Non-von-Neumann: is the Technology Transfer an Achievable Goal?

Prof. Dr.-Ing. R. W. Hartenstein
Dipl.-Inform. A. G. Hirschbiel
Dipl.-Inform. M. Weber

This paper has been submitted for a conference in 1990. However, due to proposals by reviewers, it has been changed substantially. The new version has been published under the title (same authors):

"Xputers: An Open Family of
Non-von-Neumann Architectures"

in Proceedings GI/ITG Conference on the Architecture of Computing Systems (Munich, 1990); VDE-Verlag, Berlin, 1990.

Universität Kaiserslautern
Fachbereich Informatik
Postfach 3049
D-67653 Kaiserslautern
Germany

Telefon: ++49 631 205 2606
Telefax: ++49 631 205 2640
Telex: 04 5627 unikl d
E-Mail: hartenst@rhrk.uni-kl.de

Non-von-Neumann: is the Technology Transfer an Achievable Goal?

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

Abstract: The von Neumann machine has 3 bottlenecks: its narrow bandwidth ALU, program accessing, and data accessing. A universal innovative non-von-Neumann principle eliminates 2 of them, so that acceleration factors up to several orders of magnitude can be achieved - already with a single processor. Such a machine, however, does not accept sequential code, since it needs a new kind of programming technique. That's why traditional application support software and techniques cannot be used for it. This causes severe acceptance problems. The paper introduces the new machine's principles, discusses its massive technology transfer problems, and proposes a strategy to overcome these problems.

Contents

1. Introduction

This paper discusses the severe technology transfer problems which stem from innovative non-von-Neumann computing principles. Such massive difficulties also exist for computing principles, which are highly superior to von Neumann w. r. to performance and w. r. to other aspects. An example is the Xputer. It has been shown elsewhere, that its innovative processor principles [1] 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 much faster than von Neumann for most important applications [1, 2, 3]
- its standard parts are much more simple than that of a von Neumann processor (much more easy to design than a RISC processor)
- its reconfigurable parts are available from stock at a billion dollar 18 vendor PLD market [4]

Programmable von Neumann type computation is dependent on code having been laid down in, and, being (sequentially) scanned from a program memory. Xputer programmability and universality, however, makes use of alternative computing structures: electrically configurable end re-configurable hardware, implemented with PLDs ("programmable logic devices") or similar components [4, 5, 6, 7]. That's why Xputer computation is dependent on the programmed interconnect between a cleverly prepared set of simple hardware operators.

Section 2 discusses the need for non-von-Neumann innovations. Section 3 introduces the innovative Xputer principles - an alternative to Computers. Section 4 highlights in more detail the MoM example of an Xputer architecture. Section 5 illustrates the differences between Xputer application development support environments and its traditional counterparts in computer science. Section 6 finally proposes a technology transfer strategy to cope with all these scenario-driven problems.

1.1 The Technology Transfer Problem

The von Neumann Principles are dominating almost all application areas of Computer Science. Not only practically all computers in use are of von Neumann type, but also almost all areas of developers' support software (assemblers, compilers, operating systems, CASE tools etc.) and application software are more or less directly based on von Neumann principles. Not only machine code being generated, but also high level language sources heavily reflect von Neumann principles.

Any technology transfer attempt based on a non-von-Neumann machine use would mean, that almost all of this software as well as methodologies behind it having been developed within decades could not be used for such an innovative machine. New "programming" paradigms and new hardware principles would also cause the need for training and modification of computer science curricula. Attempts to market Xputers would be extremely difficult because of massive acceptance problems. A completely new scene would have to be raised to create such a market.

submitted for the 1990 GI/ITG Conference on the Architecture of Computing Systems

Similar problems are observed in the area of data flow machines, which also use a non-von-Neumann concept. Instead of sequential machine code a data flow graph is used [8, 9]. Also understanding its innovative hardware would require retraining of users. Would not it be a consequence of all this, that for innovative computer principles there is no chance to successfully compete with all these masses of traditional support methodology and its theoretical backgrounds? Wouldn't mean this, that the technology transfer of such a drastic innovation would not be achievable at all ?

2. Why non-von-Neumann ?

The shortcomings of the von Newman machine and its "von Neumann bottle necks" have been frequently discussed throughout the decades [10, 11]. Within a von Neumann machine three major throughput 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 comparable transistor count.

Many different kinds of processor architectural remedies have been proposed or implemented, which, however, do not really result in deviations from von Neumann principles (cache memories, pipelining, vectorization, scalarization, VLIW architectures [13], RISC architectures [14, 15], Data Flow/von Neumann hybrids [16], use of very fast technologies etc.). However, only limited improvements have been achieved [17]. Disappointing results, the lack of a real break-through, motivate looking for new ideas again and again.

Sometimes it is questionable, whether the increased hardware design effort is worth the throughput improvement achieved by it. Waiting for the next technology version might bring more benefit in less time, than would be needed for the additional design effort. Also bundling several von Neumann processors to form parallel computer systems requires a expensive huge additional software overhead, which substantially increases the complexity and incomprehensibility of such a system. Parallelizing and highly optimizing compilers often produce code such, that during debugging it often is extremely difficult to understand, what the hardware really does. Similar problems also stem from optimizing compilers for RISC computers.

Better principles for a universal machine would be highly desirable: possibly more efficient (acceleration factors up to several orders of magnitude - already by a single processor), and more easily to be designed (to keep up with technology progress more rapidly). Machine principles are desirable, which achieve a very high degree of parallelism already within a single processor: fine granularity parallelism. This would be much more efficient, than the coarse granularity parallelism of concurrent processes in contemporary parallel computer systems: bundling masses of inefficient hardware, and causing a huge overhead for coordination, scheduling and dispatching.

