Are we Really Ready for the Breakthrough ?

(keynote) Reiner Hartenstein, fellow, IEEE
Kaiserslautern University of Technology
http://hartenstein.de

Abstract. *The paper outlines the key issues and fundamentals of morphware as a discipline of its own, and, as part of modern computing sciences. It discusses, what is needed for the break-fhrough.*

Embedded systems, SoC, or PC boards, with, or without morphware included, mostly means digital computing on a variety of mixed platforms. (This paper does not deal with analog circuitry.) Usually at least one von-Neumann-like micro controller or microcomputer is included in such a mix. Von-Neumann-like controllers or microprocessors are very important because of their enormous flexibility. Its simple machine paradigm is an important common model to support programming and to educate masses of programmers.

Software industry's secret of success. The von Neumann paradigm is the driving force behind the success of software industry, by its simplicity helping to focus language design and compilation techniques to a usefully narrow design space, and, by helping to educate masses of programmers. Prerequisite of this success is the fact, that its operational principles are RAM-based, yielding a seemingly almost unlimited flexibility, and scalability. Another driving force is compatibility provided by processor marketing policy and dominance.

Growing requirements. The main problem of an increasing number of application areas is the exponentially increasing demand for high performance. In an increasing number of applications the knowledge from classical computing science is less and less sufficient to meet the requirements. Since the processor / memory communication gap widens more and more, despite of the more recent availability of memory chips with faster interfaces. For many important application areas classical parallelism by multiple von-Neumann-like processors cannot provide the performance needed. For cellular wireless communication the performance requirements grow faster than Moore's law [1] [2].

Distributed Memories. The rapidly growing new R&D area and IP core market segment of distributed memory [3] provides ways to vanquish the consequences of the von Neumann bottleneck by skillfully tailored embedded system architectures. Two main directions can be distinguished: generating special architectures with application-specific address generators [4], or, more flexible architectures with general purpose address generators [5]. Example application areas are accelerators for DSP, multimedia, or wireless communication. From the SoC system level point of view such architectures and their address generators are sources and sinks of multiple ***data streams*** (fig. 1), "programmed" by ***flowware*** - in contrast to software which programs ***instruction streams***. For terminology see fig. 1 [6].

Extremely Conservative. Computing sciences are extremely conservative. After about a dozen of technology generations the dominant basic principles have mainly remained the same. Although von Neumann has not the inventor, he and his co-authors have been in the mid' 40ies the first to clearly describe the strikingly simple machine paradigm which I call the ***von Neumann paradigm.*** Other historic paradigms, like dataflow machines, or reduction engines, have never been commercially significant. Commercial break-throughs mainly stem from the progress of semiconductor technology but have hardly affected the blinders limiting the scope of conference series like ISCA or MICRO, which remained almost 100% von-Neumann-based all the time.

The Computing Architecture Crisis. Dozens of expired supercomputing companies illustrate [7], how decades of searching for the universal massively parallel computer architecture finally failed. Von-Neumann-based parallel computing really succeeded only in some special application areas. Also the area of embedded computing systems, and even PCs demonstrate, that more and more most silicon real estate is occupied by accelerators, crutches needed by the aging von Neumann processor, now being a handicapped methusela after surviving so many technology generations. The crisis of classical computing system architecture is obvious, not only from the program statistics history of the ISCA series of annual international conferences [8] (fig. 2). But also this accelerator hardware more and more moves into a crisis.

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

Fig. 1: Terminology.

Exploding design cost is increasing much faster than circuit complexity (fig. 3). Moore's Law is becoming an increasingly misleading predictor of future developments [9]. In a number of microelectronics application areas the continuing technology progress sometimes creates more problems, than it solves. More and more circuit design cleverness is required because each new technology generation comes with additional new parasitics [10] [11] [12] [13] [14] [15], like the leakage current growing exponentially with the decreasing dielectric thickness. With copper used for wires severe corrosion problems have been reported, for instance. Other examples: IBM researchers warn severe design problems with power noise at 130 nm Technology [11]. In addition to substrate coupling, capacitive coupling and mutual inductance problems, the most worrisome yet was very hard to attack. Series-RL coupling through the power supply grid is a very troubling problem, which virtually never happened at 0.25 microns. Tens or hundreds of nets are impacted by this mechanism on a single design. There are more effects, like threshold voltage changes by dopant fluctuation [16] and others.

