Stream-based Computing:
Antimatter of Computing Science

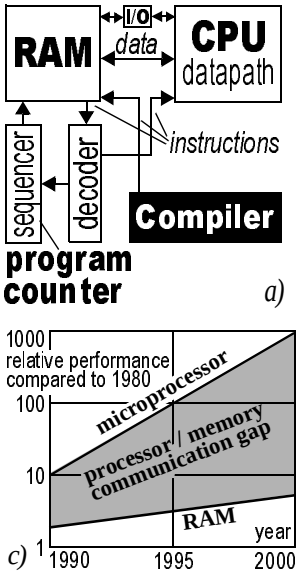
(keynote)
Reiner Hartenstein, University of Kaiserslautern, Germany
http://configware.de reiner@hartenstein.de

Abstract

The paper gives a survey on trends and recent developments in the areas of Reconfigurable Logic and Reconfigurable Computing, which could threat the basements of the current curricular building of Computer Science and Engineering [1].

1. Introduction

In 1927 Paul Dirac stated, that an electron cannot be described without having an anti electron as a partner. In 1932 this positron has been detected within cosmic radiation, what verified Paul Dirac's theory of holes assuming the existence of antimatter. A new exciting research area had been opened up. To produce antimatter for experiments billions of dollars have been spent for accelerators and storage rings like the CERN positron storage ring. Antimatter is difficult to produce. CERN's one year production of antiprotons would supply energy to light a 100 watt electric bulb for only three seconds. Aren't somewhere natural supplies of antimatter? Astrophysicists are investigating through the vastness of outer space.



b) machine category	Computer ("v. Neumann")
machine paradigm	procedural sequencing: deterministic
driven by:	control flow
RA support	no
engine principles	instruction sequencing
state register	program counter
communication path set-up	at run time
data path: resource	single ALU
data path: operation	sequential

Fig. 1: Machine paradigms: a) control-procedural "von Neumann" paradigm, b) the properties of the paradigm, c) growing processor / memory communication gap.

Meanwhile particle physics self-confidently claims, that each particle has a corresponding antiparticle, with the same mass, life span, and spin. All charges have the same amount, but with the opposite sign. An example is the electron with negative charge, and the positron as its antiparticle, however, with positive charge. But not every particle on the matter side is an exact mirror image of its antiparticle on the antimatter side. There is some asymmetry between matter and antimatter, which is no yet well known and is still under investigation.

Another fascinating phenomenon is the so-called annihilation, which happens whenever a particle and its antiparticle collide and disappear by being converted into energy. However, since our usual all-day environments consist only of "normal" matter, we don't care about antimatter. Also in computer science the overwhelming majority of people normally does not care about its antimatter, or does not have the slightest idea of its existence.

2. Matter of Computing

In Computing the world of matter is the world of instruction streams, controlled by the program counter used as state register. For simplicity I call the basic principles the von Neumann (vN) paradigm, partitioning the machine into CPU datapath, instruction sequencer (instruction fetch and branching control), RAM, and, I/O. Figure 1 a illustrates the execution mechanism of the world of matter: machine code generated by a compiler is downloaded to RAM, from where at run time under program counter control the instruction sequencer derives the instruction stream flowing between RAM, I/O and datapath. We may model this instruction-centric (control-flow-driven) machine paradigm with an atom (figure 2 a), where the instruction flow electron (location of the negative program counter state) is spinning around the positive nucleus (CPU resources like ALU etc.) .

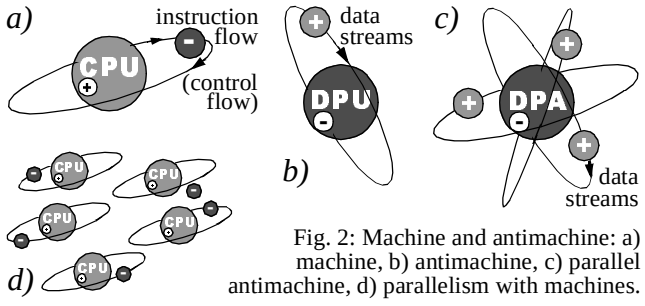


Fig. 2: Machine and antimachine: a) machine, b) antimachine, c) parallel antimachine, d) parallelism with machines.

Figure 1 b summarizes the properties of this machine paradigm: deterministic, instruction-stream-driven. Communication paths are switched at runtime (RAM2datapath, RAM2I/O, datapath2I/O, datapath2RAM, I/O2RAM, I/O2datapath). This control-procedural execution mechanism is modelled into the brain of each computing science student - as a common machine paradigm, with, or, without stack mechanism extensions. Due to the simplicity of this machine paradigm zillions of programmers can be educated.

The hardwired processor can be fabricated in volume and all personalization (programming) is achieved by downloading the fully relocatable machine code into the immensely scalable RAM. The dominating instruction

set became a quasi standard, where compatibility is managed from generation to generation of processors. All three, compatibility, having a machine paradigm, and being RAM-based, are the basis of the tremendous success of the software industry. This is the "normal" matter of computing.

2.1 Problematic matter of computing

But the world of matter in computing has a lot of problems, like, for example, the processor / memory communication bottleneck, also called von Neumann bottleneck, which is widening from generation to generation and has reached about 2 orders of magnitude (fig. 1 c). In the procedural world of computing in time the typical mind set can be specified by the submarine model. This is a vertical top-down hierarchy of layers: math formula, control/data flow graph, high level source program, assembler program, and finally the relocatable machine code. The hardware is a kind of invisible at lowest level under the surface, like a submarine.

But also the performance Software processor solutions are inefficient relative to comparable dedicated hardwired solutions (fig. 3). For a 0.13 µ technology the difference in MOPS/mW between "intel inside" and the best possible hardwired solution spans up to a factor of 3 orders of magnitude - with growing tendency for future technology generations (fig. 3). There are fundamental flaws in software architectures. First, using time multiplexing with a single piece of logic. Secondly, the overhead associated in moving data back and forth between memory back and logic. Third, control itself is overhead. Fourth, pipelining in micro-processors adds another whole level of control overhead. Chips these days are almost all memory, and the reason is that the architecture is so wrong. Only about one percent of the power is going into real logic functions and 99 percent is going into caches and other hardware overhead [2]. It is shocking to find the performance difference as a factor of 100 to 1,000 [2], even to 10.000 [3]. The metric for what is a good solution has been wrong all the time. By hardwired solutions almost 1,000 MOPS per milliwatts or almost 1,000 MOPS per square millimeter can be obtained [2].

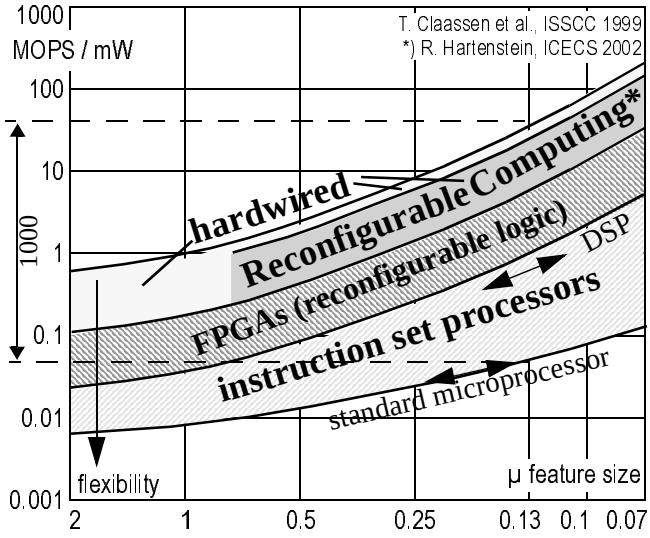


Fig. 3: Energy efficiency vs. flexibility including Reconfigurable computing.

After about a dozen of generations the microprocessor is a methusela. Not only in embedded systems the microprocessor more and more takes the role of a handicapped needing a wide variety of prostheses. so that the accelerators often occupy more silicon real estate than the processor core itself (fig. 14 b, c, d).

2.2 Matter of Parallel Computing

Parallelism by concurrent computing in the world of matter simultaneously uses several or many machines (CPUs, see fig. 5), what I would like to model as a swarm of atoms (fig. 2 d), where each has its own instruction stream electron spinning around.

