The alternative Machine Paradigm for Energy-Efficient Computing

Reiner Hartenstein, TU Kaiserslautern and KIT Karlsruhe
IEEE fellow, SDPS fellow, FPL fellow

Abstract. We are facing an unaffordable growing energy-inefficiency of instruction-stream-based computing. To prevent a coming breakdown not only of server-based infrastructures we are forced to disestablish the monopoly-like dominance of the von Neumann paradigm. We must redefine our entire discipline by massive R&D efforts and revolutionary curricula innovations.

THE POWER WALL
The Power Wall, also called “Energy Wall” (the electricity consumption of computers) cannot be ignored any longer. The energy consumption of future server farms, supercomputers and all other ICT infrastructures is heading toward astronomic dimensions and may become unaffordable towards the end of next century or even earlier. Almost about a decade ago, about 25% of Amsterdam’s electricity consumption went into server farms. At that time servers in New York City occupied a quarter square kilometer of building floor area. Because of IoT and Industry 4.0 etc. the amount of data to be processed is exploding even faster. A 2015 study predicts that by 2020 the US cyber infrastructure electricity consumption will amount to 35 or even 50% of the total US electricity production.

This study estimates that already now all computers in the US burn up more than 25% of the total US electricity consumption. Main reason of the immense electricity consumption is the predominance of the vN paradigm. Already in 2008 data centers drove more carbon emissions than the Netherlands and Argentina together.

Data centers are the largest and fastest growing groups of US electricity consumers. In 2013, U.S. data centers consumed about 91 billion kilowatt-hours, equivalent to the annual output of 34 large (500-megawatt) coal-fired power plants: enough electricity to power all households in New York City twice — on-track to reach 140 billion kilowatt-hours by the year 2020. Data center electricity consumption is predicted to increase to roughly 140 billion kilowatt-hours annually by 2020, equivalent to the annual output of 50 power plants.

A published goal is Exascale computing using 20 MW of power, yet existing circuits consume more than an order of magnitude too much power to meet this goal. The power consumption of just one "Exascale" supercomputer ("exaflop" with 1.000.000.000.000.000.000 operations per second) is expected to be possible for 2018, will be on the order of 10 GW, enough to power entire New York City twice (i.e., twice a population...
of 16 million people)\textsuperscript{5}. This illustrates \textbf{vN}-only high performance computing becoming unaffordable without use of accelerators like FPGAs, introduced later within this paper. That's why several high performance conference series changed their name by including the term “heterogeneous”.

Not only for high performance computing \textit{power-awareness} is required from programming schemes, runtime systems, resource management strategies and algorithms and novel architecting assisting in power management. Fully stressing the \textbf{vN} Paradigm makes solutions absolutely impossible. Observed trends lead to the question, whether von Neumann computing will be affordable in the future. Diagnosing and treating the von Neumann syndrome is an extremely important strategic issue.

\textbf{THE VON NEUMANN SYNDROME}

About 200 years ago Arthur Schopenhauer said: “Approximately every 30 years, we declare the scientific, literary and artistic spirit of the age bankrupt when the accumulation of errors collapses under the absurdity of its own weight.” Such developments we currently experience in several areas of computing. Mesmerized by the Gordon Moore curve, we slowed down our learning curve. Finally, after 70 years, we are witnessing the mainframe age spirit collapsing under the “von Neumann syndrome.” Opening the discussion after my keynote at IDPT’06\textsuperscript{27} Prof. C. V. Ramamoorthy has coined this term for all the problems caused by the monopoly-like dominance of the von Neumann (vN) paradigm.

Without redirecting research and education the \textbf{vN} paradigm’s inefficiency will make computer-based infrastructures unaffordable. Becoming still more and more problematic by forcing us onto a completely wrong roadmap the \textbf{vN} model of computation should urgently be removed from all its highly dominant positions because it causes the Energy Wall, the Memory Wall, the FPGA marketing wall, Nathan’s Law, and the Education Wall. However, although having been criticized already decades ago by celebrities like Edsgar Dijkstra, John Backus, Arvind and others, the \textbf{vN} paradigm still determines the main focus of CS education.