Unaffordable VLSI circuits. Full custom designed ICs have become almost unaffordable in many application areas, because of exploding mask cost and other NRE cost (fig. 4) and high cost of IC fab lines. Standard-cell-based designs yield much lower design cost, but for the price of high area-inefficiency, along with high mask cost. ASICs (gate arrays) are a way to share part of the mask cost with several products in low production quantities. But ASICs are area-inefficient.

Availability of Morphware. A viable route to overcome the design crisis and cost crisis appears to introduce morphware platforms (for terminology see fig. 1), where no specific silicon is needed, which saves mask cost and other NRE cost (fig. 4). Mapping an application onto an FPGA is a design activity at logic level (gate level), Since in fine grain morphware single bit CLBs maybe configured into a logic gates. The FPGA market has reached a volume of about 7 billion US-dollars worldwide, and is taking over a major part the gate array ASIC market [17]. Morphware is the fastest growing segment of the integrated circuit (IC) market [Dataquest]. Practically everything may be migrated onto morphware.

A second RAM-based platform. Now we have a second RAM-based platform: morphware, where structure definition is RAM-based, instead of the instruction execution schedules as known from von Neumann principles. It turns out, that von Neumann does not support such soft hardware. This is the basis of a emerging disruptive trend which has (almost) all prerequisites to repeat the success story of the software industry: by a configware industry needing the support of a revolutionary new machine paradigm.

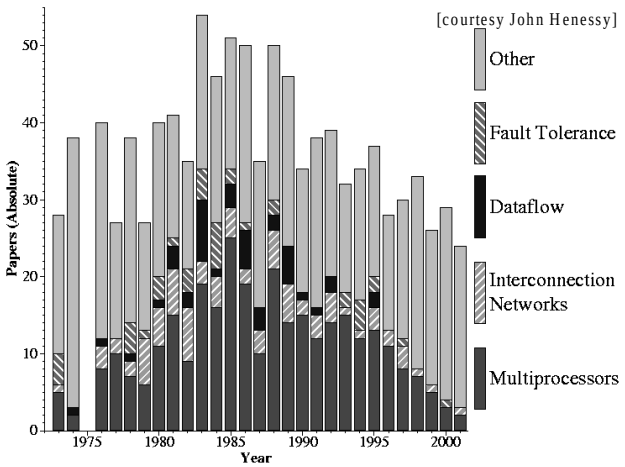

Fig. 2: Int'l Symp. on Computer Architecture paper statistic

Flexibility and time to market. Another advantage of morphware is FPGA flexibility. Since a logic gate is universal, also FPGAs are general purpose. A single FPGA type may replace a variety of IC types. Whereas turn-around times for changing hardwired IC designs takes up to several months, it is by orders of magnitude shorter on FPGAs, because personalization is done after fabrication, even at the customer's site. Patches may take only days, hours, or even minutes. So FPGAs have the potential for debugging or upgrading the last minute - even at the customer's site, or, remotely over the internet [18] or even by wireless communication [19] [20]. This means a change of the business model, compared to ASICs or other hardwired ICs [21]. Resources have been developed which support patches and even upgrades.

The programmable System on a Chip (pSoC) based on modern FPGAs does not only help to avoid application-specific silicon with its extremely high NRE cost per se. With the continuous technology progress FPGA vendors offer more and more hardwired on-board-products: micro controllers like ARM, MIPS, PowerPC, or other RISC architectures, memory, peripheral circuitry and others, together with the FPGA on board of the same chip. A not quite new 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.

