The Grand Challenge To Reinvent Computing
A new World Model of Computing

Reiner Hartenstein[1,2,3]

[1]Universidad de Brasilia, [2]TU Kaiserslautern,

[3]European Network of Excellence on High Performance
and Embedded Architecture and Compilation (HiPEAC)

Abstract. *Computers are very important for all of us. But brute force disruptive architectural developments in industry and threatening unaffordable operation cost by excessive power consumption are a massive future survival problem for our existing cyber infrastructures, which we must not surrender. The impact is a fascinating challenge to reach new horizons of research in computer science. We need a new generation of talented innovative scientists and engineers to start the beginning second history of computing. This paper introduces its new world model.*

Why Computers are Important.

Typical orders of magnitude in the computer application landscape are: hundreds of applications, consisting of ten-thousands of programs, with millions of lines of code, having been developed by expenditure of thousands of man-years investment volumes up to billions of dollars. We must maintain these important infrastructures.

Wiki "Answers pages" nicely tell us, why computers running this legacy software are indispensable in the world [1]. The Computer is an electronic device used in almost every field even where it is most unexpected. Now we cannot imagine a world without computers. These days' computers are the tools for not only engineers and scientists but also they are used by many millions of people around the world.

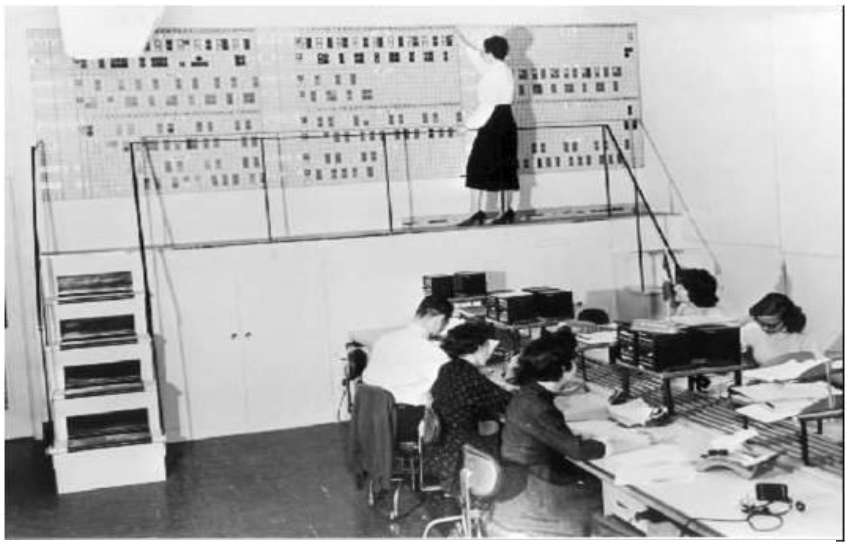

Fig. 1; Lufthansa Reservation System anno 1960 [2]

The computer has become very important nowadays because it is very much accurate, fast and can accomplish many tasks easily. Otherwise to complete those tasks manually much more time is required (fig. 1). It can do very big calculations in just a fraction of a second. Moreover it can store huge amount of data in it. We also get information on many different aspects using internet on our computer.

Banks use computers to keep record of all transactions and other calculations. It provides speed, convenience, security. Communication is another important aspect, very easy through internet and email. Computer communicates by telephone lines and wireless. Through email we can send messages to anybody in any part of the world in just a second while if we write letter then it will reach in some days. So the internet has made the earth a global village and above all saves time. This would not be

possible without computers. Internet helps to find information on every topic. It's the easiest and fastest way of research. Computer network makes the user capable of accessing remote programs and databases of same or different organizations. Without computers we also would not have any automated teller machines (ATM).

Business. Computers have now become an integral part of corporate life. They can do business transactions very easily and accurately and keep the record of all the profit and loss. Today computers can be found in every store, supermarkets, restaurants, offices etc. special software is used in these computers to calculate the huge bills within seconds. One can buy and sell things online, bills and taxes can be paid online and can also predict the future of business using artificial intelligence software. It also plays a very important role in the stock markets.

