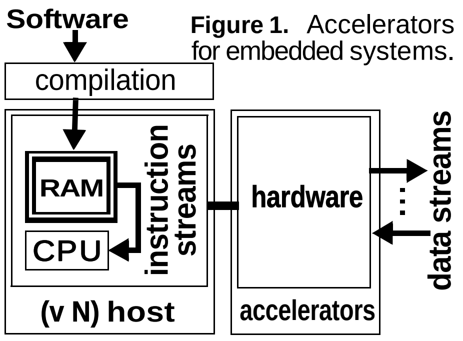


Figure 1. Accelerators for embedded systems.

Accelerator design became difficult. But soon also the design of accelerators has become increasingly difficult: the second VLSI design crisis (figure 2). With technology progress the mask cost and other NRE cost are rapidly growing (figure 2 a). Also the design cost is rapidly increasing, made worse by decreasing product life cycle length (figure 2 b). This explains the decreasing number of wafer starts (figure 2 c) as well as the rapid growth of the number of design starts based on FPGAs instead of ASICs (figure 2 d, [5]): the next paradigm shift, where the accelerator has become reconfigurable (figure 3), so that the product life cycle can be extended by upgrading the product, even at the customers site by remote reconfiguration (figure 2 e). Now both paradigms are RAM-based: the instruction-stream-based CPU, and the data-stream-based FPGA or rDPA (reconfigurable DataPath Array, see section 1.4). The accelerator's RAM is no block RAM: it's hidden inside the reconfigurable interconnect fabrics: hRAM.

Most important silicon revolution after introduction of the microprocessor. Such Reconfigurable Computing (RC) [4] [6] [7] [8] [9] is the most important revolution to silicon application after the introduction of the microprocessor [10]. It's the most important new machine paradigm with common model capability [11] [12] [13] [14] [15] [16]. Emerging in the 80ies and now having moved from a niche market to mainstream we have with FPGAs a third class of platforms filling the gap between vN-type procedural compute engines and ASIC application-specific hardware [17]. FPGAs are the fastest growing segment of the semiconductor market.

Disruptive methodology. When having been intel CEO Andy Grove claimed, that each technology providing a factor of 10 or more improvements over an established one, can be expected to become disruptive [18]. From CPU software to FPGA configware migrations speedup factors by up to more than 3 orders of magnitude have been published (fig. 4). This massively disruptive community is reporting a factor of 7.6 in accelerating radiosity calculations [19], a factor of 10 for FFT (Fast Fourier Transform), a speedup factor of 35 in traffic simulations [20]. For a commercially available Lanman/NTLM Key Recovery Server [21] a speedup of 50 - 70 has been reported. Another cryptology application reports a factor of 1305 [23]. A speedup by a factor of 304 is reported for a R/T spectrum analyzer [25]. In the DSP area [26] for MAC [26] operations a speedup factor of 100 has been reported compared to the fastest DSP on the market (2004) [27]. Already in 1997 versus the fastest DSP a speedup between 7 and 46 has been obtained [28]. In Biology and genetics (also see [29] [30]) a speedup of up to 30 has been shown in protein identification [31], by 133 [32] and up to 500 [33] in genome analysis, as well as 288 with the Smith-Waterman pattern matching algorithm at the National Cancer Institute [35]. In the multimedia area we find factors ranging from 60 to 90 in video rate stereo vision [36] and in real-time face detection [37], and of 457 for

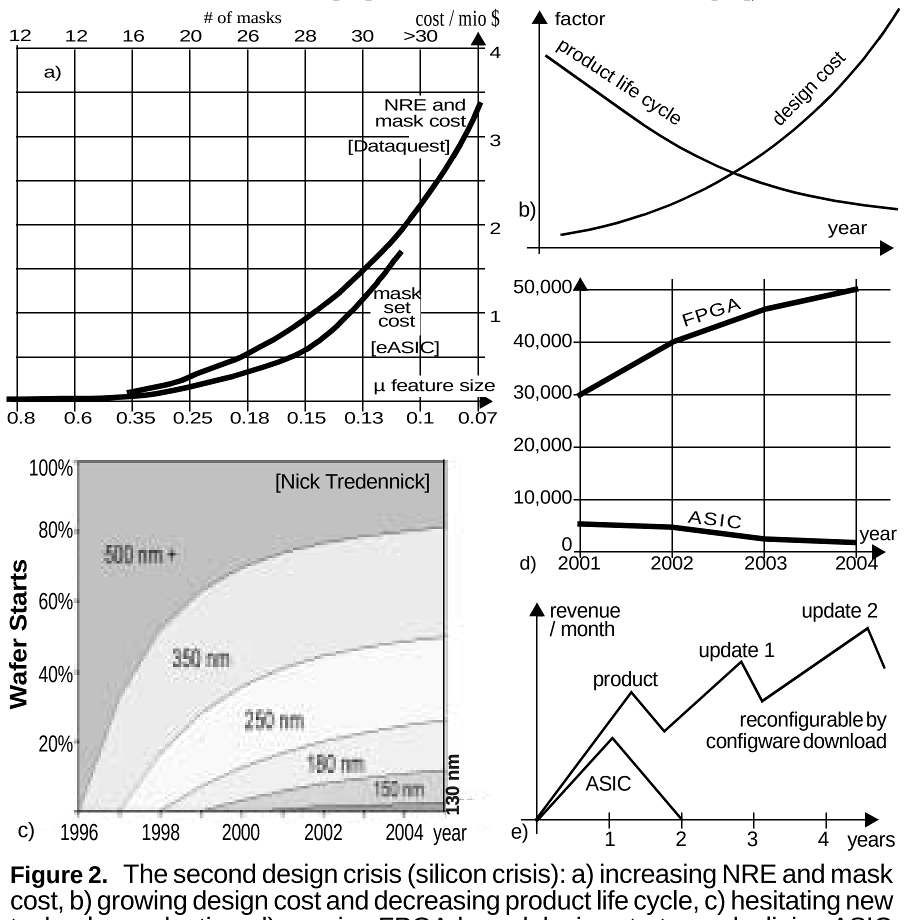


Figure 2. The second design crisis (silicon crisis): a) increasing NRE and mask cost, b) growing design cost and decreasing product life cycle, c) hesitating new technology adoption, d) growing FPGA-based design starts vs. declining ASIC design starts [5] [90], e) reconfigurability extends product life cycles.

hyperspectral image compression [38]. In communication technology we find a speedup by 750 for UAV radar electronics [39]. These are just a few examples from a wide range of publications [41] [42] [44] [45] [46] [48] [50] reporting substantial speedups by FPGAs. Fortunately, in embedded systems, highest computational requirements are determined by a small number of algorithms, which can be rapidly migrated via design automation tools [51].

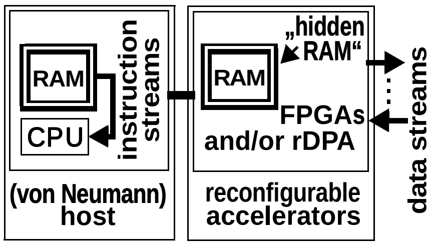


Figure 3. FPGAs replacing ASICs.

The 1st Reconfigurable Computing Paradox. These disruptive performance results are a surprise. the 1st Reconfigurable Computing Paradox [53] [54] [55] [56] [57]. The technology parameters are so bad that Andy Grove’s rule is missed by several orders of magnitude. The effective integration density really being available to the FPGA application is much worse, than this physical integration density. It is behind the Gordon Moore curve by about 4 orders of magnitude (figure 5). Wiring overhead costs about 2 orders of magnitude and reconfigurability overhead costs about another two orders of magnitude (only about one out of about 100 transistors. serves the application, whereas the other 99 transistors deserve reconfigurability). Another source of area inefficiency of FPGAs is routing congestion, which could cost a factor of two or more: not all desired connections can be routed because the supply of routing resources is exhausted. It is astonishing, how with such a bad technology such brilliant acceleration factors are obtained. For the explanation of most speed-up mechanisms involved see section 1.4, since this is more easy at the higher abstraction level given by coarse-grained reconfigurable computing.

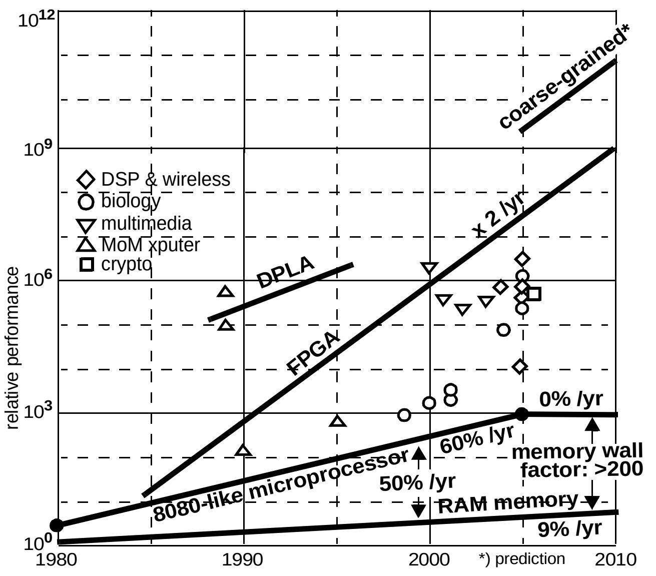


Figure 4. speed-up by software to configware migration.

Earlier alternatives. Let us have a brief look onto earlier 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 [58], and between 2,300 and 15,000 for grid-based VLSI design rule check [4] [59] [60] [61]. 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 Mead-&-Conway-style multi-university VLSI project E.I.S [62]. 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.

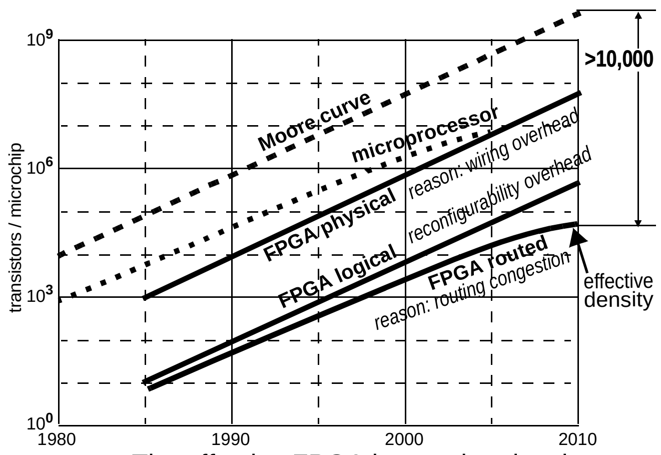


Figure 5. The effective FPGA integration density.

FPGAs became mainstream. Already many years ago FPGAs have become mainstream throughout all embedded systems areas (fig. 6 a). More recently, the FPGA-based pervasiveness of Reconfigurable Computing (RC) also spread over all application areas of scientific computing (a few example are listed in fig. 6 b). As a side effect in addition to speed-up factors of disruptive dimensions also a drastic reduction of the electric energy budget is provided down until about 10% [63] - along with a reduction of equipment cost by a similar dimension [63]. This has been reported from the supercomputing community. So far the immense electricity consumption has been considered one of the most severe obstacles on the classical way to the petaflop supercomputer. This side effect [64] extends the scope of the low power design community [65] [66] [67] [68] beyond dealing with devices powered by a tiny battery.

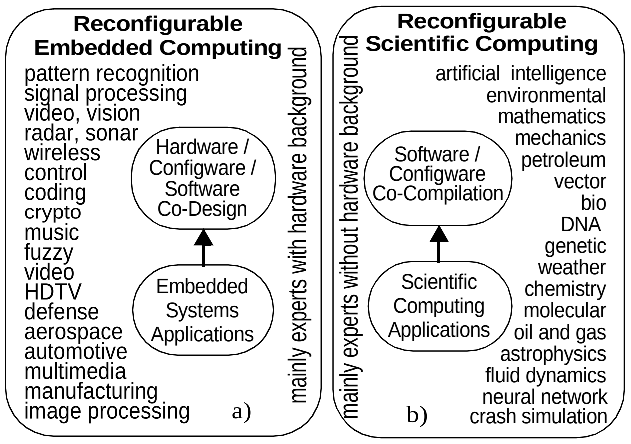


Figure 6. Two different reconfigurable computing cultures

Low power design. As leakage power and total power became a dramatic issue in very deep submicron technologies, the low power design community started in 1991 to take care of these power problems. As a spin-off of the PATMOS project funded by the Commission of the European Union [70] the annual PATMOS conference was founded as the first series on this topic area world-wide [71]. (PATMOS stands for Power and Timing Modelling, Optimization and Simulation.) Its annual sister conference series ISLPED (Int'l Symp. on Low Power Electronics and Design) has been founded four years later in the USA [72]. The low power design community started exploring new design methodologies for designing leakage tolerant digital architectures, based on architectural parameters like activity, logical depth, number of transitions for achieving a given task and total number of gates. An early proposed design method selects the best architecture out of a set of architectures (baseline, sequential, parallel, pipelined, etc.) at optimal Vdd and threshold voltages. Another design method takes as constraints given Vdd and threshold voltages.

Hardware design on a strange platform? 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. But now we have 2 reconfigurable computing scenes (fig. 6). Meanwhile FPGAs are also used everywhere (even in oil and gas [73]) for high performance in scientific computing. This is really a new computing culture - not at all a variety of hardware design. Instead of HS codesign (figure 1) we have here software / configware co-design (S/C co-design) (figure 3), 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, for running legacy code: a CPU co-processor serving to a reconfigurable main processor.