3. Xputer Principles

This section explains the non-von-Neumann *Xputer* principles in a more comprehensible way - different from having been published elsewhere [1]. It introduces Xputer principles by highlighting its

submitted for the 1990 GI/ITG Conference on the Architecture of Computing Systems

differences to the von Neumann partitioning scheme. The Harvard computer partitioning scheme in fig. 1 a illustrates, that the three following throughput bottle necks can be identified within a von Neumann machine:

(1) sequential data access (only a single data word may be 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)

For (2) for each operator selection also an instruction has to be fetched from program memory. Fig. 1 b (vs. fig. 1 a) illustrates how alternative machine ("Xputer") principles are derived from the von Neumann machine:

(a) sequencer and program memory are removed (bottle neck no. 3 has disappeared)

(b) the hardwired narrow bandwidth universal ALU is replaced by a highly parallel *reconfigurable ALU (RALU)* (bottle neck no. 2 has disappeared)

(c) bottle neck no. 1 has been left over: however, a special data sequencer (fig. b) supports highly efficient optimization of data access sequencing

Because of (a) the *Xputer* does not accept sequential code any more. That's why conventional compilers cannot be used for application development support. Because of (b) *Combinational Code* is needed

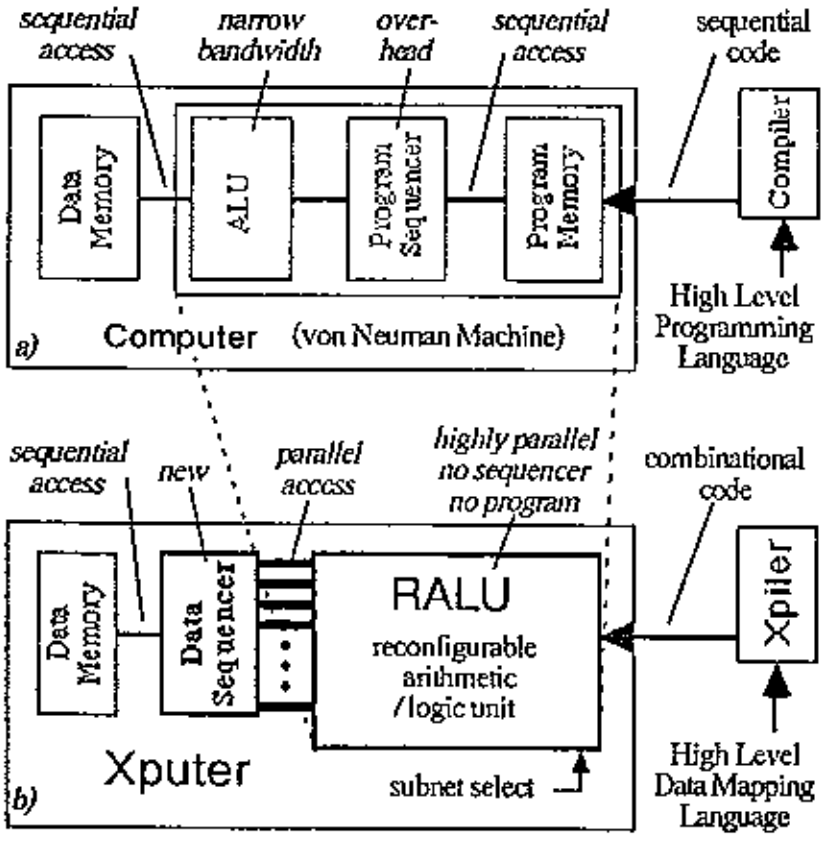

Fig. 1: Computer (Harvard Machine) vs. Xputer

instead, which (re)configures an electrically path-programmable PLD-based [4] or EGA-based [5, 7] ALU resource (called RALU) into a highly parallel network of data paths tailored to a particular application. For configuration a new type of application development software package is needed, which we call Xpiler (instead of compiler for computers). That's why a RALU does not have a hard-wired 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*)organized in a special way (see chapter 4 for more details). This register array is connected to the RALU by a very wide bandwidth data path (see fig. 1 b) in order to avoid a bottle neck. That's why the RALU (reconfigurable ALU) may execute so many data operations in parallel, so that within a single clock cycle the results of longer sequences of instruction executions on a von Neumann machine are met.

Such a highly powerful Xputer clock cycle we call an *X cycle*. No program store and no sequencer is needed to call such an *X cycle*. The set of functions carried out in parallel during such 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 the RALU by its *subnet select* key (see fig. 1 b). The source of a subnet select code may be the data sequencer (see fig. 1 b), or the host interface (see chapter 4, also for more details on the data sequencer and its cooperation with the RALU).

The data sequencer features a hardwired data address sequence generator called *MCU* (for *Move Control Unit*), which makes the data scan cache follow a particular path (*scan path*) through memory space. The MCU makes the scan cache behave like 'traveling salesman', subsequently 'visiting' different memory locations by following such a scan path. The shape of a particular scan path we call its *move pattern*. The *video scan* with its row by row move pattern through a 2-dimensional memory segment is an example of such a scan path (fig. 5 e).

At each memory location visited on a scan path 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 (*TMU*)within the data sequencer also sequences of X loops may be carried out. Such a sequence of X loops we call an *X task*. By a *task trigger* signal and 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 nor the TMU.