Business Information Systems. For the economy: business information systems are as essential as materials, energy and traffic [2]. Without business information systems the economy would be ineffective and inefficient. Business information systems are essential for globalization. Their significance for each enterprise: improving productivity of business processes (= rationalization), mastering complexity and volume, making information available fast and everywhere: for any operations, for decisions, as well as strategically for entrepreneurial planning on the creation of new business opportunities, i. e. by e-business. If automobile manufacturers would not have PPC systems (product planning & control system), cars could not be manufactured in desired wide variety.

Fig. 2; Google server farm at Dallas, Columbia river [5].

Biological And Medical Science. Diseases can be easily diagnosed with the help of computer and can also know about its cure. Many machines use computer which allows the doctor to view the different organs of our body like lungs, heart, kidneys etc. There is special software which helps the doctor during the surgery.

Education. Today the computer has become an important part of one's education because we are using computers in every field and without the knowledge of computer we cannot get a job and perform well in it. So computers can secure better jobs prospects. Knowledge about computer is must in this time.

Media. Almost every type of editing and audio-visual compositions can be made by using special software especially made for this purpose. Some software can even make three dimensional figures which are mostly used in the cartoon films. Special effects for action and science fiction movies are also created on computer.

Travel And Ticketing. Computers do all the work of plane and train reservation. It shows the data for vacant and reserved seats and also saves the record for reservation. Let us imagine, Lufthansa would handle reservations like in 1960 (Fig. 1). To-day they could not handle all their flight operations by this method.

Weather Predictions are possible by the experts using supercomputers.

Sports. It is also used for umpiring decisions. Often the umpire has to go for the decision of a third umpire where the recording is seen again and finally reaches to a fair decision. Simulation software allows sportsmen to practice and improve their skills.

Daily Life. Computers are everywhere. We operate washing machines, microwave oven and many other products using software. Moreover we can store all the information about our important work, appointments schedules and list of contacts. Computers are playing a very important role in our lives. We cannot imagine the world without computers. And this technology is advancing in both, industry and home. It is necessary for everyone to have the basic computing knowledge. Otherwise he cannot get a job as computers have invaded almost all the fields.

Fig. 2; PELAMIS wave power: creating electricity by the sea.

Why We Need To Reinvent Computing.
Brute force disruptive architectural developments in industry caused ***the many-core crisis*** and threatening ***unaffordable operation cost*** by excessive power consumption of the entirety of all von Neumann computers world-wide. Rapidly growing energy prices are predicted since the oil production has reached its peak by 2009 [3] [4]. However, the demand is growing because of developing standards of living in China, India, Brasil, Mexico and newly industrializing countries. We need "six more Saudi Arabias for demand predicted for 2030" [Fatih Birol, Chief Economist IEA]. All this will be a massive future survival problem for our cyber infrastructures, which we must not surrender because we also need it in the future. The impact is a fascinating challenge to reach new horizons of new computer science research. We need new generations of talented innovative scientists and engineers to start the second history of computing. This paper proposes its new world model.

Green Computing uses conservative methods to save energy by more efficient modules and components. For example LED flat panel displays need much less power than LCD-based displays. Also much more power-efficient power supply modules are possible. Low power circuit design is a decades old research area. Its most important conference series are about 30 years old: the ISLPED and the PATMOS series. The potential to save power by conservative green computing is less than an order of magnitude: maybe, a factor of about 4 after many years. To illustrate the order of magnitude of the benefit to expect from the subarea of low power design in the field of

term	controlled by	machine paradigm	State register	
			Type	location
software	Instruction streams	von Neumann	Program counter	in ***CPU***
configware	(configuration memory)	none	None	
flowware	reconf. address generator	datastream machine	Data counter(s)	In ***asM*** memory block

Table 1; Fundamental terminology for twin paradigm hetero computing systems.

green computing let me quote the summary of a low power design research paper [6]: "Summary: Using a Dual Vt LUT decreases power by ~13%. Predefined dual Vdd: very little effect because of routing. Fully programmable Vdd logic cells reduces power by 28.6%. Fully configurable Vdd logic cells and interconnects with power gating reduces

power by 50.55%. Tradeoffs: increase in area, increase in delay, increase in configuration time." We need a much higher potential of saving energy because "Energy cost may overtake IT equipment cost in the near future" [7]. "Green Computing has become an industry-wide issue: incremental improvements are on track" [8], "But „we may ultimately need revolutionary new solutions." since we need much higher efficiency.