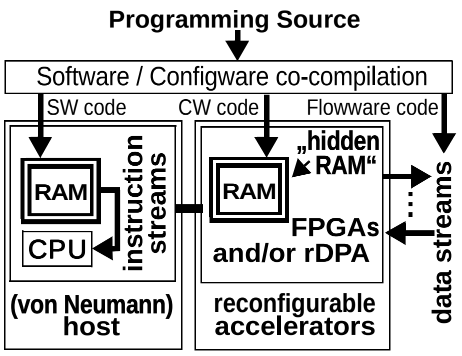


Figure 7. The dual paradigm model.

Educational deficits. 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 [75] [76] [77]. 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.

Unthinkable without FPGAs. The area of embedded systems is unthinkable without FPGAs [78]. This has been the driving force behind the commercial break-through of FPGAs. Almost 90% of all software is implemented for embedded systems [79] [81] [82] 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 such a dichotomic qualification of our graduates, urgently 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.

FPGAs and EDA. The pervasiveness of FPGAs also reaches the EDA (Electronic Design Automation) industry, where all major firms spend a substantial effort to offer a variety of application development tools and environments for FPGA-based product development [84]. Also FPGA vendors have EDA development efforts and cooperations with firms in the EDA industry and offer such tools and development environments. Since this is a highly complex market, this chapter does not go into detail because of a lack of space.

The Kress-Kung machine. 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 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* [85], which, however, is data-stream-based, and 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 7d). FPGAs or, the Kress-Kung machine, respectively, has *no „instruction fetch" at run time* (fig. 8). 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.

1.1. Configurable Cells

Since their introduction in 1984, *Field-Programmable Gate Arrays (FPGAs)* have become the most popular implementation media for application-specific or domain-specific digital circuits. For a reading source on the role of FPGAs (providing 148 references) also see [78]. The technology-driven progress of FPGAs (for key issues see [86] [87]) is faster than that of microprocessors (fig. 5). FPGAs with 50 million system gates are coming, may be, soon [88]. The FPGA is an array of gate level configurable logic blocks (CLB) embedded in a reconfigurable interconnect fabrics [87]. Its *configwar*e code (reconfiguration code [93]: fig 10) is stored in a distributed RAM memory. We may also call it *hRAM* for *„hidden RAM*", because it is hidden in the background of the FPGA circuitry.

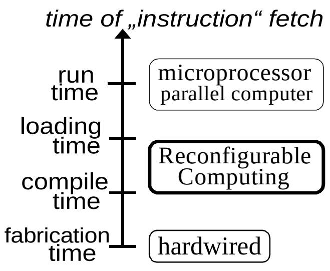


Figure 8. "Instruction Fetch"

Hidden RAM. The little boxes labelled „FF" in figure 9 are bits of the hRAM. At power-on the FPGA is booted by loading the configware code down to the hRAM, mostly from an external flash memory. Usually the FPGA includes the booting interface needed. There are efforts on the way to have the booting flash memory direct on the FPGA chip, or, to use flash technology directly for the hRAM [89], currently is less area-efficient than CMOS hRAM.

Fastest growing market segment. FPGAs are with 6 billion 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 to grow from 80.000 in 2006 to 115.000 in 2010 [90].

Two classes of reconfigurable systems. We may distinguish two classes of reconfigurable systems: *fine grain* reconfigurable systems, and, *coarse grain* reconfigurable systems (see section 1.4). Reconfigurability of fine granularity means, that the functional blocks have a datapath width of about only one bit. This means, that programming at low abstraction level is logic design. Practically all products on the market are *FPGAs* (field-

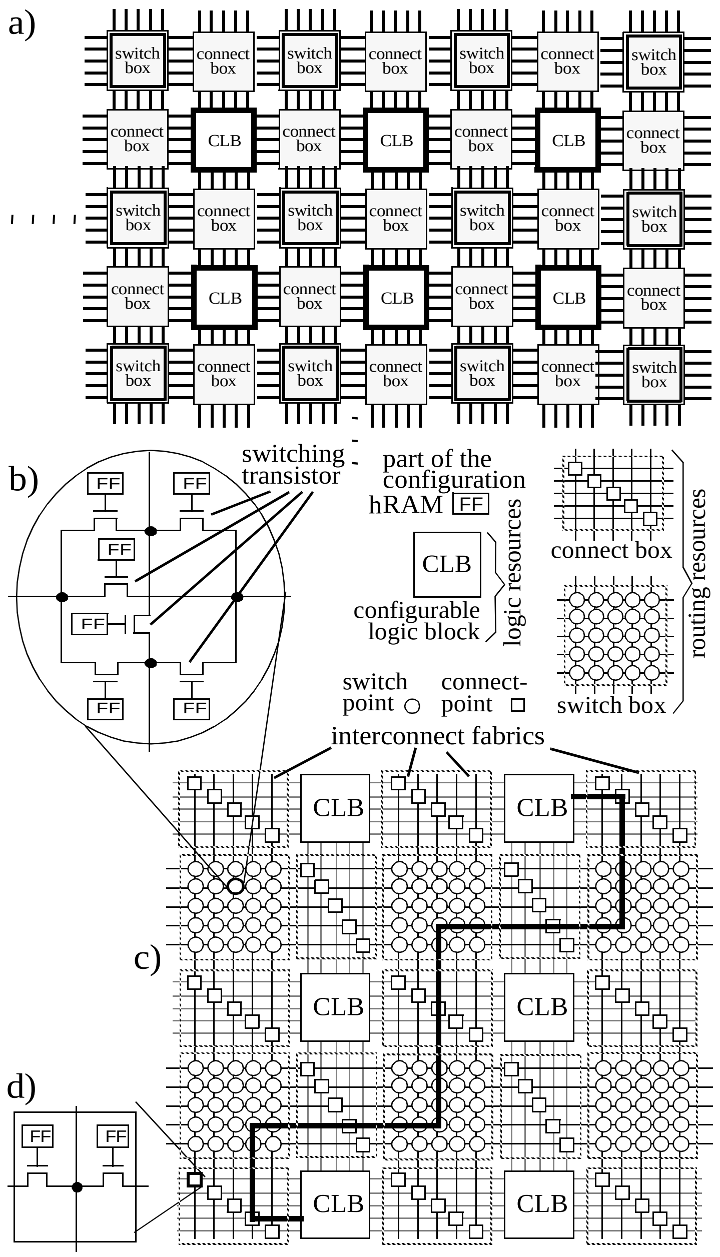


Figure 9. Illustrating FPGA island architecture fine grain reconfigurable resources: a) global view, b) switch point circuit of a switch box, c) FPGA detailed view (only one configured “wire” shown), d) „connect point“ circuit of a „connnect box“.

programmable gate arrays), although some vendors prefer different terms as a kind of brand names, like, for instance, PLD (Programmable Logic Device, or rLD for reconfigurable logic device). Reconfigurable platforms and their applications have undergone a long sequence of transitions.

FPGAs replacing ASICs. First FPGAs appeared as cheap replacements of *MPGAs* (or *MCGAs*: Mask-Configurable Gate Arrays). Still to-day FPGAs are the reason for shrinking ASIC markets (figure 2 c), since for FPGAs no application-specific silicon is needed - a dominating cost factor especially in low production volume products. Although being cheaper than for fully hardwired solutions the ASIC fabrication cost (only a few specific masks needed) is still higher than for FPGA-based solutions. Later the area proceeded into a new model of computing possible with FPGAs. Next step was making use of the possibility for debugging or modifications the last day or week, which also lead to the adoption by the *rapid prototyping* community which also has lead to the introduction of *ASIC emulators* faster than simulators. Next step is direct *in-circuit execution* for debugging and patching the last minute.

Brand names, like, for instance, PLD (Programmable Logic Device) are often confusing. Reconfigurable platforms and their applications have undergone a long sequence of transitions. First FPGAs appeared as cheap replacements of *MPGAs* (or *MCGAs*: Mask-Configurable Gate Arrays). Still to-day FPGAs are the reason for shrinking ASIC markets (figure 2 c), since for FPGAs no application-specific silicon is needed - a dominating cost factor in low production volume products. Before FPGAs have been available, ASICs have been used to replace fully hardwired solutions (full mask set needed), because fabrication cost has been lower (only a few specific masks needed). Later the area proceeded into a new model of computing possible with FPGAs. Next step was making use of the possibility for debugging or modifications the last day or week, which also lead to the adoption by the *rapid prototyping* community which also has lead to the introduction of *ASIC emulator,* expensive (million-dollar range) and bulky machines filled with hundreds or thousands of FPGAs, used as accelerators for simulators used in traditional EDA environments. Because of high cost and poor performance the time for the ASIC emulator has been over around the end of last century [5]. Because of their enormous capacity and interesting built-in features meanwhile FPGAs directly offer a viable alternative for inexpensive FPGA-based prototypes.

Terminology problems. A fine term would be „morphware" [91], which, however, is protected [92]. The historic acronyms *FPGA* and *FPL* are a bad choice, since

platform		program source	machine paradigm
hardware		not programmable	(none)
recon-figurable^1	fine-grained reconfigurable	***configware***	
	coarse-grain reconfigurable array (data-stream-based)	***configware & flowware***	anti machine
hardwired processor	data-stream-based computing	***flowware***	
	instruction-stream-based computing	software	von Neumann

1) a good general term would be ***morphware*** [91], which, however, is a registered trade mark [92]

igure 10. Terminology (also see Figure 9.Illustrating FPGA island architecture

„programming“, i. e. scheduling, is a *procedural* issue in the *time domain*. For the same reason also the term *PLD* is a bad choice and should be replaced by *rLD (reconfigurable Logic Device)*. A program determines a *time sequence* of executions. In fact the FP in FPGA and in FPL, acronym for *field-programmable*, actually means field-reconfigurable, which is a structural issue *in the space domain: configuration in space*. For terminology also see figure 24.

Island architectures. Most important architectural classes of FPGAs are [94]: island architectures (Xilinx), hierarchical architectures (Altera), and row-based architectures (Actel). A more historic architecture is *mesh-connected*, sometimes also called *sea of gates* (introduced by Algotronix) [95]. For a survey on FPGA architectures see [96]. For illustration of FPGA fundamentals we use the historic simple island architecture as an example (figure 9). Under the term „simple FPGA“ we understand a pure FPGA, which unlike modern „platform FPGAs“, do not include non-reconfigurable resources, like adders, memory blocks etc.

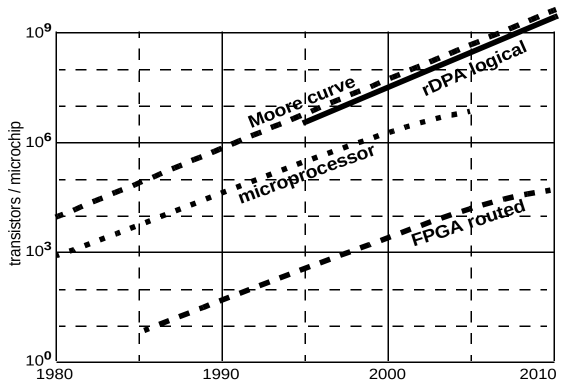


Figure 11. Area efficiency of coarse-grained RC.

An island style example. A simple island style FPGA is a mesh of many CLBs (configurable logic blocks), embedded in a reconfigurable interconnect fabrics (figure 9 a). Most CLBs are LUT-based, where LUT stands for „look-up table“. A simple example of a CLB block diagram is shown by figure 12. Its functional principles by multiplexer implementation of the LUT are shown by figure 13 a and b, where in CMOS technology only 12 transistors are needed for the fully decoded multiplexer (figure 13c). The island architecture is illustrated by figure 9 a. Fig. 9 b shows details of *switch boxes* and *connect boxes* being part of the reconfigurable interconnect fabrics. Fig. 9 c shows the circuit diagram of a *cross point* in a switch box, and, figure 9 d from within a connect box. The thick wire example in figure 9 b illustrates, how these interconnect resources are configured to connect a pin of one CLB with a pin of another CLB The total configuration of all wires of an application is organized by a *placement and routing* software. Sometimes more interconnect resources are needed than are available, so that for some CLB not all pins can be reached. Due to such *routing congestion* it may happen, that a percentage of CLBs cannot be used.

Fine grain reconfigurable systems products. A wide variety of fine-grained reconfigurable systems products [94] [95] is available from a number of vendors, like the market leader Xilinx [97], the second largest vendor Altera [98], and many others. Also a variety of evaluation boards and prototyping boards is offered. COTS (commodity off the shelf) boards for FPGA-based developments are available from Alpha Data, Anapolis, Celoxica, Hunt, Nallatech, and others, to support a broad range of in house developments. As process geometries have shrunk into the deep-submicron region, the logic capacity of FPGAs has greatly increased, making FPGAs a viable implementation alternative for larger and larger designs. Deep submicron FPGAs are available in many different sizes and prices per piece ranging from 10 US-Dollars up to FPGAs with much more than a million usable gates for more than 1000 US-dollars. E. g. Xilinx offers the Virtex-4 multi-platform FPGA family on 90nm process technology, the 550 MHz Virtex-5 family of FPGAs on 65nm technology providing four platforms (high-performance logic, serial connectivity, signal processing, and embedded processing) and has pre-announced FPGAs with 50 million system gates for about 2005 [88]. Modern FPGAs support mapping entire systems onto the chip by offering on board all components needed, like several memory banks for user data, one, or several microprocessors like ARM, PowerPC, MIPS, or others, a major number of communication interfaces (WAN, LAN, BoardAN, ChipAN etc.) supporting contemporary standards, up to several GHz bandwidth, JTAG boundary scan circuitry to support testing, sometimes even multipliers. Low-cost FPGA development boards are available for universities [99]. A base version board is priced at $90 US and contains a 400,000-gate Xilinx Spartan-3 FPGA. Other boards are available with up to 2 million gates. Each board comes bundled with free development software.

