# The von Neumann Syndrome*

(slightly modified version)

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

Arthur Schopenhauer: "Approximately every 30 years, we declare the scientific, literary and artistic spirit of the age bankrupt. In time, the accumulation of errors collapses under the absurdity of its own weight."
RH: "Mesmerized by the Gordon Moore Curve, we in CS slowed down our learning curve. Finally, after 60 years, we are witnessing the spirit from the Mainframe Age collapsing under the von Neumann syndrome."

***Abstract.*** *Because of high energy consumption our computer-based infrastructure may become unaffordable - without reinventing the entire computing discipline, also due to cope with the manycore programming crisis. The paper highlights facts, trends, and a roadmap to by-pass this crisis and to reach new horizons.*

The term *von Neumann syndrome* has been coined by C. V. „RAM“ Ramamoorthy in reply to my talk at San Diego [1]. Most problems are caused by the *Energy Wall*, the *Memory Wall*, and the *Education Wall*. Still main focus of CS education, the *von Neumann (vN)* basic common model [2] lost its dominance decades ago [3], also having been criticized for overhead [5] [6]: *its principles are fundamentally wrong, since data processing targets data streams - not instruction streams.* In industry it has been replaced by a cooperation of vN CPU and non-VN accelerators (fig. 1). To-day, most MIPS equivalents are running on FPGAs [7] (Field-Programmable Gate Arrays [8]: the fastest growing segment of the semiconductor market), where the microprocessor has become the tail wagging the dog and the basic accelerator model is *data-stream-based* - not instruction-stream-based. However, most published documentations of such symbiotic systems use a confusing and/or selfish terminology and are structured like stirred up Spaghetti Bolognese, not at all straightening out an underlying twin paradigm common basic model.

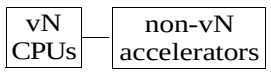


Fig. 1. Basic model replacing von Neuman (vN)

**von Neumann computing: will it be still affordable throughout next decade?**

**The most disruptive revolution since the mainframe:** it's *Reconfigurable Computing (RC)* [9] [10] [12] [13] [14] [15], mostly FPGA-based. Its pervasiveness is obvious. It comes with a second machine paradigm: the *anti-machine*, counterpart of vN [16] [17]. Meanwhile RC has become mainstream, not only in embedded systems. More than 170 international conference series deal with RC and its applications [18]. Not only in digital consumer electronics getting momentum from market convergence, RC is the key for future architectures: *field-programmability is a must* [19]. Since 2006, RC is also a hot spot in supercomputing [20] [21]: The personal supercomputer is near [22].

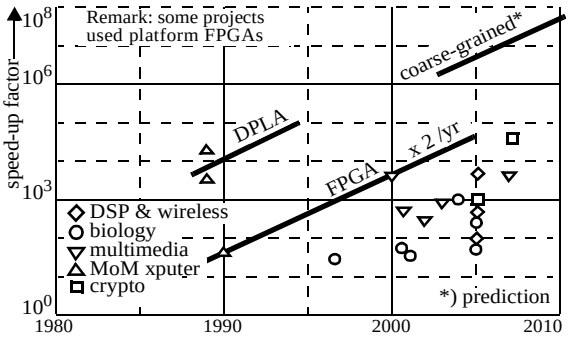


Fig. 2. Speedup by software to configware migration.

**The 1st Reconfigurable Computing Paradox.** From software to configware migrations, speedup factors by up tp 4 orders of magnitude (e.g. 3000 in image processing [24] and 38514 in DES breaking [25]) have been published (fig. 2), although FPGA technology parameters are very bad [10]. The effective integration density of a large FPGA is tremendously behind the Gordon Moore curve (fig. 3). Compared to speedup *the discrepancy* is up to 8 orders of magnitude. What explains such excellent results by such a bad technology? (Instead of „simple FPGAs“ (fig. 3) some more recent projects from fig. 2 used platform FPGAs having a better integration density by including a domain-specific mix of hardwired module blocks embedded in FPGA fabrics.) Part of the explanation is the *Gordon Moore gap*. The Moore curve does not show the actual computational density (effective MIPS per area unit) of microprocessor chips which has drastically decreased with the sequence of generations [10] [11]. The discrepancy also indicates,

Energy Wall, Memory Wall and Education Wall fuel the von Neumann syndrome.

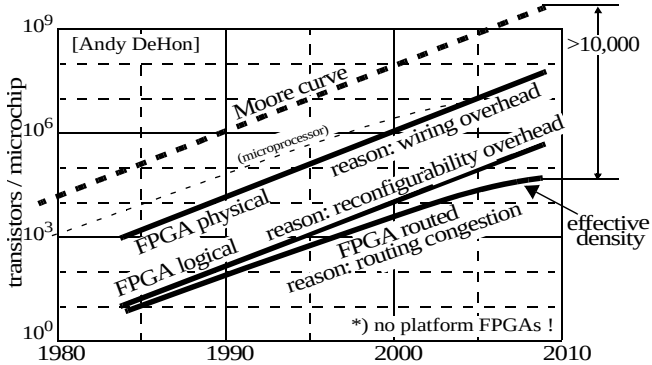


Fig. 3. Simple FPGA effective integration density*.