Immense Energy Consumption Of The Internet

From the total electricity consumption of Amsterdam 25% went into server farms already in 2005. In New York City Server-Farms occupied a quarter square kilometer of building floor space. The electricity bill is a key issue: also for Google, Microsoft, Yahoo and Amazon with their huge datacenters at Columbia River (fig. 2), and ORNL benefits from Tennessee Valley Authority. Google has a patent for a "water-based data centers, using the ocean to provide power and cooling (fig. 3). It makes sense, to measure the performance of computing devices not just by MIPS (million instructions per second), but by MIPS/watts instead

Already in 2005 Google's annual electricity bill was higher than 50,000,000 US-$. Last year the internet edition of TIME reported, that Google causes 2% of the world-wide electricity consumption. Google denied. What does this mean? Google does not pay these 2%. About 90 % of it is payed by customers accessing the Google search engines. Other estimations claim, that the greenhouse gas emission from power plants generating the electricity needed to run the internet is higher than that of the total world-wide air traffic. G. Fettweis estimates, [9] that – if current trends continue – the electricity needed for internet and the communication equipment used by it, will ***increase by a factor of 30*** by the year 2030: this is much more than currently the total electricity consumption of the entire world.

SGI Altix 4700 w RC 100 RASC vs. Beowulf cluster	speed-up factor	save factor		
		power	cost	size
DNA & Protein sequencing	8723	779	22	253
DES braking	28514	3439	96	1116

Table 2; Recent speed-up and power saving examples of software to configware migration

Massively Saving Energy By Reconfigurable Computing

At least a partial paradigm shift promises to save electricity by orders of magnitude. Dozens of papers (ref. in [10]) have been published on speed-ups obtained by migrating applications from software running on a CPU, over to configware for programming FPGAs [11]. For Bioinformatics applications [12] speed-ups have been obtained by 2 to 4 orders of magnitude. For digital signal processing and wireless communication, as well as image processing and multimedia, speed-ups by 2 to almost 4 orders of magnitude, and for cryptography by 3 to >5 orders of magnitude. More recently also energy saving factors have been reported, roughly one order of magnitude lower than the speed-up. For example, Tarek El-Ghazawi et al. [13] have reported more recently for DES breaking (a crypto application): 28,500 (speed-up) vs. 3439 (saving energy) and for DNA sequencing 8723 vs. 779 etc. (table 2).

Fig. 3; The geo-centric Aristotelian world model

Toward A New World Model Of Computing

Not only for the definition of the term ***"Reconfigurable Computing" (RC)*** [12, 13, 14] it makes sense, to use a clear terminology – not only to improve education about how to reinvent computing. It is a sluttish use of terms if ***"soft"*** or ***"software"*** is used for

everything, which is not hardware. Within this paper we use the term "software" only for instruction streams and their codes. However, we generalize the term ***"programming"*** (fig. 5) such, that ***procedural programming*** (in time domain) creates sequential code, like ***instruction streams (software)***, or ***data streams***, which we call ***"flowware"***, and, that "***structural programming"*** (programming in space) creates ***"structural code",*** which we call ***"configware",*** since it can be used for the configuration of ***FPGAs (Field-Programmable Gate Arrays)*** [11] or other reconfigurable platforms. Summary: table 3.

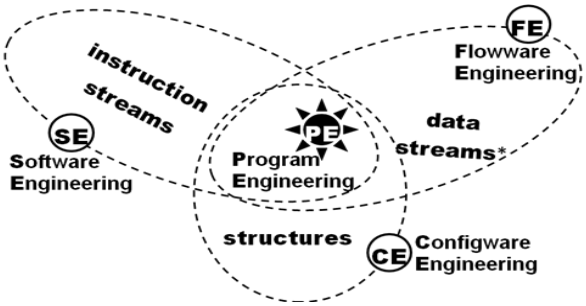

Fig. 5; Generalization of Software Engineering.

