A PSEUDO PARALLEL ARCHITECTURE FOR SYSTOLIC ALGORITHMS

R.W. Hartenstein, A.G.Hirschbiel, M.Weber

Universität Kaiserslautern, Postfach 3049, D - 6750 Kaiserslautern, FRG
phone: (+49-631) 205-2606 or (+49-7251) 3575, fax: (+49-631) 205-3200

Abstract

The map-oriented machine (MoM) is a flexible non-von-Neumann accelerator having been developed at Kaiserslautern University. This machine uses a 2-dimensional map-oriented data memory, over which a variable-sized window cache can be moved in arbitrary schemes to analyze and change data via the problem-oriented logic unit, which delivers powerful, programmable pattern matching mechanisms. The MoM can be used to speed up signal processing, image processing, VLSI layout processing and many other applications, and it may serve as a systolic array simulator and evaluator. Moreover it can be used as a low-cost, simple, flexible and programmable array emulation computer. In contrast to systolic arrays, where **data streams are moving** through an array of PEs, the MoM keeps **data at fixed locations** in its memory and moves its window cache in an application-specific manner across this memory space.

1 Introduction

This paper introduces a non-von-Neumann innovative computing device, called *Xputer*. It has been shown in [1], that its innovative processor principles fill the technology gap between universal, but slow von Neumann hardware and inflexible, but powerful tailored hardware solutions. The significance of the Xputer goes even further beyond just filling a gap:

- it is as universal as von Neumann principles
- it is faster than von Neumann for most important applications [1-3]
- its standard parts are much simpler than those of a von Neumann processor (much easier to design than a RISC processor)
- its reconfigurable parts are available from stock at a billion dollar 18 vendor PLD market [4]

Von Neumann type computation is dependent on sequential code scanned from a program memory. Xputer programmability and universality, however, makes use of alternative computing structures: electrically reconfigurable hardware, implemented with PLDs or similar components [4 - 6]. That's why Xputer computation is dependent on the programmed interconnect between a cleverly prepared set of simple hardware operators.

1.1 Why non-von-Neumann ?

The shortcomings of the von Neumann machine and its 'bottle necks' have been frequently discussed throughout the decades [7, 8]. Within a von Neumann machine three major bottle necks can be identified which result in any lack of parallelism. That's why von Neumann processors are by orders of magnitude less powerful, than what may be achieved by tailored ASIC or full custom VLSI solutions of com- parable transistor count. Many different kinds of processor architec- tural

remedies have been proposed or implemented, which, however, do not really result in deviations from von Neumann principles [9 - 12]. Only limited improvements have been achieved [13]. The con- currency approach, bundling several von Neumann processors to form parallel computer systems, requires an expensive additional software overhead, which substantially increases the complexity and incompre- hensibility of such a system. Better principles for a universal machine would be desirable: possibly more efficient (acceleration factors up to several orders of magnitude), and more easily to be designed (to keep up with technology progress rapidly). Machine principles are desirable, which achieve a very high degree of parallelism already **within** a single processor: fine granularity parallelism. This would be more efficient than the coarse granularity parallelism of concurrent proces- ses in contemporary parallel computer systems: bundling masses of hardware and causing a huge overhead for coordination and scheduling.

2 Xputer Principles

This section explains the *Xputer* principles by highlighting its differences to the von Neumann partitioning scheme. The Harvard computer partitioning scheme in fig. 1a illustrates, that three throughput bottle necks can be identified within a von Neumann machine:

1) sequential data access (only a single word is accessed at a time)
2) "sequential" ALU (narrow bandwidth ALU: only a single data operator can be activated at a time)
3) program scanning overhead (not every instruction being scanned is a data manipulation instruction)

In 2) for each operator selection also an instruction has to be fetched from program memory. Fig. 1b illustrates how alternative machine principles are derived:

a) sequencer and program memory are removed (bottle neck no. 3 has disappeared)
b) the hardwired narrow bandwidth ALU is replaced by a highly parallel *reconfigurable ALU (RALU)* (bottle neck 2 has disappeared)
c) bottle neck no. 1 has been left over: however, a special data sequencer supports optimization of data access sequencing

Because of a) the *Xputer* does not accept sequential code. That's why conventional compilers cannot be used for application development support. Because of b) *combinational code* is needed instead, which configures an electrically path-programmable PLD- or PGA-based [4 - 6] ALU resource, called RALU (**r**econfigurable ALU), into a highly parallel network of data paths tailored to a particular application. For configuration a new type of application development software is needed, called **X**piler. A RALU does not have a hardwired instruction set: Xputers do not have a fixed word format. That's why extensions by adding more memory planes are possible conveniently.