Contemporary RISC core IP cells are available so small, that 64 or more of them would fit onto a single chip to form a massively parallel computing system. But this is not a general remedy for the parallel computing crisis [4], indicated by rapidly shrinking supercomputing conferences and dying supercomputing industries (fig. 4). For many application areas process level parallelism yields only poor speed-up improvement per processor added. Amdahls law explains just one of several reasons of inefficient resource utilization. Another dominating problem is the instruction-driven late binding of communication paths, which often leads to massive communication switching overhead at run-time (fig. 5). R&D in the past has largely ignored, that the vN paradigm is not a communication paradigm. Processor architecture has reached a dead end with masses of research projects around speculative execution pipelining cache-related strategies, and others, usually achieving only marginal improvements.

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

Fig. 4: Gordon Bell's "Dead Supercomputer Society" [keynote at ISCA 2000].

Not only sequential computing, but also parallel computing still computing in time, but concurrently. Locality is not known at machine level, since it is only a physical detail at lower levels - within the responsibility of completely different people, the hardware people. These seem to come from a different world, what is indicated not only by communication barriers. Hardware people seem to have their experiences from the world of antimatter. The submarine model does not support hardware / software co-design, since it cannot represent hardware and software as alternatives, nor any hardware / software interrelation.

3. Antimatter of Computing

Meanwhile a world of antimatter of computing is emerging, which is a world of data streams, controlled by data sequencers - in contrast to the instruction streams of the world of matter. From this point of view the data counter is the antiparticle of the program counter. In this world the von Neumann Computer machine finds its antimachine, the Xputer machine paradigm (X paradigm). In contrast to the world of instruction streams controlled by program counters, we have data streams controlled by data counters within data sequencers.

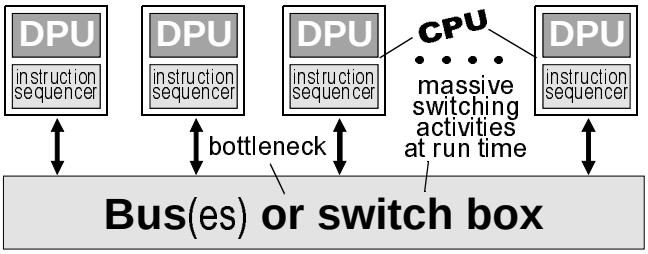
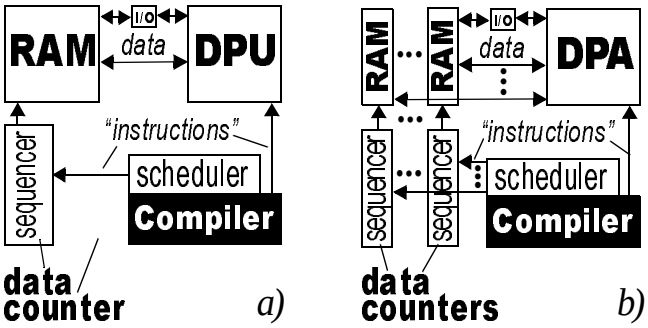


Fig. 5: Typical concurrent computing platform (memory not shown)

In the world of antimachines programming means data scheduling, instead of instruction scheduling needed for vN. We may simply model this data-stream-centric machine paradigm with an atom (figure 2 c), where the data flow positron (location of the positive data counter state) is spinning around the negative nucleus (data path resources) DPU stands for "datapath unit".

This chapter just introduces reconfigurable resources for the nucleus of data-stream-based machines. The machine paradigm around this nucleus will be treated in detail in chapter 5 of this paper. Note, to avoid confusion: we are not talking about the indeterministic "dataflow machines" [6], a dying exotic research area. This nice term has been annexed decades ago by these people for a very special concept (compare fig. 8 or fig. 20). So we should prefer the term "data streams", instead.



c)

machine category	Xputer [32] (no transputer!)
machine paradigm	deterministic
driven by:	(**no** *dataflow* [6]) data **stream**(s)
RA support	yes
engine principles	data sequencing
state register	(multiple) data counter(s)
communication path set-up	at load time
data path: resource	array of ALUs
data path: operation	parallel

Fig. 6: Machine paradigms :
a) data-procedural Xputer paradigm
b) parallel Xputer: general model of all data-stream-based machines - counterpart of ("antimatter to") the "von Neumann" machine,
c) properties of the machine paradigm.

3.1 Reconfigurable Logic

In contrast to the "von Neumann" machine, which does not support reconfigurable datapaths, its antimachine, the Xputer paradigm, supports both, hardwired and reconfigurable datapaths. To implement reconfigurable logic and reconfigurable datapaths, a technology is commercially available, had been a niche technology for quite a time. Now, however, a rapidly increasing number and attendance of conferences on re-configurable computing (the three most important ones, FCCM, FPGA, and FPL ([8], the eldest and largest), and workshops (RAW, RSP, ENREGLE etc.) as well as the adoption of this topic area by congresses like DAC, ASP-DAC, DATE, ISCAS, SPIE, and many others indicate, that reconfigurable platforms are heading for mainstream, supported by a rapidly growing large user base of HDL-savvy designers with FPGA experience. The echo on my double time slot embedded tutorials [9] [10] confirmed this by a quite number of further invitations ([9] et al.).

Reconfigurable platforms bring a new dimension to digital system development and have a strong impact on SoC Design. If the state of the art of the design flow would be as desired, their flexibility could support turn-around times of minutes instead of months for real time in-system debugging, profiling, verification, tuning, field-maintenance, and field upgrades.

term	granularity (path width)	configurable blocks
Reconfigurable Logic	fine grain (~1 bit)	CLBs: configurable logic blocks
Reconfigurable Computing	coarse grain (example: 16 or 32 bits)	rDPUs: reconfigurable data path units (for instance: ALU-like)
	multi-granular (supports slice bundling)	rDPU slices (example: 4 bits)

Fig. 7: How to avoid confusion with the term of "reconfigurable"

Terminology is an important aspect since we are far from consensus. In Reconfigurable Systems (RS) we should be clearly distinguish (fig. 7) between the areas of Recon-figurable Logic (RL) also called field-programmable logic (FPL), and, Reconfigurable Computing (RC). RL, also called field-programmable logic (FPL), where the typical product name is FPGA (field-programmable gate array, sometimes called PLD: for programmable logic device), deals with fine-grained reconfigurable circuits and systems. A typical fine-grained reconfigurable circuit consists of an array of CLBs (configurable logic blocks) with a path width of 1 bit, which are embedded in a reconfigurable interconnect fabrics. From an EDA point of view this RL level appears as a methodology of logic design for hardwired logic, but "on a strange platform", which is not really hardwired.

3.2 FPGAs

Figure 9 illustrates a typical FPGA architecture. FP (field-programmable) stands for the reconfigurability of CLB logic function select and for the reconfigurable interconnect "fabrics" made from switch boxes, tap switch boxes, wire pieces, as well as the configuration code input wrappers (not shown in this figure). GA (gate array) stands for the array of CLBs (programmable gates), which are embedded in the reconfigurable interconnect fabrics.

environment	platform
dataflow machines	indeterministic, dying R&D area
control-procedural programming (instruction stream-based computing	von-Neuman-like machine architectures
data-procedural programming (data stream-based computing)	Xputer architectures using DPU or DPA: coarse grain reconfigurable
- - - - - - -	hardwired
logic design	configurable GAs
	FPGAs: re-configurable
	dynamically reconfigurable

Fig. 8: Segmentation of digital system implementation disciplines

FPGA Vendors are stepping forward rapidly. FPGA vendors on the market are: Actel, Altera, Atmel, Cypress, Lattice, Lucent, Quicklogic, Triscend, and Xilinx. The PLD market is poised to grow, according to many industry watchers. Currently FPGA vendors have a relatively fast growing large user base of HDL-savvy designers. Cost differences between volume FPGAs and (volume) ASICs are shrinking. Driven by a growing large user base innovations occur more and more rapidly. Altera and Xilinx are currently the leading FPGA vendors (figure 10 a), both with a volume of sales almost 1.5 Bio US-$ in the year 2000. Advantages of PLDs are becoming apparent to the marketplace. Dataquest calls programmable logic the fastest growing segment of the entire semiconductor market. Mostly driven by telecom and wireless-communications applications growing 20% annually, PLD revenue will jump to $7.04 billion in 2004 [IC Insights].