Low power dissipation and radiation tolerance. Also FPGAs featuring low power dissipation [66] or better radiation tolerance (for aerospace applications) are offered. Several major automotive corporations contracted FPGA vendors to develop reconfigurable devices optimized for this branch of industry. Some FPGAs commercially available also support partial column wise reconfiguration, so that different talks may reside in it and may be swapped individually. This may also support *dynamic reconfiguration* (*RTR*: run time reconfiguration), where some tasks may be in the execution state, whereas at the same time other tasks are being reloaded. Dynamic reconfiguration, however, tends to be tricky and difficult to understand and to debug. But static reconfiguration is straight forward and more easy to understand. Because reconfiguration is slow, also multi-context reconfigurable systems has been discussed, but is not yet available commercially. Multi-context reconfigurable systems features several alternative internal reconfiguration hRAM memory banks, for example 2 or 4 banks, so that reconfiguration can be replaced by an ultra fast context switch to another hRAM memory bank.

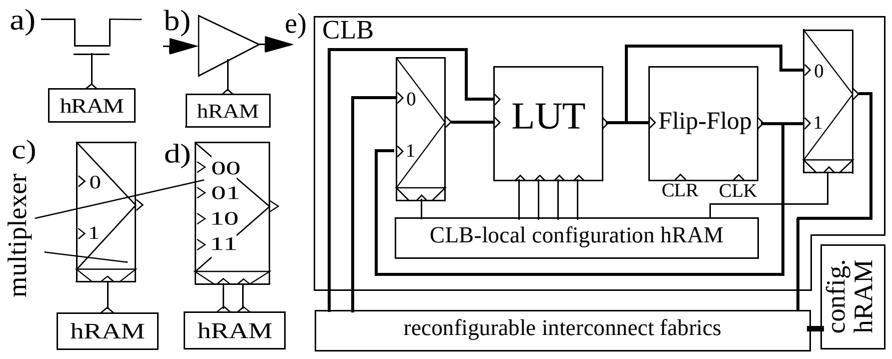
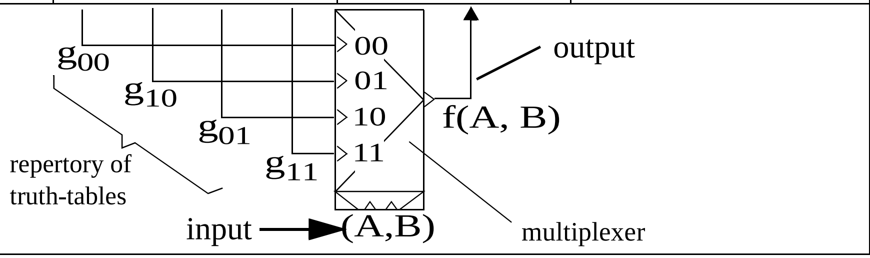


Figure 12. Reconfiguration resources in FPGAs: a) pass transistor, b) tri-state buffer, d) 2-way multiplexer, d) 4-way multiplexer, e) simplified CLB example.

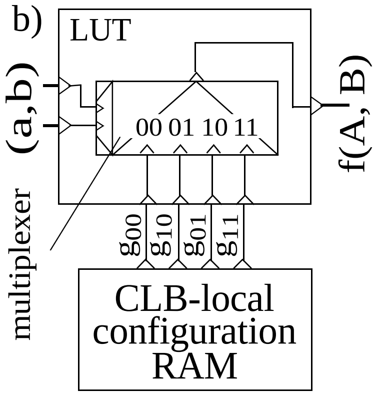
1.2. von Neumann vs. Reconfigurable Computing Paradigm

It is well known that the growth rate of the integration density of microprocessors is slower than Moore's law. Because of the high degree of layout regularity the integration

a)	configuration bits g_{ab}				f(A, B)	function
#	g_{00}	g_{01}	g_{10}	g_{11}		
0	0	0	0	0	0	constant 0
1	0	0	0	1	A and B	and
2	0	0	1	0	B disables A	if B then 0 else A
3	0	0	1	1	A	identity A
4	0	1	0	0	A disables B	if A then 0 else B
5	0	1	0	1	B	identity B
6	0	1	1	0	A exor B	antivalence
7	0	1	1	1	A or B	or
8	1	0	0	0	not(A or B)	nor
9	1	0	0	1	A coin B	equivalence
10	1	0	1	0	not(B)	negation of B
11	1	0	1	1	B implies A	if A then 1 else ¬ B
12	1	1	0	0	not(A)	negation of A
13	1	1	0	1	A implies B	if B then 1 else ¬ A
14	1	1	1	0	not(A and B)	nand
15	1	1	1	1	1	constant 1



b)



c)

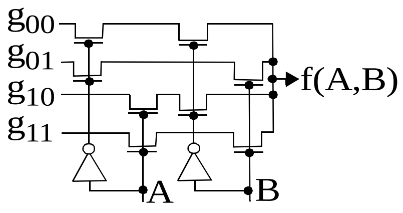


Figure 13. Illustrating LUT implementation by multiplexer: example for functions of 2 variables: a) illustration of the function generator, b) illustration of LUT (look-up table) block use within CLB (compare figure 12 e), c) multiplexer circuit.

density of FPGAs, however, is going roughly by the same speed as with Moore's law [4] (figure 5). But because of the high percentage of wiring area the transistor density of FPGAs are behind the Gordon Moore curve two orders of magnitude [4]. However, the number of transistors per chip on FPGAs has overhauled that of microprocessors already in the early 90ies and now is higher by two orders of magnitude [4].

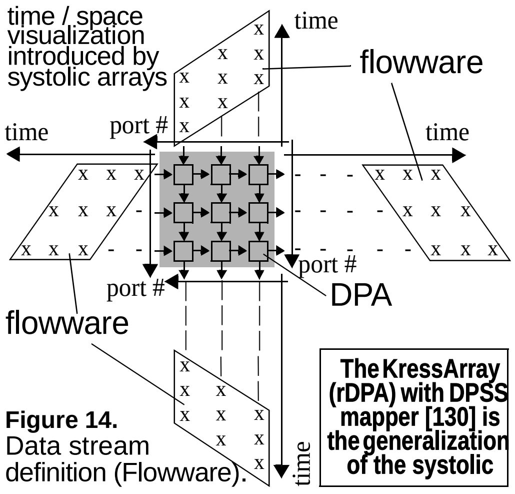

time / space visualization introduced by systolic arrays
time
flowware
time
port #
time
port #
port #
DPA
flowware
time
The KressArray (rDPA) with DPSS mapper [130] is the generalization of the systolic

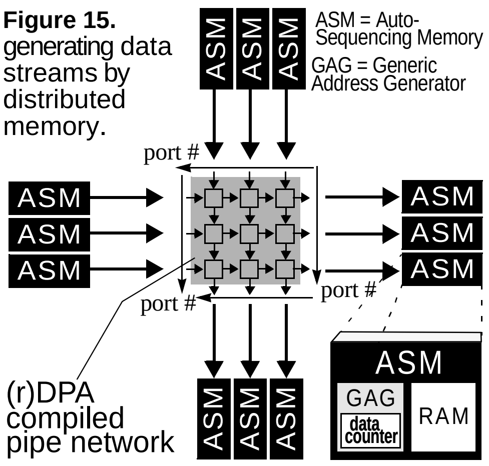
Figure 14. Data stream definition (Flowware).

The end of Moore's Law. Increasing the architectural complexity and the clock frequency of single-core microprocessors has come to an end (e. g. see, what has happened to the intel Pentium 4 successor project). Instead, multi-core microprocessor chips are emerging from the same vendors (e. g. AMD: 32 cores on a chip by 2010 [101]). But just more CPUs on the chip is not the way to go for very high performance. This lesson we have learnt from the supercomputing community paying an extremely high price for monstrous installations by having followed the wrong road map for decades. Such fundamental bottlenecks in computer science will necessitate new breakthroughs [103]. Instead of hitting physical limits we found, that further progress is limited by a fundamental misconception in the theory of algorithmic complexity [104]. *Not processing* data is costly, *but moving* data. We have to rethink the basic assumptions behind computing. Such a change should take place, when the old paradigm is in crisis and cannot explain compelling new facts, like recently illustrated by the reconfigurable computing paradox. But this is not the first crisis of the old paradigm [105] [106]. However, the substantial investments of the past decades in the old paradigm causes resistance to a paradigm change, because knowledge accumulated up to that point may loose its significance ([107] as quoted by [108]). Those who have a vested interest in the old paradigm will always attempt to extend that paradigm to accommodate new facts. Thus the final victory of the new paradigm can only be guaranteed by a generation change [108]. But research on Innovation is less pessimistic [109].

Transdisciplinary Approaches needed. Instead of the traditional reductionism we need transdisciplinary approaches, such as heralded by the current revival of Cybernetics, e. g. labelled as Integrated Design & Process Technology [110], or Organic Computing [111]. To reanimate the stalled progress in HPC for a break-through in very high performance computing we need a transdisciplinary approach for bridging the hardware / software chasm, which meanwhile has turned into a configware / software chasm. For much more successful efforts we need a transdisciplinary paradigm shift, over to a new fundamental model, such as available from the Reconfigurable Computing community dealing with configware engineering as a counterpart to software engineering.

Classical parallelism does not scale - [101] [104] a very expensive lesson which we have already learnt from the supercomputing community with massively increasing the number of processors when going cheap COTS (commodity off the shelf). With the growing degree of parallelism, the programmer productivity goes down drastically („The Law of More“ [113]). It is an illusion to believe, that scalability would get massively better, when all these processors will be resident on a single chip - as long as the reductionistic monopoly of the von Neumann mind set will not be relieved, where the classical fundamental paradigm is still based on concurrent sequential processes and message passing through shared memory, both being massively overhead-prone and extremely memory-cycle-hungry. Rescue should not be expected from threads, although intel pre-announced some tools intended to avoid, that the programmers shy away. In his cover feature article [114] Edward A. Lee from UC Berkeley claims, that for concurrent programming to become mainstream, we must discard threads as a programming model. Nondeterminism is the overhead-prone problem, not only hidden behind methodologies attributed „speculative“. By the way, complications by threads perfectly illustrate the von-Neumann-based software paradigm trap.

Escape the Software Development Paradigm Trap, said IRIS director Mark Bereit [116], who refutes the assumption that software development will always be difficult and bug-ridden, noting that this is due „solely to the software development paradigm that we've followed, unchallenged, for decades". One of the consequences of this paradigm trap crisis is demonstrated by (not only) the history of programming languages [117] [119], where 2500 different languages are seen as counted by Bill Kinnersley [120]. Diarmuid Piggott counts the total even higher with more than 8,500 programming languages [121]. What an immense waste of manpower to make sure to stay caught inside the paradigm trap.



ASM = Auto-Sequencing Memory
GAG = Generic Address Generator
ASM
port #
(r)DPA compiled pipe network
GAG
data counter
RAM

Figure 15. generating data streams by distributed memory.

Bad scalability. The software development paradigm is also the reason of bad scalability and bad programmer productivity in classical parallelism. Mark Bereit proposes reworking the model and studying other engineering disciplines for inspiration. He proposes to study mechanical engineering. But much better is studying Reconfigurable Computing. Tensilica senior vice president Beatrice Fu said, that reconfigurable computing offers the option of direct processor-to-processor communications without going through memory nor through a bus. This paradigm shift is an old hat, but until recently mostly ignored, not only by the supercomputing community. Buses cause multiplexing overhead [123] and the dominating instruction-stream-based-only fundamental model is extremely memory-cycle-hungry [125]: the reason of the „memory wall“. The alternative offered by reconfigurable computing is data stream parallelism by highly parallel distributed fast local memory.

More simple memory parallelismand more straight forward than e. g. interleaved memory access known from vector computers. Crooked Labelling. The difference between

Parallel Computing and Reconfigurable Computing is often blurred by projects labelled "reconfigurable", which, in fact are based on classical concurrency on a single chip. To avoid confusion: switching the multiplexers or addressing the registers at run time is not „reconfiguration“. At run time, real Reconfigurable Computing never has an instruction fetch: only data streams are moving around (This should not be confused with dynamically reconfigurable systems: a mixed mode approach switching back and force between reconfiguration mode and execution mode, which should be avoided for introductory courses.)

The data stream machine paradigm has been around 3 decades as a niche. Software used it indirectly by inefficient instruction-stream-based implementations.

CPUs outperformed by FPGAs. The world-wide total running compute power of FPGAs outperforms that of CPUs. Most total MIPS running worldwide have been migrated from CPUs to accelerators, often onto FPGAs. The FPGA market with almost 4 billion US-Dollars (2006) is the fastest growing segment of the integrated circuit market. Gartner Dataquest predicts almost 7 billion US dollars for the year 2010. Xilinx and Altera currently dominate this market with a share of 84%.