The data sequencer includes a register array (called *data scan cache*) or- ganized in a special way. It is connected to the RALU by a very wide bandwidth data path (fig. 1b) to avoid a bottle neck. That's why the RALU may execute many data operations in parallel, that within a single clock cycle the results of longer instruction sequences on a von Neumann machine are met. Such a powerful Xputer clock cycle we call an *X cycle*. No program store and no sequencer are needed to call such an *X cycle*. The set of functions carried out in parallel during an X cycle we call an *X function*. Several X functions may reside simultaneously in a RALU. A particular X function may be activated from by

its *subnet select* key (fig. 1b). The source of a subnet select code may be the data sequencer or the host interface. The data sequencer features a hardwired address sequence generator called *MCU* (*Move Control Unit*), which makes the data scan cache follow a particular *scan path* through memory space. The shape of a scan path we call its *move pattern*. The *video scan*'s row by row move pattern through a 2-D memory segment is an example of such a scan path (fig. 4e). At each memory location visited an X cycle may be evoked from the RALU. The sequence of X cycles thus associated with a scan path we call an *X loop*. By a stack mechanism within the data sequencer sequences of X loops,*X tasks*, may be carried out. By a *task select key* the execution of an X task may be evoked from outside the MCU. No program-driven controller is needed for the MCU.

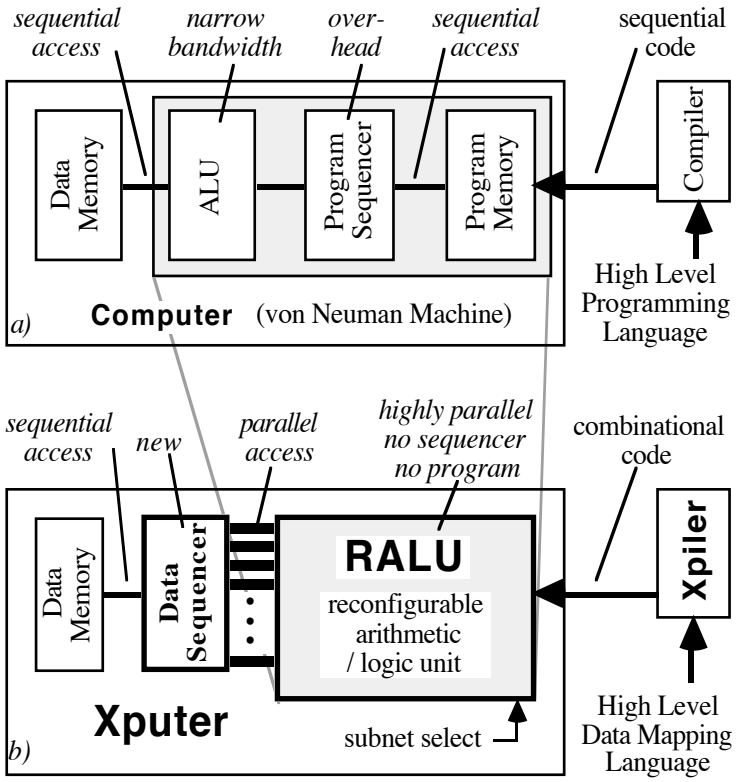

Fig. 1: Computer (Harvard Machine) vs. Xputer

2.1 Programmable vs. Partly Customized Xputer

Currently there are two large flexibility gaps between universal computer solutions and specialized solutions (1) and beyond von Neumann solutions (2) (fig. 2). The Xputer concept offers three environmentally different classes of solutions:

- programmable Xputer (1)
- partly customized Xputer (2a)
- embedded Xputer (2b)