3.1 A Few Application Examples

Xputers can carry out any kind of data processing, since being as universal as von Neumann computers (also see section 4.4). However, Xputers achieve extraordinarily much better performance in processing such problems, where data dependencies can be efficiently mapped onto a two- or more-dimensional memory space.Such problems we call *map-oriented* problems.

Fig. 2 gives a few examples of such "map-oriented" problems. For Xputers aping systolic arrays is of fundamental importance because all (pseudo-)systolizable problems are *map-oriented* problems. That's why Xputers are also very efficient for all kinds of problems, which may be efficiently implemented on systolic arrays [1, 40].

VLSI layout processing	design rule check, circuit extraction, compaction [18, 19]
Image processing	pattern recognition, pattern matching, shrink, expand, contour following, segmentation, set operations, ... [19, 20]
Minimum-cost path	Lee algorithm [21, 22]
Other non-numerics	sorting [24], searching, bit map graphics [31]
Arithmetic processing	[24], matrix operations, convolution ...
Signal processing	[23, 24, 25, 26, 27, 28] Fourier transform, other transforms, Viterbi algorithm [32], filtering, ...
Ape Systolic Systems	[1, 40] many applications .... [13, 14, 15]
Emulate Neuro Nets	and other cellular systems [33, 34]
Encryption	[35, 36]
Lattice gas simulation	[6, 37, 38] and .... other
grid-based algorithms	[39]

Fig. 2. Good Xputer application area examples

Such systolic data dependence mappings are relatively mature techniques (for more details see chapter 5). This is also he reason why many of the references in fig. 2 point to the area of systolic array application. A number of suitable application areas is listed in fig. 2. There are more such application areas [23, 24, 25]:

motion detection, maximum likelihood decoders,

least square estimation, least mean square problems,

graph problems, path problems, emulation of shuffle arrays,

sorting, string matching, hashing,

relational data base operations, cryptosystems,

adaptive beam forming, synthetic aperture radar,

differential equations, partial differential equations,

linear state equations, sets of linear equations, recurrence equations,

yield modeling, correlation, eigenvalue problems, and many others

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 data array stored at fixed memory location. On an Xputer not the data, but a data access location is moving along the *scan path* such, that the personalized RALU serves as a single systolic "basic processing element" (here used in a time multiplex mode). Instead of an array of processing elements working in parallel on the entire array of data stream variables only a single processing element's locus of activity travels through memory space to visit one variable after the other. Thus by the Xputer the systolic parallelism is sequentialized. However, compared to a von Neumann approach still a dramatic acceleration is achieved, since the RALU is operating combinationally (i. e.: not sequentially) and also all other von Neumann bottle necks and overheads are avoided. Fig. 9 shows a few acceleration factor examples.

3.2. Xputer: One Principle - Many Architectures

The innovative Xputer computation principles have explained above. These general principles give room to develop many other Xputer architectures. This is similar to the von Neumann domain where also many processor architectures and computer architectures have been proposed and developed through the decades. One such an architecture example is the MoM ("Map-oriented Machine", which we also call "MoMputer"), which will be introduced by next chapter.

Considering the speed benefits and the range of applications to be covered, we could develop different Xputer architectures of the data scanner (compare fig. 1 b) featuring alternative repertories of data accessing sequences (compare fig. 5). Also the basic architecture of the RALU may be subject of architectural considerations. For example a special reconfigurable integrated circuit has been manufactured, which has been optimized for high hardware utilization in highly parallel pattern matching [20, 41].

Also different RALU architectures may be combined for high universality of an Xputer architecture. A *gate level Xputer* uses low level (gate level) PLD resources. Its Xpiler is a kind of logic synthesis system. Such an architecture is extremely flexible, since it does not have any data manipulation instructions at all. All functions used in a particular application are defined entirely by Xpiler generated code. Also, for instance, several adders and several multipliers within one RALU could be configured to operate in parallel.

A *functional level Xputer* uses higher level (functional level, also called: RT level) RALU resources. Its RALU provides a predefined ALU-like operator repertory. Such an architecture could e. g. provide a von-Neuman-like repertory of pseudo instructions. However, the narrow bandwidth of a von Neumann ALU is avoided, since its interconnect used in a particular application is not hardwired, but is defined entirely by Xpiler generated code.

Also *hybrid Xputer architectures* by mixing features of gate level Xputers and functional level Xputers are feasible, e. g. for an optimum combination of universality and very high acceleration (also see next section 3.3). The selector input *subnet select* of the RALU (fig. 1 b) may be used to switch between different such architectures, but also between different applications simultaneously being resident in an Xputer.

3.3 The Universal Xputer

Von Neumann has proven the generality of his machine principles [42]. He has proven, that such a machine can solve any problem, which is computable at all. The repertory of operations available within the ALU is an essential ingredient of this generality. The same generality can be achieved and also be proven for an Xputer architecture, the RALU of which provides a similar repertory of operations. The same generality may be achieved also with RALU concepts which, however, feature high internal parallelism - in contrast to von Neumann. .

Also control instructions like 'fetch next', branch and jump instructions are an essential ingredient of the generality of the von Neumann machine [42, 43]. Also these conditions can be easily met by an Xputer architecture, if it is connected to a programmable controller which supports the data sequencer in a top-down manner. Such a controller , however, does not sequence conventional machine

instructions (very low level: fine granularity sequencing); but it is used at much higher level: it sequences entire tasks (very large granularity sequencing). So a simple controller of only very low capacity needed. This controller is evoked at a very low rate, so that it eats up only a very low percentage of computation time. So this definitely does not mean not a return to von Neumann through the rear entrance.