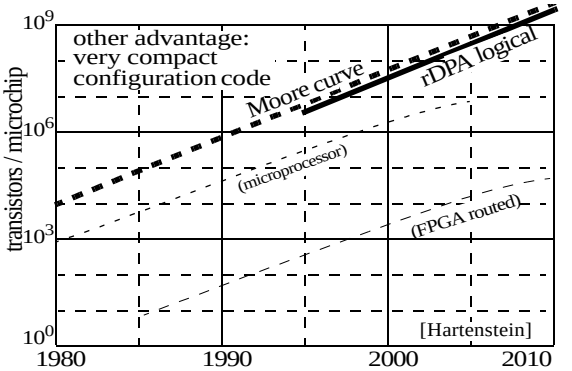


Fig. 4. Area efficiency of coarse-grained RC.

*) invited paper, Stamatis Vassiliadis Symposium „The Future of Computing“, Delft, The Netherlands, Sept. 28, 2007

that our common models and implementation principles are fundamentally wrong: *the von Neumann syndrome*. Following the vN-centric spirit from the mainframe age in using technology of the silicon age we are navigating with a completely wrong road map. We need to think out of the box. Reconfigurable Computing is leading us to new roads we need to escape from the vN paradigm trap,tunnel of horror. The M'soft [instruction-stream-based] programming model for upcoming *manycore microprocessors* has been ***predicted to be 10 years off*** [23]. For an earlier solution we need a twin paradigm approach: instruction-stream-based solutions coordinated with data-stream-based antimachine concepts. Each core should have the option to run as a vN CPU, or, as an antimachine's *DPU (DataPath Unit: has no program counter)*, s. fig. 5.

von Neumann syndrome: an important strategic issue - at least at national level.

| type of PU (processing unit) | instruction sequencer included? | execution triggered by |
|---|---|---|
| CPU | yes | instruction fetch |
| DPU or rDPU | no | arrival of data* |

*) transport-triggered

Figure 5. Duality of paradigms von Neumann vs. antimachine

**Missing the point: because of tunnel view.** For a hammer everything looks like a nail. For some mathematicians many phenomena look like an algebraic problem. Around 1980 it has been found out, that systolic arrays are beautiful schemes to solve complex algebraic problems [26] [27] [28], and, have come up with an excellent definition of the term *data stream* - nicely illustrated by a time/space schematics (fig. 6). Their several synthesis methods to derive the pipe network of a systolic architecture from a mathematical formula has been, of course, algebraic. Since this means linear projection yielding only uniform linear pipes, these synthesis methods have supported only applications with strictly *regular data dependencies*, which is a far-ranging limitation. It took almost 15 years, to overcome this limitation, which stems from the tunnel view of the algebraic perspective. In 1995 Rainer Kress replaced their algebraic synthesis methods by simulated annealing [29], which in fact, means a generalization of the systolic array. The resulting *supersystolic array* methodology supports all kinds of irregular pipe schemes, including (not only) spiral, zigzag and even much more wild schemes, even also with fork and join features. By the way, the original conference series on systolic arrays [30] [31] extended its scope [32] [33].

„data stream" ≠ „dataflow".

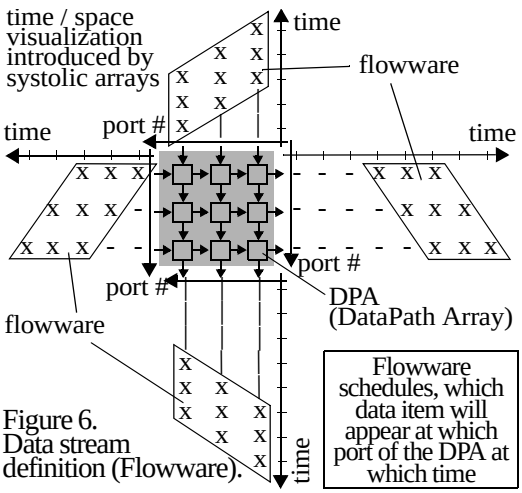


Figure 6. Data stream definition (Flowware).

Flowware schedules, which data item will appear at which port of the DPA at which time

**Anti machine: delayed by paradigm trap.** Who organizes the *data streams* to run systolic arrays? The reply of that systolic array scene around 1980 has been: „this is not our job. [This is the job of hardware people.]" Since computing resources without a sequencer are *not a computational „machine"* in the sense of a machine paradigm, those mathematicians have missed to invent the new machine paradigm of the antimachine [34] [35] [36] - the counterpart of the von Neumann machine. This „transdisciplinary" break-through had been delayed by their reluctance due to their algebraic paradigm trap. Each individual data stream is generated by an *auto-sequencing memory (ASM)* block including a data counter. For the array example in fig. 6 we need 12 ASMs, 6 of them as data stream sources, and 6 of them as data stream sinks (fig. 7). So this machine has 12 data counters in total. Compared to von Neumann the main differences of the antimachine are: (mostly multiple) *data counters*, co-located with memory - in contrast to vN's single *program counter*, co-located with the single datapath unit (DPU). The anti machine does not have a CPU: it only has (mostly multiple) DPUs, i. e. without program counters. (*Datastreams ≠ dataflow.* Note: only use the term *datastreams*, but avoid to use the obsolete term *dataflow*! [37]) The two machine paradigms are twins, because to express sequencing the same language primitives are used (fig. 7) [38]. Both paradigms have the same syntax rules. Their sequencers use the same circuity. Their semantics is only slightly different. The only external asymmetry is the fact, that data stream loops can be internally parallel at this level, whereas instruction stream loops cannot. A von Neumann machine can have only a single DPU (inside the CPU), whereas an antimachine can have multiple DPUs.