Xputer availability adds two more dimensions of flexibility: for many applications the universal *programmable Xputer* (using PLDs or PGAs) offers much more throughput without loss of flexibility (1). Another dimension of flexibility is the choice of alternative techni- ques (gate arrays or even full custom ASICs for higher throughput) for RALU implementation (2): combination of tailored RALU with standard ICs for data sequencer and host interface (2a), or tailored RALU

combined with data sequencer and interfacing cells taken from a cell library (2b). The advantage of the *partially customized Xputer* over the 'classical' ASIC approach is, that only the RALU part has to be designed, whereas for the rest of the hardware standard ICs may be used. Also the embedded Xputer solution substantially reduces design effort: only the RALU has to be designed, while the rest is made up from library cells provided by the technology vendor. All this helps to save a substantial amount of design time and cost, compared to a fully customized solution. This is an important aspect, since micro- processor development has entered a second phase of design crisis. So the Xputer concept is not only a contribution to the area of computer structures, but also to the area of VLSI design practice [1].

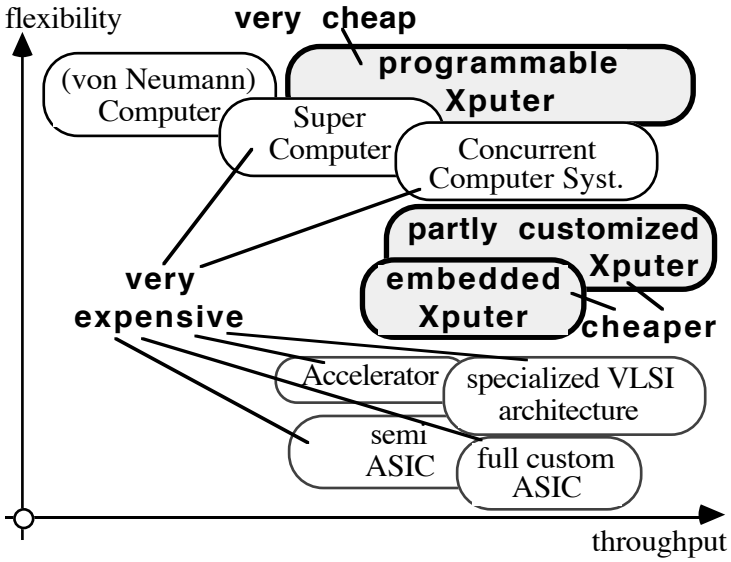

Fig. 2. Xputers: filling the flexibility / performance gap between von Neumann and specialized solutions

3 The MoM Xputer Architecture

The general Xputer principles give room to develop many architectures. This is similar to the von Neumann domain where also many architectures have been developed through the decades. One such Xputer architecture, implemented at Kaiserslautern, is called 'Map- oriented Machine' *(MoM)* or *MoMputer* [31]. Its name is derived from the fact, that this architecture provides a powerful hardware support by the design of the data sequencer for mapping an important class of data dependencies [29, 30, 32] onto the data memory space.

3.1 The Data Scan Cache of the MoM

The MoM's *data scan cache* is a register file for an efficient communication between RALU and data memory. All words in the cache can be accessed in parallel by the RALU. Its size is adaptable to different applications (fig. 3) and can be reconfigured at run-time. It maps a scan window onto the data memory [2, 3, 16] where it holds and updates copies of a few neighbor words from memory for read / modify / write access. This buffer is called *scan cache*, because the MoM architecture supports efficient cache updating while scanning a memory segment. This support is available for a repertory of move patterns (fig. 4) being hardwired in the MCU.

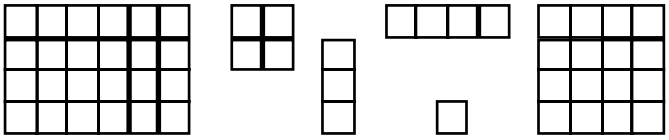

Fig. 3. Data Scan Cache: 2-D Size Adjustment Examples

A 4 by 4 cache format has been used together with a video scan move pattern for design rule check demo [15, 33]: at each X cycle a 4 by 4 segment of grid-based layout is read into the cache, matched in parallel with 800 reference patterns and written back into data memory. For most image processing examples a 3 by 3 scan cache size would do [16]. The above example includes the highly parallel read / modify / write path as the only result path. As a second result a feedback, for data-dependent cache movements, to the MCU may be generated.