Meanwhile a wide variety of hardwired IP cores is delivered on board of the same chip with the FPG. Due to Moore's law the FPGA vendors offer more and more products having microcontrollers like ARM, MIPS, PowerPC, or other RISC architectures, memory, peripheral circuitry and others, together with the FPGA on board of the same chip (fig. 14 b). A Xilinx FPGA, for example, has 4 PowerPCs on board. and 24 Conexant 3.125 Gb/s serial link cores providing a total of 75 Gb/s/chip link bandwidth. Such a symbiosis between FPGA and microprocessor corresponds fig. 14 c. Including the interface hardware it corresponds fig. 14 d.

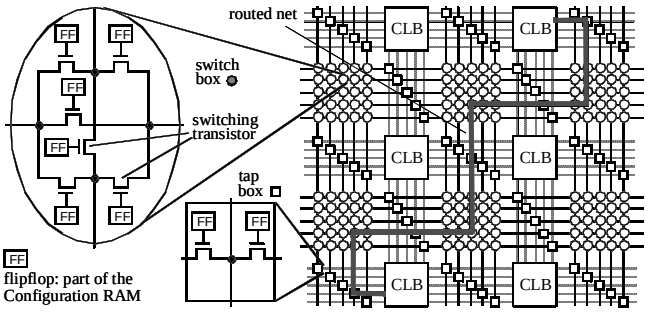


Fig. 9: FPGA architecture example (CLB: configurable logic block).

3.3 FPGA applications and R&D areas

Mostly driven by telecommunication networks and wireless-communications applications a growth has been predicted of 20% annually, and FPGA vendor revenue will jump to $7.04 billion in 2004. In a number of application areas like multimedia, wireless telecommunication, data communication and others, the throughput requirements are growing faster than Moore's law [12]. The current state of the art in FPGAs does not yet provide sufficient performance. Other application areas are image processing digital signal processing, encryption, automotive electronics, etc. (also see [13] within this conference proceedings volume). An emerging market is runtime reconfiguration, and, may be, or may be not, evolvable hardware.

One of the eldest FPGA markets is for ASIC emulation. Rapid Prototyping / ASIC Emulation is an important application of FPGAs. Simulation is the most time consuming step during the IC design flow, which may take even days or weeks. This can be replaced by Rapid Prototyping: much faster emulation on large FPGA arrays. By acquisitions the 3 major EDA vendors offer ASIC emulators, along with compilers: has acquired Quickturn (Cadence), IKOS (Synopsys), and Celaro (Mentor Graphics), also offering such service over the internet. For smaller designs less complex emulation boards may be used, like Logic emulation PWB (based on Xilinx Virtex, can emulate up to 3 million gates), and, the DN3000k10 ASIC Emulator from the Dini Group. The area of FPGA use for system prototyping has its own international workshop series on Rapid System Prototyping (RPS) [14]).

rank 2000	vendor	%	URL
1	Xilinx	42	xilinx.com
2	Altera	37	altera.com
3	Lattice	15	latticesemi.com
4	Actel	6	actel.com

a)

vendor	URL	remark
PACT	pactcorp.com	commercial
Quicksilver Technology	qstech.com	commercial
KressArray	kressarray.de	academic

b)

Fig. 10: a) coarse grain reconfigurable array IP core and EDA tool developer, b) top 4 FPGA manufacturers 2000 (total of 3.7 bio $).

Run time reconfiguration (RTR) provides a powerful advantage of FPGAs over ASICs [15]: smaller, faster circuits, simplified hardware interfacing, fewer IOBs; smaller, cheaper packages, simplified software interfaces. Exploding design cost and shrinking product life cycles of ASICs create a demand on RA usage for product longevity [16]. Performance is only one part of the story. The time has come to fully exploit their flexibility to support turn-around times of minutes instead of months for real time in-system debugging, profiling, verification, tuning, field-maintenance, and field-upgrades.

The new area of research in Evolvable Hardware (EH), or Evolutionary Methods (EM), based on Darwinistic methods, uses FPGAs to develop biologically inspired electronic systems [18]. Being a kind of revival of cybernetics or bionics, this is stimulated by the progress technology. Labels like „evolutionary" and the „DNA" metaphor helped to create a widely spread awareness, research funding programs in the EU, in Japan, Korea, and the USA, and many international conferences. In this highly visionary scene freaks do almost everything with genetic algorithms, even where simulated annealing is by orders of magnitude more efficient. Shake-out phenomena should be expected.

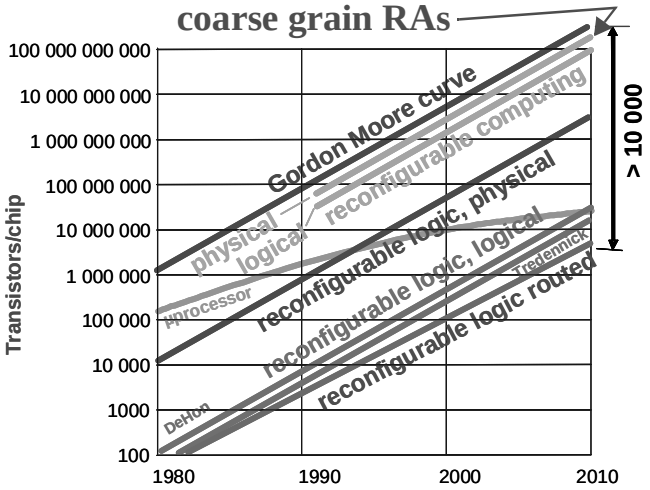


Fig. 11: Integration density of coarse grain reconfigurable computing arrays.

3.4 FPGA efficiency

Due to higher degree of regularity the growth rate of FPGA integration density (number of transistors per chip) is higher than that of general purpose microprocessors (compare fig. 11). The growth rate is about the same as that of semiconductor memory. But this is only the physical integration density. But the logical integration density, i. e. of the parts which directly serve the application, is another factor of 100 behind, so that in total it is 4 orders of magnitude behind the Gordon Moore curve. (see fig. 11). Compared to memory and other full-custom-style integrated circuits, fine grain reconfigurable circuits are highly area-inefficient [19]. Due to rough estimations [20] only about one percent of the chip area serves the real application, whereas the other 99 percent are reconfigurability overhead. About 10% area are needed for storage of configuration code, and, about 90% area are needed for routing resources like wire pieces and switches. Nick Tredennick estimates, that for each transistor serving directly the application, about 200 more transistors are needed for reconfigurability.

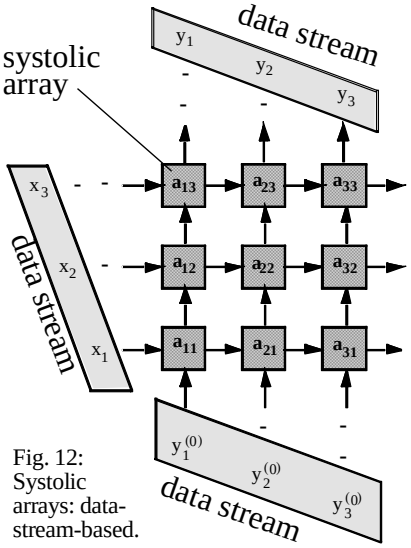


Fig. 12: Systolic arrays: data-stream-based.

The consequence of recon-figurability overhead is, compared to hardwired solutions, a higher power consumption (roughly by a factor of 10, see fig. 3) and lower switching speed or clock frequency (about a factor of 3 to 5). By re-design efforts reducing the clock speed a highly progressive improvement of power dissipation may be obtained, since reducing clock frequency by a factor of n yields a reduction of power dissipation by a factor of n^3 [21]. The design has to be re-optimized, since just tuning the clock would not yield this result. However, the reconfigurable solution is an order of magnitude more efficient than a software solution on a microprocessor (see fig. 3).

4. Configware Industry

Software does not run on antimachines. Generating classical procedural code the traditional compilers cannot be used to program antimachines, since structural code including data schedules is needed. This is not software as we know it. What we need is the anti part of software which is called configware (yet some people call it "IP cores").

Traditional procedural computing systems are run by software code downloaded into its RAM, what is the basis of basis of the extreme flexibility. This is the secret of success of the software industry: it is RAM-based (figure 15). But also using reconfigurable platforms (FPGAs or rDPAs) are RAM-based: structural code (configuration code) is downloaded into the "hidden" RAM of FPGAs. Are there any chances, that configware may repeat the success story of software?