A new route to fault tolerance. Interesting research results about fault-tolerant FPGAs are available [22] [23] [24] [25] [26] [27] [28]. Also experiences from using FPGAs in space missions could be useful [19] [20]. FPGAs are the most effective way and almost the only viable way to achieve reliability by defect tolerant integrated circuits. A kind of automatic repair after error detection is achieved by partial RTR (run time reconfiguration) which resembles a partial re-design at run time. A first step should be a feasibility study in cooperation with an FPGA vendor by to investigate a road map toward a fault tolerant FPGA architecture and related new EDA tools to be integrated in the overall design flow.

Soft CPUs. Configware providers meanwhile offer CPUs as soft IP cores (configware versions of CPUs or micro controllers) also called *FPGA CPU*, or, *soft CPU*, to be mapped onto an FPGA, like MicroBlaze (32 bits 125 MHz, [Xilinx]), the Nios (multi-granular [Altera]), Leon (32 bit RISC, SPARC V8 compatible, public domain). Using the usual FPGA design flow such soft CPU IP cores can be also generated from VHDL or Verilog sources originally targeted at a hardwired implementation.

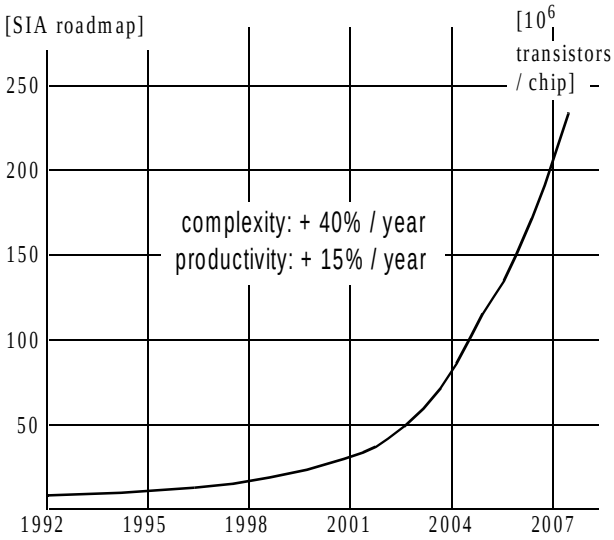

Fig. 3: IC design cost, growing faster then complexity.

Old fab line + algorithmic cleverness. More recently the research is heading toward low power FPGAs from the algorithms side. Jan Rabaey et al. have shown, that by transfer of an algorithm from a DSP to an experimental low power FPGA an improvement of factors between 5 and 22 have been obtained [29] [30]. When also the optimum technology is selected, a reduction of clock frequency by a factor of n yields a reduction of power dissipation by a factor of n^3 [31]. The only unsolved question is, how long this technology will be still available.

Configware industry is emerging as a counterpart to software industry. Part of the configware is provided by the FPGA vendors for free. But the number of independent configware houses (soft IP core vendors) and design services is growing. **A** good designer productivity and design quality cannot be obtained without good configware libraries with soft IP cores from various application areas.

New business model. Like microprocessor usage, also programming reconfigurable platforms is RAM-based, but by structural programming instead of procedural programming. Now both, host and accelerator are RAM-based and as such also available on the same chip: a new approach to SoC design. Also morphware is RAM-based and supports personalization at the customer's site, which means a change of the business model, providing a flexibility, which is not possible by classical ASICs. Since the program code is fundamentally different from software, it is called **„Configware"** (fig. 1 [6]). Now for patches and upgrades both can be downloaded: software and configware.

Structured Configware Design. With current design flows the still unsolved scalability problem of FPGAs will end up in a severe interconnect congestion crisis in proceeding toward the GigaFPGA mentioned above. The amount of reconfigurable interconnect resources and the average relative distance to be bridged is increasing with technology progress. Very sophisticated architectures including several different classes of wires like, nearest neighbor, short distance wires, medium distance wires, long distance wires, and very long distance wires, are no convincing solutions, since it is very difficult or even impossible to find efficient routing and placement algorithms for them. The only way to really solve this problem is *structured configware design*, using a wiring by abutment strategy similar to that known from structured VLSI design [32]. Experiences with the KressArray Design Space Xplorer indicate, that such a structured configware design strategy have been efficiently implemented by routing and placement using simulated annealing [33] [34].