In contrast to 1-dimensional von Neumann memory space, the *map-oriented data memory* is primarily 2-dimensional. But at run-time it can be switched to 1-D, 3-D or more dimensions. Inside this memory segments of arbitrary number and size can be defined to provide a memory map similar to floor plans in VLSI. Also nesting of segments is hardware supported. Modern user interfaces encourage graphical presentation of such memory maps.

3.2 The Data Sequencer of the MoM

The data sequencer provides hardwired move patterns for a repertory of scan paths. The MCU supports all kinds of generic move patterns, which are completely defined by their name, parameters and segment limits [31]. The shuffle pattern in fig. 4l is completely defined by its name, its stepwidth (here 3) and the segment length (here 12). Two major cache movement strategies are available and may be combined:

- independent move patterns (e.g. video scan or shuffle)
- data-dependent move patterns (e.g. in curve following)

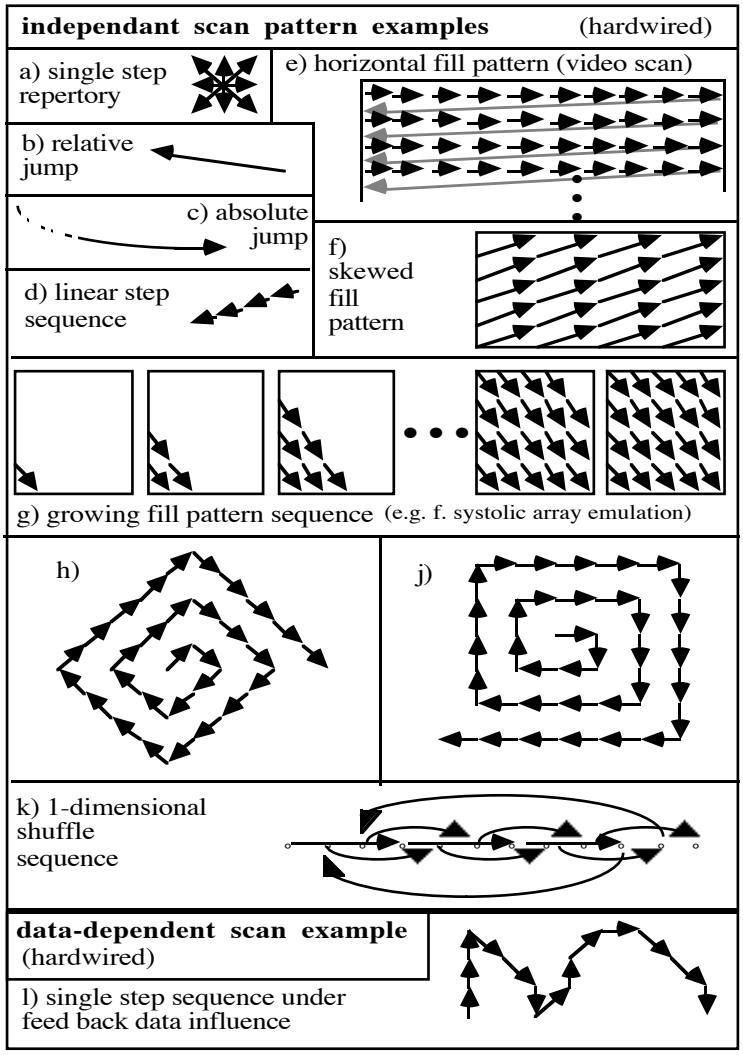

Fig. 4: Some MoMputer Scan Pattern Scheme Examples

Cache movements are controlled by the *move pattern generator* within the MCU using a structured jump generator hardware. It generates the address sequences needed to move the scan cache along a particular scan path. Fig. 4 illustrates part of the repertory of move patterns, such as for single steps and jumps, as well as sequences for video scan, skewed fill patterns, circular scans, useful for the Lee algorithm, shuffles useful for a large number of algorithms [34], or other *travel paths*. By this powerful repertory of hardwired scan paths the data sequencer may carry out long *sequences of X cycles* without the need of a programmable controller. The removal of controllers driven from control code in a program memory strictly avoids overhead and bottle necks at fine granularity level. The benefit is very high throughput and still a high degree of flexibility by:

- hardwired support of regular interconnect schemes (shuffle, butterfly, trellis, spiral and fill scan, and others)
- hardwired support beyond nearest neighbour schemes
- extremely high parallelism at low level in pattern matching
- simple straight forward data sequencing optimization strategy is highly efficient

A third part of the data sequencer is the *task manager unit (TMU),*

which controls the coordination of X loops, linked together to form an *X task*. Due to the architecture of the TMU, the MoM minimizes host interaction for the benefit of speed. Also a partly or fully sequential MCU implementation inside the host would be possible. If typical ALU operators are added to the RALU a hybrid architecture may be created, which adds von Neumann features to the MoM.

3.3 Advantages of Multiple Data Scan Caches

An Xputer can have 2 or more scan caches moving independently at the same time. Such simultaneous cache movements may be synchronized by a common clock. E.g. cache 1 reads operands from data memory in a linear step sequence and cache 2 writes results to data memory in a shuffle sequence, as in fast Fourier transform applications. This multiple scan cache strategy may result in substantial speed up. For a 10 by 10 matrix multiplication dual cache use yields an acceleration factor of sixteen over the single cache version (fig. 5e).

3.4 MoM Implementation

Xputers can carry out any kind of data processing, since being as universal as von Neumann computers. However, Xputers achieve extraordinarily better performance in such problems, where data dependencies can be efficiently mapped onto a two- or more-dimensional memory space. Such problems we call *map-oriented* problems. Xputers aping systolic arrays is of fundamental importance because all systolizable problems are *map-oriented*. That's why Xputers are very efficient for all problems, which may be implemented on systolic arrays.

row #	Operation (512x512 sized memory)	MoM* msec	VAX 11/750 sec	VAX 11/750 acceleration factor	68000 sec	68000 acceleration factor
a)	Grey-Image operations Binary-Image operations	60	7.8	>130	14.7	>240
b)	Erosion, Dilation, Skeleton, Edge Detection	210	38	>180	64	>300
c)	Design Rule Check	260	300	>1000	600**	>2000
d)	Minimum-cost Path (Lee)	150	20	>130	40**	>160
e)	10x10 Matrix multiplication ☐ aping systolic array ☐ using 2 caches	 16 1	 0.04	 (>2) 40	 0.15	 (>9) 150
f)	Viterbi Algorithm	42	1**	>20	1.8**	>40
g)	bubble sort algorithms	100	4.2	>40	8.0**	80

* conservative TTL demo set-up, which has not been tuned ** estimated

Fig. 5: Some MoM acceleration factor examples

The speed benefit of the MoM is illustrated by comparison of benchmarks of a few physical demos having been set up. The comparison (fig. 5) is intended to be fair: a conservative technology MoM demo set-up is compared with a VAX 11/750, which also is technologically not up to date. The benchmarks include areas such as: VLSI layout processing [14, 15], image processing [16], Lee algorithm [17], sorting [19, 40], searching, bit manipulation [24], matrix operations, convolution, signal processing [23], Fourier transform, Viterbi algorithm [25], filtering, ape systolic systems [26], emulation of neuro nets and other cellular systems [27], encryption, grid-based algorithms [28] and other applications. Dramatic improvements have been achieved in design rule check, routing and image processing. E.g. the check of a one million square CMOS design with given design rules takes one second compared to minutes or hours using conventional mini computers. Most benefit by Xputer use will be achieved e.g. in digital

signal processing and image processing.

The MoM architecture has been developed and implemented, using an *eltec 68k system* and an *IBM PC-AT* as user interfaces. Essential standard hardware parts in nMOS technology have been designed, fabricated and successfully tested: scan cache and data sequencer. Hardware and software of two host interfaces have been implemented.