One such solution is the use of a controller outside the Xputer: the Xputer is used as a co-processor to a conventional processor (see fig. 3 a, also compare fig. 1 b). Conventional control instructions of the host can be used to evoke sequences of X loops within the Xputer. In most important applications there would be an extremely low Xputer to host interaction rate, so that Xputer power would be dominant over von Neumann slowliness, as long as satellite operations are concerned.

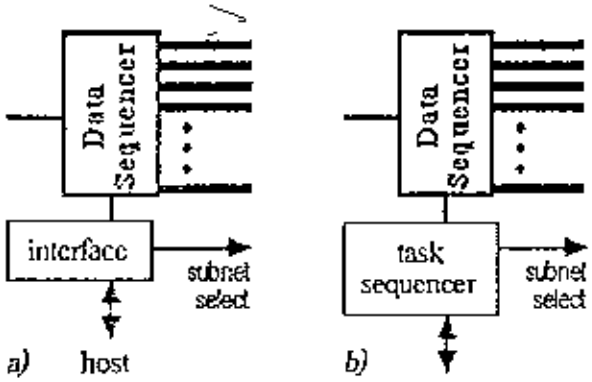

Fig. 3. Xputer control part versions: a) task sequencing by host, b) universal Xputer with internal controler

Another concept would be the use of a *task sequencer* within the Xputer (see fig. 3 b, compare fig. 1 b). (This solution is used by the MoM, see section 3.). This task sequencer includes a small programmable sequencer to evoke sequences of X tasks. This task sequencer can be tailored to meet generality requirements in a similar way, than known from von Neumann. This concept leads to really universal stand-alone Xputer architectures, not needing any support by a host for this generality.

3.4 Programmable vs. Partly Customized Xputer

Where universal computers cannot meet real time requirements sometimes more specialized acceleration machines [44] or special purpose VLSI architectures [45] often are very expensive, providing limited improvement and insufficient flexibility only. Tailored ASIC solutions may provide much more throughput, while being even more expensive and even much less flexible. Currently there are two large gap flexibility gaps between universal computer solutions and these specialized solutions (1) and beyond von Neumann solutions (2). (See fig. 4.) The Xputer offers three environmentally different classes of solutions:

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

Xputer availability adds two more dimensions of flexibility: for many, many applications the universal *programmable Xputer* (using PLDs or programmable gate arrays) offers much more throughput

without loss of flexibility (1). Another dimension of flexibility is the choice of alternative techniques (gate arrays or other ASIC technologies, or even full custom for even higher throughput) for RALU implementation (2): combination of tailored RALU with standard ICs for data sequencer and host interface (2 a), or tailored RALU combined with data sequencer and interfacing cells taken from a cell library (2 b).

The advantage of the *partially customized Xputer* over the "classical" ASIC approach is it, that only the RALU part has to be designed, whereas for the rest of the hardware standard integrated circuits may be used. Also the embedded Xputer solution substantially reduces design effort: only the RALU has to be designed, whereas the rest is made up from library cells provided by the technology vendor or

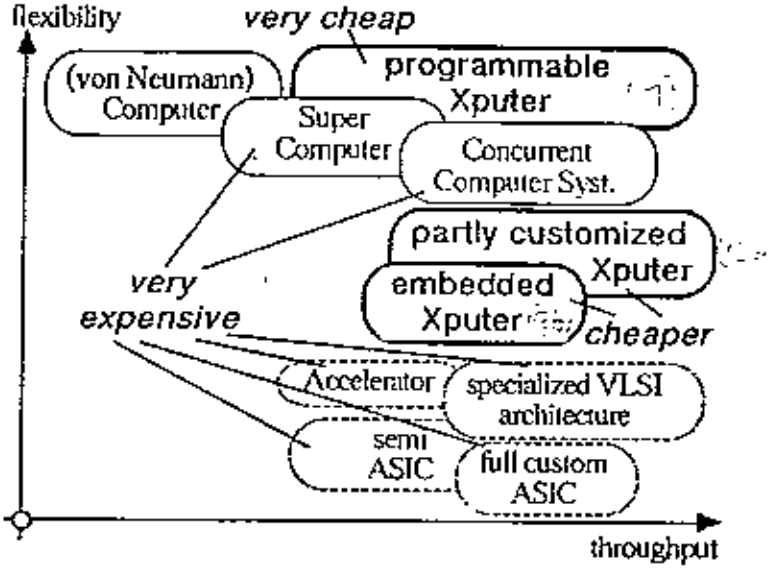

Fig. 4. Xputers: filling the flexibility/performance gap between von Neumann and specialized solutions

All this helps to save a substantial amount of design time and design cost, compared to a fully customized solution. This is an important aspect, since microprocessor development has entered a second phase of design crisis [46]. 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].

4. The MoM Xputer Architecture

Above we have pointed out the possibility to create many different Xputer architectures. One such Xputer architecture, having been implemented at Kaiserslautern is called "Map-oriented Machine" *(MoM)* or *MoMputer* [49, 50, 51]. This name is derived from the fact, that this architecture provides a powerful hardware support for mapping comprehensibly an important class of data dependencies [47, 48, 53] onto Xputer data memory space. Such a mapping technique is achieved by the design of the data sequencer (fig. 1 b). Essentials of the mapping support implementation are its subunits (see fig. 8): the *data scan cache,* and the *move control unit (MCU).*

4.1 The Data Scan Cache of the MoM

The MoM's data *Data Scan Cache* is a reconfigurable register file for an efficient communication between RALU and data memory, such as e.g. for a read/modify/write communication. All words in this cache can be accessed in parallel by the RALU. The scan cache size is adaptable to different applications; it can be reconfigured at run time under parameter control. During window mode it maps a vari-size scan window onto the MoM data memory space [2, 3, 20]. It then holds and updates copies of a few neighbor words from memory for read/modify/write access. Fig. 5 shows some 2-dimensional cache size adjusument examples.