The first electrical computer suitable for mass production was readily prototyped in 1884: the Hollerith Tabulator. With its manually programmed plug board (figure 1) it is a \textbf{first reconfigurable computer} not driven by instruction streams. Driven by data streams (punched card sequences) its paradigm was non-von-Neumann. Discarding this paradigm in the mid 40ies by adoption of the \textbf{vN} paradigm (by ENIAC etc.) was one of the biggest mistakes in the history of computing, meanwhile having caused users and vendors altogether to waste quadrillions of dollars.

Instruction-stream-based computing has been hit by 3 software crises. Enormous advances in hardware and dramatic reductions in hardware costs did not help to avoid software crisis 2.0. From high level languages like PASCAL back to assembly languages (OO programming, C++, C#,
Java, etc.) made it even worse. Software crisis 3.0 came along with programming multicores because instruction stream parallelism is hard, cannot be abstracted or layered away, and is very hard to debug. Data movement is not the bottleneck. It is very important, that Computer Architecture is now at a cross road.

Von Neumann paradigm principles are fundamentally wrong. “Data processing” finally targets data streams — not instruction streams. However, doing this by instruction streams is an expensive very long detour. This paper illustrates vN syndrome symptoms and highlights facts, trends, and a roadmap to cope with this crisis and to reach new horizons. Still main focus of CS education, the von vN basic common model began to loose its helpful efficiency already decades ago. But vanquishing vN will be extremely difficult, since much more than a half century of software sits squarely on top. Solutions are migrations over to the almost 20 years old Xputer paradigm\textsuperscript{22,23}. Since a rapid complete solution everywhere is impossible we must go heterogeneous by a twin-paradigm approach for quite a time. For this massive challenge: we must re-invent CS education and severely re-orient CS research.

To-day, a growing number of MIPS equivalents is running on accelerators like FPGAs (\textbf{Field-Programmable Gate Arrays}) where the basic paradigm is data stream-based, i.e. non-vN. However, most publications about hetero systems use a confusing terminology hardly illustrating the underlying twin paradigm common model. Standardization is completely missing. (For terminology see the chapter "THE XPUTER PARADIGM").

\textbf{THE MEMORY WALL}

The term “Memory Wall” has been coined by Sally McKee\textsuperscript{15} for the growing disparity of speed between CPU and memory outside the CPU chip. This “Dave Patterson’s law” (figure 2) shows the memory chip bandwidth by several orders of magnitude behind microprocessor clock frequency. This is a main reason of this tremendous instruction stream overhead. This bandwidth gap grows about 50\% per year and has reached a factor of $>1000$. Because of vN-based huge Software code size, it mostly cannot be stored at on-chip memory, so that a hierarchy of different memory units is needed, including also hard discs being much slower than RAM.

Here Nathan’s law (“Software is a gas, which expands to fill all its containers.”) meets the memory wall causing inefficiency by additional orders of magnitude since software packages often reach astronomic dimensions and running them requires large power-hungry slow extra memory capacity. The software code size unit is “MLOC” (Million Lines Of
Code): for Windows XP (2001) with 40 MLOC, for Mac OS X 10.4 (2006) with 86 MLOC, and for SAP NetWeaver (2007) even with 238 MLOC. This makes running instruction streams by software again much more inefficient in contrast to data streams (flowware) from configware, tending to be massively more compact. However, flowware does not at all suffer from Nathan’s law. Commercially available platform FPGAs can have more than 700 local on-chip memory blocks, so that data streams for many applications can be scheduled to/from fast local memory.

The predominance of tremendously inefficient software-driven vN-type computers, employing a fetch-decode-execute cycle to run such gigantic programs from memory hierarchies, is a cause of the massive waist of energy. Here the term „software“ stands for extremely memory-cycle-hungry instruction streams — coming with multiple overhead levels: the von Neumann Syndrome which is also a reason of the software crises.