4 Xputer (and MoM) Principles of Operation

Comparing with systolic array operation is a good illustration of Xputer principles of operation. For Xputer execution each moving systolic data stream is converted into a data array stored at fixed me- mory location. On an Xputer not the data, but a data access location is moving along a *scan path* such, that the RALU serves as a single 'processing element' (used in time multiplex mode). Instead of an array of PEs working in parallel on the entire array of data streams only a single PE's locus of activity travels through memory space to visit a variable after the other. Thus by the Xputer the systolic paral- lelism is sequentialized. Compared to a von Neumann approach still a dramatic acceleration is achieved, since the RALU is operating combi- nationally and also other von Neumann bottle necks are avoided. The mapping of systolic arrays onto the MoM takes advantage of existent systolic array synthesis methods. Their output serves as input to the MoM mapping procedure. This procedure falls into four steps:

1. Determine the cache size by the number of local and external registers of one processing element.
2. Determine the memory map of the systolic array by tiling the cache onto the map-oriented memory.
3. Map the work of one processing element onto the MoM's RALU.
4. Sequentialize the parallel work of the PE-array by using a move pattern of the cache over the two-dimensional memory.

An elaborate Xputer application theory is already existing, which provides a rich repertory of methods and tools. It just has to be adapted to the Xputer principles: it is the theory of systolic processing. For a wide variety of application areas [18 - 23] this scene has developed a powerful methodology of data dependency mapping onto the 2-dimensional implementation space of VLSI resources. So this also is a kind of map-oriented processing. This mapping is relatively simple, so that the derivation of the theory of MoM and Xputer processing from the theory of systolic processing is an easy task. The conclusion is, that the theory of MoM application support is almost ready for use. We in Kaiserslautern, for instance, have implemented a systolic array synthesizer SYS^3 [35]. A modified version of SYS^3 also serves as part of our MoM development environment.

4.1 MoM-DE: an Xputer Development System

The code to be generated by a MoMpiler is fundamentally different from the code generated by conventional compilers. While a compi- ler generates fine-grain sequential code to be laid down in a program store, a MoMpiler generates combinational code for path-program- ming the RALU and only small amounts of large-grain sequential code for Xputer task sequencing. For MoM application development *MoM-DE* (MoM Development Environment) has been implemented [36]. Also a special high level language *MoPL* (Map-oriented Pro- gramming Language) has been implemented as an extended Pascalish procedural language. It has been extended by features which are useful for Xputer programming, as to concisely express generic move pat- terns. MoM-DE includes a *MoMpiler* accepting MoPL, which pro- vides CAE tools needed for RALU personalization and loading MCU task register sets. It also includes a high level part for optimization by pseudo-systolic

data dependence remapping [32], which uses an extended version of the systolic array synthesizer *SYS*^{3}. MoPL includes *PaDL* (Pattern Description Language) as a sublanguage, which efficiently supports RALU programming for pattern matching applications. Image processing, Lee routing and design rule check have been implemented with these tools. MoPL uses the traditional procedural style so that training effort to introduce Xputer program- ming principles is minimized. This helps to smoothly embed Xputer paradigms into a conventional programming scene that it may be con- veniently introduced to programmers with conventional background.

4.2 MoM and Xputer Paradigm

A computation problem can be expressed as a dependence graph (DG) with each node describing an operator and the directed edges describing its data dependency [30]. One can convert a DG into a computation graph (CG) spanning over the time-space domain, where DG nodes are annotated with time/space subscripts indicating when and where each of the operation nodes is mapped onto a processor resource. Each CG vertex represents a variable and each edge describes the data dependency of a vertex pair. Phase I of the MoM-DE converts systolizable DGs and many other type DGs into pseudo-systolic MoM-optimal data memory maps which make use of the hardwired MoM repertory of generic move patterns [32]. A pseudo-systolic memory map is a CG that features optimum data sequencing: duality of memory map and move pattern: (re)arrange location of data within a memory seg- ment such, that an optimum move pattern can be achieved. Most effi- cient are move patterns, which are hardwired into the Xputers data se- quencer. The domain of the duality concept of pseudo-systolic memo- ry map and generic data sequencing schemes, supported by Xputer data sequencing hardware and MoM-DE, by far exceeds the domain of systolizable DGs, since it also features individual jumps throughout data memory space. With a high level task sequencer or a conventional host the universality of the von Neumann machine is reached.