Bad Parameters. However, FPGAs have even more comparably bad parameters: the clock frequency is substantially lower than that of microprocessors, FPGAs are power-hungry and expensive. We would like to call it the „Reconfigurable Computing Paradox“, that despite of these dramatically worse technology parameters such enormous speed-up factors can be obtained by software to configware migration (figure 4). The explanation of this paradox is the machine paradigm shift coming along with such a migration.

The third Reconfigurable Computing Paradox. It may be called the third paradox of Reconfigurable Computing, that despite of its enormous pervasiveness, most professionals inside computer Science and related areas do not really understand its paradigm shift issues. This massively affects implementer productivity. 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.

Gradually wearing off. Currently the dominance of the basic computing paradigm is gradually wearing off with growing use 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 [128]. To obtain the payoff from RC we need a new understanding of computing and supercomputing. To bridge the translational gap, the software / configware chasm, we need to think outside the box.

Going toward the dual paradigm mind setis the current strong trend (figure 7): the duality of the instruction-stream-based CPU paradigm, and its counterpart, the data-stream-based anti machine paradigm based on data counters instead of a program counter. The von Neumann paradigm using a program counter [1] [2] will not be explained by this chapter. Since about the early 80ies one of the two the roots of the anti machine paradigm has been hidden inside the systolic array community [128] [129], mainly a group of mathematicians, who have nicely defined the concept of „data streams“, coming along with a beautiful visualization by time/space diagrams (see fig 14). Systolic arrays have been popularized in 1981 by H. T. Kung [129]. But for mathematicians a systolic array has only been a data path, a pipe network, but it has not been a computational machine, because the sequencer has been missing. But by the mathematicians, development of a methodology for generating these data streams at run time has been considered being somebody else’s job. Refusing a transdisciplinary perspective this has been a typical result of the „formal“ reductionism having been fashionable at that time.

Mathematicians’ paradigm trap. The classical systolic array suffered from a severe restriction. It could be used only for applications with strictly regular data dependencies. For a hammer many things in the world look like a nail. About 25 years ago for the mathematicians working on systolic arrays all applications looked like algebraic problems. From this point of view their algebraic synthesis algorithms generating a systolic architecture from a mathematical formula have been based only on linear projections, which yield only uniform arrays with linear pipes: usable only for applications regular data dependencies.

Generalization of the systolic array. Later Rainer Kress holding an EE degree discarded the mathematician’s synthesis algorithms and used simulated annealing instead, for his DPSS (Data Path Synthesis System) [130]. This means the generalization of the systolic array: the KressArray [131], or super-systolic array (we may also call it Kress-Kung-Array), also supporting any wild forms of pipe networks, including any non-linear pipes like spiral, zigzag, branching and merging, and many other forms. Now reconfigurability makes sense. Fig. 15 shows how the sequencing part is added to the systolic array data paths, so that we get a complete computational machine. Instead of a program counter we have multiple data counters supporting parallelism of data streams. The GAG (generic address generator, [132] [134] [135]) is a data sequencer connected to a RAM block for data storage. GAG and RAM block are parts of the ASM (Auto-sequencing Memory), a generalization of the DMA (Direct Memory Access). The following paragraphs explain the machine principles and software / configware co-compilation techniques for the duality of machine paradigms.

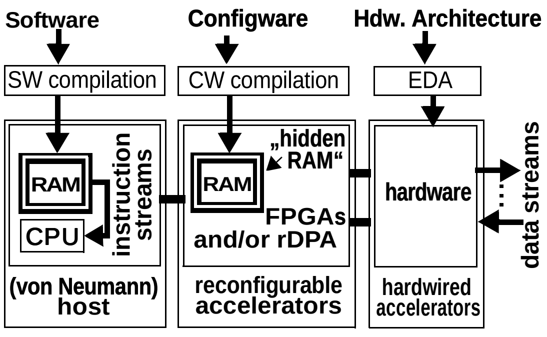


Figure 16. De facto common model of contemporary computing systems: von-Neumann-only is obsolete.

Discussing Software Engineering vs. Configware Engineering. In total we have 3 different kinds digital subsystems (fig 16) and 3 different kinds of programming sources (fig 16). The dual-paradigm model can be illustrated by contrasting via Nick Tredennick’s model of computer history (fig 17). The algorithm of early machines like the Digital Differential Analyzer (DDA) could not be programmed: no source code was needed (fig 17 a). With the von Neumann’s classical software processor only the algorithm is variable, whereas the resources are fixed (hardwired), so that only one type of source code is needed: software (fig 17 b), from which the compiler generates software machine code to be downloaded into the *instruction RAM* - the instruction schedule for the *software processor* (fig 17 b). 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 sources (fig 18): *Configware* and *Flowware* (fig 17 c):

1) Configware [93] deserves structural programming of the resources by the *„mapper“* using *placement and routing* or similar mapping methods (for instance by simulated annealing [131] [130] [137] [138] [139]) (fig 19 b).

2) Flowware [141] deserves programming of the data streams by the *„data scheduler“* (fig 19 b), which generates the flowware code needed for downloading into the generic address generators (GAG) within the ASM auto-sequencing memory blocks (fig 15).

The dichotomy of language principles. These two different fundamental machine principles, von Neumann software machine vs. the Kress-Kung machine, the configware machine, are contrasted by the dichotomy of languages (fig. 23) and by fig. 18 [56] [68]. Flowware Languages are easy to implement [141] [142]. A comparison with software programming languages is interesting [87]. 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".

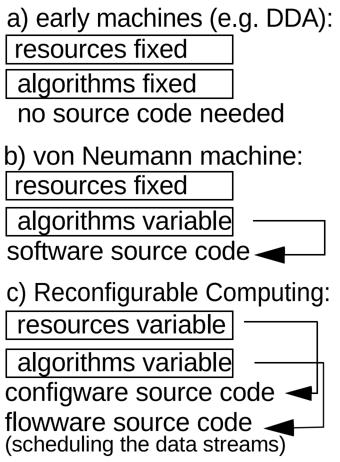


Figure 17. Nick Tredennick' machine classification scheme

Some remarks on Terminology: Since the basic paradigm is **<u>not</u>** instruction-stream-based, necessarily the term „Configware“ should be used for program sources, instead of the term „Software“, which would be confusing (fig 10). The term „software“ must be unconditionally restricted to traditional sources of instruction-stream-based computing. In fact this paradigm relies on data streams, however, not on instruction streams.

Equivocalities of the term „data stream“ are a problem, not only in education. 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 1980 by the systolic array scene [128] [129], where data streams enter and leave a datapath array (a pipe network: illustrated by fig 14). In fact, there a set of data streams is a data schedule specifying, which data item has to enter or leave which array port at which point of time.

The tail is wagging the dog. Because of their memory-cycle-hungry instruction-stream-driven sequential mode of operation microprocessors usually need much more powerful accelerators [4]: 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) *reconfigurable system age* with a third basic model, where the accelerator has become programmable (reconfigurable).

What high level source language for HPC and supercomputing users? Useful for application development are *Co-Compilers* (fig 7), automatically partitioning from the programming source into software and configware and accepting, for instance a C-like language [143]. The methodology is known from academic co-compilers [143] [144], easy to implement since most of their fundamentals have been published decades ago [146]. Fig. 18 contrasts the difference of intermediate sources to be generated by such a co-compiler. Although high level programming languages like C come closer to the such a user's mind set, than classical FPGA synthesis environments, it is still confusing. Such imperative languages seem to have an instruction-stream-based semantics and do not exhibit any dual paradigm features, because data-stream-based constructs are out of reach. It would be desirable to have a source language at one abstraction level higher than imperative languages, like the domain of mathematical formula. Such a system can be implemented by using a term rewriting system (TRS) [148] for dual paradigm system synthesis [149].

#	Source	is compiled into:
1	Software	instruction schedule
2	Flowware	data schedule
3	Configware	a pipe network by placement and routing

Figure 18. Sources by compilation targets

FPGA main processor with auxiliary CPU. 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 [150]: Is it time for vN to retire? The RAMP project, for instance proposes to run the operating system on FPGAs [151]. In fact, in some embedded systems, the CPU has this role already now. But often the awareness is missing.

T**he 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 10). 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 24), and, its processing unit *is not a CPU* (fig. 24). Instead, it is only a *DPU (Data Path Unit)*: without an instruction sequencer (fig. 24).

The enabling technology of the Kress-Kung machinehas one or mostly several *data counters* as part of the *Generic Address Generators* (*GAG*) [132] [134] [135] within data memory banks called *ASM* (*Auto-Sequencing Memory*, see fig 15). ASMs send and/or receive data streams having been programmed from *Flowware* sources [133] (fig 14). An ASM is the generalization of the DMA circuit (Direct Memory Access) [154] [155] for executing block transfers without needing to be controlled by instruction streams inside. ASMs, based on the use of distributed memory architectures [156] are very powerful architectural resources, supporting the optimization of the data storage schemes for minimizing the number of memory cycles [135]. The MoM Kress-Kung machine based on such generic address generators has been published in 1990 [157] [158]. The use of data counters replacing the program counter has first been published in 1987 [159].

Hardwired Kress-Kung machines. There are also hardwired versions of the Kress-Kung machine.We may distinguish 2 classes of Kress-Kung machines (fig 10): programmable ones (reconfigurable systems: reconfigurable Kress-Kung machine, needing 2 types of programming sources (see next paragraph and fig. 18): *Configware* for structural programming, and *Flowware*, for data scheduling. However, also hardwired Kress-Kung machines can be implemented for instance, (the BEE project [160]), 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 5 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 [161] [162]. On such a basis even *fault tolerance* by self-repair can be implemented [163] [164], as well as reconfigurable artificial neuronal networks [165] [166]. The electronics within the Cibola satellite [167] scheduled to be launched by October 2006 uses such fault tolerance mechanisms to cope with fault introduced by cosmic radiation [169]. Dynamic reconfigurability can be confusing for beginners and should be introduced not earlier than at graduate courses.

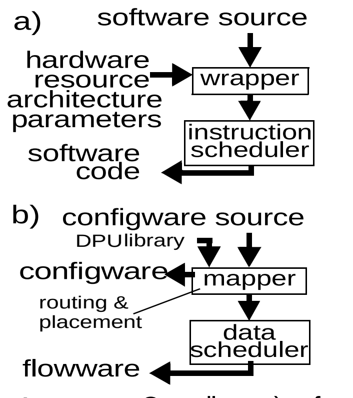


Figure 19. Compilers: a) software compilation, b) configware/ flowware co-compilation.

New educational approaches are 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, micro-processors (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.

1.3. Future of FPGA (Technologies)

The scaling of CMOS technology will come to an end It is unclear whether CMOS devices in the 10-20 nanometer range will find a useful place in semiconductor products [170]. Timothy S. Fisher (Purdue) said. "But before we can even think about using nanotubes in electronics, we have to learn how to put them where we want them." [171]. New silicon-based technologies (e.g., silicon nanowires) and non-silicon based (e.g., carbon nanotubes) show the promise of replacing traditional transistors [172] [174] [176] [177]. However, there are multiple challenges to face, like the production of nanoscale CMOS with reasonable yield and reliability, the creation of newer circuit structures with novel materials as well as the mixing and matching of older and newer technologies in search of a good balance of costs and benefits [170]. Opportunities will be driven by the ability of designing complex circuits. Problems related to defect density, failure rates, temperature sensitivity can be expected. Our ability to define the right design technology and methodology will be key in the realization of products of these nanotechnology. It also will depend on the direction that the semiconductor road will take [170].

Lithographic patterning has long been the primary way of defining features in semiconductor processing [178]. Nano designs, however, will not simply be an extension of classical VLSI design. We may not be able to directly pattern complex features, but rather need self-assembly to create ordered devices, and post-fabrication reconfigurability. This creates new challenges for design and motivates different architectures than found on classical silicon. The crossbar may to play a prominent role in these new developments. Its regular structure is well suited to nanotechnology-based assembly. A nanometer-scale crossbar is useful only if wires in each of its dimensions can be addressed individually. Research examines methods of stochastic addressing of nanowires [180] as well as the efficient data storage in crossbar-based memories [180].

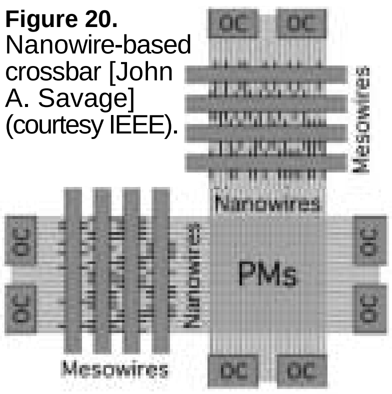


Figure 20. Nanowire-based crossbar [John A. Savage] (courtesy IEEE).

Nanoscale features. For nanoscale features there seems to be a sufficient set of building blocks for fully nanoscale systems (fig. 20 shows a crossbar [172]). Scientists are developing a growing repertoire of techniques which allow us to define features (e.g. wire widths, spacing, active device areas) without lithography, requiring self-assembly for structures from individual atoms with tight dimensions, with wires just a few atoms wide [172]. Semiconducting nanowires with diameters down to 3 nm (about 6 atoms wide) have been grown [181] [182], composed of different materials or selectively doped [183] along their length using timed growth [184]. Diode junctions built by crossing P-doped and N-doped nanowires use field-effects to control conduction to implement switchable crosspoints or memory bits [185] [186]. Nanowires can be sandwiched into array blocks including programmable OR planes, memory planes, and signal restoration or inversion planes [180] [187] [188]. Flow techniques can be used to align a set of nanowires into a single orientation, close pack them, and then transfer them onto a surface [189] [190]. Switchable molecules can be assembled in a crossed array [185] [188], providing sublithographic scale, programmable junctions. It is very difficult to predict, how many more years it will take to a large scale market introduction.