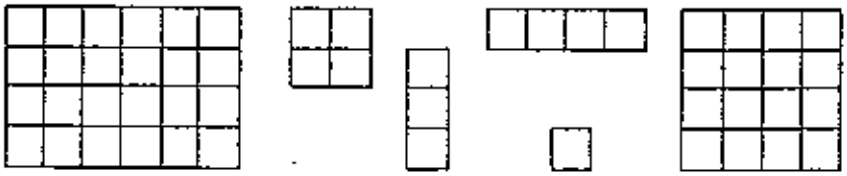

Fig. 5. Data Scan Cache: 2-dimensional Size Adjustment Examples

This buffer is called a *scan cache* or *scan window*, because the MoM architecture supports efficient cache updating [54, 55, 56, 57] when the cache is scanning a memory segment. This hardwired support is available for a repertory of scan patterns or move patterns (for illustration see e. g. fig. 6 a, d through j and l), being hardwired into the MCU (see section 4.2).

By a 4 by 4 cache format, for example, 16 operands are directly connected to the RALU (see the right end of fig. 5). This cache format has been used together with a video scan move pattern for design rule check acceleration demo [19, 58, 59]: at each X cycle a 4 by 4 pixel segment of (grid-based) layout including an error layer is read into the cache, is matched with in parallel with 800 reference patterns [59] (within the RALU, where its error layer is updated), and written back into data memory. For most image preprocessing examples also a 3 by 3 scan cache size adjustment would do [20].

The above design rule check example includes the highly parallel read/modify/write path as the only result path. As a second result path a feedback to the move control unit may be generated (see *decision data path* in fig. 8). This feedback is used for data-dependent cache movements (as e.g. in curve following, fig. 6 l, also see section 4.2).

In contrast to the 1-dimensional von Neumann memory space, the MoM's *map-oriented data memory* is primarily 2-dimensional. But at run-time the dimensionality of the memory also can also be switched to 1-D, 3-D, 4-D or more dimensions [19]. Inside the data memory segments of arbitrary number and size can be defined to provide a memory map similar to floor plans in VLSI (if 2-dimensional mode is used). Also nesting of segments is hardware supported by a stack mechanism. Modern user interfaces encourage a graphical presentation of such memory maps [60].

4.2 The Data Sequencer of the MoM

The data sequencer of the MoM (including the *Move Control Unit (MCU)* [19, 20, 61, 62, 63]) provides hardwired move patterns for a repertory of scan paths (for terminology definition see last paragraphs of section 3.2). The MCU hardware supports all kinds of