5 Conclusions

Xputer principles are a promising alternative to von Neumann: the narrow bandwidth standard ALU is replaced by a very wide bandwidth hardware resource. In many important application areas Xputer use yields dramatic acceleration factors. The MoM Xputer architecture has been developed and implemented at Kaiserslautern, along with a number of demo applications and an experimental hardware demo setup. Essential standard hardware parts in nMOS technology have been designed, fabricated and successfully tested. Many other Xputer architectures are feasible. The MoM architecture example combines von Neumann flexibility and generality with speed advantages of specialized hardware solutions. It is much cheaper and much more universal than fully customized hardware solutions. Its standard parts are much more simple, than a von Neumann microprocessor: the design is much less costly than designing a RISC processor.

New theories and methodologies needed for Xputer application support have been easily derived from the theories of systolic processing as well as from the methodology of integrated circuit design. An experimental application development environment has been developed and implemented. The data-dependence-driven Xputer paradigm is an excellent subject of modern visualization techniques. This paradigm and its typical way of memory space mapping may be more easily supported by visual presentations, than the string-oriented von Neumann world of sequential processes.

The universality from von Neumann computers has been achieved and proven for Xputer architectures [41]. Xputer principles still have a high potential for future improvements and continuous product inno- va-

tions and thus for dynamic long range marketing strategies based on growing product families. Already today it will not be difficult to create a market for Xputer hardware and software. By introducing Xputer principles a new academic discipline in computer science re- search and engineering is supported, much of the theoretical and prac- tical background of which is available to be adapted from the areas of digital signal processing, systolic arrays, ASIC and VLSI design. Also a new direction of research in compilers, programming langua- ges and their architectural support, which stresses low level paralle- lism and data dependence analysis [37 - 39] is supported. So also pro- gress in conventional computing will benefit from Xputer research.

6 Literature

[1] R.W. Hartenstein, A.G. Hirschbiel, M.Weber: MoM - a partly custom-designed architecture compared to standard hardware, *W.E. Proebster, H. Reiner: CompEuro '89,* IEEE Press, 1989.

[2] R.W. Hartenstein, A.G. Hirschbiel, M.Weber: MoM - Map Oriented Machine; *E. Chiricozzi, A. D´Amico: Parallel Processing and Applications;* North Holland, 1988.

[3] R.W. Hartenstein, A.G. Hirschbiel, M.Weber: MoM - Map-oriented Machine; *T. Ambler, P. Agrawal, W. Moore: Hardware Accelerators;* Adam Hilger, Bristol 1988.

[4] N.N.: Programmable Logic Devices, The Second Generation; *Electronics,* May 1988.

[5] R.Freeman: User-programmable Gate Arrays; *IEEE Spectrum*, p. 32-35, December 1988.

[6] Xilinx Corp.: *The Programmable Gate Array Design Handbook;* San José, 1986.

[7] H.W. Lawson, B. Magnhagn: Advantages of Structured Hardware; *Proc. 2nd Symp. on Computer Architecture,* 1975.

[8] R.W. Hartenstein, G. Koch: The Universal Bus Considered Harmful; *R.Hartenstein, R.Zaks: Microarchitecture of Computer Systems;* North Holland, 1975.

[9] G.S. Sohi, S. Vajepeyam: Tradeoffs in Instruction Format Design for Horizontal Architectures; *Proc. ASPLOS-III (Boston 1989),* IEEE Computer Soc. Press, Washington, 1989.

[10] N. N.: Inside Technology; *Electronics,* April 1988.

[11] D.A. Patterson: Reduced Instruction Set Computers; *Comm. ACM, **28**, 1,* 1985.

[12] R.S. Nikhil, Arvind: Can Dataflow Subsume von Neumann Computing?; *Proc. 16th ISCA,* IEEE CS Press, 1989.

[13] N.P. Youpi, D.W. Wall: Available Instruction Level Parallelism for Superscalar and Superpipeline Architectures; *Proc. ASPLOS-III,* IEEE Computer Society Press, 1989.

[14] W. Nebel, *CAD-Entwurfskontrolle in der Mikroelektronik*. Teubner, Braunschweig, FRG, 1985.

[15] R. Hartenstein, R. Hauck, A. Hirschbiel, W. Nebel, M. Weber: PISA - A CAD package and special hardware for pixel oriented layout analysis, *Proc. ICCAD 1984,* IEEE CS Press, 1984.