Reconfigurable Computing. From a decade of world-wide research on Reconfigurable Computing another breed of reconfigurable platforms is an emerging commercial competitor to FPGAs. Whereas *RL* based on fine grain morphware (FPGAs) uses single bit wide CLBs,

Reconfigurable Computing (RC) based on coarse-grained morphware, uses rDPUs (reconfigurable data path units), which, similar to ALUs, have major path widths, like 32 bits, for instance - or even rDPAs (rDPU arrays) [35] [36]. In contrast to logic design level of mapping onto fine grain morphware, mapping applications onto coarse-grained morphware belongs to functional abstraction levels - using compilation techniques instead of logical synthesis. Important applications in areas like telemetries, multimedia and communication, are derived from the performance limits of the "general purpose" vN processor. For very high throughput requirements RC morphware is the drastically more powerful and more area-efficient alternative, also about one order of magnitude more energy-efficient than fine grain, and has drastically reduced reconfigurability overhead [35]. Commercial versions are available from PACT Corp. [37].

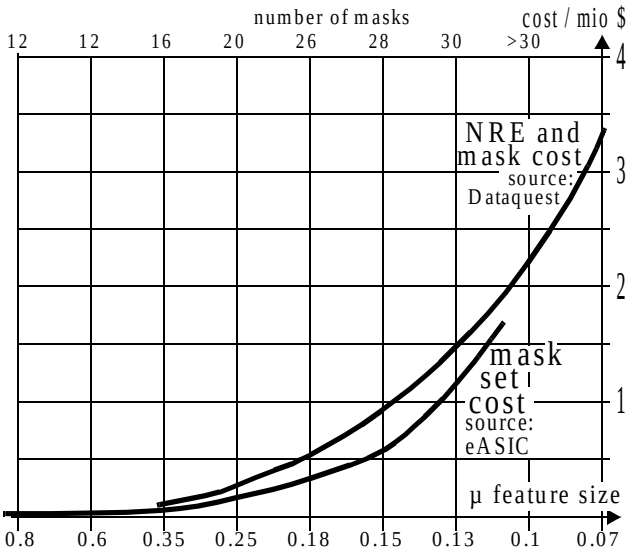

Fig. 4: Increasing mask set cost and total NRE cost.

Flowware. We can distinguish two categories of RC platforms [35] [36]: microminiaturized versions of traditional parallel computing systems (not subject of this paper), and, datapath networks driven by data streams. Data streams have been popularized by systolic and super-systolic arrays, and projects like SCCC [38], SCORE [39], ASPRC [40], BEE [41], and the KressArray Xplorer [33] [34]. In a similar way like instruction streams can be programmed from *software sources*, data streams are programmed from *flowware sources* (also see fig. 1). High level programming languages for flowware [42] and for software join the same language principles. A von Neumann machine needs *just software* as the only programming source, since its hardwired resource is not programmable. A reconfigurable data-stream-based machine, however, *needs two programming sources*: *configware* to program (to reconfigure) the operational resources, and, *flowware* to schedule the data streams (see fig. 1).

The Makimoto / Tredennick model. To fully understand the role of morphware within modern SoC design it is useful to model the history of IC application. Tredennick's classification scheme (fig. 5 a, b, and, c) [35] goes conform with Makimoto's wave model [17] [46]. This also explains why morphware needs two different programming sources (fig. 6 b): configware and flowware, whereas instruction-driven platforms need only a single source: software (fig. 6 a). This refines Tredennick's scheme into the one which is illustrated by fig. 5 d and e. The data-stream-based anti machine paradigm ([44] [45] fig. 6 a, and, fig. 5 e) is needed, because the instruction-stream-based von Neumann paradigm only supports hardwired platforms, but no morphware platforms [35]. Morphware has no instruction streams at run-time, since configuration which replaces the instruction fetch, is carried out before run time [35]. Even run time configurable systems distinguish between configuration mode and run time mode. Configuration cannot be derived from software, because it is a structural issue and not a procedural one. It is interesting, that even hardwired machine platforms can be driven by data-streams [41]. This is modelled by the Tredennick / Broderson Scheme (fig. 5 f) - another refinement of Tredennick's scheme.