We must reinvent computing **[41].**

**The von Neumann tunnel of horror.** The vN paradigm is preferred by rationally bounded humans for reasons of Denkoekonomie ([Ernst Mach] [39]). Scarce resources (intelligence) are substituted as soon as possible. vN's beneficiaries Intel and Microsoft gain from the fact that the programmer does not need to think a lot about many difficult aspects of computing [40]. However, our von-Neumann-centric cyber-infrastructure is a tunnel of horror: astronomic code sizes cause *a massive array of overhead phenomena*, due to the von Neumann syndrome. 1000 processors running in parallel means that 1000 instruction streams with all their overhead phenomena yield a drastic programmer productivity decline. [41]: „In practice we are limited to a few instructions per clock cycle." Traditional software engineering problems are now topped by *the manycore programming crisis*. The human wave approach toward improving components not being the main system bottlenecks to come up with a variety of speculative and other indeterministic methodologies, recently topped by transactional memory efforts. A huge waste of researcher capacity to obtain mostly marginal results: the mountain screamed and bore a mouse. Also

(invited paper in:) K. Bertels et al. (ed.): The Future of Computing - Essays in memory of Stamatis Vassiliadis; TU Delft, 2007
ISBN: 978-90-807957-3-0 NUR 958

multithreading is speculative and is not the silver bullet. Overhead phenomena also lead to microprocessor chips featuring a highly disappointing computational density [11]. This is only an incomplete list of indications to the von Neumann syndrome. The table in fig 9 lists some of the von Neumann overhead phenomena which can be avoided by software to configware migration, i. e. by RC.

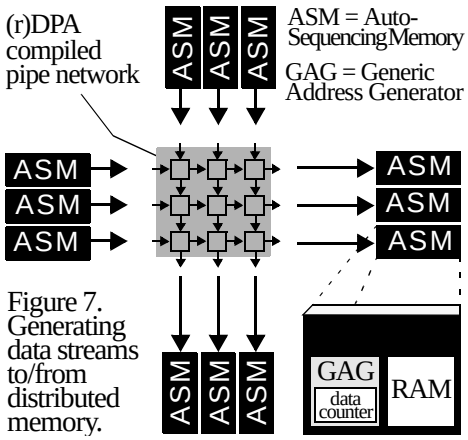


Figure 7. Generating data streams to/from distributed memory.

**A new paradigm cannot be avoided.** Most researchers and implementers in HPC are reluctant to go for a paradigm extension. This is (not only) illustrated by following panelists' statements from SC06. „It is feared that domain scientists will have to learn how to design hardware. Can we avoid the need for hardware design skills and understanding?" [42], „A leap too far for the existing HPC community" [42]. *Trying to avoid paradigm revisions leads to a completely wrong road map*. The following statement sounds somewhat better: „We need a bridge strategy by developing advanced tools for training the software community to think in fine grained parallelism and pipelining techniques." [42]. Such a bridge strategy also makes sense, because of the manycore programming crisis running in parallel with the break-through of Reconfigurable Computing.

**Everything we know is wrong.** The vN syndrome reminds of a Freudian repression of material parallelism. RC is more equivalent to our natural unconscious intelligence. However, the revolution is painful, since consciously parallel thinking is hard, is rarely systematically trained, and is badly supported by established tools [40]. Since parallelism now becomes increasingly ubiquitous, HPC should be able to exploit and extend mainstream programming languages, operating systems, development tools, libraries, and even applications intended for smaller scale systems [41]. Also grid computing requests delivering its functionality through far simpler programming interfaces: „The Grid is sometimes its own worst enemy. Grid computing misses the point." [43] [44]: "If we want to enable new science then we need to empower the user." To make it much easier for developers to implement parallel software and systems, we urgently must reinvent not only computing but also the computing profession [41]. Everything we know is wrong [46]. This requires a paradigm revision for execution and programming models. [47].

Reconfigurable Computing means: no instruction fetch at run time.

| | instruction stream languages | data stream languages |
|---|---|---|
| sequencing primitives | read next instruction<br>goto (instruction address)<br>jump (to instruction address)<br>instruction loop<br>loop nesting<br>escapes<br>instruction stream branching<br>***no loops internally parallel*** | read next data item<br>goto (data address)<br>jump (to data address)<br>data loop<br>data loop nesting<br>escapes<br>data stream branching<br>***yes: loops int'y parallel*** |

Figure 7. Why vN and antimachine language paradigms are twins

**Twin paradigms approach is needed.** The von Neumann syndrome tells us, that instead of physical limits, *fundamental misconceptions of algorithmic complexity theory* limit the progress and will necessitate new breakthroughs. *Not processing is costly, but moving and storing data and messages.* We've to completely re-think basic assumptions behind computing. Moving data (at run time) in the vN domain turns for the antimachine domain into (at compile time) *moving the locality of execution* by pipe network synthesis (line 7 and 8 in fig. 8). Another example are *reconfigurable address generator*s moving address computation overhead (lines 9 and 10 in fig. 9) from run time to compile time [48] [49] [50]. Supporting compilation techniques, also featuring automatic software / configware partitioning, have been demonstrated, accepting C language sources [51] [52] [53] [54], and mathematical formula input sources [55]. *Cores of manycore systems should be heterogeneous*, also including reconfigurable cores like FPGAs etc. Because we cannot afford to discard non-von-Neumann accelerators, *we have to support a twin paradigm approach:* a well and clearly organized dual model supporting both, von Neumann and the antimachine paradigm.