Configware industry is emerging as the anti industry of the software industry. Being RAM-based config-ware industry is taking off to repeat the success story known from soft-ware. Tsugio Makimoto has predicted this more than ten years ago [22] [23]. Like software, also configware may be downloaded over the inter-net, or even via wireless channels. FPGA or rDPA functionality can be defined and even upgraded later at the customer's site - in contrast to the hardware it replaces: configware use means a change of the business model - providing shorter time to market and (FPGA) product longevity. Many system-level integrated products without reconfigurability will not be competitive.

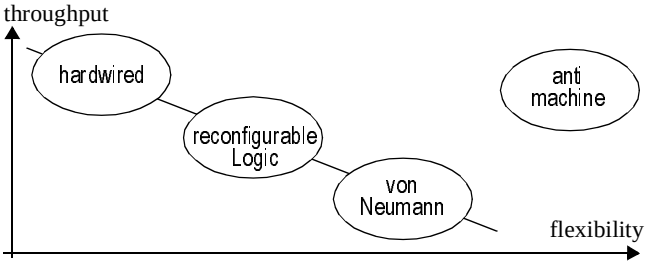


Fig. 13: Closing the gap - and beyond: by antimachine.

FPGAs are going into every type of application. Designer productivity and IP reuse are the key issues. Customers implementing more complex systems and even entire SoC (systems on chip) are looking how to shorten design times with so many gates, even entire systems on a chip - by predefined functional blocks to deal with more mundane or standard functions from various application areas. This led vendors to focus on intellectual property and the use of soft IP core code blocks. Currently most of the configware (reusable soft IP cores) is provided by the FPGA vendors. But the number of independent configware houses (soft IP core vendors) and design services is growing.

4.1 Configware Tools

A separate Configware tools market, comparable to the compiler and OS market in computers, separate from the (reconfigurable) hardware market already exists, since Cadence, Mentor Graphics and Synopsys just jumped into it by closing the back end gap down to creating configware code. The battle for market shares between EDA vendors and FPGA vendors has just been started.

The software market distinguishes applications and system software like operating systems, compilers etc. This also holds for the configware market, where we have application configware (soft IP cores) and systems configware (configware tools). This terminology should maintain the awareness of the dichotomy of informatics and anti informatics. "Configware tools" we mean all kinds of configware development and operation support tools and environments - not only software but also configware with FPGA operating system role (like for configuration management, runtime reconfiguration, interfacing the FPGA to its environment or to the internet, etc. ###

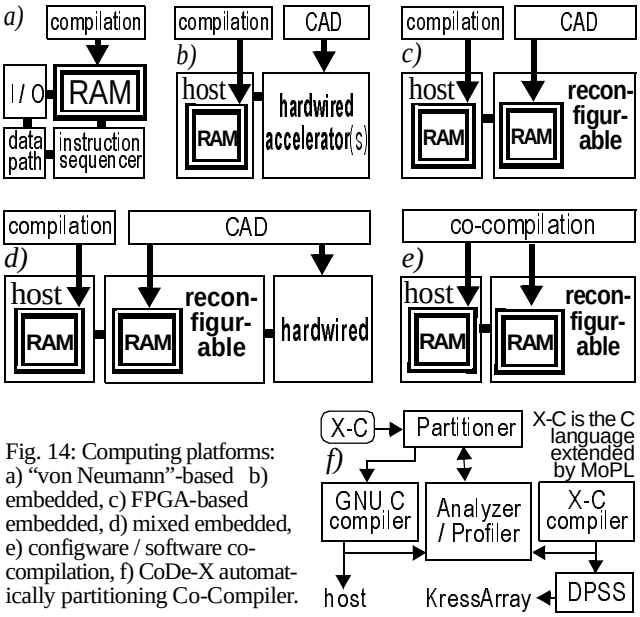


Fig. 14: Computing platforms: a) "von Neumann"-based b) embedded, c) FPGA-based embedded, d) mixed embedded, e) configware / software co-compilation, f) CoDe-X automatically partitioning Co-Compiler.

The mind set for the paradigm shift in configware implementation is still waiting for the broad break-through because of educational background problems. Historically driven R&D in this area is still dominated by hardware people. For this reason system configware (tools for configware implementation) are still mainly called EDA software (electronic design automation).

Being fabless the FPGA vendors Xilinx and Altera spend most of their higher qualified manpower in configware tool development, IP core development, application development, and related design services. Xilinx and Altera are more and more morphing into EDA companies. Also the major EDA companies (Cadence, Mentor Graphics, or Synopsys), show increasing awareness of the high volume market of the new needs for low cost, bullet-proof and self-supporting high quality zero maintenance configware tools for masses of designers, with a low EDA budget. Configware tools are the key enabler for the customer to obtain high quality FPGA-based products with good designer productivity. A good FPGA architecture is useless, if it is not efficiently supported by configware tools.

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

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

Fig. 15: The secret of success.

5. Asymmetry between matter and antimatter

Particle physicists have stated, that there is an asymmetry between matter and antimatter. Also in computing we have such an asymmetry. In our physics-like model the antimachine may have one (fig. 2 b) or several data stream positrons spinning around a negative nucleus (fig. 2 c). DPU stands for datapath unit and DPA for datapath array (an array of DPUs). The vN machine can have only one program counter, whereas its antimachine, the Xputer, may have one (figure 6 a) or multiple data counters (figure 6 b). On the antimatter side the "instructions" (here called "configuration code") and the data schedule are not fetched at run time, but downloaded already at loading time, or at design time, if DPU and DPA are not reconfigurable, There are no instruction streams read anymore ar run time. The data schedule is downloaded into the data sequencer [24] [25] [26] [27] [28], especially designed for the data-stream-based Xputers [29] [30] [31] [32]. Several Xputer architecture examples have been implemented [3] [30] [31] [32]. The table in fig. 18 gives a survey over the asymmetry between the worlds of matter and antimatter in informatics.

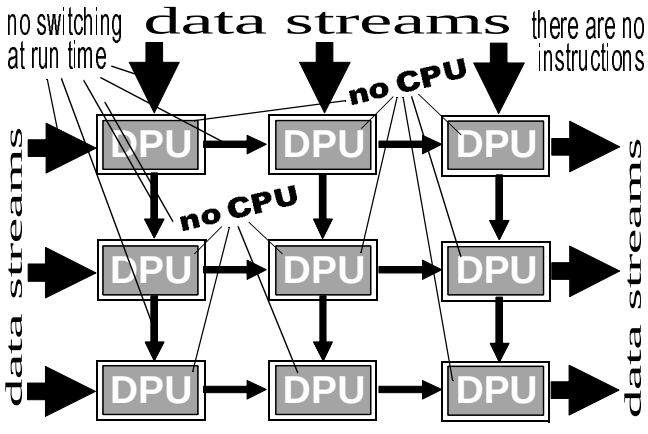


Fig. 16: Stream-based computing array example: transport-triggered exec.

5.1 The Antimachine

Why does the antimachine paradigm make sense? The von Neumann machine is no more the most flexible computing device (fig. 13). For reconfigurable datapaths von Neumann is the wrong machine paradigm, because its instruction set changes whenever a different configuration code is fetched, so that always a different instruction sequencer would be needed. In using reconfigurable DPUs or DPAs von Neumann architectures fall apart. Von Neumann is no more the most flexible computing device. For this case, like using FPGAs or other reconfigurable platforms (chapter 6.1) as a datapath, the antimachine makes sense. Instructions from the old world of normal matter are used for procedural programming (programming in the time domain), whereas for the antimachine configuration means structural programming by rearranging the internal structure of the datapath (programming in the space domain).

However, we do not need reconfigurable datapaths for running an Xputer. An antimachine can also run very well with a hardwired datapath. On old example of such a hardwired data-stream-based system is the systolic array [33] [34], having been a subject of research mainly during the eighties. A systolic array is a fully regular and uniform pipe network (example in fig. 12). Systolic arrays can be designed only for applications with strictly regular data dependencies. (For generalization of the systolic array into the supersystolic DPA or rDPA see chapter 6.1.)

The antimachine does not have a von Neumann bottleneck.

The systolic array people just specified the data streams they needed to run a particular array, but did not care how it is created. Because the classical systolic array people did not use a machine paradigm there are still many unsolved problems. Theoreticians often do not care about application. There have been examples, where unscrambling and scrambling the data streams took more time than the computation within the array. We need a machine paradigm as a general model for inclusion of runtime organization like for data streams between array, I/O, and memory. Such a common antimachine paradigm [30] [31] [32], architecture examples [35] [3] [29], and related implementation tools [36] [37] [38] [39] have been published up to more than a decade ago.