David Patterson explained, that our old innovation paradigm has changed. It used to be that power was free, but transistors were expensive. But now power is expensive and transistors are cheap. The old wisdom was that innovation came through compiler optimization. Now we learnt that we need wide-ranging innovation efforts which will take many years.

**RECONFIGURABLE COMPUTING (RC)**

The massively more efficient alternative to instruction streams are data streams via Reconfigurable Computing: to implement accelerators. A kind of pipelined network of operators is configured so that operators are transport-triggered, since each data word connection is coming along with handshake wires. Earlier high level reconfigurable computing has been pioneered with systolic arrays, introduced by H. T. Kung in 1980\(^8\). What synthesis method? “Of course algebraic!” (he is a mathematician). This means: linear projections only. This can be used only for applications with strictly regular data dependencies. This is a massive handicap, since only a very, very small part of very few possible application areas is supported by this. On the question, who generates their data streams, he
replied “This is not our job”. So he missed to come up with the new machine paradigm running data streams instead of instruction streams.

My student Rainer Kress solved problems by simulated annealing instead of linear projections\textsuperscript{36} for synthesis, and directly coupling this super-systolic “KressArray” to data memory with multiple data counters instead of a program counter (fig. 5). In 1984 FPGAs\textsuperscript{16} came to market with an array of programmable logic blocks (CLBs), and a hierarchy of reconfigurable interconnects allowing the blocks to be "wired together”. Now Reconfigurable Computing is a quite mature research and engineering area, also covered by masses of international conferences\textsuperscript{33}.

Instead of completely replacing a microprocessor based process by a hardware implementation it often makes sense to connect a specialized accelerator hardware to it. However, the production cost of such ASIC custom chips is steadily rising. And because of the design sophistication more and more project teams are looking here for more competitive ways without having to create custom chips. The entry barrier is substantially lowered with off the shelf FPGAs being perfect for rapid prototyping and easily upgradable.

FPGA are ICs being easily configurable by a customer or a designer. Meanwhile RC also supports functions above logic design level. Performance improvements by up to several orders of magnitude\textsuperscript{35} (fig. 3\textsuperscript{6}) can be obtained by migration from software running on vN computers over to FPGAs (fig. 4), depending on the application area, on the implementation method and the skills of implementers. For the VLSI design multi-university E.I.S. project\textsuperscript{21} our design rule check on a VAX 11/750 was extremely slow. By the PISA project\textsuperscript{11} we implemented an Xputer-based version with a speed-up factor of 15,000 (fig. 3). A PLA called “DPLA”\textsuperscript{7} designed by my group at Kaiserslautern replaced 256 FPGAs, since processing canonical Boolean expressions was sufficient
and the better flexibility of FPGAs was not needed. The DPLA was fabricated by MPC (multi project chip) facilities of the E.I.S. project\textsuperscript{21}.

**THE RECONFIGURABLE COMPUTING PARADOX**

The massive speed-up and energy saving results (fig. 2\textsuperscript{12}) are a surprise, since the integration density of FPGAs is by several orders of magnitude behind the Gordon Moore curve, because of massive wiring overhead: reconfigurability overhead and routing congestion, so that only about 3 percent of the transistors directly serve the application whereas the other 97 percent are used for routing. The clock frequency is substantially lower and 30 – 50 times more transistors are needed per operation than in a classical microprocessor. So a several orders of magnitude worse technology delivers a several orders of magnitude better performance. The reason of this “Reconfigurable Computing Paradox” is the imbelievably harmful “von Neumann syndrome”. Reconfigurable Computing as basic accelerator model is so extremely more efficient since it is data stream-based and not instruction-stream-based\textsuperscript{12}.