Also for converging consumer electronics markets field-programmability is a must [19]

**The 2nd Reconfigurable Computing Paradox.** Compared to hardwired accelerators, FPGAs have bad technology features, such as slow clock frequency, and, higher energy and space requirements. Many publications report speedup factors obtained from software to configware migration (onto FPGAs [7] [10]) - up to a factor of 6000 (fig. 2). But only one of those publications reports slashing the electricity bill and space needed down to less

| # | feature | von Neumann machine | hardwired antimachine (e.g. [56]) | reconfigurable antimachine |
|---|---|---|---|---|
| 1 | machine code schedules: | instruction stream | data stream(s) | |
| 2 | # of programming sources | 1 | | **2** |
| 3 | programming source 1* | none | | configware |
| 4 | programming source 2** | software | flowware | |
| 5 | (source 2) sequenced by: | 1 program counter | 1 or more data counter(s) | |
| 6 | counter co-located with: | DPU (data path) „CPU" | memory block(s): „ASM" | |
| 7 | inter PU communication: | via common memory | piped through | |
| 8 | data meet PU (processing unit) | move data at run time | move execution locality at compile time | |

*) to set up resources
**) for scheduling

Figure 8. von Neumann vs. Anti machine: Partitioning scheme.

(invited paper in:) K. Bertels et al. (ed.): The Future of Computing - Essays in memory of Stamatis Vassiliadis; TU Delft, 2007
ISBN: 978-90-807957-3-0 NUR 958

than 10 percent, for a speedup factor of only 17 [45]. This means a discrepancy factor of substantially more than 100 in terms of Watts per effective MIPS. The (semi-RC) GRAPE machine shows, that the electricity consumption per MIPS might go down to about one tenth of a percent [64] [65] [66]. A recent talk reports a power factor of 3439 [25] for a 38514x speedup factor at DES breaking, also causing a massive impact on security overhead: a Megabit key needed soon?

Software to configware migration promises tremendous speedups and energy savings: at much lower cost

**The von Neumann syndrome: the Energy Wall.** The electricity consumption of computers has been mainly ignored (fig. 10). But recently a discussion has been kicked off within the HPC scene [21]. The energy consumption of future supercomputers is heading for astronomic dimensions. The discussion now also includes server farms. Google's annual electricity bill amounts to 50 million US-Dollars - more than the value of its computing equipment. And, about 25% of Amsterdam's electricity consumption goes into server farms. Servers in New York city occupy a quarter square kilometer of building floor area. A study predicts, that - from currently more than 20% - by the year 2020 the electricity consumption of the entire cyber infrastructure in the US will amount to 35 - 50% of the US electricity production (fig. 10) [58]. The crude oil price development (fig. 11) and market predictions (fig. 12) lead to the question, wether von Neumann computing will be affordable in the future. Conclusion: *the von Neumann syndrome is a strategic issue, at least at national level.*

| # | type of run time overhead | v. Neumann machine | hardwired antimachine | reconfigurable antimachine |
|---|---|---|---|---|
| 9 | state address computation | instruction stream | none | no instruction fetch at run time |
| 10 | data address computation | instruction stream | none | |
| 11 | inter PU communication | instruction stream | none | |
| 12 | instruction fetch | instruction stream | none | |
| 13 | data meet PU | instruction stream | none | |
| 14 | synchronization | instruction stream | none | |
| 15 | multi-threading | instruction stream | none | |
| 16 | transactional memory | instruction stream | none | |
| 17 | message passing | instruction stream | none | |
| 18 | multiplexing [57] | instruction stream | reduced or none | |

Figure 9. Overhead avoided by antimachine w. distr. on-chip memory.

**The Memory Wall.** Dave Patterson's law shows the memory chip bandwidth orders of magnitude behind microprocessor clock frequency: main reason of the tremendous instruction stream overhead. Because of usual vN-based software code size, it mostly cannot be stored at on-chip memory - in contrast to configware code tending to be massively more compact. Commercially available platform FPGAs can have up to more than 700 local on-chip memory blocks, so that data streams for many applications can be scheduled to/from fast local memory (fig. 8) [59]. This also helps to *explain the first reconfigurable computing paradox.*

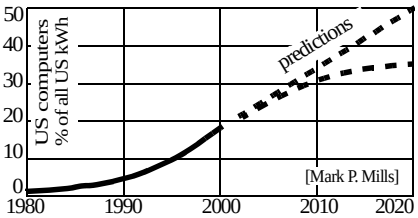


Figure 10. Electricity consumption.

**The von Neumann syndrome: very high cost.** From software to configware migration a hardware cost reduction to ~10% has been reported, also reducing the number of racks needed to ~10% [45]. Much more is expected for the future. A drastic reduction of floor area needed is a substantial cost factor because of the building size needed. Also, if needed at all, the cost of air conditioning equipment and the electricity bill coming with it is massively reduced: another motivation for a paradigm revision.

MIPS/watt are more important than MIPS/$.

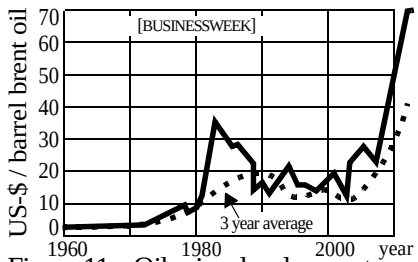


Figure 11. Oil price development.

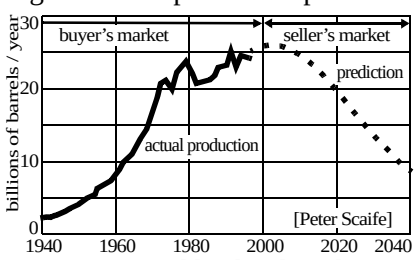


Figure 12. Worldwide oil production.