The dichotomy of common architectural models. Going one abstraction level down from the Makimoto / Tredennick models yields a dichotomy of common architectural models based on the two machine paradigms: the von Neumann machine (fig. 7 a) and the anti machine. All models consist of two blocks: a functional block (FB) and a memory block (MB). On the von Neumann side (fig. 7 a), exactly one sequencer resides in the FB and the MB has only one memory bank. The interface between FB and MB is the "von Neumann bottleneck" not permitting any parallelism (fig. 7 a). On the anti machine side (fig. 7 b, c, and d) one (fig. 7 b,) or several sequencers (fig. 7 c, and d) reside in the MB, whereas the FB has no sequencers. Note the asymmetry between machine and anti machine (compare fig. 7 a with fig. 7 b). The anti machine side has a much wider design space. It also supports also multiple banks and multiple sequencers within the FB, like one datapath with multiple memory banks (fig. 7 c), or multiple datapaths with multiple memory banks (fig. 7 d).

Data-stream-based Computing. The models having been introduced above are important models to alleviate understanding implementation flows. The *Anti Machine* Paradigm [44] [45] (based on *data sequencers* [46] [5] [3]) is for morphware [47] and even for hardwired data-stream-based machines the better counterpart (fig. 7 b) of the von Neumann paradigm (based on an instruction sequencer, fig.

7 a). Instead of a CPU the anti machine has only a DPU (datapath unit) or a rDPU (reconfigurable DPU) without a sequencer (fig. 7 b). The anti machine model locates data sequencers on the memory side (fig. 7 b). Unlike "von Neumann" the anti machine has no von Neumann bottleneck, since it also allows multiple sequencers (fig. 7 c) to support multiple data streams interfaced to multiple memory banks (fig. 7 c), allowing operational resources much more powerful than ALU or simple DPU: major DPAs or rDPAs (fig. 7 d). There is a lot of similarities, so that each of the two models is a kind of mirror image of the other model - like matter and antimatter

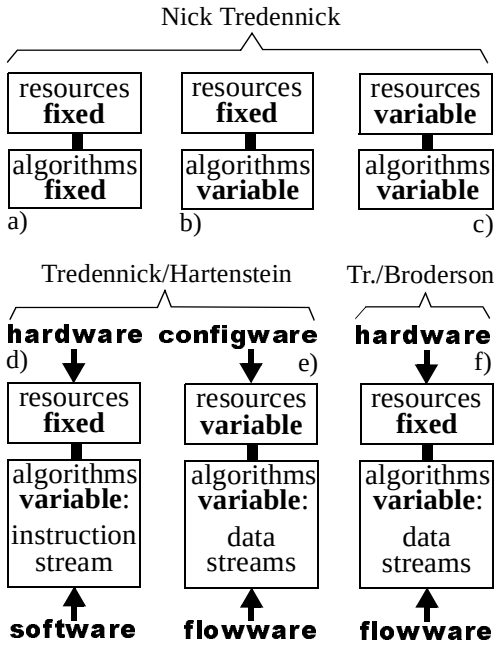

Fig. 5: Nick Tredennick's digital system classification scheme: a) hardwired, b) programmable in time, c) reconfigurable**;** d) von-Neumann-like machine paradigm, e) reconfigurable anti machine paradigm, f) Broderson's hardwired anti machine.

The coming configware industry? The success of software industry is RAM-based, supporting rapid personalization by procedural programming at the customer's site. Already a single morphware device provides massive parallelism at logic level or at operator level, which usually is much more efficient than process level parallelism possible only with multiple von Neumann (vN) processors being affected by a severe crisis [7] [8]. But this paradigm shift is still widely ignored: FPGAs and RC do not repeat the RAM-based success story of the software industry. There is not yet really a major configware industry, since mapping applications onto FPGAs mainly uses hardware synthesis methods. Because of lacking awareness of this paradigm switch there is not yet a configware industry.