5.2 Stream-based Computing

New machine principles are needed for RC. Classical parallel processing relying on concurrent processes is not the way to go, since its fundamental architecture relies on a uniprocessor [40]. Generally classical parallel processing has not been successful (see fig. 4). Arrays or other ensembles of CPUs are too difficult to program, and often the run-time overhead is too high, except for a few special application areas favored by Amdahl's law. All these problems stem from the fact, that the operation of CPUs or of arrays of CPUs are control-flow-based. We need an alternative paradigm which is not control-flow-based. For details see section 8.

Stream-based ALU arrays or DPU arrays (DPU stands for Data Path Unit). Its alternative, (locally) distributed computing, uses arrays of ALUs (or other DPUs) instead of arrays of CPUs. The DPUs within such an array are interconnected form a pipe network, i. e. a network of multiple pipelines in terms of multiple DPUs (Data Path Units) without program controllers, not multiple CPUs. Tailored multiple data streams are pumped from outside through this pipe network. That's why these arrays are called "stream-based" arrays. The KressArray is an early stream-based DPU array, which is reconfigurable [19] [36] [37]. There's no CPU. There's nothing "central". It's fully distributed, with lots of different DPUs containing adders, registers, multipliers -- just what's needed for direct mapping of the algorithm onto the architecture. As soon as the architecture is defined, the data streams needed are obtained by using a scheduling algorithm. For data stream creation see section 5.3 on the memory communication gap

5.3 Stream-based Memory Organization

Theoreticians often do not care about application. The classical systolic array people just specified data streams, but did not care about their implementation. Meanwhile generation and organization of data streams between memory and the nucleus processing resources has become a maturing design and optimization methodology (characterized by terms like data transfer and storage exploration: DTSE) to cope with memory bandwidth and power dissipation problems.

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

There are two kinds of scenarios to cope with the traditional memory communication gap still widening. First, in streaming data applications like in DSP the data streams can be split up into parallel streams to be interfaced with multiple I/O ports of rDPAs (e. g. fig. 17). Second, artificial multiple "data streams" from/to multiple memory banks (fig. 17) can be generated by multiple data sequencers, being distributed over the rDPU array [24]. Data sequencer principles have been developed, which avoid control overhead [40]. (r)DPAs provide a much more efficient way to cope with memory communication bandwidth problems than classical concurrent computing.

Caches do not improve throughput, since in data-stream-based computing there are mainly hardwarized loops, but no instruction streams. The processor / memory communication bandwidth gap may be dramatic in extremely data-intensive RA applications. The demand stimulates the development of “embedded memory optimization” methods using data transfer and storage optimization and data reuse transformation [41] [42] [43], loop transformations [39] [44] [45], smart memory interface, also with 2-D memory organization [46] [3], generic address generators [25] [47] [26], as well as a mapping method, where both, data sequencer DPUs and application rDPUs can be mapped onto the same rDPA [24] [9].

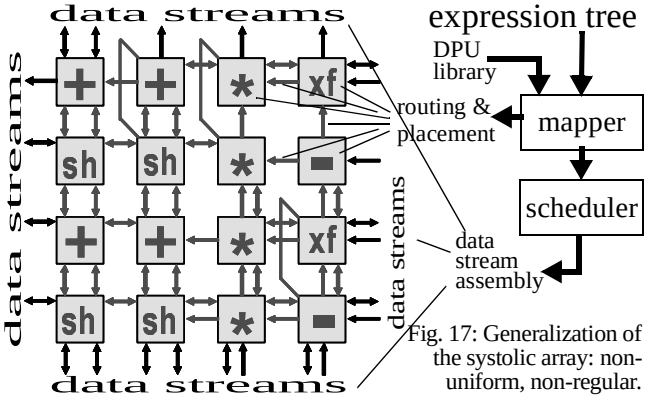


Fig. 17: Generalization of the systolic array: non-uniform, non-regular.

5.4 Compilers for the Antimachine

Multiple data streams being piped through a rDPU array are the key to massive parallelization. This has been proven by systolic array researchers who used linear projections as a synthesis method. However, for rDPA synthesis no linear projection is used, but simulated annealing instead, to avoid restrictions to applications only with regular data dependencies. By turning away from linear projections we obtain a generalization of the systolic array also to support inhomogenous irregular arrays (fig. 17). This completely removes the restriction to applications with only regular data dependencies. After (r)DPA synthesis a scheduling algorithm prepares the data streams (more details in section 6.2).

Such a compiler including a mapper and scheduler DPSS (Data Path Synthesis System) has been published in 1995 [36]. In 2000 a KressArray design space Xplorer has been implemented, which supports a generically defined KressArray family covering any path width and a wide variety of inter-rDPU interconnect resources [37]. The Xplorer also supports mapping memory communication resources together with the application onto the same rDPA (fig. 21) [24] [26].

Also experimental high level programming languages for that new paradigm have been implemented, within the context of MoM (map-oriented machine) architectures [35] [3] [29]. Instead of a "control flow" sublanguage a "data stream" sublanguage like *MoPL* [48] recursively defines *data goto, data jumps, data loops, nested data loops,* and *parallel data loops*. Also to solve the memory bandwidth problem the Xputer paradigm is much more promising than "von Neumann".

	matter	antimatter
machine paradigm	von Neumann (vN)	Xputer
programming	procedural	structural (re)configuration (super "instruction" fetch)
		data scheduling
"instruction" fetch	at run time	at loading time
program execution at run time	instruction schedule	data schedule
operation spin	instruction flow	data stream(s)
nucleus resources	CPU	(r)DPU, or, (r)DPA (nothing central - no CPU)
	hardwired	hardwired or reconfigurable
parallelism	multiple machines	single machine

Fig. 18: Asymmetry between matter and antimatter (compare fig. 2).

5.5 Antimachine Synthesis

On hardwired DPU array basis the BWRC [2] is developing an entire "chip in a day" design methodology for data-stream-based machines. by direct mapping of algorithms onto high-level, pre-characterized macros (library elements, parameterized blocks, like FFTs of different sizes, and a Viterbi decoder), wired together from a Simulink data-flow diagram. An automated flow goes through module generation, synthesis and layout. The easiest place to make the most profound optimizations is at the system level, where one needs to know what the implications are of the algorithmic choices you're making. Compared to full custom microprocessor design, BWRC people give up easily a factor of two or three in speed - but for gaining a factor of 100 in area efficiency. BWRC got rid of difficult problems by using relatively low clock rates. This flow gives a much more efficient way to solve the problem. Adapting the goals or Broderson's group to reconfigurable computing, where no physical design has to be done: could this end up with a FPGA-based "chip-in-one-hour" implementation?

6. Reconfigurable Computing

Reconfigurable Computing (RC) [9] [10] [11] [12] is the reconfigurable form of parallel computing, where (coarse-grained) DPUs or DPAs and the interconnect resources are reconfigurable, so that the pipe network is configured into a rDPA like the KressArray [36], based on the most straight-forward method of stream-based computing.

Stream-based (r)DPA use (example in fig. 16) is more efficient than normal matter concurrency (fig. 1 d). Classical parallel processing relying on concurrent processes (example in fig. 5) is not the way to go, since its fundamental architecture relies on a uniprocessor, where pipelining brings only marginal improvements. Its alternative, (locally) distributed computing, i. e. DPU array computing, however, means parallelism by an application-specific pipe network in terms of multiple DPUs, not multiple CPUs (fig. 16).

Stream-based RC (with rDPU arrays: rDPAs) is urgently needed, since in application areas like multimedia, wireless telecommunication, data communication and others, the throughput requirements are growing faster than Moore's law [12]. The current state of the art in FPGAs does not yet provide sufficient performance. For flexibility and low power the only viable solution is one with rDPAs like offered by providers like PACT [50].

6.1 Coarse-Grained Reconfigurable Datapaths

Reconfigurable computing, where typical products are reconfigurable arrays (RA) or reconfigurable data path arrays (rDPA), deals with coarse-grained reconfigurable circuits and systems. A typical coarse-grained reconfigurable circuit consists of an array of CFBs (configurable functional blocks), also called reconfigurable datapath unit (rDPU), with a wide path width like, for instance, 16 or 32 bit. Figure 19 shows a 32 bit example with a SNN filter algorithm mapped onto a 160 DPU KressArray. Here we may also say, the RA granularity is 16 or 32 bit. Also RAs with multiple granularity are known, where medium grain rDPU slices of pathwidth W (4 bits, for example) may be bundled for rDPUs with a path width of N x W, where N is a non-negative integer.