**Mapping from time to space: the Education Wall.** In contrast to vN, the anti machine is data-stream-based: *no instruction fetch at run time*. Mapping an application from software to configware means *mapping from time to space* - a domain hated by people with a software-only background. A Japanese CTO when introduced [60] to rDPAs [61]: „But you cannot implement decisions!" We immediately see the Education Wall in his brain. How to map a flowchart's decision box into the space domain? It turns into a demultiplexer controlled by the decision bit running on an extra wire instead of sitting in a CPU's register: it's *branching in space*. (The *Register Transfer Module* system (1972) sold by Digital Equipment [62], and a similar system (1967) from academia [63], are based on such a mapping of flowcharts.) See, how this important finding was commented in the 70ies by the HDL scene: „This is so simple. Why did it take 25 years to find out?" This backlog was due to the *tunnel view of the software-only mind set*. Meanwhile time to space mapping is an old hat: we should stop ignoring it. This request is urgently addressed to our curriculum recommendation groups like the joint ACM/IEEE-CS task force, still hopelessly reluctant to discuss RC issues. Such a software-only mind set misses the IT job market. To create better visibility of these problems and to encourage activities to cope with the Education Wall the *International Annual Conference Series on RC education* has been founded. [67] [68].

Undiscovered, the antimachine paradigm has been around for decades: used indirectly via highly inefficient instruction streams.

(invited paper in:) K. Bertels et al. (ed.): The Future of Computing - Essays in memory of Stamatis Vassiliadis; TU Delft, 2007
ISBN: 978-90-807957-3-0 NUR 958

**Dynamically Reconfigurable FPGAs** [69] are partially reconfigurable where parts of it can be running while other parts are being reconfigured. Dispatching, scheduling and swapping configware macros are the job of a configware OS (configware operating system [70]). „No instruction fetch at run time“ still holds, when the model used here follows a clean definition. This is a bit difficult to explain to beginners without some advanced FPGA background.

The von Neumann paradigm is tremendously overhead-based - in contrast to the Antimachine.

| Source | is compiled into: |
|---|---|
| Software | an instruction schedule |
| Flowware | a data schedule |
| Configware | a pipe network by placement and routing |

Figure 13. Sources by compilation targets.

**Fully fledged paradigm shift not needed.** Classical CS knowledge is still important (also to run legacy software). We need only the additional adoption of a second paradigm, which is only partially different from the von Neumann mind set. The second paradigm, the antimachine, is a twin brother of the von Neumann paradigm. The control syntax is mainly the same (fig. 7). Only the semantics is different: controlling data streams instead of instruction streams. This helps us to find a good bridge strategy to go from von Neumann single paradigm to a twin paradigm methodology. Most ingredients of the antimachine methodology are rather old stuff - mostly ignored by the CS community. Most of the enabling technologies of RC and to cope with the von Neumann syndrome, have been published at least 20 years ago, like for instance, about loop transformations, routing algorithms, languages to express parallelism, compilation techniques, data streams, systolic and supersystolic arrays, software to hardware migration etc. The point of view may be slightly different to-day.

We see a trend away from a bit-level FPGA hardware mind set, over to functional level with MAC, ALUs, DPUs, CPU-cores etc [40].

**Understandable modelling scheme needed.** A global system view is required for grasping the principles and essential issues of the contemporary heterogeneous twin paradigm systems, not only in undergraduate education, *we need an intuitive terminology and an understandable common modeling scheme*. A style of schematics with a clear distinction between von Neumann subsystems and antimachine blocks helps a lot. We distinguish 3 different types of programming sources (fig. 13) [71]. Instead of only type of source („software“) in von-Neumann-only systems (fig 14 b), two more kinds of sources are added by RC (fig. 14 c), **which we should <u>not</u> call *software***: it's *configware* to set up the structure of reconfigurable resources, and *flowware* for scheduling the data streams, according to configware compilation results. To create and understand such schemes we should clearly distinguish different types of programming sources: source type 1 (row 3 in fig 8,) ***to set up resource***s (not needed for hardwired machines), and source type 2 (row 4 in fig 8) ***for scheduling*** (instruction streams by *software* for von Neumann machines, or, by *flowware* for data streams at antimachines). This twin paradigm approach means *the interweavement of 2 cultures: a transdisciplinary approach* (fig. 16), affiliating the instruction-stream-based mind set (computing in time: procedural semantics) with the data-stream-based mind set (computing in space: structural semantics). It is easy, since the syntax is mainly the same (fig. 7). Affiliating should not mean mixing. Although the semantics is different, it should avoid confusion by navigating with a clean coordinate system: this challenge to educators can definitely be mastered.

Using rDPAs (coarse-grained reconfigurable arrays) is the best strategy to bridge the education wall - by raising the abstraction level.

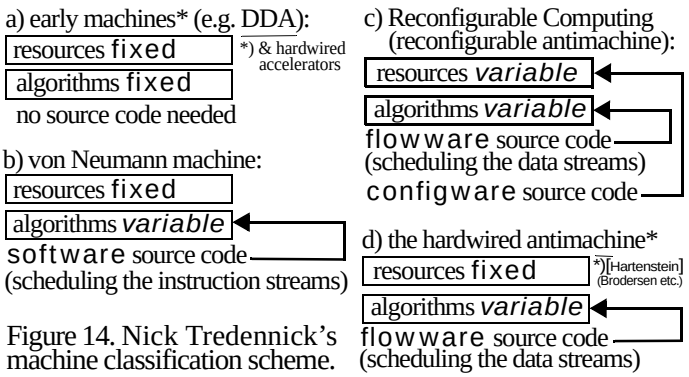


Figure 14. Nick Tredennick's machine classification scheme.