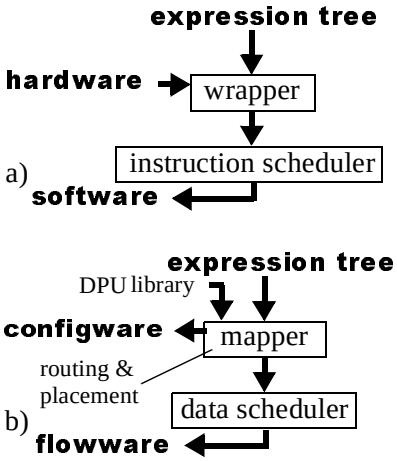

Fig. 6: Compilation: a) von-Neumann-based, b) for anti machines.

A new mind set is needed. However, the new trend is coming along with the challenge to overcome the incising limitations of users' dominant "procedural-only" mind set of computing science in the von Neumann era. There is a lot of similarities between the worlds of von Neumann and its anti machine paradigm. But like in particle physics, there are asymmetries - making it a bit inconvenient to teach the coming dichotomy of computing sciences. A new mind set is needed to overcome our inconvenience and to trigger a mass movement.

No, we are not ready for the break-through. It has been predicted, that by the year 2010 about 95% of all programmers will implement applications for embedded systems. However, the algorithmic cleverness and other important skills are missing. Not only computing sciences are not yet ready. Our basic curricula do not teach, that hardware and software are alternatives, and do not teach how hardware / software partitioning is carried out. Our basic curricula still widely ignore the existence of reconfigurable platforms and their use for embedded digital system design.

Missing algorithmic cleverness. In low power IC design (integrated circuit design), the leakage current and other technological phenomena, getting much worse with each new technology generation, are a major design problem. For instance, it may be better to implement a high performance application on 100 processors running at 200 MHz than on one processor running at 20 GHz. But from the EDA (electronic design automation) and programming point of view it is extremely difficult to achieve, that only 100 processors are needed to implement this parallel version. The algorithmic cleverness is missing - in practice and in our curricula.

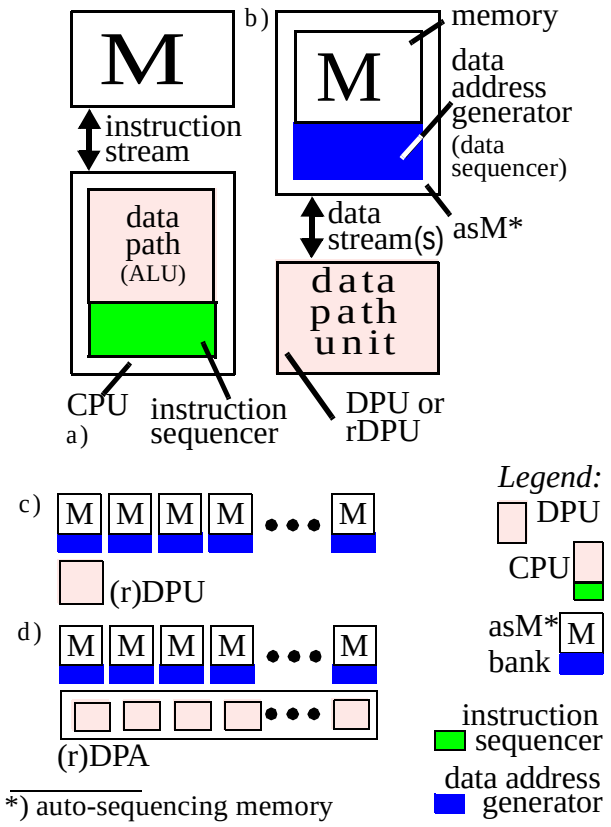

Fig. 7: Illustrating basic machine paradigms: a) von Neumann, b) data-stream-based anti machine (Xputer) with simple DPU, c) w. rDPU and distributed memory architecture, d) w. DPU array (DPA or rDPA).