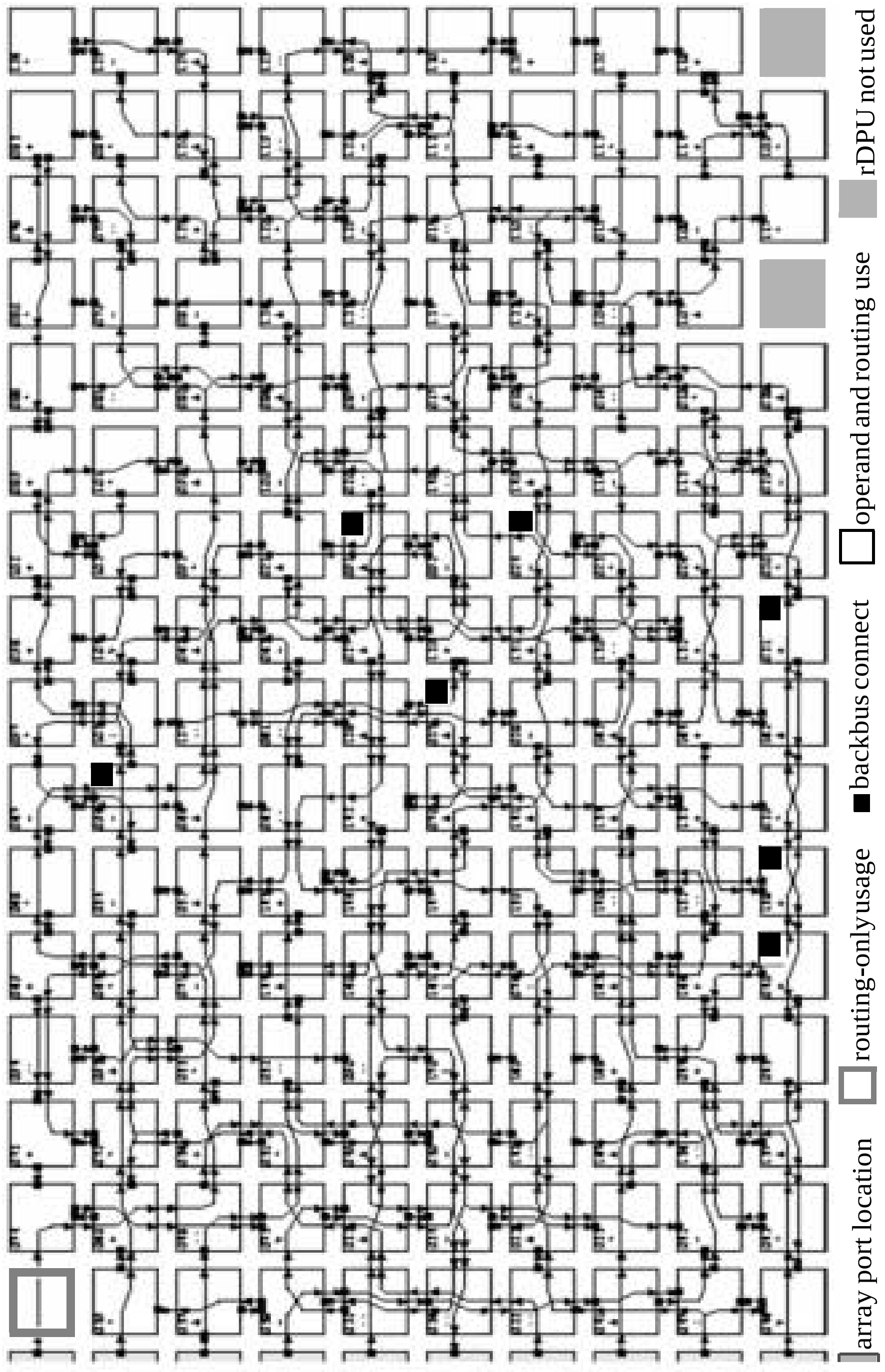


Figure 21. Example of mapping an application (image processing: SNN filter) onto a (coarse grain) KressArray.

1.4. Coarse Grained vs. Fine Grained Reconfigurability

We may distinguish two classes of reconfigurable microchips [191]: *fine grain* reconfigurable devices like FPGAs (see section 1.1), and, *coarse grain* reconfigurable devices: rDPAs (reconfigurable DataPath Arrays) [192]. The table in fig 27 gives a survey on differences between coarse-grained and fine-grained. Instead of the up to hundreds of thousands of about 1 bit wide CLBs (Configurable Logic Blocks) of modern FPGAs, a rDPA has a few (about up to 128 or 256 or much less) larger rDPU blocks (reconfigurable DataPath Units), for instance, 32 bits wide. For illustration a rDPU could be compared with the ALU of CPU (fig 27).

Advantages of Coarse-grained Reconfigurability. Coming along with functional level rDPUs as configurable blocks of rDPAs [192] - physically at a more convenient abstraction level much closer to a software user's mind set than that of FPGAs difficult to reach by design tools or compilation techniques, coarse-grained reconfigurability makes the educational gap smaller. To software people the configuration of FPGAs looked more like logic design on a strange platform, however, not like computation. Another advantage of coarse-grained reconfigurability is the much higher computational density than coming with FPGAs. In contrast to the very bad effective integration density of FPGAs (fig 5), this density of coarse-grained reconfigurable arrays (rDPAs) almost reaches the Gordon Moore curve [4] (fig 11, also see row 8 in fig 27): it is better by 4 orders of magnitude. Because a coarse-grained array has only a few larger configurable blocks (about a few hundred rDPUs or much less) with very compact configuration codes the configuration time is reduced to microseconds - in contrast to milliseconds as known from FPGAs.

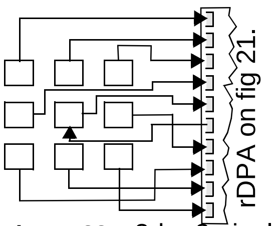


Figure 22. 3-by-3 pixel window of SNN algorithm

Free form pipe networks. In contrast to a CPU, a DPU is not instruction-stream-driven and has no program counter and its operation is transport-triggered by the arrival of operand data. This new machine paradigm (the diametrical counterpart of von Neumann) is based on free form large pipe networks of rDPUs (without memory wall & compilation is easy), but not on concurrent sequential processes. There is no instruction fetch overhead at run time since these pipe networks, generalizations of the systolic array, are configured before run time. This new paradigm is based on data-streams generated by highly parallel distributed on-chip local small but fast memory which consists of auto-sequencing memory (ASM) blocks (figure 15) using reconfigurable generic address generators (GAG), providing even complex address computations not needing memory cycles [132]. This kind of memory parallelism is more simple and more straight forward than interleaved memory access known from vector computers. The ASM is reconfigurable. This means its address computation algorithm is not changed at run time. It does not need an instruction stream at run time. In embedded systems often a symbiosis of both models, instruction-stream-based and data-stream-based, practices a duality of machine paradigms (fig. 7). Figures 19 and 24 explain this by showing the differences.

Improved designer productivity. The even physically high abstraction level or coarse-grained arrays dramatically improves the designer productivity. This is utilized by the KressArray Xplorer [137] [138] design environment (fig 29), including a design space explorer capable to compare and profile within a few hours a variety of different rDPU architectures and rDPA architectures, supporting the entire KressArray family concept [137] [138] featuring individual numbers of ports of individual path width at each side (example in fig 29 b) and many layers of background interconnect: mixture of buses and long distance point to pint. Why does this make sense? Since logic gates are general purpose elements, a basic FPGA is a universal device, whereas performance-optimized coarse-grained arrays tend to be more or less domain-specific, depending on the collection of functions available on a rDPU like, for instance, the physical availability of floating point operations. Fig. 21 shows a rDPA example having been compiled by the KressArray Xplorer for the SSN filter of an image processing application, which computes the center pixel from the 3-by-3 pixel input as shown by fig. 22. This rDPA consists of 160 rDPUs, each 32 bits wide. The highly optimal solution is demonstrated by the following result properties: Only 2 rDPUs are not used, and only 1 rDPU is used for rout-through-only. Almost all rDPUs are directly connected by nearest-neighbor interconnect, except 6 of them, which need a few additional backbus wires.

type of functional unit	instruction secquencer included?	execution triggered by
CPU	yes	instruc-tion fetch
DPU, or, rDPU	no	arrival of data[1]

1).transport-triggered

Figure 24. Duality of paradigms: von Neumann vs. Kress-Kung machin

Data-stream-based. The Kress-Kung array non-von-Neumann mode of operation, known from pipelines and from systolic arrays and Kress-Kung Arrays, is data-stream-based (fig. 14), instead of instruction-stream-based. The (r)DPU does not have an instruction sequencer, and, its execution is kind of transport-triggered, i. e. it's triggered by the arrival of its operand data. The data streams are generated by local ASM distributed reconfigurable memory [132] [193] (fig 15). The ASM (Auto-Sequencing Memory) is reconfigurable. This means that its address computation algorithm is not changed at run time. It does not need an instruction stream at run time. I. e. the Kress-Kung machine paradigm is a non-von Neumann paradigm: it is the diametrical counterpart of the von Neumann principles.

TTAs are von Neumann. We should not be confused with „transport-triggered architectures“

language category	Software Languages	Flowware Languages
sequencing managed 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
data manipulation specification	yes	not needed (specified by configware only)
state register	program counter	multiple data counters
instruction fetch	memory cycle overhead	no overhead
address computation	massive memory cycle overhead	no overhead

Figure 23. Software languages versus flowware languages.

(TTA) [194] [195] [196] such as heralded by the TTA community: using buses driven by an (instruction-stream-based) move processor [197]. This is not really data-stream-based: it is not an anti machine architecture. It's instruction-stream-based instead of a pipe network driven by a locally distributed ASM memory architecture [132] [193]. TTAs are von Neumann architectures. In contrast to a pipe not changed at run time, the interconnect of a bus is usually frequently changed at runtime - overhead-prone [123] and memory-cycle-hungry, since being under the control of instruction streams. For their data TTAs do not use (reconfigurable) ASM memory architectures. For this reason TTAs are von Neumann architectures.

FPGA-based Speed-up mechanisms better explained here. The astonishingly high acceleration factors obtained by CPU to FPGA migration not only in embedded systems (fig. 4, also see section 1.1), but also in supercomputing, are more easily explained within the higher abstraction level of coarse-grained reconfigurable systems here in this section. Fig. 26 a shows the execution (500 nanoseconds in total) of a conditional add operation on a simple hypothetical instruction set processor. Fig. 26 b illustrates its execution by a small coarse-grained array, where no memory cycles are needed and storing the final result to the rDPA's output buffer (slow technology) takes only 5 nanoseconds. The speed-up factor is 1000. We may summarize via this example, that avoiding or minimizing memory access is the main secret of success of reconfigurable computing: with FPGAs or coarse-grained. Also the GAG reconfigurable generic address generator of the ASM helps a lot (fig.15): at run time it does not need any memory cycle for address computation. To access bulk data storage the GAG methodology also helps to find storage schemes with minimal memory cycles [132] [135].

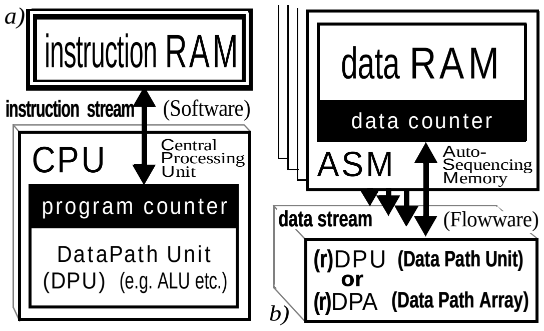


Figure 25. Basic paradigms: a) von Neumann machine, b) anti machine (reconfigurable or hardwired).

Communication effort at classical supercomputing. The communication complexity is growing exponentially because of typical bottlenecks, mainly determined by slow memory cycles [198] (fig. 28). Bus systems and other switching resources tend to require memory-cycle-hungry high administrative efforts: the von Neumann bottleneck [123]. Data transport at run time is a predominating problem, whereas we have an abundance of cheap CPU resources. Thinking in communicating concurrent processes like with the MPI (Message Passing Interface) standard is dominant. The communication effort tends to grow exponentially with the number of CPUs involved. The scalability of the performance of an application often drastically misses the theoretical peak performance values, which the platform seems to offer [199]. Amdahl's law explains only one of several reasons of this inefficiency [200]. However, the Kress-Kung machine does not have a von Neumann bottleneck.

MPI (Message Passing Interface). For reader outside the supercomputing community this is explained as follows. MPI is a standard implementing the distributed memory programming model by exchanging messages (also data) between several processors of shared memory architectures. All functions are available in a FORTRAN or a C version library with internal data structures, which are mainly hidden from the user. MPI is based on the *Communicating Sequential Processes* model (***CSP*** [201] [202]) by Tony Hoare [203] for quasi-parallel computing auf distributed, heterogeneous, loosely coupled computer systems. The programming language Occam is an implementation of CSP. JCSP combines CSP and Occam concepts in a Java-API. Herewith parallel MPI programs are executable on PC clusters and dedicated parallel computers (e.g. communicating via common main memory). Most FPGA users are not familiar with MPI and members of the supercomputing community do not understand the languages used by FPGA-savvy hardware experts. But there are efforts on the way to cope with the language problem [204]. For instance, also co-compilation from a C-like language is an interesting approach [143] [144].