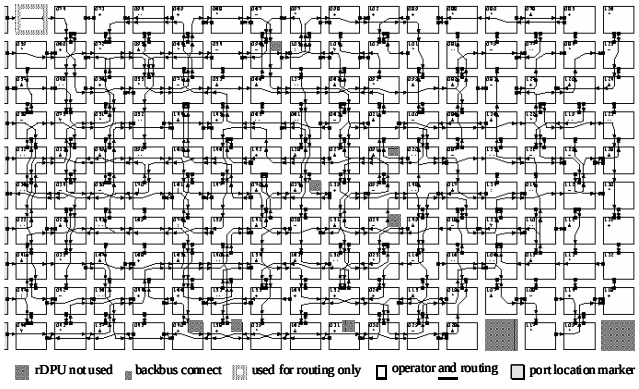


Fig. 19: 32 bit KressArray with SNN filter algorithm mapped onto it.

A major benefit is massive reduction of configuration memory and configuration time, and drastic complexity reduction of the placement and routing problem because only a few CFBs are needed. Several architectures are briefly outlined in [9]. Because (r)DPAs and even their routing switches have regular structure potential, a drastically higher integration density can be obtained (see fig. 3), than with FPGAs. Physical and even the logical integration density come close to the Gordon Moore curve (figure 11). Performance may be substantially faster. Energy efficiency practically reaches that of hardwired solutions (figure 3) - about two orders o magnitude better than with software on a microprocessor (figure 3). Also (r)DPAs may have microcontrollers (fig. 14 c) and hardwired interfacing circuitry on board of the same chip (fig. 14 d).

On non-reconfigurable DPU array basis the BWRC [2] is developing a similar design methodology by direct mapping of algorithms onto high-level, pre-characterized library elements, wired together from a Simulink data-flow diagram and passed to module generation, synthesis and layout ?

6.2 Configware tools for rDPAs

Along with a mapper and scheduler the early DPSS (Data Path Synthesis System) has been published in 1995 [36]. In 2000 a KressArray design space Xplorer has been implemented [37] [38] [35], which supports a generically defined KressArray family covering any path width and a wide variety of inter-rDPU interconnect resources.

There are four different routes to implement DPAs: (1) design of fully universal rDPAs (highly area-inefficient), (2) mapping the rDPA onto a sufficiently large FPGA or combination of several FPGAs, if available, (3) design of an application-domain-specific rDPA, (4) design a hardwired DPA (not reconfigurable: application-specific).

Configware tools (EDA) for DPAs or rDPAs are a key issue. All four routes have in common, that mainly the same design flow front end may be used for all of them. The tendency goes toward stream-based DPU arrays and there is no principle difference, whether the DPU array is hardwired or reconfigurable. The only important difference is binding time of placement and routing: before fabrication, or, after fabrication.

machine category		Computer ("v. Neumann")	Xputer [32] (no transputer!)	dataflow machine [6]
machine paradigm		procedural sequencing: deterministic		arbitration-driven
				indeterministic
driven by:		control flow	data stream(s)	token flow
reconfigurability support		no	yes	no
engine principles		instruction sequencing	data sequencing	arbiter decides order of execution
state register		program counter	(multiple) data counter(s)	none
communication path set-up		at run time	at load time	at run time
data path	resource	single ALU	FPGA or (r)DPU array	single ALU
	operation	sequential	parallel	sequential

Fig. 20: Machine paradigms: comparing Computer and Xputer.

7. No Annihilation in Computing

Whenever in an accelerator or elsewhere a particle and its antiparticle collide, they disappear, what particle physicists call annihilation. But in computing science we do not crash matter into antimatter. We can avoid annihilation by arranging such meetings very carefully. We do it all the time in hardware / software co-design (figure 14 b) and configware / software co-design (figure 14 c), or, in hardware / configware / software co-design (figure 14 d).

About 20 billion of microprocessors are used in embedded systems and 90% of all existing software is running on them [5]. Probably 90% of all programmers will write software for embedded applications by 2010. This growing market share of embedded system applications stimulates a strong tendency to overcome separate design flows for configware development and software development. First versions of EDA tools integrating the flows into configware / software co-design are emerging from academia, and soon from FPGA or EDA vendors.

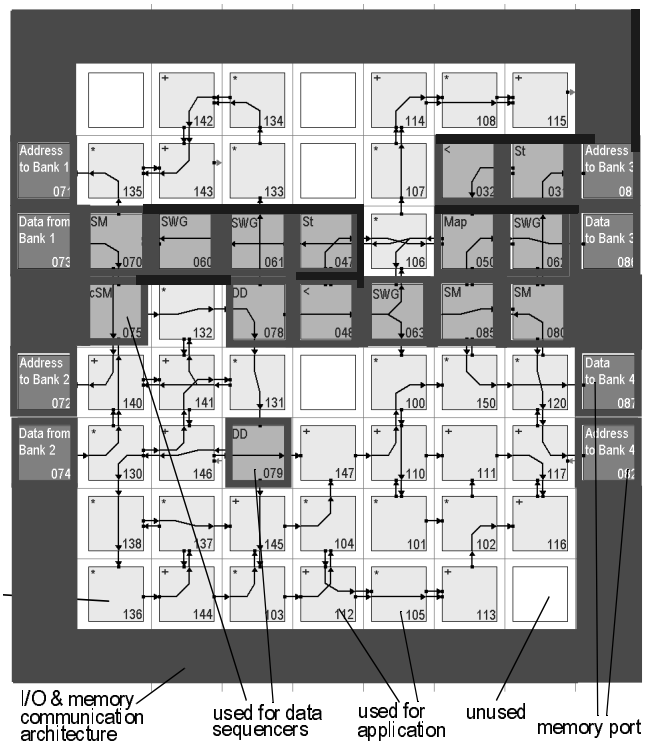


Fig. 21: Mapping application (linear filter) and memory communication architecture (dark background) onto the same KressArray, including the address ports and the data ports to 4 different memory banks (5 of 8 memory port connects are routed through application DPUs).

8. Co-compilation Techniques for RC

Newer commercial platforms include all on a single chip, combining a core processor (ARM, or MIPS), embedded memory and reconfigurable logic. Sequential code is downloaded to the host's RAM. But accelerators are still implemented by CAD (figure 23 c), and a C compiler is only an isolated tool, and, software / configware partitioning is still done manually [51] [53] [54] [55] [45] [57], so that massive hardware expertise is needed to implement accelerators.

Like microprocessor use, also reconfigurable device application is RAM-based, but by structural programming (also called "(re)configuration") instead of procedural programming. Now both, host and accelerator are RAM-based (fig. 14 c): a new approach to SoC design. But the "von Neumann" paradigm does not support soft datapaths because "instruction fetch" is not done at run time, and, since most reconfigurable computing arrays do not run parallel processes, but multiple pipe networks instead. A transition from CAD (fig. 14 e) to compilation is needed, and from hardware/software co-design to configware/software co-compilation, supporting the emerging trend to platform-based SoC design.

From Makimoto's wave model point of view [19] [22] [23] this reveals a huge historical gap: first wave methods "stone age" or "rubylith age" methods (fig. 23 a) for designing third wave platforms (fig. 23 c). Due to this huge gap (missing the second phase which as happened with the microprocessor: fig. 23 b): the RAM-based nature of the physical platform is mainly ignored (at this level). To close this EDA methodology gap is the dramatic challenge to EDA industry: more and more using high level programming languages, or, yet using HDLs (hardware description languages) as sources to describe design problems.

8.1 Co-Compilation

Using rDPAs or antimachines as accelerators supports a change of market structure by migration of accelerator implementation from IC vendor to customer. Since "normal" compilers do not support soft datapaths nor stream-based machines (fig. 20 d compares the properties of both paradigm)s, this demands automatic compilation from high level programming language sources onto both, host and RA: *co-compilation* including automatic software/configware partitioning. CoDe-X is the first such co-compilation environment having been implemented ([39] fig. 14 f), which partitions mainly by identifying loops suitable for parallelizing transformation [19] [39] [45] into code downloadable to the MoM accelerator Xputer. The MoM (Map-oriented Machine) is an Xputer architecture for data-stream-based computing [35] [3] [29].