**Platform FPGAs** include a domain-specific mix of hardwired module blocks like LUTs, multipliers or DSP cells, memory objects, ALUs, etc., specialized ALUs, vN processors, sometimes with customizable instruction set processors, and even analog components. Such a heterogeneous mix poses significant new challenges for programmers and for synthesis software. This makes quantification of the performance and capacity of modern FPGAs – and particularly comparison of various arrays – almost impossible.

**Coarse-grained Reconfigurable Computing: for mastering the education wall.** The mental models of hardware and software engineers are unnecessarily set apart from each other [40]. But the market trend goes away from a bit-level FPGA hardware mind set, over to functional level with MAC, ALUs, DPUs and CPU-cores inside platform FPGAs [40] and rDPAs. A bridge strategy to cope with programmer's reluctance is the use of *coarse-grained reconfigurable data path arrays (rDPAs)* [72] [61]. Their cores are *rDPUs (reconfigurable data path units)* not having a program counter (fig. 5). In contrast to using FPGAs the modeling with rDPUs reaches functional level, coming much closer to the software-based mind set. Much better than any kind of FPGAs, the rDPAs (e.g. [73]) are the best educational

rDPAs: methodology is ready - users are not (delaying the break-through).

(invited paper in:) K. Bertels et al. (ed.): The Future of Computing - Essays in memory of Stamatis Vassiliadis; TU Delft, 2007
ISBN: 978-90-807957-3-0 NUR 958

| # | moving data between | data transport | execution triggered by | strategy |
|---|---|---|---|---|
| 1 | vN CPU cores | by instruction streams via common memory | instruction streams | moving data at run time |
| 2 | rDPU cores within rDPA (coarse-grained array) | piped through: directly from rDPU to rDPU | arrival of data (transport-triggered) | moving the locality of execution at compile time |

Figure 15. Avoiding the memory wall by coarse-grained reconfigurable array instead of many core CPU array.

strategy to bridge the software/configware chasm. Well-known and easily understandable loop transformations are smoothly mappable into pipe networks to be configured on rDPAs [51] [52]. Configware code size (much faster configuration time) and effective technology parameters of rDPAs (fig. 4) are much better than those of FPGAs (fig. 3) by about 4 orders of magnitude. Techniques for rDPA-based co-compilation and design space exploration, including automatic software/configware partitioning and interface generation, have been demonstrated [51] [61] [74].

The personal supercomputer is near.

**rDPA coarse-grained arrays to avoid the memory wall.** Fig. 15 illustrates the performance advantage of a coarse-grained rDPA array (row 2) over a manycore array of CPUs (row 1). Moving data between CPUs goes through common memory needing instructions slowed down by the memory wall for both: moving the data and to evoke executions on the CPUs (line 1). However, via compilation techniques for a coarse-grained reconfigurable array (placement and routing) the interconnect between rDPUs is configured to form a pipe network such, that data are directly pushed without needing common memory. This avoids data memory cycles. The execution within each rDPU is triggered by handshake, i. e by the arrival of data piped through directly from another rDPU. This avoids instruction memory cycles.

RC should be established as a vehicle to fascinate learning for the manycore future and to reverse the enrolment downtrend.

**rDPA coarse-grained arrays vs. platform FPGAs.** rDPUs inside rDPAs are reconfigurable, whereas hardware blocks in platform FPGAs are not. Following a less understandable model, platform FPGAs are less suitable for a bridge strategy than rDPAs. In contrast to FPGAs, rDPAs like the XPP array from PACT [73] provide higher speedups and lower energy consumption - coming with a compilation environment including tools for automatic interfacing CPUs with rDPUs - closing a zipper (fig. 16).

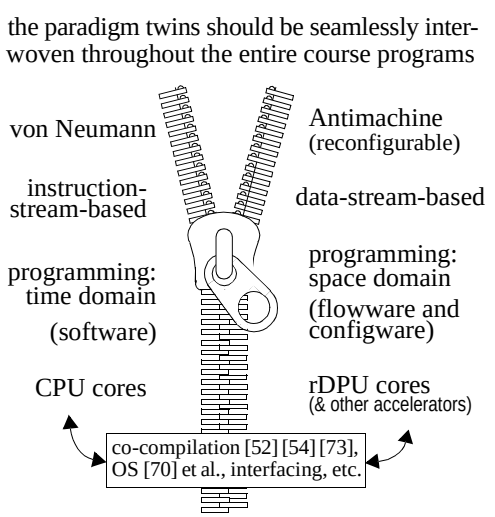


Figure 16. Twin paradigm zipper strategy to jazz up CS education by Reconfigurable Computing.

**Conclusions.** *RC is an essential qualification for the manycore future [40], HPC and supercomputing.* The performance of von Neumann computing systems is dramatically behind expectations. Gordon Moore's curve is far away from indicating computing performance. Von Neumann-based computing comes along with a tremendous array of instruction stream overhead phenomena, which are not coming with RC methodologies having no instruction fetch at run time. Von-Neumann-based programmer productivity progressively declines with an increasing number of processor cores involved. The equipment cost and energy consumption of von-Neumann-based computing are too high by more than an order of magnitude. Available RC methodology to cope with this syndrome is mainly ignored. Exceptions are many areas in embedded systems. Curriculum recommendations fail to hit this present and future IT job market, missing to consider that most software is written for embedded systems and most MIPS run on FPGAs. Educational deficits hamper the development of better development tools for better acceptance. We urgently need (1) Reconfigurable Computing education and training for the entire CS and IT community. and (2) to update CS curricula by a twin paradigm zipper strategy throughout entire course programs. RC should be established as a vehicle for fascinating learning for the manycore future [40] and to reverse the CS enrolment down trend. The potential performance gains and massive reductions of equipment cost and energy consumption by RC are by far too high to pass up such golden opportunities.