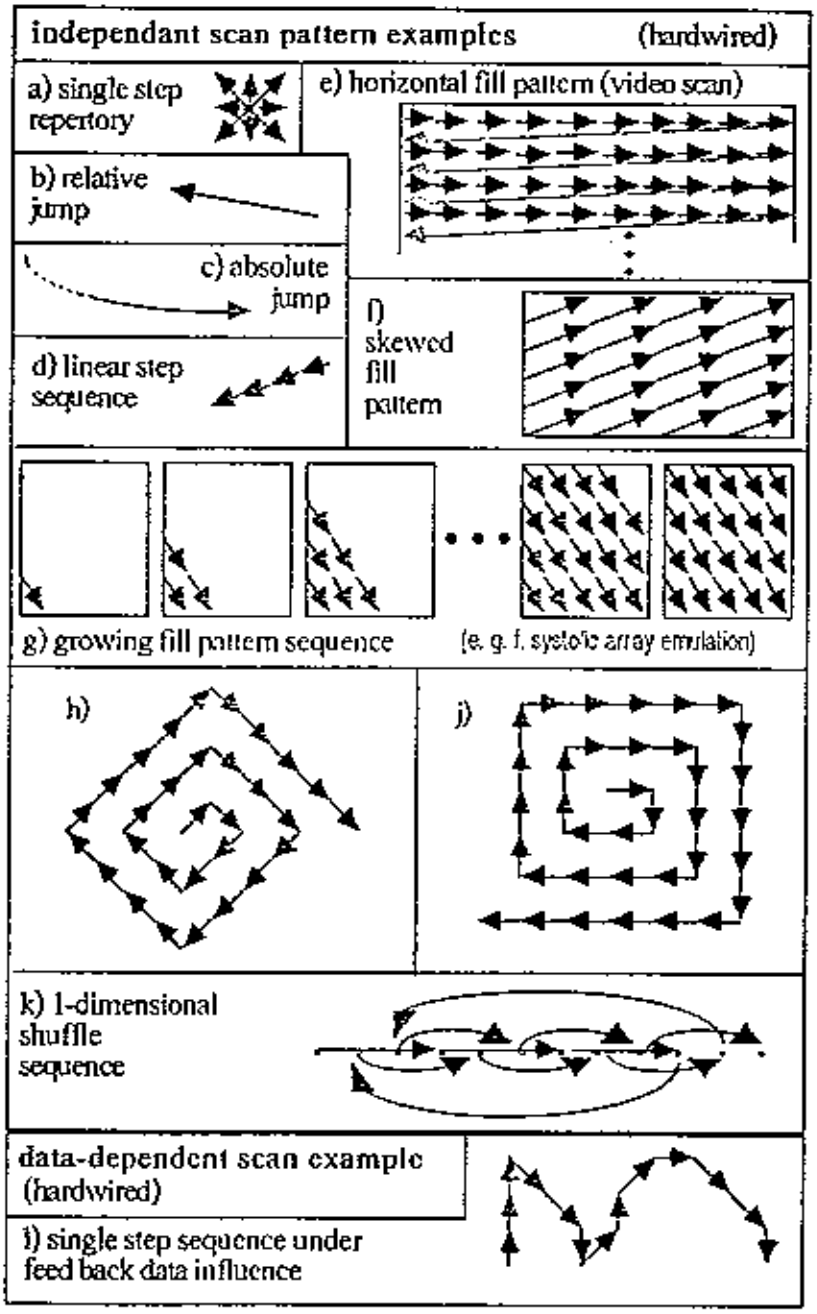

Figure 6: Some MoMputer Scan Pattern Scheme Examples

generic move patterns, which are completely defined by the name (or key) of a pattern, by a parameter, and by the segment limits [51]. The shuffle pattern in fig. 6 l, for instance, is completely defined by its name, its step width parameter [52] (which is 3), and by the segment length (which is 12.) Two major cache movement strategies are available from the MCU and may also be combined:

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

Cache movements are controlled by the move pattern generator [61] within the MCU (see fig. 8) using a structured jump generator hardware [62, 63]. It generates the address sequences needed for moving the data scan cache along a particular scan path Fig. 5 illustrates part of the repertory of move patterns, such as e. g. for single steps to one of the 8

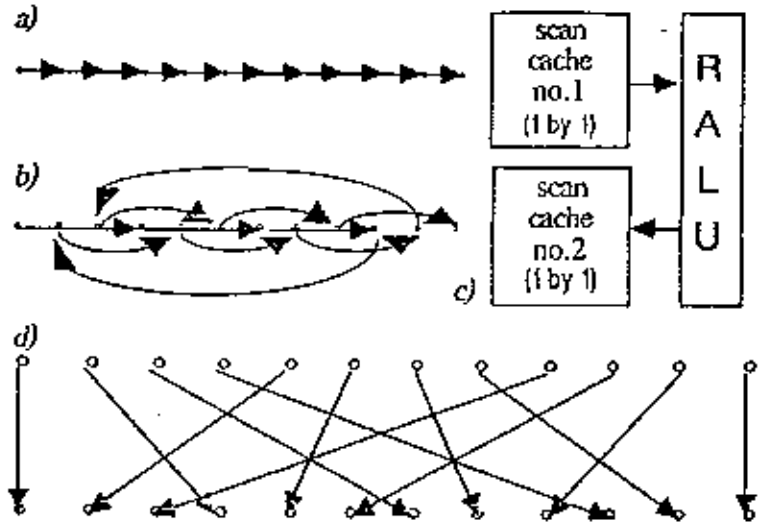

Fig. 7. Shuffle transfer (d) via RALU using double scan cache (c) by two different move patterns (a and b) running in parallel

nearest neighbors (a), for relative (b) and absolute jumps (c) in memory space, as well as step sequences for a 'video scan' (e), skewed fill patterns (f, g), circular scan patterns (h, j) such as useful for the Lee algorithm [21, 22], shuffle sequences (k) useful for a large number of algorithms [64] (see fig. 6 for a double cache shuffle transfer example), or other *'travel paths'* through memory space.

By this powerful repertory of hardwired data scan paths the data sequencer may carry out long *sequences of X cycles*, also without the need to use a programmable controller. The removal of controllers driven from control code laid down in a program memory strictly avoids overhead and bottle necks at fine granularity level. The benefit of this is a very high throughput and still a high degree of flexibility by:

- hardwired support of regular interconnect schemes (shuffle, butterfly, trellis, data stream, spiral scan, n-dimensional: fill scan, and others, also see figs. 6, 7)
- hardwired support far beyond nearest neighbor 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 (fig. 8) is the *task manager unit (TMU)*, which also controls the coordination of X loops [65], which may be linked together to form an X task. Due to the architecture of the TMU, which is tightly coupled to the host interface [66, 67, 68, 69], the Kaiserslautern MoM architecture minimizes MoM/host interaction for the benefit of speed.

However, also a partly or fully sequential MCU implementation inside the host would be possible. If typical ALU operators are added to the RALU (standard circuit or taken from a cell library) even a hybrid architecture may be created, which adds von Neumann features to the MoM (also see section 3.3).

4.3 Advantages of Multiple Data Scan Caches

An Xputer can have 2, 3 or more scan caches, capable of moving independently at the same time. But such simultaneous cache

movements may be synchronized by a common clock, i. e. by X cycles, such, that the coordination results in very attractive functionality. Fig. 6 illustrates an example: cache no. 1 reads operands from data memory in a linear step sequence and cache no. 2 writes results to data memory in a a shuffle transfer sequence, such as e. g. in fast Fourier transform applications. This multiple scan cache strategy may result in substantial speed up. Also see the performance figures of the application demo example in fig. 9 e: for a 10 by 10 matrix multiplication dual scan cache use yields in an acceleration factor of sixteen over the single cache version. By the way: the combination of several X loops, including any mixes of parallel loops and series of loops, we call an *X task*.

4.4 MoM Implementation

The speed benefit of the MoM varies from application to application. This is illustrated by comparison of the benchmarks of a few physical demos having been set up at Kaiserslautern (fig. 9) using the MoM architecture. The comparison is intended to be fair: a conservative technology MoM demo set-up is compared with a VAX 11/750, which also is technologically no more up to date.