This established terminology reveals (table 1), that a software to configware migration means ***a paradigm shift***, away from the traditional programmer's CPU-centric world model of computing, resembling the geo-centric Aristotelian world model (Fig. 4). To reinvent computing we need a multi paradigm hetero system world model of computing science (fig. 5), which models the co-existence of, and the communication between: (1.) the CPUs running by software (instruction streams), (2.) the reconfigurable modules to be structurally programmed by configware, and (3.) the data stream machines programmed by flowware for generating and accepting data streams (asM in table 1 stands for "auto-sequencing Memory", also containing the data counter inside a reconfigurable address generator). Fig. 5 shows this triple-paradigm "Kopernican" world model replacing the obsolete "Aristotelian" tunnel view perspective of classical software engineering. This model will help us to come up with a new horizon of programmer education which masters overcoming the hardware / software chasm, having been a typical misconception of the ending first history of computing.

#	moving data between	data transport	execution triggered by	strategy
1	von Neumann CPU cores	via common memory	instruction stream	moving data at run time
2	(r)DPU cores within (r)DPA	piped thru directly from (r)DPU to (r)DPU	arrival of data ***(transport-triggered)***	moving at compile time the locality of execution

Table 3; How to move data: von Neumann machine vs. pipe network

The Reconfigurable Computing Paradox

Why does software to configware migration yield such massive improvements in speed and power consumption, although FPGAs are a much worse technology? The loop to pipe conversion model illustrates it (fig. 8). FPGAs have an enormous wiring overhead, and massive reconfigurability overhead (from a hundred transistors maybe about 5 or less serve the application, whereas the other 95 or more provide the reconfigurability), and, have a much slower clock speed than a CPU. Routing congestion may even further degrade FPGA efficiency. To answer, it is the von Neumann syndrome [17], looks a bit unfair. It's the typical environment which is so inefficient, that the much better processor technology is left behind the leading edge by orders of magnitude. It's a software engineering issue, that multiple levels of overhead lead to massive code sizes which hit the memory wall [18]. Nathan's law by Nathan Myhrvold, a former CTO of Microsoft, sais that software is a gas, which fills any available storage space (on-chip memory, extra semiconductor memory located outside the processor chip), as well as hard disks, and even the internet. The memory wall is partly a technology issue – not fully the paradigm's fault.

```
loop i = 2...N
  loop j = 2...N
    if key[j-1] > key[j] then
      swap (key[j-1], key[j])
    endif;
  endloop j;
endloop  i;
```

Fig. 6; Bubble Sort Algorithm

A Migration Example. Software to configware migration and software to hardware migrations depend on the same principles, since both are time to structure mappings. The difference is binding time: before fabrication (hardware), or after fabrication time (configware). For illustration we show the migration of the well-known $O(n^2)$ running time bubble sort algorithm [19] (fig. 7), fully based on memory-cycle-hungry CPU instruction streams, also for reading and storing the data. Our bubble sort example's first migration step by a first parallelization attempt is shown in fig 7a. The little square boxes are registers for keys k_i. The ensemble of all key registers is a bidirectional