RC is an essential qualification for the manycore future, **HPC** and supercomputing.

vN principles are fundamentally wrong, since data processing targets data streams - not instruction streams.

## Literature


[1] R. Hartenstein (keynote): The Transdisciplinary Responsibility of CS Curricula; Proc. IDPT'06, June 26 -29, 2006, San Diego, USA

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

[3] R. Hartenstein (invited paper): The Microprocessor is no more General Purpose; Proc. IEEE ISIS, Austin, Texas, 1997

[4] http://www.fpl.uni-kl.de/papers/paper097.pdf

[5] J. Backus: Can programming be liberated from the von Neumann style?; Communications of the ACM, August 1978, 20(8).

[6] Arvind et al.: A critique of Multiprocessing the von Neumann Style; Proc. ISCA 1983

[7] R. Hartenstein (opening keynote): Reconfigurable Computing and the von Neumann Syndrome; ICA3PP, June 2007, Hangzhou, China


(invited paper in:) K. Bertels et al. (ed.): The Future of Computing - Essays in memory of Stamatis Vassiliadis; TU Delft, 2007
ISBN: 978-90-807957-3-0 NUR 958

[8] http://helios.informatik.uni-kl.de/FPGAbooks/index.htm
[9] Christphe Bobda: Introduction to Reconfigurable Computing Systems; Springer-Verlag, 2007
[10] R. Hartenstein (invited chapter): Basics of Reconfigurable Computing; in: J. Henkel, S. Parameswaran (editors): Designing Embedded Processors. A Low Power Perspective; Springer Verlag, March 2007
[11] BWRC, UC Berkeley, 2004
[12] R. Hartenstein (invited chapter): Morphware and Configware; in Albert Zomaya (ed.): Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies; Springer, 2006
[13] R. Hartenstein: A Decade of Research on Reconfigurable Architectures - a Visionary Retrospective; DATE 2001, Munich, Germany
[14] J. Becker (keynote): Adaptive Reliable Chips: Challenges in Reconfigurable and Organic Computing in the Nano Era; Proc. IDPT 2007, June 2007, Antalya, Turkey
[15] Special issue on Reconfigurable Computing: SDPS Journal of Integrated Design and Process Science; December 2007
[16] R. Hartenstein (keynote): Software or Configware? About the Digital Divide of Computing; 18th International Parallel and Distributed Processing Symposium (IPDPS), April 26–April 30, 2004, Santa Fe, New Mexico, USA
[17] R. Hartenstein (keynote): The Impact of Morphware on Parallel Computing; 12th EUROMICRO Conference on Parallel, Distributed and Network based Processing (PDP04); February, 11-13, 2004. A Coruña, Spain
[18] http://hartenstein.de/NewJournal.pdf
[19] T. Makimoto (keynote): Impact of Chip Innovations Driving the Computing Power; ISC 2007, June 26 - 29, 2007, Dresden, Germany
[20] R. Hartenstein (opening keynote): Supercomputing goes Reconfigurable; Winter International Symposium on Information and Communication Technologies (WISICT 2005), Cape Town, South Africa, Jan. 3 - 6, 2005
[21] R. Hartenstein (invited paper): Reconfigurable Supercomputing: Hurdles and Chances; Proc.ISC'06, June 28-30, 2006, Dresden, Germany
[22] R. Hartenstein: Configware für Supercomputing: Aufbruch zum Personal Supercomputer; PIK, 2006
[23] R. Merrit: M'soft: Parallel programming model 10 years off; EE Times (07/23/07)No. 1485, P. 4; http://www.eetimes.com
[24] http://www.pse.siemens.at/apps/sis/ge/pseinternet.nsf/0/PKFC681010482920B2C12572A40059C972
[25] T. El-Ghazawi (panelist): Supercomputing on Exotic Architectures; SC07, Reno, NV, November 2007
[26] M. Foster, H. Kung: Design of Special-Purpose VLSI Chips: Example and Opinions. ISCA 1980
[27] H. T. Kung: Why Systolic Architectures? IEEE Computer 15(1): 37-46 (1982)
[28] N. Petkov: Systolic Parallel Processing; North-Holland; 1992
[29] R. Kress et al.: A Datapath Synthesis System for the Reconfigurable Datapath Architecture; ASP-DAC'95
[30] First International Workshop on Systolic Arrays, Oxford University on 2-4 July 1986. The proceedings: see [31]
[31] W. Moore, A. McCabe and R. Urquhart (eds.): Systolic Arrays, Bristol: Adam Hilger, 1987.
[32] IEEE 18th Int'l Conf. on Application Specific Systems, Architectures, and Processors (ASAP), Montreal, CANADA, Jul. 2007.
[33] http://asap-conference.org/
[34] A. Hirschbiel et al.: A Flexible Architecture for Image Processing; Microprocessing and Microprogramming, vol 21, 1987
[35] M. Weber et al.: MOM - Map Oriented Machine; in: E. Chiricozzi (ed.): Parallel Processing and Applications, North-Holland, 1988
[36] M. Weber et al.: MOM - a partly custom-design architecture compared to standard hardware, Proc. IEEE Compeuro, Hamburg 1989.
[37] The dataflow machine does not have a sequencing counter: arbiter-driven its operation is indeterministic