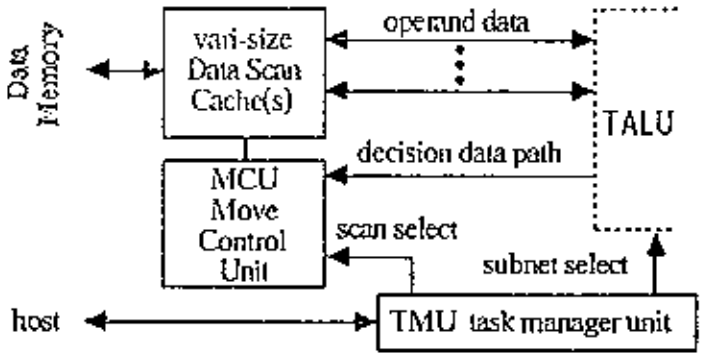

Fig. 8: Data Sequencer of the MoM

A reconfigurable PLD-based RALU resource has been implemented [72]. Major hardware parts have been designed in nMOS technology, manufactured and successfully tested: scan cache [54, 55, 56, 57] and data sequencer [61, 62, 63, 65]. Also hardware and software of two host interfaces [66, 67, 68, 69, 70] have been implemented.

#	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 (λbased)	260	300	>1000	600**	>2000
d)	Minimum-cost Path (Lee)	150	20	>130	40**	>160
e)	10x10 Matrix multiplication aping systolic array using 2 cycles	16 1	0.04	(>2) 40	0.15	(>9) 150
f)	Viterbi Algorithm	42	1**	>20	1.8**	>40
g)	bubble sort algorithms	140	2.2	>15	4.2	30

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

Fig. 9: Some MoMputer acceleration factor examples

Dramatic improvements have been achieved in design rule check, routing and image preprocessing applications, for instance. E.g. the check of a one million square lambda NMOS design with grid-based design rules takes only one second, compared to many minutes or hours using mini computers of the von Neumann type. This is an acceleration by several orders of magnitude. There are many, many other application areas, where Xputer use yields such dramatic acceleration factors (e. g. see above list). Most benefit by Xputer use will be achieved e. g. in the areas of digital signal processing and image processing.

5. New Computation Theory needed ?

Tens of thousands of papers, representing millions of man years in research and development, have contributed to the theory of using the von Neumann architecture and its application development and the methodology has grown over 4 decades. Revolutionarily new processor principles such as the MoM have the disadvantage, that this know-how in existing application support tools (compilers, environments, operating systems) cannot be used any more. Even portations are impossible, since the underlying principles and their theoretical fundamentals do not fit any more.

So a new theory of computation and application development would be needed. This would be a major obstacle in trying to find users. To avoid massive Xputer acceptance problems we propose the following strategy. We believe, that an elaborate Xputer application theory is already existing, which provides a growing rich repertory of methods and tools. It just has to be adapted to the Xputer principles: it is the rapidly developing theory of systolic processing (fig. 7 b).

For a surprisingly wide variety of application areas (e. g. see in [23 - 28]) this currently very active scene has developed a powerful methodology of data dependency mapping onto the mostly 2-dimensional implementation space of VLSI resources. So this also is a kind of map-oriented processing, similar to the principles of the MoM. For further application problem preparation, needed for efficient MoM execution, these results have mainly to be remapped into a more refined time step scale.

This mapping is relatively simple, so that the derivation of the theory of MoM (and Xputer) processing from the theory of systolic processing is a relatively 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 [74, 75], based on this theory. A modified version of SYS^3 also serves as part of our MoM development environment (MoM-DE, see next section).

5.1 MoM-DE: an Xputer Development System

The code to be generated by a MoMpiler (fig. 10 b) is fundamentally different from the code generated by a conventional compiler (fig. 10 a). Whereas a compiler generates fine granularity sequential code to be laid down in a program store, a MoMpiler generates 2 kinds of code: combinational code for path-programming (configuring) the RAL, and only small amounts of large granularity sequential code for Xputer task sequencing.

For MoM application development *MoM-DE*(MoM Development Environment) has been implemented [83]. Also a special high level language *MoPL* (Map-oriented Programming Language) [82] has been implemented, which is an extended Pascalish procedural language. It has been extended by adding procedural features which are useful for Xputer programming, such as e. g. to concisely express generic data sequencing patterns (scan cache move patterns). MoM-DE includes a *MoMpiler* (Phase II, see fig. 10 b) accepting MoPL, and, which provides CAE tools needed for RALU personalization [76, 77, 78, 80] and loading MCU task register sets (for more details also see [65, 79]). It also includes a high level part (phase I) for optimization by pseudo-systolic data dependence remapping [53]. Phase I uses a modified version *SYS³-MoM* of a systolic array synthesizer *SYS³* [74, 75] having been implemented at Kaiserslautern.

MoPL includes *PaDL* (Pattern Description Language) [79, 80, 81] as a sublanguage, which efficiently supports Xputer programming for pattern matching applications. Image preprocessing [20], Lee Routing [21, 22] and Design Rule Check [19, 58, 59] demo applications on the MoM have been implemented with these tools. The PISA Pattern Editor [80, 81], a simple graphical editor, supports convenient editing and inspection of sets of reference patterns. For design rule check and similar applications also an automatic reference pattern generator and optimizer has been implemented [79], which accepts design rules expressed in a simple separation rule language. For Mead & Conway nMOS design rules [58] it has generated 256 reference patterns, for Lambda-based CMOS design rules [59] it has generated 800 reference patterns (which are RALU-executed completely in parallel, see section 3).