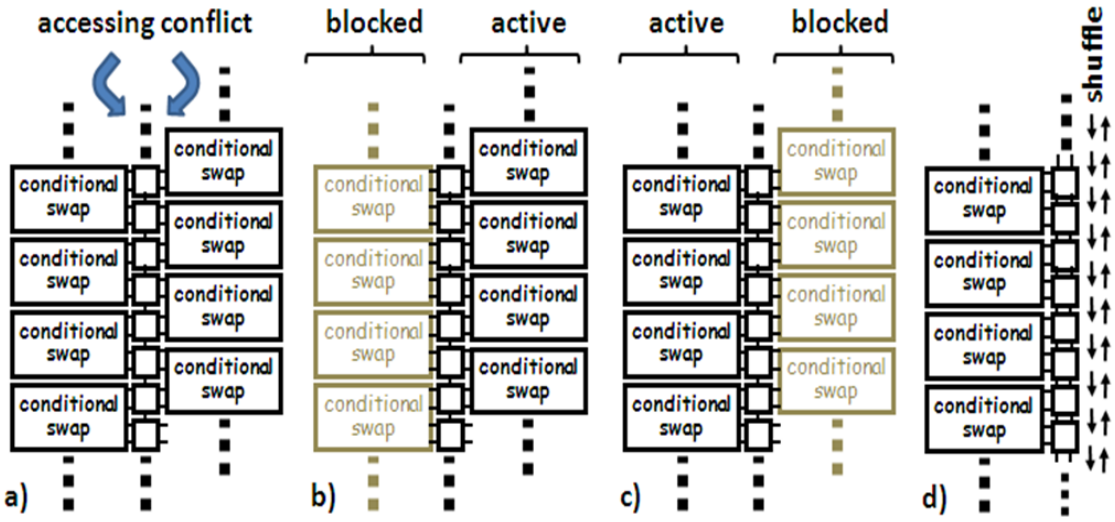

Fig. 7; The bubble sort algorithm: a) conflict by straight-forward parallelization, its solution: each execution step split into 2 phases: see b) and c); saving 50% conditional swap units by d) shuffle move

pipeline register array. Before sorting all data k_i for all I can be piped down into this array. But this solution comes with accessing conflicts, since always 2 conditional swap units access the same key register k_i at the same time. This "bug" can be removed by splitting the operation into 2 phases (see fig 7 b and 7 c). But this solution wastes resources, since at any time only 50% of the conditional swap units are buzy. For optimization we change the algorithm into the "shuffle sort" algorithm (fig. 6 d) [20, 21, 22] having only half as many conditional swap units. But here only half the number of k_i / k_j pairs is processed. For avoiding to break bubbling up we move the contents of the k register pipeline up and down, each time by a single step: a shuffle movement. The algorithmic complexity turns from $O(n^2)$ (fig. 6) into O(n). In a similar manner, other well-known algorithmic methods can be transformed to explore parallelism and locality, like in dynamic programming as presented in [12]. Needing no CPU instructions brings additional massive speed-up. Since software is usually stored outside CPU on-chip memory, the memory wall [18] and overhead-phenomena typical to software cause performance by additional orders of magnitude worse than that of the migrated version.

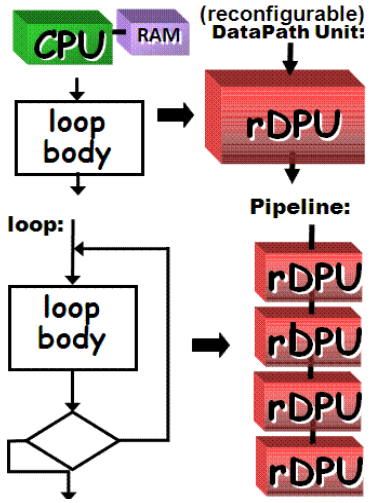

Fig. 8; Loop to pipe conversion.

1How Data Are Moved is a key issue. CPUs usually move data between memories and requires instruction streams to carry it out (first line, table 3). This means the movement of data is evoked by execution of instructions due to the von Neumann paradigm. Also the execution of operations inside a CPU requires reading and decoding of instructions. On a reconfigurable platform, however, which can be modeled as a pipe network, data are moved directly from DPU to DPU. This means, that operation execution inside a DPU (not having a program counter) is "transport-triggered" (second line, table 3). It is triggered by the arrival of the data item, not needing an instruction to call it. Not looking at dynamically reconfigurable systems ([23] only for advanced courses) we see, that

reconfigurable fabrics ***don't perform any instruction sequencing at run time.***

Neurocomputing. Another opportunity for massively saving energy is a paradigm shift to neurocomputing, pioneered by Karl Steinbuch [24, 25] having invented the „Lernmatrix", the first technically fully usable artificial neural network [26, 27]. Meanwhile neurocomputing is booming with an increasing number of also interdisciplinary research areas [28, 29]. The Lernmatrix is e. g. the basis of an artificial retina developed by Carver A. Mead for his company Foveon [29, 30] (meanwhile acquired by camera specialist SIGMA). Meanwhile several other neurocomputing network architectures are around [28, 29].