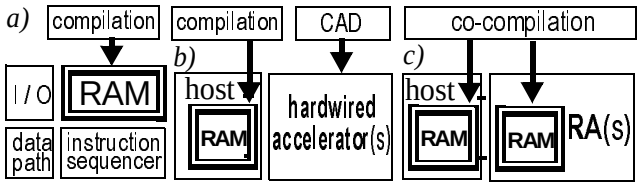


Fig. 22: History of computing platforms: a) "von Neumann"-based, b) embedded, c) emerging.

9. The Giga FPGA

The Giga FPGA is emerging. It can be predicted, that within some more years FPGAs with 100 million gates or more will be available commercially. Onto a single chip of such a platform up to about a hundred soft processor cores can be mapped, leaving sufficient area to other on-board resources like RAM, register files, peripheral circuits and miscellaneous others. Because of Moore's law we may finally end up with FPGAs of almost infinite size, so do not need to care a lot about efficiency.

9.1 Soft CPUs

The success of companies like Tensilica indicate that there is a growing market ASIPs (Application-Specific Instructionset Processors). Even more flexibility can be obtained by a soft CPU. Configware providers meanwhile have discovered CPUs having been developed as soft IP cores to be mapped onto an FPGA, also called FPGA CPU, or, soft CPU, like the 32 bit MicroBlaze (125 MHz, Xilinx), the Nios (16-bit instruction code, Altera), which can be configured as (8-), 16- and 32-bit data paths, and Leon, a SPARC clone by Gaisler Research. Using the usual FPGA design flow the soft CPU IP cores can be generated from VHDL or Verilog originally targeted at a hardwired implementation. The table in fig. 23 lists more soft CPU examples. Soft CPUs are also a well accepted academic research area. Some soft CPU core examples have been implemented by universities, like UCSC (already 1990), Märaldalen University, Eskilstuna, Chalmers Univ., Goeteborg, Cornell, Georgia Tech, Hiroshima City University, Michigan State, Universidad de Valladolid, Virginia Tech, Washington University, St. Louis, New Mexico Tech, UC Riverside, Tokai University.

soft CPU core	architecture	platform / remarks
MicroBlaze 125 MHz 70 D-MIPS	32 bit standard RISC 32 reg. by 32 LUT RAM-based reg.	Xilinx - Virtex-II fabric (possible: up to 100 on one FPGA)
Nios	16-bit instr. set	Altera - Mercury
Nios 50 MHz	32-bit instr. set	Altera 22 D-MIPS
Nios	8 bit	Altera - Mercury
gr1040	16-bit	Altera - Mercury
gr1050	32-bit	FLEX10K30 or EPF6016
DSPuva16	i8080A	Spartan-II
Leon 25 Mhz	SPARC	Xilinx XCV800-6
ARM7 clone	ARM RISC	
uP1232 8-bit	CISC, 32 reg.	200 XC4000E CLBs
REGIS	8 bit instr. + ext. ROM	2 XILINX 3020 LCA
Reliance-1	12 bit DSP	Lattice 4 isp30256, 4 isp1016
1Popcorn-1	8 bit CISC	Altera, Lattice, Xilinx
Acorn-1		1 Flex 10K20
YARD-1A	16-b.RISC 2-opd. Instr.	old Xilinx FPGA Board
xr16	RISC integer C	SpartanXL

Fig. 23: Soft CPU IP cores currently available.

9.2 Soft rDPUs and rDPAs

Soft rDPAs will be an alternative. With coming Giga FPGAs even soft DPAs are within reach. For soft DPAs we don't need to wait for the coming Giga FPGA. Already now it is conveniently possible to map 32 MicroBlaze or 64 Nios onto a FPGA. The Giga FPGA could mean, that even coarse grain reconfigurable systems like those from PACT Corp. [50] can be mapped onto a (fine grain) FPGA (fig. 10). It is quite promising, that the performance disadvantage of lower clock frequency can be fixed by utilizing the better flexibility and by a much higher degree of parallelism (compare fig. 13).

9.3 Bleeding Edge Designs

Excessive optimization will be needed for Giga FPGAs. If there is an infinite amount of gates available on a chip just compilation techniques can be used in front of the (gate level) configuration code generator. But for FPGAs one million gates (state of the art, or 10 million gates: may be in 2003) is far away from "infinite resources". The consequence is, that for closing the gap excessive optimization is required. This means, that leading edge designs ("bleeding edge designs") are achievable only partly with sophisticated EDA tools, so that hardware expertise is inevitable for the designer.

9.4 New FPGA architectures needed

A major problem of configware industry in competing with software industry is the fact, that no FPGA architecture is available which is fully scalable and which supports fully relocatable configuration code (fig. 15). The consequence is the need for excessive re-compilation and re-debugging as soon as another FPGA type is used. It is an unanswered question, wether such a FPGA architecture is physically feasible. But it seems to be feasible with an innovative FPGA in connection with a new configware tool tailored to solve this problem. Without solving this problem it is very difficult for configware industry to repeat the success story of the software industry.

10. Conclusions

The paper has given a survey on reconfigurable logic and reconfigurable computing and its R&D branches and has pointed out future trends driven by technology progress and innovations in EDA. Deep submicron allows SoC implementation, and the silicon IP business reduces entry barriers for newcomers and turns infrastructures of existing players into liability [16]. The availability of reconfigurable platforms and related EDA tools

Many system-level integrated future products without reconfigurability will not be competitive. Instead of technology progress better architectures by reconfigurable platform usage will be the key to keep up the current innovation speed beyond the limits of silicon.

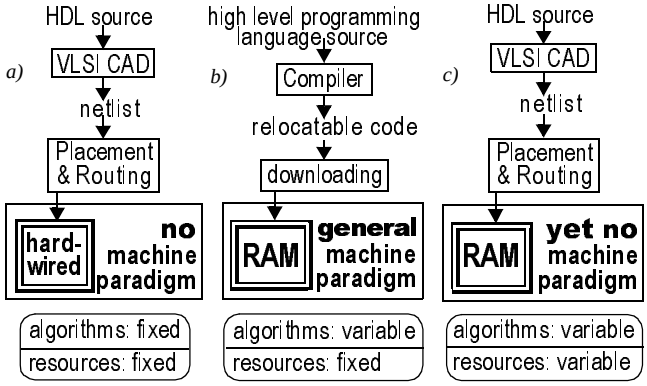


Fig. 23: Synthesis a) hardwired, b) "von Neumann", c) reconfigurable.

11. Literature

1. for viewgraphs of an earlier version of this talk follow link on page http://kressarray.de
2. C. Chang, K. Kuusilinna, R. Broderson, G. Wright; The Biggascale Emulation Engine; FPGA 2002, 10th Int'l Symp. on Field-programmable Gate Arrays; Monterey, CA, Feb 24 - 26, 2002
3. R. Hartenstein, A. Hirschbiel, M. Weber: MoM - a partly custom-design architecture compared to standard hardware; IEEE CompEuro 1989
4. R. Hartenstein (invited paper): High-Performance Computing: über Szenen und Krisen; GI/ITG Workshop on Custom Computing, Dagstuhl, June 1996
5. F. J. Rammig: Entwurf eingebetteter verteilter Systeme; Anniversary Conference 10 Jahre EAS und DaSS 2002; Dresden, Germany, March 21-22, 2002
6. D. Gajski et al.: A second opinion on dataflow machines; Computer, Feb 1982
7. http://xputers.informatik.uni-kl.de/xputer/history.html
8. http://fpl.org
9. R. Hartenstein (embedded tutorial): A Decade of Research on Reconfigurable Architectures - a Visionary Retrospective; DATE 2001, Munich, March 2001
10. R. Hartenstein (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; ASP-DAC'01, Yokohama, Japan, Jan 30 - Feb. 2, 2001
11. R. Hartenstein: Trends in Reconfigurable Logic and Reconfigurable Computing; ICECS 2002, Dubrovnik, Croatia, Sept 15-18, 2002
12. R. Hartenstein (invited embedded tutorial:): Reconfigurable Computing: the Roadmap to a New Business Model and its Impact on SoC Design; SBCCI 2001 - 14th Symposium on Integrated Circuits and Systems Design, Pirenopolis, DF, Brazil, September 10-15, 2001

13J. Becker: Configurable Systems-on-Chip: Commercial and Academic Ap-proaches; ICECS 2002, Dubrovnik, Croatia, Sept 15-18, 2002