**TIME TO SPACE MAPPING**

The migration of an implementation from software over to Reconfigurable Computing is based on time to space mapping by turning a program (figure 5a). into a pipe network. The result is shown by fig. 5b. The decision box of a flowchart from the time domain is directly implemented by a multiplexer, whereas the 3 operation boxes are implemented as flip-flops. For its traditional application a flowchart needs its software implementation inefficiently running on a computer, whereas after mapping into space (this means: into morphware) it runs by itself. When this method concept came up in the context of very early hardware description languages a colleague asked: “This is so simple! Why did it take 25 years to find out?” This historic methodology has been published already in the 70ies\textsuperscript{14} and has been supported by hardware.
modules offered by Digital Equipment. Normally we get a pipe network where the operations are efficiently “transport-triggered” via handshake by arriving new data values. It automatically runs by itself so that no instruction streams are needed here (fig. 5b). To benefit from this we urgently need massively innovative reconfigurable computing education to fix the widely spread lack of qualification of people doing synthesis jobs and people designing application development and synthesis tools.

THE XPUTER PARADIGM

There are terminology conflicts: dataflow versus data streams. Originally stressing differences to „control-flow“ an area called „Dataflow“ computers was started in the mid' 70ies and attracted a great deal of attention at that time. This term is misleading since its „I-Structure“ is "Tagged-Token-Flow"-centered and didn’t really use data-flow-driven code. But the power efficiency break-thru did not happen here\(^{17}\). It came much later by the Xputer.\(^7\) With respect to terminology Xputer area people are forced to sidestep by using less snappy terms like „data-driven“ or „data streams“.

![Diagram of Xputer](image)

**Figure 5; Time to space mapping.**

Now let us look at the Xputer. It is not a „dataflow machine“. It is a “data stream machine“. In contrast to the vN computer running instruction streams with one program counter, an Xputer has several data counters (fig. 6) to run the parallelism of multiple data streams including reconfigurable address generator accelerators (to avoid needing instruction streams for complex address computations). Onto this paradigm quite a number of applications have been completely migrated from software to morphware (fig. 6). BTW: Xputer operations are deterministic. To avoid use of lengthy instruction streams we use reconfigurable address generators accelerate the address computation by providing an additional speed-up of much more than an order of magnitude. Result is generalization\(^{10}\) of the systolic array.

![Diagram of Xputer machine block structure](image)

**Fig. 6; Xputer machine block structure.**

It’s a machine paradigm efficiently supporting reconfigurable computing: the antimachine paradigm called “Xputer”: counterpart to the von Neumann paradigm by using several data counters instead of a program counter\(^9\) (fig. 4 b). Now reconfigurability efficiently makes sense: use morphware instead of hardware. For massive deficits in standardization and terminology let us stress the following...
explanatory terms (fig. 3):

- **Morphware**: reconfigurable circuitry like FPGAs etc.
- **Configware**: program notation for configuration of Morphware
- **Flowware**: program notation for scheduling data streams
- **Software**: program notation for scheduling instruction streams
- **Computer**: processor running Software by a program counter
- **Xputer**\(^7,22\): processor running Flowware by multiple data counters

Figure 7 is Nick Tredennick’s classification scheme of required resources: a) for vN with only one variable resource, b) for Xputers\(^7\), requiring two programmable resources, c) twin-paradigm like with “all programmable” requires three programming resources. Software runs instruction streams on vN computers (figure 4 c) whereas Flowware is running data streams through Morphware like FPGAs (figure 4 b). By the vN paradigm data and instructions are moved by Software i.e. by memory-cycle-hungry instruction streams which fully hit the memory wall: heavily slowed down by Dave Patterson’s Law (figure 2). So computing sciences are in a severe crisis. We urgently must shape a reconfigurable computing revolution for going toward incredibly promising new horizons of affordable highest performance computing. This cannot be achieved by the classical Software-based mind set. We urgently need a new dual paradigm approach everywhere by re-definition of research and education.