"The human brain needs only a few watts for about 10,000,000,000,000,000 (ten million billion) compute operations per second. Von-Neumann-paradigm-based it would need more than a MegaWatt, the electric power needed for a small town and it would sizzle off in a fraction of a second." [Alfred Fuchs].

Memristors. From the existence of resistor, capacitor and inductor it has been reasoned by symmetry arguments that there should be a fourth fundamental element (fig. 9) "memristor" (short for memory resistor) [31]. Already in 1961 when introducing the Lernmatrix Karl Steinbuch proposed such a variable resistor adjusting itself during learning phases of this artificial neural network [26]. For about 3 years such memory-capable resistors not needing an internal power supply had been available from Memistor Corp. founded in 1962 by Prof. Widrow (Stanford). However, its technology did not survive, since it could not follow Gordon Moore's law. Recently Hewlett Packard reported technologically survivable nano-scale memristors based on a relatively simple metal / oxide / metal layered nano structure from titanium dioxide and platinum electrodes [32] which also supports direct synapse imitations (like the nodes of a Lernmatrix) avoiding the need for very complex extremly time-consuming and power-consuming digital simulations. The memristor is promising a break-through not only in neurocomputing but also for computing in general. HP people try to boost Moore's law by higher performance at lower power consumption since a single memristor can execute the same logic functions as several transistors. Memristors can also be used for more compact faster more energy-efficient alternatives for flash memory. At HP they have demonstrated this also by a memristor-based FPGA which needed substantially less transistors.

Problems We Must Solve: (1.) A mass migration from software to configware – benefit: massively saving energy, much higher performance, gaining very high flexibility. (2.) From time to time only a smaller part of legacy software can be migrated. For optimization we need to develop a migration priority list of legacy software to identify the most promising candidates. Should neurocomputing also be part of this list? The result will mostly be hetero systems, where programmers with an instruction-stream-based sequential-only mind set are not qualified. (3.) So another obstacle to be removed is, that a qualified programmer population missing. We have to cope with a massive programmer productivity decline for two reasons: (a.) to cope with the many-core crisis where more parallel programming qualifications are needed to an increasing extent, and (b.) to an increasing extent hetero systems will have to be programmed (like for modern FPGAs featuring all 3 on the same chip: reconfigurable fabics, hardwired function blocks, and CPUs) which requires twin paradigm programming skills. As a consequence we need innovative undergraduate programming courses [33] which also teach a sense for locality. The extension of the non-sequential part of education should be optimized not to scare away undergraduate students. Lab courses should be MathWorks-supported model-based, mainly at the abstraction level of pipe networks.

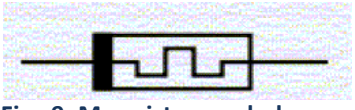

Fig. 9; Memristor symbol

New Horizons Of Research And Development. We need a new generation of talented innovative scientists and engineers to start the beginning second history of computing, not only for the survival of our important computer-based cyber infrastructures, but also

for developing and integrating exciting new innovative products for the transforming post PC era global information and communication markets [34]. Masses of highly qualified new kinds of jobs must be created to meet the fascinating challenges of reinventing computing sciences.

Literature

[1] http://wiki.answers.com/Q/Why_are_computers_important_in_the_world

[2] Ernst Denert: Erfahrungen und Einsichten eines Software-Unternehmers; 28. Okt 2004, TU Kaiserslautern.

[3] W. Blendinger: Post Peak – Abstieg vom Peak Oil; ASPO annual conf., Berlin, 18 Mai 2010

[4] K. Aleklett: Post Peak – The Future of the Oil-Production; ASPO ann. conf., Berlin, 18 Mai 2010

[5] Randy H. Katz: Tech Titans Building Boom; IEEE Spectrum, February 2009

[6] C. Freeman: "Power Reduction for FPGA using Multiple Vdd/Vth"; Monday April 3, 2006;

[7] Albert Zomaya (keynote): Reconfigurable Architectures Workshop, RAW 2009, Rome, Italy

[8] Horst Simon (invited presentation): Leibniz-Rechenzentrum, TU Munich, 2009, Garching, Germany