The stool is moved and not the grand piano. With the instruction-stream-based mind set of the classical supercomputing, data transport often resembles moving the grand piano over to the stool of the pianist. However, the data-stream-based mind set of the reconfigurable computing community follows the inverse approach: the stool is moved and not the piano. Here MPI is unknown and concurrent processes are only a secondary aspect. Primarily the data are not moved by memory-cycle-hungry communication resources, but the locality of operations is assigned to the right places within the data streams, generated by ASM distributed memory (fig. 25 b and 26). The communication paths between rDPUs as pipe networks - without any administrative overhead and usually without intermediate storage. Since the number of processors is usually much lower than the number of data items, there is not much to move - during the less precious compile time. At run time also no instructions are transported: another acceleration aspect.

Taxonomy of Algorithms missing. Now it is time to discuss time to space mapping methods for software to configware migration: with coarse-grained arrays and also in general. Some algorithms are easy to map, and some others are difficult to map. Let us look at examples from the embedded systems area. Classical DSP algorithms, for instance, are easy to map form time to space and the result does not need much interconnect. It is well known, that some DSP algorithms even result just in simple a linear pipe. Other examples are extensively error-correcting (de)coding algorithms for a bad signal to noise ratio in wireless communication, like for turbo codes or the Viterbi algorithm and others [205],

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

		memory cycles	nano seconds
if C then read A endif	instruction fetch	1	100
	decode instruction		
	operand fetch	1	100
	execute instruction		
if not C then read B endif	instruction fetch	1	100
	decode instruction		
addieren	instruction fetch	1	100
	decode instruction		
	execute instruction		
Speichern	store result	1	100
total		**5**	**500**

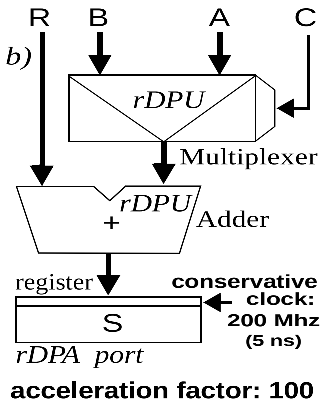


Figure 26. Illustrating acceleration: a) instruction-stream-based execution on a simple hypohetical processor (C = 1), b) Data-stream-based execution in a rDPA.

where the result requires an immense amount of interconnect resources: by far much more than available with only nearest neighbor connect within the (r)DPU array.

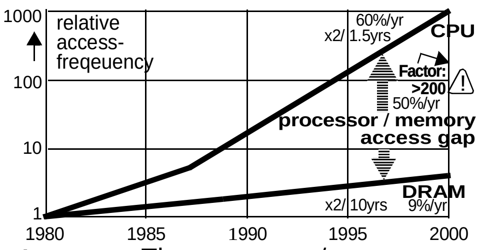


Figure 28. The processor / memory access gap (Patterson's law [198]).

Missing transdisciplinary perspective. On such samples of algorithmic cleverness we notice, that each special application domain keeps its own trick box. But what is really missing in education and elsewhere is an all-embracing dual-paradigm taxonomy of algorithms. A reductionistic attitude forcing a fragmentation into many special areas keeps us from the benefit of transdisciplinary understanding the algorithms taxonomy problem. But an all-embracing taxonomy of algorithms is not even available from a single-paradigm perspective. Even in classic high performance computing and supercomputing there is a fragmentation into special application domains. For education also here a transdisciplinary perspective is missing.

The personal supercomputer is near. The Munich-based PACT Corp. [206] with its coarse-grained reconfigurable XPA (Xtreme Processing Array) product (also see fig. 30) has demonstrated, that a 56 core 16-bit rDPA running at less than 500 MHz can host simultaneously everything needed for a world TV controller, like multiple standards, all types of conversions, (de)compaction, image improvements and repair, all sizes and technologies of screens, and all kinds of communication including wireless. More high performance by less CPUs, by reconfigurable units instead of CPUs. By this coarse-grained methodology also a single-chip game console is feasible. A highly promising vision would be a polymorphic multi core super pentium with multiple dual-mode PUs, which under control of a mode bit could individually run in CPU mode or in rDPU mode (not using the program counter). Choosing the right distributed on-chip memory architecture and the tight reconfigurable interconnect between these PUs is the key issue. But CPU vendors have a lack of reconfigurable computing experience, whereas the rDPA vendors lack concurrent

#		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	about 0.5 GHz	about 1 - 3 GHz
8	typical effective integration density compared to the Gordon Moore curve	reduced by a factor of about 10,000 (fig. 5)	reduced only by a few percent (fig. 11)

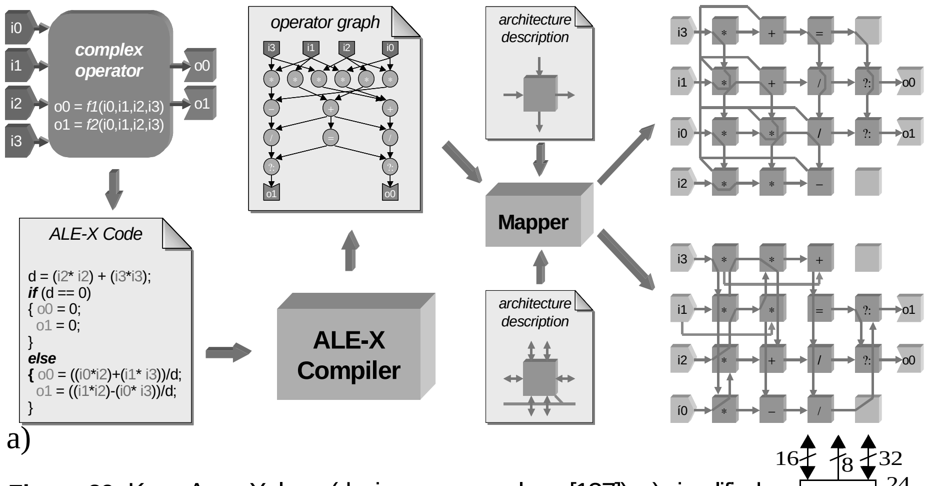
Figure 27. fine-grained vs. coarse-grained reconfigurability.

computing experience. But because of educational deficits it is not very likely, that one of the major microprocessor vendors or one of the major FPGA vendors will go toward multi core microprocessors adopting the coarse-grained reconfigurable computing array methodology. From a different point of view this would also be a proposal for PACT Corp. to develop such a modified polymorphic version of the XPU array.

The 2nd Reconfigurable Computing Paradox: although for many application areas again more orders of magnitude in speed-up are expected for coarse-grained arrays, compared to the orders of speed-up having already been obtained with FPGA-based solutions (see curve „coarse-grained“ in fig 4), there is no indication of efforts going into this direction, where future laptops reach a performance, which to-day requires a supercomputer. Another vision would be, that e. g. Xilinx would insert a rDPA onto a new platform FPGA targeting the scientific computing market. The RAMP project [151] having proposed to run the operating system on an FPGA would sound like: „Xilinx inside“ replacing „intel inside“. But there are also doubts wether Xilinx will understand such a vision of a future, where a dual-paradigm HPC and supercomputing will be found within embedded systems, achieved the performance, making unnecessary hangars full of monstrous expensive equipment unbelievably intensive guzzling electric power (also see [152]).

1.5. History of FPGAs

About in the mid’ 70ies the field programmable logic array (FPLA) was introduced by Signetics and Programmable array logic (PAL) was introduced by Monolithic Memories, Inc. (MMI), both for implementing combinational logic circuits -forerunners of the FPGA. A programmable logic array (PLA) is a programmable device used to implement combinational logic circuits. The PLA has a set of programmable AND



```
d = (i2* i2) + (i3*i3);
if (d == 0)
{ o0 = 0;
  o1 = 0;
}
else
{ o0 = ((i0*i2)+(i1* i3))/d;
  o1 = ((i1*i2)-(i0* i3))/d;
}
```

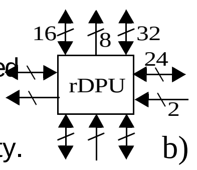


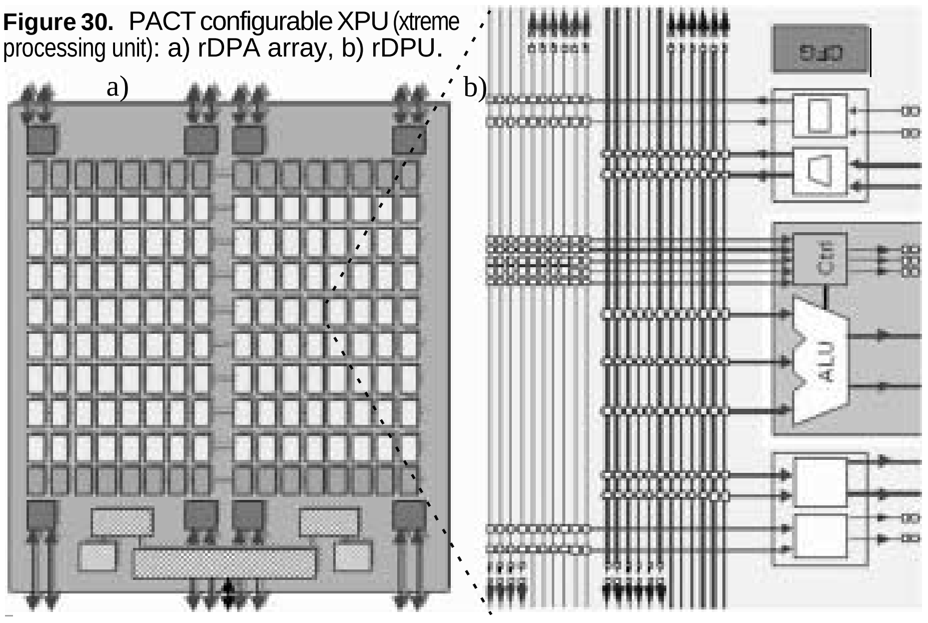
Figure 29. KressArray Xplorer (design space explorer [137]): a) simplified example to illustrate platform space exploration by finding an optimized array depending on rDPU architecture (1 or 2), b) KressArray family rDPU example architecture illustrating flexibility.

planes, which link to a set of programmable OR planes, which can then be conditionally complemented to produce an output. This layout allows for a large number of logic functions to be synthesized in the sum of products (and sometimes product of sums) canonical forms. The PAL introduced 1978 and second sourced by National Semiconductor, Texas Instruments, and Advanced Micro Devices was a huge success. The programmable elements connect both the true and complemented inputs to AND gates, also known as product terms, which are ORed together to form a sum-of-products logic array. Before PALs were introduced digital designers used SSI (small-scale integration) components, such as 7400 series nand gates and D-flip-flops. But one PAL device replaced dozens of such 'discrete' logic packages in many products, such as minicomputers [212] [213], so that the SSI business went into decline.

First Xilinx FPGA. Xilinx released the first FPGA in 1985, the XC2064 chip with a 1,000 gate size. Two decades later, in the year 2004 the size of an FPGA was 10,000 times larger. In 1985 this was a new technology for implementing digital logic. These new FPGAs could be viewed either as small, slow gate arrays (MPGAs) or large, expensive PLDs [214]. The major deviation from PLDs was the capability to implement multi-level logic. Their main advantage was in low NRE cost for small volume designs and prototyping. In the late 80s and early 90s an explosion of different architectures was observed [96] [214]. A good reference on the introduction of FPGA Architecture is [96].

Across the 100,000 gate threshold. As long as design size remained within the range of several thousand gates, schematic design entry and a good gate-level simulator were enough to create and verify the entire design. Hardware descriptions languages started to sneak into schematic designs in the shape of HDL macros, and as designs migrated into tens of thousands of gates they gained importance. By the time FPGAs crossed the 100,000 gate threshold it became obvious that HDLs would have to eliminate schematic entry and gate-level simulators.

Figure 30. PACT configurable XPU (xtreme processing unit): a) rDPA array, b) rDPU.



FPGAs in HPC mark the 'Beginning of a New Era [127]. It is time to look for alternatives to the classical von Neumann computer architecture. The old recipe of increasing the clock speed, the number of CPUs, the cores per chip and the threads per core just doesn't give us enough sustained computing performance. It seems that the von Neumann architecture has come to a dead end in HPC [127]. Things are now changing dramatically. FPGAs are being recognized as viable alternative to the massively power-consuming large scale computers [127].

Programming problems. FPGAs have been mostly hidden in embedded systems [127]. Their programming was cumbersome and required specialists using some obscure programming languages like VHDL or Verilog. You might think of these languages as a kind of assembler language, but that's wrong. It's even worse: Programming is done at the gate level, that is, at the very lowest level of information processing with NAND and NOR gates.

Educational Deficits.A less welcome side effect of the paradigm shift are educational deficits needing a training on the job, since typical CS or CE curricula ignore Reconfigurable Computing - still driving the dead road of the von-Neumann-only mind set [126]. A new IEEE international workshop series on Reconfigurable Computing Education has been founded to attack this problem [41]. Within the several hundreds of pages of all volumes of the 2004 joint ACM / AIS / IEEE-CS curriculum recommendations [208] by the find and replace function found zero encounters of the term „FPGA“ and all its synonyms or other terms pointing to reconfigurable computing. This is criminal. (For the 2005 version see [211].) These recommendations completely fail to accept the transdisciplinary responsibility of computer science to combat the fragmentation into many application-domain-specific tricky reconfigurable computing methodologies.