[38] A. Ast, et al.: Data-procedural Languages for FPL-based Machines; FPL'94
[39] Robert S. Cohen (ed.): Ernst Mach. Physicist and Philosopher; Kluwer Academic Publishers, 1975,
[40] A. Fuchs (Siemens AG, Vienna, Austria): personal communication
[41] Burton J. Smith (keynote): Reinventing Computing; ISC 2007, June 26 - 29, 2007, Dresden, Germany
[42] T. El-Ghazawi et al.(panelists): Reconfigurable Supercomputing: is High-Performance, Reconfigurable Computing the Next Supercomputing Paradigm? SC06, Int'l Conf. on HPC, Networking, Storage and Analysis, Nov. 11-17, 2006, Tampa, Florida
[43] D. De Roure (keynote): eScience is about Scientists too; eResearch Australasia 2007, June 26 - 28, 2007, Brisbane, Australia
[44] M. Knights: Grid computing misses the point, says academic; OS and Servers News, June 28, 2007, www.techworld.com
[45] Herb Riley, R. Associates: http://www.supercomputingonline.com/article.php?sid=9095
[46] J. Shalf: Overturning the Conventional Wisdom for the Multicore Era; ISC 2007, June 26 - 29, 2007, Dresden, Germany
[47] Th. Sterling: HPC Achievement and Impact - 2007: Q & A; ISC 2007, June 26 - 29, 2007, Dresden, Germany
[48] M. Herz et al. (invited paper): Memory Organization for Data-Stream-based Reconfigurable Computing; ICECS 2002,
[49] M. Herz, et al.: A Novel Sequencer Hardware for Application Specific Computing; Proc. ASAP'97
[50] H. Reinig et al.: Novel Sequencer Hardware for High-Speed Signal Processing; Proc. DMM, Smolenice, Slovakia, Sept.1995
[51] J. Becker et al.: Parallelization in Co-Compilation for Configurable Accelerators; Proc. ASP-DAC´98
[52] J. Becker: A Partitioning Compiler for Computers with Xputer-based Accelerators, Ph. D. diss., Univ. of Kaiserslautern 1997
[53] E. Moscu Panainte: The Molen Compiler for Reconfigurable Architectures; Ph. D. dissertation, 2007, TU Delft
[54] E. Moscu Panainte, K. Bertels, S. Vassiliadis: The Molen compiler for reconfigurable processors; ACM Trans. TECS, Febr. 2007)
[55] M. Ayala-Rincón et al.: Prototyping Time and Space Efficient Computations of Algebraic Operations over Dynamically Reconfigurable Systems Modeled by Rewriting-Logic; ACM Trans. on Design Automation of Electronic Systems (TODAES), 2006
[56] R. Broderson et al.: The Biggascale Emulation Engine; Proc. FPGA 2002
[57] G. Koch et al.: The Universal Bus Considered Harmful; Proc. 1st EUROMICRO Symp.; Nice, France, 1975; North Holland, 1975
[58] M. P. Mills: The Internet Begins with Coal; 1999, Green Earth Society, USA
[59] M. Herz: High Performance Memory Communication Architectures for Coarse-grained Reconfigurable Computing Systems; Ph. D. thesis, Kaiserslautern, 2001
[60] M. Herz et al.: On Reconfigurable Co-Processing Uhnits; in: J. Rolim (editor): Parallel ad Distributed Processing; 1998
[61] U. Nageldinger et al.: Generation of Design Suggestions for Coarse-Grain Reconfigurable Architectures; FPL 2000
[62] C. G. Bell et al: The Description and Use of Register-Transsfer Modules (RTM's); IEEE Trans-C21/5, May 1972
[63] W. A. Clark: Macromodular Computer Systems; 1967 SJCC, AFIPS Conf. Proc. vol. 30, 1967, Washington, DC.
[64] R. Männer, R. Spurzem et al.: AHA-GRAPE: Adaptive Hydrodynamic Architecture - GRAvity PipE; Proc. FPL 1999
[65] T. Narumi, et al.: 46 Tflops Special-purpose Computer for Molecular Dynamics Simulations: WINE-2; ISCP2000, Beijing, 2000.
[66] T. Narumi et al.: Molecular dynamics machine: Special-purpose computer for molecular dynamics simulations; Molecular Simulation, 1999
[67] http://www.fpl.uni-kl.de/RCeducation
[68] R. Hartenstein: Why we need Reconfigurable Computing Education: Introduction; RCeducation, March 1, 2006, Karlsruhe,Germany
[69] T. Pionteck, C. Albrecht, R. Koch, E. Maehle, M. Huebner, J. Becker: Communication Architectures for Dynamically Reconfigurable FPGA Designs; Proc. RAW 2007, Long Beach, California USA
[70] M. Platzner et al.: Operating Systems for Reconfigurable Embedded Platforms; IEEE Trans. on Computers. 53(11), Nov. 2004.
[71] N. Tredennick: Technology and Business: Forces Driving Microprocessor Evolution; Proc. IEEE Dec. 1995
[72] J. Becker, M. Vorbach: An Industrial/Academic Configurable System-on-Chip Project (CSoC): Coarse.grain XPP/Leon-based Architecture Integration; DATE 2003
[73] http://www.pactxpp.com
[74] J. Becker et al.: Automatic Parallelism Exploitation for FPL-based Accelerators; Proc.HICSS'98, Big Island, Hawaii,1998

(invited paper in:) K. Bertels et al. (ed.): The Future of Computing - Essays in memory of Stamatis Vassiliadis; TU Delft, 2007
ISBN: 978-90-807957-3-0 NUR 958