Qualification Deficit. We have a lack of experience using FPGAs for computationally intense applications, lack of algorithmic cleverness to translate into morphware, immature FPGA-based design tools. We urgently need a consensus on terminology, and need to fight, that reconfigurable computing and morphware are not treated as a contaminant which when entering computing disciplines only meets troops of anti bodies ready to keep us out.

1. Literature

[1] J.Becker: Configurable Systems on Chip; ICECS 2002

[2] J. Rabaey (keynote): Silicon platforms for the next generation wireless systems; Proc. FPL 2000

[3] M. Herz et al.: (invited): Memory Organization for Data-Stream-based Reconfigurable Computing; ICECS 2002,

[4] F. Catthoor et al.: Custom Memory management Methodology; Kluwer 1998

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

[6] http://morphware.net - http://configware.org - http://flowware.net - http://data-streams.org

[7] Gordon Bell's "dead supercomputing society"; keynote at Int'l Symp. Computer Architecture (ISCA) 2000

[8] J: Hennessy (keynote): Int'l Symp. on Computer Architecture (ISCA) Barcelona, Spain. June 1998;

[9] I. Tuomi: The Lives and Death of Moore's Law; http://firstmonday.org/issues/issue7_11/tuomi/index.html

[10] J. McPherson: Scaling-Induces Reductions in CMOS Reliability Margins and the Escalating Need for Increased Design-In Reliability Efforts; Int'l Symp. on Quality Electronic Design 2001 (ISQED '01),

[11] R. Bergamaschi, J. Cohn (embedded tutorial): The A to Z of SoCs; ICCAD 2002

[12] D. Goodman: Prognostic Methodology for Deep Submicron Semiconductor Failure Modes; IEEE Trans. Components and Packaging Technologies, Vol. 24, No. 1, March 2001,.

[13] T. Nigam et al.: A Fast and Simple Methodology for lifetime prediction of ultra-thin Oxides, Proc. 1999 IRPS

[14] F. d'Heurle, P. Ho: Thin Films: Interdiffusion and Reactions; John Wiley & Sons, 243 (1978)

[15] H. B. Bakoglu: Circuits, Interconnections, and Packaging for VLSI; Addison-Wesley, 1990

[16] D. Burnett, J. Higman, A. Hoefler, B. Li, P. Kuhn: Variation in Natural Threshold Voltage of NVM Circuits Due to Dopant Fluctuations and its Impact on Reliability; Int'l Electronic Devices Meeting (IEDM), Dec. 9 - 11, 2002, San Francisco.

[17] T. Makimoto (keynote): The Rising Wave of Field-Programmability; Proc. FPL 2000

[18] S. Guccione et al.: JBits Java-based interface for reconfigurable computing; Proc. MAPLD 1999

[19] A. Dawood, N. Bergmann: Enabling Technologies for the Use of Reconfigurable computing in Space; 5th Int'l. Symp. on Signal Processing and its Applications, vol. 2

[20] G. Donohoe, J. Lyke: Adaptive Computing in Space; 42th Midwest Symp. on Circuits and Systems, vol. 1,

[21] T. Kean (keynote): It's FPL, Jim - but not as we know it! Market Opportunities for the new Commercial Architectures; Proc. FPL 2000

[22] T. Bartzik et al.: Design of a Fault-tolerant FPGA; Proc. FPL 2000

[23] F. Hanchek, S. Dutt: Methodologies for Tolerating Cell and Interconnect Faults in FPGAs; IEEE Trans on Computers; vol 47, 1 January 1998

[24] M. Renovell, J. Portal, J. Figueras, Y. Zorian: Test pattern and test configuration generation methodology for the Logic of RAM-based FPGAs; European Test Workshop,

[25] J. Lach, H. W. Mangione-Smith, M. Potkonjak: Enhanced FPGA Reliability Through Efficient Run-Time Fault Reconfiguration; IEEE Trans on Reliability; vol 49, Sept 2000