1.6. Literature

[1] A. Burks, H. Goldstein, J. von Neumann: Preliminary discussion of the logical design of an electronic computing instrument; US Army Ordnance Departmentt Report, 1946.

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

[3] M. J. Smith: Application Specific Integrated Circuits; Addison Wesley 1997

[4] R. Hartenstein (invited): The Microprocessor is no more General Purpose; Proc. IEEE International Symposium on Innovative Systems (ISIS), Oct.9-11 1997, Austin, Texas

[5] L. Rizzatti: Is there any future for ASIC emulators? (or How FPGA-Based Prototypes Can Replace ASIC Emulators); EDA Alert e-Newsletter PlanetEE, Febr. 18, 2003 www.planetee.com

[6] T. Makimoto (keynote): The Rising Wave of Field-Programmability; FPL 2000, Villach, Austria, Aug. 27 - 30, 2000; Springer Verlag, Heidelberg/New York, 2000

[7] J. Becker (invited tutorial): Reconfigurable Computing Systems; Proceedings Escola de Microeletrônica da SBC - Sul (EMICRO 2003), Rio Grande, Brasil, September 2003

[8] J.Becker, S. Vernalde (editors): Advances in Reconfigurable Architectures - Part 1; special issue, International Journal on Embedded Systems, 2006

[9] J.Becker, S. Vernalde (editors): Advances in Reconfigurable Architectures - Part 2; special issue, International Journal on Embedded Systems, 2006

[10] F. Faggin, M. Hoff, S. Mazor, M. Shima: The history of 4004; IEEE Micro, Dec. 1996

[11] R. Hartenstein (invited talk): Reconfigurable supercomputing: What are the Problems? What are the Solutions? Seminar on Dynamically Reconfigurable Architectures, Dagstuhl Castle, Wadern, Germany, April 2 - 7, 2006

[12] R. Hartenstein (opening keynote): From Organic Computing to Reconfigurable Computing; 8th Workshop on Parallel Systems and Algorithms (PASA 2006), March 16, 2006, Frankfurt/ Main, Germany - in conjunction with the 19th International Conference on Architecture of

Computing Systems (ARCS 2006), March 13 - 16, 2006, Frankfurt/Main, German

[13] R. Hartenstein (opening keynote): Supercomputing goes Reconfigurable - About key issues and their impact on CS education; Winter International Symposium on Information and Communication Technologies (WISICT 2005), Cape Town, South Africa, Jan. 3 - 6, 2005

[14] R. Hartenstein (plenum keynote address): Software or Configware? About the Digital Divide of Computing; 18th International Parallel and Distributed Processing Symposium (IPDPS), April 26–April 30, 2004, Santa Fe, New Mexico, USA

[15] R. Hartenstein (invited presentation): The Digital Divide of Computing; 2004 ACM International Conference on Computing Frontiers (CF04); April, 14-18, 2004. Ischia, Italy

[16] R. Hartenstein (invited keynote address): The Impact of Morphware on Parallel Computing; 12-th Euromicro Conference on Parallel, Distributed and Network based Processing (PDP04); February, 11-13, 2004. A Coruña, Spain

[17] D. Chinnery, K. Keutzer: Closing the Gap between ASIC & Custom; Kluwer 2002

[18] A. Grove: Only the Paranoid Survive; Currence 1996

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

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

[21] F. Dittrich: World's Fastest Lanman/NTLM Key Recovery Server Shipped; Picocomputing, 2006 URL: [22]

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

[23] K. Gaj, T. El-Ghazawi: Cryptographic Applications; RSSI Reconfigurable Systems Summer Institute, July 11-13, 2005, Urbana-Champaign, IL, USA URL: [24]

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

[25] 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

[26] ***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*** = Multiply and Accumulate; ***PU*** = Processing Unit ***rDPA*** = reconfigurable Data Path Array; ***rDPU*** = reconfigurable Data Path Unit; ***rE*** = reconfigurable Element

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

[28] J. Rabaey: Reconfigurable Processing: The Solution to Low-Power Programmable DSP; ICASSP 1997

[29] Y. Gu, et al.: FPGA Acceleration of Molecular Dynamics Computations; FCCM 2004 URL: [30]

[30] http://www.bu.edu/caadlab/FCCM05.pdf

[31] A. Alex, J.Rose et al.: Hardware Accelerated Novel Protein Identification; FPL 2004

[32] N. N. Nallatech, press release, 2005

[33] H. Singpiel, C. Jacobi: Exploring the benefits of FPGA-processor technology for genome analysis at Acconovis; ISC 2003, June 2003, Heidelberg, Germany URL: [34]

[34] http://www.hoise.com/vmw/03/articles/vmw/LV-PL-06-03-9.html

[35] N. N. (Starbridge): Smith-Waterman pattern matching; National Cancer Institute, 2004

[36] A. Darabiha: Video-Rate Stereo Vision on Reconfigurable Hardware; Master Thesis, Un. Toronto, 2003

[37] R. McCready: Real-Time Face Detection on a Configurable Hardware Platform; Master thesis, University of Toronto, 2000

[38] T. Fry, S. Hauck: Hyperspectral Image Compression on Reconfigurable Platforms; Proc. FCCM 2002

[39] P. Buxa, D. Caliga: Reconfigurable Processing Design Suits UAV Radar Apps; COTS J., Oct. 2005: [40]

[40] http://www.srccomp.com/ReconfigurableProcessing_UAVs_ COTS-Journal_Oct05.pdf

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

[42] R. Porter: Evolution on FPGAs for Feature Extraction; Ph.D. thesis; Queensland Un., Brisbane, Australia, 2001 : [43]

[43] http://www.pooka.lanl.gov/content/pooka/green/Publications_files/imageFPGA.pdf

[44] E. Chitalwala: Starbridge Solutions to Supercomputing Problems; RSSI Reconfigurable Systems Summer Institute, July 11-13, 2005, Urbana-Champaign, IL,USA

[45] S. D. Haynes, P. Y. K. Cheung, W. Luk, J. Stone: SONIC - A Plug-In Architecture for Video Processing; Proc. FPL 1999

[46] M. Kuulusa: DSP Processor Based Wireless System Design; Tampere Un. of Technology, Publ. # 296: [47]

[47] http://edu.cs.tut.fi/kuulusa296.pdf

[48] B. Schäfer, S. Quigley, A. Chan: Implementation Of The Discrete Element Method Using Reconfigurable Computing (FPGAs); 15th ASCE Engineering Mechanics Conf., June 2-5, 2002, New York, NY: [49]

[49] http://www.civil.columbia.edu/em2002/proceedings/papers/126.pdf

[50] G. Lienhart: Beschleunigung Hydrodynamischer N-Körper-Simulationen mit Rekonfigurierbaren

Rechenystemen; 33rd Speedup / 19th PARS Worksh.; Basel, Switzerland, March 19 - 21, 2003

[51] G. Steiner, K. Shenoy, D. Isaacs, D. Pellerin: How to accelerate algorithms by automatically generating FPGA coprocessors; Dr. Dobbs Portal, August 09, 2006 URL: [52]

[52] http://www.ddj.com/dept/embedded/191901647

[53] R. Hartenstein (keynote): The Transdisciplinary Responsibility of CS Curricula; 8th World Conf. on Integrated Design & Process Technology (IDPT), June 25-29, 2006, San Diego, CA, USA

[54] R. Hartenstein (invited presentation): Reconfigurable Computing; Computing Meeting, Commission of the EU, Brussels, Belgium, May 18, 2006,

[55] R. Hartenstein (keynote): New horizons of very high performance computing (VHPC) - hurdles and chances; 13th Reconfigurable Architectures Workshop (RAW 2006), Rhodos Island, Greece, April 25 - 26, 2006

[56] R. Hartenstein: Configware für Supercomputing: Aufbruch zum Personal Supercomputer; PIK - Praxis der Informationsverarbeitung und Kommunikation, 2006; KG Saur Verlag GmbH, Munich, Germany - title in English: see [57]

[57] Configware for Supercomputing: Decampment for the Personal Supercomputer

[58] J. Becker, et al.: An Embedded Accelerator for Real Time Image Processing; 8th EUROMICRO Workshop on Real Time Systems, L'Aquila, Italy, June 1996

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

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

[61] J. Becker et al.: High-Performance Computing Using a Reconfigurable Accelerator; CPE Journal, Special Issue of Concurrency: Practice and Experience, Wiley, 1996

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

[63] Herb Riley, R. Associates: http://www.supercomputingonline.com/article.php?sid=9095

[64] R. Hartenstein (opening keynote): The Re-definition of Low Power Design for HPC: a Paradigm Shift; 19th Symp. on Integrated Circuits and System Design (SBCCI 2006), Aug. 28t - Sept 1, 2006, Ouro Preto, Minas Gerais, Brazil

[65] Ch. Piguet (keynote): Static and dynamic power reduction by architecture selection; PATMOS 2006, Sept. 13 - 15, Montpellier, France

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

[67] R. Hartenstein (opening keynote address: The Re-definition of Low Power Digital System Design - for slashing the Electricity Bill by Millions of Dollars; The 19th Symposium on Integrated Circuits and System Design (SBCCI 2006), Aug. 28 - Sept. 1, 2006, Ouro Preto, Minas Gerais, Brazil

[68] R. Hartenstein (invited contribution): Higher Performance by less CPUs; HPCwire, June 30, 2006, in conjunction with ISC 2006, Dresden, Germany - [69]

[69] http://xputers.informatik.uni-kl.de/staff/hartenstein/lot/HartensteinLessCPUs06.pdf

[70] R.Hartenstein, A. Nuñez, et al.: Report on Interconnect Modelling: Second Periodic Progress Report; PATMOS Project, Universität Kaiserslautern, Dec. 1991

[71] http://www.patmos-conf.org/

[72] http://www.islped.org/

[73] 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 - URL: [74]

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

[75] R. Hartenstein (invited presentation): Reconfigurable Computing (RC) being Mainstream: Torpedoed by Education; 2005 International Conference on Microelectronic Systems Education, 12 - 13 June 2005, Anaheim, California, co-located with DAC (Design Automation Conf.) 2005

[76] R. Hartenstein (invited opening keynote address): Reconfigurable HPC: torpedoed by Deficits in Education?; Workshop on Reconfigurable Systems for HPC (RHPC); July 21, 2004, to be held in conjunction with HPC Asia 2004, 7th International Conf. on High Performance Computing and Grid in Asia Pacific Region, July 20 - 22, 2004, Omiya (Tokyo), Japan

[77] R. Hartenstein (keynote): The Changing Role of Computer Architecture Education within CS Curricula; Workshop on Computer Architecture Education (WCAE 2004) June 19, 2004, in conj. w. 31st Int'l Symp. on Computer Architecture (ISCA), Munich, Germany, June 19-23, 2004

[78] S. Hauck: The Role of FPGAs in Reprogrammable Systems; Proc. IEEE, 1998

[79] F. Rammig (interview): Visions from the IT Engine Room; IFIP TC 10 - Computer Systems Technology, URL: [80]