[9] G. Fettweis: ICT Energy Consumption - Trends and Challenges; WPMC'08, Lapland, Finland, 8 –11 Sep 2008

[10] R. Hartenstein: Why we need Reconfigurable Computing Education; 1st Int'l Workshop on Reconfigurable Computing Education (RC education 2006), March 1, 2006, KIT Karlsruhe Insitute of Technology, Germany

[11] N. Conner: FPGAs for Dummies - FPGAs keep you moving in a fast-changing world; Wiley, 2008

[12] R. Jacobi, M. Ayala-Rincón, L. Carvalho, C. Llanos, R. Hartenstein: Reconfigurable systems for sequence alignment and for general dynamic programming; Genetics and Molecular Research 2005

[13] T. El-Ghazawi et al.: The Promise of High-Performance Reconfigurable Computing. IEEE Computer Feb 2008

[14] Joao Cardoso, Michael Huebner (Editors): "Reconfigurable Computing" Springer Verlag 2010

[15] Voros, Nikolaos; Rosti, Alberto; Hübner, Michael (Eds.): *"Dynamic System Reconfiguration in Heterogeneous Platforms - The MORPHEUS Approach"*; Springer Verlag, 2009

[16] Ch. Bobda: Introduction to Reconfigurable Computing - Architectures, Algorithms, Applications; Springer, 2007

[17] R. Hartenstein: The von Neumann Syndrome; Stamatis Vassiliadis Memorial Symp., Sep 2007, Delft, NL

[18] S. McKee: Reflections on the memory wall; Proc. 1st conf. on Computing Frontiers, Ischia, Italy, 2004

[19] D. Knuth: The Art of Computer Programming, Vol. 3, Sorting and Searching, Addison-Wesley, 1973

[20] R. Hartenstein, A. G. Hirschbiel, M. Weber: MOM-map-oriented machine-a partly custom-designed architecture compared to standard hardware; Proc. IEEE CompEuro, Hamburg, Germany, May 1989

[21] Fig. 13. in R. Hartenstein: The History of KARL and ABL; in: .J. Mermet (editor): Fundamentals and Standards in Hardware Description Languages; ISBN 0-7923-2513-4, Kluwer, 1993.

[22] M. Duhl: Incremental Development and Description of a Shuffle Sort Array Circuit in hyperKARL from the Algorithm Representation of the Bubble Sort Algorithm; Projektarbeit, Informatik, Univ. Kaiserslautern 1988

[23] M. Huebner, D. Goehringer, J. Noguera, J. Becker: Fast dynamic and partial reconfiguration Data Path with low Hardware overhead on Xilinx FPGAs; Proc. RAW 2010, Atlanta, USA , April, 2010

[24] R. Hartenstein: Steinbuch; Neue Deutsche Biographie (NDB), vol. 25, Duncker & Humblot, Berlin 2010

[25] B. Widrow et al.: 1917 Karl Steinbuch 2005 - Eulogy; IEEE Computational Intelligence Society Newsl. Aug. 2005

[26] K. Steinbuch: Die Lernmatrix; Kybernetik 1 (1961), S. 36-45

[27] K. Steinbuch, U. Piske (1963): Learning matrices and their applications" IEEE Trans EC, Dec 1963

[28] R. Rojas: Neural Networks; Springer Verlag, 1996

[29] R. Hecht-Nielson: Neurocomputing, Addison-Wesley, 1990;

[30] C. A. Mead: Analog VLSI and Neural Systems, Addison-Wesley, 1989

[31] L. Chua: Memristor—The missing circuit element. IEEE Trans on Circuits Theory 18 (1971), pp. 507–519

[32] R. Stanley Williams: How We Found the Missing Memristor; IEEE Spectrum, Dec. 2008

[33] R. Hartenstein (keynote): Reconfigurable Computing: boosting Software Education for the Multicore Era; IV Southern Programmable Logic Conference (SPL 2010), Porto Galinhas Beach, Pernambuco, Brasil, 24-26 March 2010

[34] P. Cowhey, J. Aronson: Transforming Global Information and Communication Markets, MIT Press 2009