14. 13th IEEE Int'l Workshop on Rapid System Prototyping; Darmstadt, Germany, July 1-3, 2002 http://www.rsp-workshop.org
15. S. Guccione (keynote): Reconfigurable Computing at Xilinx; DSD 2001, Digital System Design Symposium, Warsaw, Poland, Sep 4-6, 2001
16. T. Kean (keynote): It's FPL, Jim - but not as we know it! Market Opportunities for the new Commercial Architectures; in [17]
17. R. Hartenstein, H. Grünbacher (Editors): The Roadmap to Reconfigurable computing - Proc. FPL2000, Aug. 27-30, 2000; LNCS, Springer-Verlag 2000
18. M. Sipper et al:: Proc: "2nd Int'l Conf. on Evolvable Systems: From Biology to Hardware (ICES98), LNCS vol. 1478, Springer-Verlag, 1998
19. R. Hartenstein: The Microprocessor is no more General Purpose (invited paper), Proc. ISIS'97, Austin, Texas, USA, Oct. 8-10, 1997.
20. A. DeHon: Reconfigurable Architectures for General Purpose Com-puting; rep. no. AITR 1586, MIT AI Lab, 1996
21. M. J. Flynn, P. Hung, K. Rudd: Deep Submicron Microprocessor Design Issues; IEEE Micro July-Aug '99
22. T. Makimoto: The Rising Wave of Field-Programmability; in: [17]
23. T. Makimoto, D. Manners: Living with the Chip; Chapman & Hall, 1993
24. M. Herz, R. Hartenstein, M. Miranda, E. Brockmeyer, F. Catthoor: Memory Addressing Organization for Stream-based Reconfigurable Computing; ICECS 2002, Dubrovnik, Croatia, Sept 15-18, 2002
25. M. Herz, et al.: A Novel Sequencer Hardware for Application Specific Computing; Proc. ASAP'97, Zurich, Switzerland, July 14-16, 1997
26. M. Herz: High Performance Memory Communication Architectures for Coarse-Grained Reconfigurable Computing Architectures; Ph. D. Dissertation, Universitaet Kaiserslautern, January 2001
27. R. Hartenstein, A. Hirschbiel, M. Riedmüller, K. Schmidt, M.Weber: A High Performance Machine Paradigm Based on Auto-Sequencing Data Memory; HICSS-24, Hawaii Int'l. Conference on System Sciences, Koloa, Hawaii, 1991
28. Reiner W. Hartenstein, Helmut Reinig: Novel Sequencer Hardware for High-Speed Signal Processing; Workshop on Design Methodologies for Microelectronics, Smolenice Castle, Slovakia, September 1995
29. R. Hartenstein, A. Hirschbiel, M. Weber: MOM - Map Oriented Machine; in: E. Chiricozzi, A. D'Amico: Parallel Processing and Applications, North-Holland, 1988
30. R. Hartenstein et. al.: A Novel ASIC Design Approach Based on a New Machine Paradigm; IEEE J.SSC, Volume 26, No. 7, July 1991.
31. R. Hartenstein et al.: A Novel Paradigm of Parallel Computation and its Use to Implement Simple High Performance Hardware; InfoJapan'90, 30th Anniversary of the Computer Society of Japan, Tokyo, Japan, 1990.
32. R. Hartenstein, A. Hirschbiel, K. Schmidt, M. Weber: A Novel Paradigm of Parallel Computation and its Use to Implement Simple High-Performance-HW; Future Generation Computer Systems 7, 91/92, North Holland - invited reprint of [31]
33. N. Petkov: N. Petkov. Systolic Parallel Processing; North-Holland, 1993
34. P. Quinton et al.: Systolic Algorithms & Architectures; Prentice Hall, 1991
35. R. Hartenstein, M. Herz, T. Hoffmann, U. Nageldinger: Generation of Design Suggestions for Coarse-Grain Reconfigurable Architectures; in [17]
36. R. Kress et al.: A Datapath Synthesis System for the Reconfigurable Datapath Architecture; ASP-DAC'95, Chiba, Japan, Aug. 29 - Sept. 1, 1995
37. U. Nageldinger et al.: KressArray Xplorer: A New CAD Environment to Optimize Reconfigurable Datapath Array Architectures; ASP-DAC, Yokohama, Japan, Jan. 25-28, 2000.
38. U. Nageldinger: Design-Space Exploration for Coarse Grained Reconfigurable Architectures; Dissertation, Universitaet Kaiserslautern, June 2001
39. J. Becker: A Partitioning Compiler for Computers with Xputer-based Accelerators; Ph. D. dissertation, Kaiserslautern University, 1997.

AM	antimachine
ASIP	application-specific instruction set processor
ASPP	application-specific programmable product
CFB	configurable function block
CLB	configurable logic block
cSoC	configurable SoC
DPSS	datapath synthesis system
DTSE	data transfer and storage exploration
DPA	DPU array
DPU	datapath unit
EDA	electronic design automation
FPGA	field-programmable gate array
FPL	field-programmable logic
GAG	generic address generator
IP	intellectual property
MOPS	million operations per second
OS	operating system
PLD	programmable logic device
RA	reconfigurable array
RC	reconfigurable computing
rDPU	reconfigurable DPU
rDPA	reconfigurable DPA
RC	reconfigurable computing
RL	reconfigurable logic
am	reconfigurable machine
SBC	stream-based computing
SoC	system on a chip
vN	von Neumann

Fig. 24: Acronyms used in this paper.

40. R. Hartenstein (invited post conference tutorial): Enabling Technologies for Reconfigurable Computing 3rd Workshop.on Enabling Technologies for System-on-Chip Development November 21, 2001, Tampere, Finland
41. F. Cathoor et al.: Custom Memory Management Methodology; Kluwer, 1998
42. N. D. Zervas et al.: Data-Reuse Exploration for Low-Power Realization of Multimedia Applications on Embedded Cores; PATMOS'99, Kos Island, Greece, Oct 1999
43. S. Kougia et al.: Analytical Exploration of Power Efficient Data-Reuse Transformations on Multimedia Applications; ICASSP'2001, Salt Lake City, May 2001
44. D. Kulkarni, M. Stumm: Loop and Data Transformations: A tutorial; CSRI-337, Computer Systems Research Institute, University of Toronto, June 1993
45. K. Schmidt: A Program Partitioning, Restructuring, and Mapping Method for Xputers; Ph.D. Thesis, University of Kaiserslautern 1994
46. R.W. Hartenstein, A.G. Hirschbiel, M. Weber: A Flexible Architecture for Image Processing; Microprocessing and Microprogramming, vol 21, pp 65-72, 1987
47. R. Hartenstein, A. Hirschbiel, K. Schmidt, M. Weber: A Novel ASIC Design Approach based on a New Machine Paradigm; Proc. ESSCIRC`90, Grenoble, France
48. A. Ast, J. Becker, R. Hartenstein, R. Kress, H. Reinig, K. Schmidt: Data-procedural Languages for FPL-based Machines; in [49]
49. R. Hartenstein, M. Servit (Editors): Proc. FPL'94, Prague, Czech Republic, Sept. 7-10, 1994, LNCS, Springer Verlag, 1994
50. http://pactcorp.com
51. T. J. Callahan and J. Wawrzynek: Instruction-Level Parallelism for Reconfigurable Computing; in [52] pp. 248-257.
52. R. Hartenstein, A. Keevallik (Editors): Proc. FPL'98, Tallinn, Estonia, Aug. 31- Sept. 3, 1998, LNCS, Springer Verlag, 1998
53. M. Budiu and S. C. Goldstein: Fast Compilation for Pipelined Reconfigurable Fabrics; Proc. FPGA'99, Monterey, Feb. 1999, pp. 135-143.
54. M. Weinhardt, W. Luk: Pipeline Vectorization for Reconfigurable Systems; Proc. IEEE FCCM, April 1999
55. M. Gokhale, J. Stone: NAPA C: Compiling for a hybrid RISC / FPGA architecture; Proc. IEEE FCCM April 1998
56. R. Hartenstein (invited talk): Stream-based Arrays: Converging Design Flows for both, Reconfigurable and Hardwired; IFIP Int'l Conf. on Very Large Scale Integration (VLSI-SoC 2001), Dec. 2- 4, 2001, Montpellier, France
57. J. Becker et al.: A General Approach in System Design Integrating Reconfigurable Accelerators; Proc. IEEE ISIS'96; Austin, TX, Oct. 9-11, 1996