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

[81] S. Hang: Embedded Systems – Der (verdeckte) Siegeszug einer Schlüsseltechnologie; Deutsche Bank Research, Jan. 2001, [82], URL: [83]
[82] S. Hang ([81] title in English): Embedded Systems - The (subsurface) Triumph of a Key Technology;
[83] http://xputers.informatik.uni-kl.de/VerdeckterSiegeszug.pdf
[84] P. Gillick: State of the art FPGA development tools; Reconfigurable Computing Workshop, Orsay, France, Sept. 2003
[85] http://antimachine.org
[86] V. Betz, J, Rose, A. Marquardt (editors): Architecture and CAD for Deep-Submicron FPGas; Kluwer, 1999
[87] R. Hartenstein: Morphware and Configware; (invited chapter) in: A. Zomaya (editor): Handbook of Innovative Computing Paradigms; Springer Verlag, New York, 2006
[88] S. Hoffmann: Modern FPGAs, Reconfigurable Platforms and their Design Tools; Proc. REASON summer school; Ljubljana, Slovenia, August 11-13, 2003
[89] N.N.: Pair extend Flash FPGA codevelopment pact; http://www.electronicstalk.com/news/act/act120.html
[90] B. Lewis, Gartner Dataquest, October 28, 2002
[91] http://morphware.net
[92] http://www.morphware.org/
[93] http://configware.org
[94] D. Soudris et al.: Survey of existing fine grain reconfigurable hardware platforms; Deliverable D9, AMDREL consortium (Architectures and Methodologies for Dynamically Reconfigurable Logic); 2002
[95] J. Oldfield, R. Dorf: Field-Programmable Gate Arrays: Reconfigurable Logic for Rapid Prototyping and Implementation of Digital Systems; Wiley-Interscience, 1995
[96] J. Rose, A. E. Gamal, and A. Sangiovanni-Vincentelli: Architecture of FPGAs; Proceedings of IEEE, vol. 81, no. 7, July 1993
[97] http://www.xilinx.com
[98] http://www.altera.com
[99] N. N.: FPGA Development Boards and Software for Universities; Microelectronic System News, August 2, 2006, URL: [100]
[100]http://vlsi1.engr.utk.edu/~bouldin/MUGSTUFF/NEWSLETTERS/DATA/1278.html
[101]S. Davidmann: Commentary: It's the software, stupid; El. Bus. online; Aug. 15, 2006 URL: [102]
[102]http://www.reed-electronics.com/eb-mag/index.asp?layout=articlePrint&articleID=CA6360703
[103]Michio Kaku: Visions; Anchor Books, 1998
[104]U. Rüde: Technological trends and their impact on the future of supercomputers; in: H. Bungartz, F Durst, C. Zenger, C. (editors): High Performance Scientific and Engineering Computing: Int'l FORTWIHR Conf. on HPSEC, Munich, Germany, March 16 - 18, 1998 - URL: [112]
[105]F. Brooks: No Silver Bullet; COMPUTER, 20, 4 (April 1987)
[106]F. Brooks: The Mythical Man Month (Anniversary Edition w. 4 new chapters) Addison Wesley, 1995
[107] T. S. Kuhn: The Structure of Scientific Revolution; Univ. of Chicago Press, Chicgo, 1996
[108]V. Rajlich: Changing the Paradigm of Software Engineering; C. ACM 49,8 (Aug. 2006)
[109]E. M. Rogers: Diffusion of Innovations - 4th edition; The Free Press, New York, 1995
[110]http://www.sdpsnet.org/Other_PDFs/IDPT2006FinalProgram.pdf
[111]http://www.organic-computing.de/
[112]http://www10.informatik.uni-erlangen.de/Publications/Papers/1998/lncse_vol8_98.pdf
[113]V. Natoli: A Computational Physicist's View of Reconfigurable High Performance Computing; Reconfigurable Systems Summer Institute, Urbana, IL, July 2005, URL: [115]
[114]Edward A. Lee: The Problem with Threads; COMPUTER, May 2006
[115]http://www.ncsa.uiuc.edu/Conferences/RSSI/2005/docs/Natoli.pdf
[116]Mark Bereit: Escape the Software Development Paradigm Trap; Dr. Dobb's Journal (05/29/06)
[117]B. Hayes: The Semicolon Wars; American Scientist, July/Aug. 2006 URL: [118]
[118] http://www.americanscientist.org/content/AMSCI/AMSCI/ArticleAltFormat/20066510252_307.pdf
[119]E. Levenez: Computer Languages History; http://www.levenez.com/lang/
[120]B. Kinnersley: The Language List - Collected Information On About 2500 Computer Languages, Past and Present; http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm
[121]D. Piggott: HOPL: An interactive roster of programming languages; 1995-2006. URL: [122]
[122]http://hopl.murdoch.edu.au/
[123]G. Koch, R. Hartenstein: The Universal Bus considered harmful; in: [124]

[124]R. Hartenstein R. Zaks (editors): Microarchitecture of Computer Systems; American Elsevier, New York 1975

[125]R. Hartenstein (opening keynote address): Why we need Reconfigurable Computing Education: Introduction; The 1st International Workshop on Reconfigurable Computing Education (RCeducation 2006), March 1, 2006, Karlsruhe, Germany - in conj. with the IEEE Computer Society Annual Symp. on VLSI (ISVLSI 2006), March 2-3, 2006, Karlsruhe, Germany [126]

[126]http://hartenstein.de/RCedu.html

[127]A. Reinefeld (interview): FPGAs in HPC Mark „Beginning of a New Era"; HPCwire June 29, 2006 - http://www.hpcwire.com/hpc/709193.html

[128]N. Petkov: Systolic Parallel Processing; North-Holland; 1992

[129]H. T. Kung: Why Systolic Architectures? IEEE Computer 15(1): 37-46 (1982)

[130]R. Kress et al.: A Datapath Synthesis System (DPSS) for the Reconfigurable Datapath Architecture; Proc. ASP-DAC 1995

[131]http://kressarray.de

[132]M. Herz et al. (invited p.): Memory Organization for Data-Stream-based Reconfigurable Computing; 9th IEEE Int'l Conf. on Electronics, Circuits and Systems (ICECS), Sep 15-18, 2002, Dubrovnik, Croatia

[133]http://flowware.net

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

[135]M. Herz: High Performance Memory Communication Architectures for Coarse-grained Reconfigurable Computing Systems; Dissertation, Univ. Kaiserslautern, 2001, URL: [136]

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

[137]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, Austria

[138]U. Nageldinger: Coarse-grained Reconfigurable Architectures Design Space Exploration; Dissertation, University of Kaiserslautern, 2001, Kaiserslautern, Germany URL: [139]

[139]http://xputers.informatik.uni-kl.de/papers/publications/NageldingerDiss.html

[140]R. Hartenstein (invited plenum presentation): Reconfigurable Supercomputing: Hurdles and Chances; International Supercomputer Conference (ICS 2006), June 28 - 30, 2006, Dresden, Germany - http://www.supercomp.de/

[141]http://flowware.net

[142]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, Tschechia

[143]J. Becker et al.: Parallelization in Co-Compilation for Configurable Accelerators; Asian-Pacific Design Automation Conference (ASP-DAC), 10. - 13. February, 1998, Yokohama, Japan

[144]J. Becker: A Partitioning Compiler for Computers with Xputer-based Accelerators, Dissertation, University of Kaiserslautern, 1997, URL: [145]

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

[146]K. Schmidt: A Program Partitioning, Restructuring, and Mapping Method for Xputers, Dissertation, University of Kaiserslautern, 1994 URL: [147]

[147]http://www.shaker.de/Online-Gesamtkatalog/Details.asp?ID=930601&CC=41546&ISBN=3-8265-0495-X&Reihe=15&IDSRC=4&LastAction=Search

[148]J. C. Hoe, Arvind: Hardware Synthesis from Term Rewriting Systems; VLSI'99, Lisbon, Portugal

[149]M. Ayala-Rincón, R. Hartenstein, C. H. Llanos, R. P. Jacobi: Prototyping Time and Space Efficient Computations of Algebraic Operations over Dynamically Reconfigurable Systems Modeled by Rewriting-Logic; accepted for ACM Transactions on Design Automation of Electronic Systems (TODAES), 2006

[150]A. Cantle: Is it time for von Neumann and Harvard to Retire?; RSSI Reconfigurable Systems Summer Institute, July 11-13, 2005, Urbana-Champaign, IL,USA

[151]http://bwrc.eecs.berkeley.edu/Research/RAMP/

[152]Horst Simon[1] (interview): A Petaflop Before Its Time? HPCwire, June 28, 2006 URL: [153]

[153]http://www.hpcwire.com/hpc/705881.html

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

[155]S. Heath: Embedded Systems Design, Elsevier, 2003

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

1. associate laboratory director for computing sciences at Berkeley Lab, also the director of NERSC

[157] A. G. Hirschbiel et al.: A Novel Paradigm of Parallel Computation and its Use to Implement Simple High Performance Hardware; InfoJapan'90 - Int'l Conf. memorating 30th Anniversary, Computer Society of Japan; Tokyo, Japan, 1990

[158] Invited reprint of [157] in Future Generation Computer Systems 7 91/92, p. 181-198, North Holland

[159] W. Nebel et al: Functional Design Verification by Register Transfer Net Extraction from Integrated Circuit Layout Data; IEEE COMPEURO, Hamburg, 1987

[160] C. Chang et al: The Biggascale Emulation Engine (Bee); summer retreat 2001, UC Berkeley

[161] H. Simmler et al.: Multitasking on FPGA Coprocessors; International Conference on Field Programmable Logic and Applications (FPL), 28. - 30. August 2000, Villach, Austria

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

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

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

[165] J. A. Starzyk, J. Guo, Z. Zhu: Dynamically Reconfigurable Neuron Architecture for the Implementation of Self-organizing Learning Array; International Journal on Embedded Systems, 2006

[166] M. Glesner, W. Pochmueller: An Overview of Neural Networks in VLSI; Chapman & Hall, London, 1994

[167] N. Ambrosiano: Los Alamos and Surrey Satellite contract for Cibola flight experiment platform; Los Alamos National Lab, Los Alamos, NM, March 10, 2004 URL: [168]

[168] http://www.lanl.gov/news/releases/archive/04-015.shtml

[169] M. Gokhale et al.: Dynamic Reconfiguration for Management of Radiation-Induced Faults in FPGAs; IPDPS 2004

[170] G. De Micheli (keynote): Nanoelectronics: challenges and opportunities; PATMOS 2006, Sept. 13 - 15, Montpellier, France

[171] E. Venere: Nanoelectronics Goes Vertical; HPCwire, Aug. 4 2006, Vol. 15, No. 31 - http://www.hpcwire.com/hpc/768349.html

[172] J. E. Savage: Research in Computational Nanotechnology; Link: [173]

[173] http://www.cs.brown.edu/people/jes/nano.html

[174] A. DeHon et al.: Sub-lithographic Semiconductor Computing Systems; HotChips-15, August 17--19, 2003, link: [175]

[175] http://www.cs.caltech.edu/research/ic/pdf/sublitho_hotchips2003.pdf

[176] L. Gottlieb, J. E. Savage, A. Yerukhimovich: Efficient Data Storage in Large Nanoarrays; Theory of Computing Systems, Vol. 38, pp. 503-536, 2005.

[177] J. E. Savage: Computing with Electronic Nanotechnologies; 5th Conference on Algorithms and Complexity, May 28-30, 2003 Rome, Italy

[178] International Technology Roadmap for Semiconductors. link: [179]

[179] http://public.itrs.net/Files/2001ITRS/ , 2001.

[180] A. DeHon, P. Lincoln, J. Savage: Stochastic Assembly of Sublithographic Nanoscale Interfaces; IEEE Transactions in Nanotechnology, September 2003.

[181] Y. Cui et al.: Diameter-Controlled Synthesis of Single Crystal Silicon Nanowires; Applied Physics Letters, 78(15):2214–2216, 2001.

[182] A. M. Morales, C. M. Lieber: A Laser Ablation Method for Synthesis of Crystalline Semiconductor Nanowires; Science, 279:208–211, 1998.

[183] M. S. Gudiksend et al.: Epitaxial Core- Shell and Core-Multi-Shell Nanowire Heterostructures; Nature, 420:57–61, 2002.

[184] M. S. Gudiksen et al.: Growth of Nanowire Superlattice Structures for Nanoscale Photonics and Electronics;. Nature, 415:617–620, February 7 2002.

[185] C. Collier et al.: A Catenane-Based Solid State Reconfigurable Switch; Science, 289:1172–1175, 2000.

[186] C. P. Collier et al.: Electronically Configurable Molecular-Based Logic Gates. Science, 285:391–394, 1999.

[187] A. DeHon: Array-Based Architecture for FET-based Nanoscale Electronics; IEEE Transactions on Nanotechnology, 2(1):23–32, March 2003.

[188] C. L. Brown et al.: Introduction of [2] Catenanes into Langmuir Films and Langmuir-Blodgett Multilayers. A Possible Strategy for Molecular Information Storage Materials;

Langmuir, 16(4):1924–1930, 2000.

[189]Y. Huang, X. Duan, Q. Wei, C. M. Lieber; Directed Assembley of One-Dimensional Nanostructures into Functional Networks. Science, 291:630–633, Jan. 26 2001.

[190]D. Whang, S. Jin, C. M. Lieber; Nanolithography Using Hierarchically Assembled Nanowire Masks; Nano Letters, 3, 2003. links: [207]

[191]R. Hartenstein (invited embedded tutorial): A Decade of Reconfigurable Computing: A Visionary Retrospective; Design, Automation and Test in Europe, Conference & Exhibition (DATE 2001), March 13 - 16, 2001, Munich, Germany

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

[193]R. Hartenstein et al.[1]: A High Performance Machine Paradigm Based on Auto-Sequencing Data Memory; HICSS-24 Hawaii Int'l Conf. on System Sciences, Koloa, Hawaii, Jan. 1991

[194] J. Hoogerbrugge, H. Corporaal, H. Mulder: Software pipelining for transport-triggered architectures: Proc. MICRO-24, Albuquerque, 1991

[195]H. Corporaal: Microprocessor Architectures from VLIW to TTA; John Wiley, 1998

[196]http://en.wikipedia.org/wiki/Transport_Triggered_Architectures

[197]D. Tabak, G. J Lipovski: MOVE architecture in digital controllers; IEEE Transactions on Computers C-29, pp. 180--190, 1980

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

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

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

[201]C. A. R. Hoare: Communicating Sequential Processes, Prentice-Hall, 1985 - URL: [202]

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

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

[204]Th. Steinke: Experiences with High-Level-Programming of FPGAs; International Supercomputer Conference (ICS 2006), June 28 - 30, 2006, Dresden, Germany

[205]N. Wehn (invited keynote): Advanced Channel Decoding Algorithms and Their Implementation for Future Communication Systems; IEEE Computer Society Annual Symposium on VLSI, March 2-3, 2006, Karlsruhe, Germany

[206]http://pactcorp.com

[207]http://www.cs.caltech.edu/research/ic/abstracts/sublitho_hotchips2003.html

[208] Joint Task Force for Computing Curricula 2004: Computing Curricula 2004. Overview Report; s. [209]

[209]http://portal.acm.org/citation.cfm?id=1041624.1041634

[210]http://hartenstein.de/ComputingCurriculaOverview_Draft_11-22-04.pdf

[211]http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf

[212]Tracy Kidder: The Soul Of A New Machine; Little Brown and Company, 1981- reprint: [213]

[213]Tracy Kidder (reprint): The Soul Of A New Machine; Modern Library, 1997

[214]S.-L. Lu: The FPGA paradigm; http://web.engr.oregonstate.edu/~sllu/fpga/lec1.html

1. best paper award