[16] R. Hartenstein, A. Hirschbiel, M. Weber: A Flexible Architecture for Image Processing; *Microprocessing and Microprogramming 21*, 1987.

[17] C.Y. Lee: An Algorithm For Path Connections And Its Applications; *IEEE Trans. EC-10*, September, 1961.

[18] W. Moore, A. McCabe, R. Urquhart: *Proc. 1st Int`l Conf. on Systolic Arrays (Oxford, UK, 1986);* Adam Hilger, 1986.

[19] K. Bromley, S.Y. Kung, E. Swartzlander jr.: *Proc. 2nd Int`l Conf. on Systolic Arrays;* IEEE Computer Soc. Press, 1988.

[20] J.G. McCanney, J. McWhirter, E. Swartzlander jr.: *Systolic Array Processors;* Prentice-Hall, 1989.

[21] E. Swartzlander jr.: *Systolic Signal Processing Systems;* Marcel Dekker, New York 1987.

[22] S.Y. Kung: *VLSI Array Processors;* Prentice-Hall, 1988.

[23] S.Y. Kung, H.J. Whitehouse, T. Kailath: *VLSI and Modern Signal Processing;* Prentice-Hall, Englewood Cliffs 1985.
[24] N. Charachorlo, et al.: A Million Transistor Systolic Array Graphics Machine; in: [24] p. 193 - 202.
[25] K.A. Wen, et al.: Implementation of Array Structured Maximum Likelihood Decoders; in: [24], p. 227 - 236.
[26] S.Y. Kung: Parallel Architectures for Artificial Neural Nets; in [24], p. 163 - 174.
[27] F. Blayo, P. Marchal: Generic Systolic VLSI Chip for Cellular Automata; in [25], p. 655 - 664.
[28] K. Stüben, U. Trottenberg: Multigrid Methods: Fundamentals, Algorithms, Models, Problem Analysis and Applications; report, SFB 72, Univ. Bonn, F.R.G., Sept 1982.
[29] R.W. Hartenstein, A.G. Hirschbiel, M.Weber: Mapping Systolic Arrays onto the Map-Oriented Machine; in: [25]
[30] W.T. Lin, J.P. Hwang: A High Speed Shuffle Bus for VLSI Arrays; *IEEE J. Solid-State Circuits* ***1*** *(1)* 41-68 (1983).
[31] A.G. Hirschbiel, M. Weber: *The Kaiserslautern MoM Architecture and its Implementation,* report, Kaiserslautern 1989.
[32] M. Weber: Transforming Systolic Algorithms into MoPL Programs; internal report; Univ. Kaiserslautern, 1989.
[33] C. Mead, L. Conway: *Introduction to VLSI Systems*. Addison Wesley, 1980.
[34] H.S. Stone: Parallel Computing with the Perfect Shuffle; *IEEE TC-20 (1971),* p. 153 - 161.
[35] R. Hartenstein, K. Lemmert: SYS^3 - A CHDL-Based CAD System for the Synthesis of Systolic Architectures; *J. Darringer, F. Rammig: Hardware Description Languages and their Applications;* North Holland, 1989.
[36] A.G. Hirschbiel, M. Weber: *Methodology of MoM application support,* internal report, Univ. Kaiserslautern, 1989.
[37] P. Tang, P.-C. Yew, C. Q. Zhu: Impact of Self-Scheduling Order on Performance of Multiprocessor Systems; *1988 Int'l Conf. on Supercomputing*, (July 1988) p. 593 - 603.
[38] D. Kuck, et al.: Dependence Graphs and Compiler Optimizations; *Conf. on Princ. of Program. Languages,* 1981.
[39] D. Kuck, et al.: The Effects of Program Restructuring, Algorithm Change, and Architecture Choice on Program Performance; *Int`l Conf. on Parallel Processing;* 1984.
[40] M. Weber: Using the MoPL Language for Programming the Bubble Sort Algorithm for the MoM; report, Universität Kaiserslautern, 1989.
[41] R.W. Hartenstein, A.G. Hirschbiel, M. Weber: Non-von-Neumann: is the Technology Transfer an Achievable Goal?; internal report, Univ. Kaiserslautern, 1989.