**TWIN PARADIGM COMPUTING**

If for major efforts like supercomputers a full migration is blocked by limited manpower, heterogeneous systems are set up including some accelerators embedded into a software-based system. Meanwhile "All Programmable FPGAs" go heterogeneous by additional vN processors on board. Such hetero systems are twin paradigm approaches (fig.7c). Since a programmed accelerator like on an FPGA is special purpose it should be easy to re-load Morphware and Flowware. For such systems fig. 8 shows the structure of a partitioning compiler (1997 implemented at Kaiserslautern). The Hardware/ Software chasm turns into Configware/Software interfacing: heterogeneous for generalization of software engineering. High performance conferences changed their names by including the term “heterogeneous”.

![Diagram](image.png)
THE FPGA MARKETING PARADOX

There is a second paradox. Although migrations to FPGAs provide speed-ups by up to many orders of magnitude, the market success of FPGAs is by orders of magnitude worse than that of other semiconductor products (fig. 9): by selling by orders of magnitude more efficient products. It was going that way for almost twenty years. The problem is the synthesis process and the qualification of people doing the synthesis job and of people implementing the application development systems. It’s never really clear who’s ahead, who’s behind, or even what the rules were.

WE MUST REDIRECT RESEARCH AND EDUCATION

Til 2017 IDC predicts for the world 8.6 million data centers, and a future substantial growth of their sizes, huge already now. If trends continue till the end of next decade, their energy consumption will be higher than the overall consumption of the entire world to-day. It will become unaffordable without a paradigm shift: replacing vN by heterogeneous. The entire computer science research and education world must be completely redefined to create widely spread twin-paradigm qualifications and facilities, since the massive long term tremendous damage to be expected by the worldwide monopoly-like massive dominance of the von Neumann paradigm can not be tolerated any longer.

Everywhere in the world morphware is well available with low cost FPGAs off the shelf. But we learn from the FPGA marketing wall that this does not...
yet help (fig. 9). Progress is blocked by severe educational problems. Sufficiently qualified populations of application implementers and well qualified EDA tool implementers are here far from being available.

FPGA synthesis is a high-stakes game, but, despite of fierce competition and some technological advances in the tools, the scoreboard is still basically a riddle [Kevin Morris]. Can software development engineers program FPGAs without learning HW design? Can programming languages be compiled down to FPGAs or twin-paradigm systems? No, it is it coming with a too sophisticated compilation / programming model and interpretation overhead: as the programmer’s night mare!

This educational dilemma urgently needs potent solutions. We fear the worst: that because of the manycore crisis the indispensability of programmable accelerators may be ignored. To avoid this we must merge both areas in education: manycore parallelism and reconfigurable computing. CS people must learn the twin paradigm mind set. The creation of innovative courses and updated curricula is urgently needed.

The real outcome of FPGA synthesis efforts depends on how well the tools interpret the code, guess the user’s intent, and infer some circuitry that will honor the user’s requests while following design rules and optimization goals the user may not even know. Only a few customers see FPGA-vendor-supplied tools as almost good enough, and other customers relied on the frequently changing markets of third party EDA solutions.

Also new software-centric design environments are needed that can tremendously boost the productivity of designers and open up FPGA-based acceleration to the masses of software engineers.

A major obstacle to wider FPGA use in mainstream computing is the complexity of the design process. Could this be simplified so far that the incredible compute power of modern FPGAs would be accessible to ordinary software developers? The implementers of application development tools at FPGA vendors or at related EDA firms are not sufficiently qualified. Also elsewhere we are very far away from programming by flowware instead of software being easy to learn and not needing a fully fledged hardware expert to program configware.

Already in 2006 Burton Smith criticized that University research on important areas has dried up. Research (and education) is urgently needed in reconfigurable and heterogeneous systems (and parallel computing) on languages, compilation techniques, debugging and performance tuning, and operating systems (and computer architecture).

The Xputer as the 2nd computing paradigm is the key to curricular innovation: a Trojan horse introducing Reconfigurable-Computing-based issues to the classical programmer mind set. We must merge research and curricula development into a twin paradigm environment since research
is useless if its results cannot be used. A new workshop series has been recently started: “Reconfigurable Computing for the Masses, Really? New opportunities and long-standing challenges of disruptive technology”\textsuperscript{18}.