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

[27] N. Mahapatra, S. Dutt: Efficient Network-Flow Based Techniques for Dynamic Fault Recon-figuration in FPGAs; FTCS-29 - T29th Int'l Symp. on Fault-Tolerant Computing - Madison, Wisc, June 15-18, 1999

[28] W. Feng, F. J. Meyer, F. Lombardi: Two-Step Algorithms for Maximal Diagnosis of Wiring Interconnects; FTCS-29 - The 29th Int'l Symp. on Fault-Tolerant Computing - Madison, Wisc., June 15-18, 1999

[29] V. George, J. Rabaey: Low-Energy FPGAs; Kluwer, 2001

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

[31] M. Flynn et al.: Deep Submicron Microprocessor Design Issues; IEEE Micro July-Aug '99

[32] C. Mead, L. Conway: Introduction to VLSI Design; Addison Wesley, 1980

[33] U. Nageldinger et al.: KressArray Xplorer: A New CAD Environment to Optimize Reconfigurable Datapath Array Architectures; Proc. ASP-DAC 2000.

[34] U. Nageldinger et al.: Generation of Design Suggestions for Coarse-Grain Reconfigurable Architectures; FPL 2000

[35] R. Hartenstein (embedded tutorial): A Decade of Research on Reconfigurable Architectures - a Visionary Retrospective; DATE 2001, Munich, March 2001

[36] R. Hartenstein (invited embedded tutorial): Coarse Grain Reconfigurable Architecture; ASP-DAC 2001

[37] http://pactcorp.com

[38] J. Frigo, et al.: Evaluation of the streams-C C-to-FPGA compiler: an applications perspective; FPGA 2001

[39] E. Caspi, et al.: Stream Computations Organized for Reconfigurable Execution (SCORE): Proc. FPL '2000

[40] T. Callahan, J. Wawrzynek: Adapting Software Pipelining for Reconfigurable Computing; CASES 2000

[41] C. Chang, et al.: The Biggascale Emulation Engine; summer ret'2001, UC Berkeley

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

[43] R, Hartenstein (invited paper): The Microprocessor is no more General Purpose; Proc. ISIS`97, Austin, TX, Oct'77

[44] R. Hartenstein (keynote address): Disruptive Trends by Custom Compute Engines. Proc. FPL 2002

[45] R. Hartenstein (keynote address): Reconfigurable Computing: urging a revision of basic CS curricula; Proc. ICSENG02, Las Vegas, USA, 6-8 August, 2002

[46] J. Becker et al.: A General Approach in System Design Integrating Reconfigurable Accelerators; Proc. IEEE ISIS'96; Austin, TX, Oct. 9-11, 1996

[47] R. Hartenstein et. al.: A Novel ASIC Design Approach Based on a New Machine Paradigm; IEEE J.SSC, 26/7, July '91

AM	anti machine	FA	function array	rDPU	reconfigurable DPU
anti machine	machine using data counters*	flowware	source code for data streams*	rFA	reconfigurable FA
ASIC	application-specific IC	FPGA	field-programmable GA	rFB	reconfigurable function block
asM	auto-sequencing memory	FPGA CPU	soft CPU	rLB	reconfigurable logic block
cFB	configurable function block	FPFA	field-programmable FA	RISC	reduced instruction set computer
cLB	configurable logic block	GA	gate array	rSoC	reconfigurable SoC
configware	source code for morphware*	IC	integrated circuit	RTR	run time reconfiguration
cSoC	configurable SoC	KressArray	area-efficient rDPA family	soft CPU	CPU, emuated on an FPGA
DP	data path	NRE (cost)	non-recurrent expense	SoC	System on a Chip
DPA	DPU array	pSoC	programmable SoC	vN	von Neumann*
DPU	DP unit	RC	Reconfigurable Computing	vN machine	machine using program counter*
DSP	digital signal processing	rDPA	reconfigurable DPA	Xputer	AM pradigm

Fig. 8: Glossary

*) see fig. 1