Classical software engineering education or software design education should be disestablished everywhere and replaced by dual paradigm programming education and system engineering education. The entire computer science research and education world must be fully redefined to create widely spread twin paradigm qualifications and facilities.

\textbf{THE ROLE OF ACM AND IEEE}

Rebooting Computing?\textsuperscript{32} Within 519 pages on computing curricula recommendations\textsuperscript{29} the word "reconfigurable" does not exist and the keyword "FPGA" does not at all appear within the official part. (It appears there only once, but inside a personal discussion section.). It claims: "The CS2013 guidelines include a redefined body of knowledge, a result of rethinking the essentials necessary for a Computer Science curriculum". Slogans like "The Great Principles of Computing" or "The Beauty and Joy of Computing" are no more up to date as long as the consequences of the von Neumann syndrome are strictly put under taboo. So my submission to Communications of the ACM was rejected within less than 8 hours without reviewing\textsuperscript{30}. More recently I had taboo experiences with IEEE Computer. Maybe, in "Computer Science Curriculum" the term "Computer" excludes xputers? What about "Data Processing Science Curriculum"? Would this conflict with "instruction processing"? Who has a good terminology idea?

\textbf{CONCLUSIONS}

We are witnessing the mainframe age spirit collapsing under the “von Neumann syndrome". The introduction of von Neumann (vN) principles 70 years ago was one of the biggest mistake in history. The Reconfigurable Computing Paradox is a highly convincing illustration of the extremely massive inefficiency of the classical vN-only mind set.

ACM / IEEE Curriculum recommendations\textsuperscript{29} strictly put this completely under taboo. So it is no surprise, that a sufficiently qualified population of application implementers and tool developers and powerful design environments to boost designer productivity is very far from being available. A quantum leap in computing efficiency is possible only when we all are experts also in Reconfigurable Computing\textsuperscript{12}. A rapid full replacement of the cumbersome vN paradigm is not feasible and partial replacement will take a very long time since more than a half century of software sits squarely on top. The von Neumann area should relinquish its dominance but should not be discarded since we must go heterogeneous, also for stepwise accelerator inclusion.
Innovative research is urgently needed for both: methodology and curricula. To support the dual paradigm approach the entire CS research and education world must be completely redefined as soon as possible. We urgently need friends starting a mass movement against the taboo.

Uma reforma total no ensino de computação é necessária!

@ TU Kaiserslautern, August 18, 2016

References

6. Matthias Fuchs: Geschichte und Einführung in Aufbau und Arbeitsweise von FPGAs; 2003, Univ. Ulm http://www.informatik.uni-ulm.de/ni/Lehre/SS03/ProSemFPGA/Einfuehrung_FPGAs.pdf
9. R. Hartenstein: Karl Steinbuch about how to invent something; http://www.fpl.uni-kl.de/staff/hartenstein/lot/Invent_something.pdf
10. R. Hartenstein et al.: KressArray Xplorer: a new CAD environment to optimize reconfigurable datapath array architectures; Proc. ASP-DAC January 2000, Yokohama, Japan
21. The E.I.S. Project; http://www.fpl.uni-kl.de/staff/hartenstein/eishistory_en.html
23. Reiner Hartenstein (PATMOS 25th anniversary keynote): How to Cope with the Power Wall; CHIP IN BAHIA 2015, Salvador, Bahia, Brasil, August 31 to Sep 4 http://hartenstein.de/salvador.htm
24. N. N.: Number and Size of Data Centers; http://www.fpl.uni-kl.de/papers/Data_station_images_English.pdf
26. Tarek:


32. IEA: Energy Efficiency of the Internet of Things; Technology and Energy Assessment Report; Prepared for IEA 4E EDNA; April 2016; click here


34. N.N.: FPGA and Reconfigurable Computing Conferences; August 18, 2016, http://www.fpl.uni-kl.de/FPGAconf.html