Most parts of the MoPL language use the traditional procedural style, so that training effort for introducing Xputer programming principles and practices is greatly minimized. This helps to smoothly embed Xputer supported paradigms into a conventional programming scene, so that it may be conveniently introduced to programmers with conventional background.

5.2 MoM and Xputer Paradigm Summary

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 [47]. One can convert a DG into a computation graph (CG) spanning over the time-space domain, where DG nodes are annotated with time-space indices indicating when and where each of the of the operation nodes is mapped onto (a) processor resource(s). Each CG vertex represents a variable and each edge describes the data dependency of a vertex pair. Phase I of the MoM-DE converts systolizeable DGs and many other type DG schemes into pseudo-systolic MoM-optimal data memory maps which make use of to the powerful hardwired MoM repertory of generic data sequencing schemes *(move patterns)*[53].

A pseudo-systolic memory map is a computation graph (CG) that features optimum data sequencing: duality of memory map and move pattern: (re)arrange location of data within a memory segment such, that an optimum move pattern can be achieved. Most efficient are generic move patterns, which are hardwired into the Xputers data sequencer. The domain of the duality concept of pseudo-systolic memory map and generic data sequencing schemes - supported by Xputer data sequencing hardware and MoM-DE - by far exceeds the domain of systolizeable DGs, since it also features individual jumps

throughout data memory space. With a high level task sequencer use (see section 3.3) or a conventional host the universality of the von Neumann machine is reached.

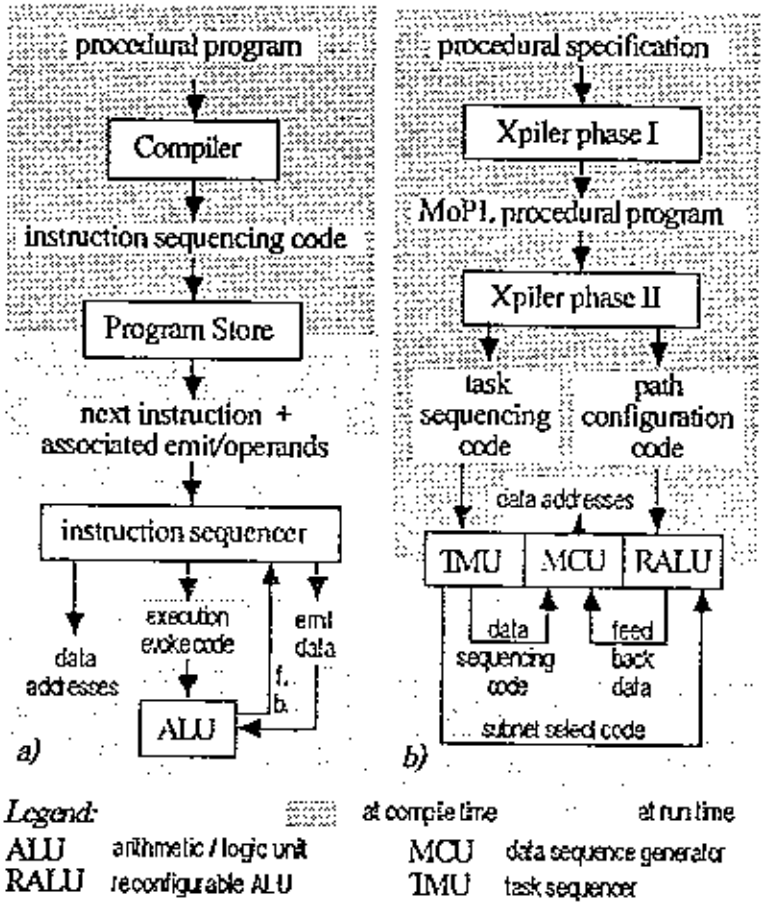

Fig. 10. comparison of compiler targets: a) von Neumann, b) Xpiler for Xputer systems

It may be summarized, that for computation based on Xputers the development of a new theory is not needed (available). Experiments with MoM-DE have shown, that fundamentals for powerful application development support environments can be derived from scientific disciplines already existing. Normal programmers do not need to learn this theory. Due to its universality the Xputer may mainly be programmed also from conventional procedural programs expressed in languages like Pascal. That's why for standard applications no substantial retraining of programmers is required.

For highly advanced Xputer applications, however, some knowledge about the existing theory and about Xputer principles is quite usful, if very high acceleration factors are desired. The portation of existing non-critical software is normally not required, since an Xputer may be conveniently integrated as a co-processor to one or more conventional computing environments. So to-day the time is ready for at least precompetitive industrial prototype developments. The strategy discussed next chapter helps to ease the technology transfer und to rapidly develop markets for Xputer hardware and software. For the future improvement of support tools and for development of better Xputer architectures and for more efficient teaching a research and development scene should be set up to refine the existing theory and ito make it more independent of its roots.

6. Solving the Technology Transfer Problem

The solution of the technology transfer problem associated with non-von-Neumann universal computer applications would be achievable at best in such application areas (1), where it would offer really substantial benefits over von Neumann machine application as well as over ASIC solutions. Substantial accelerations should be achieved already by monoprocessor use only. A second requirement would be, that the development of a completely new science of computation will not be needed (2). The new programming style needed for it should be highly comprehensible and easy to teach (3).

It has been shown, that all these requirements can be met for quite a number of application areas. We will find an optimum Xputer application area:

- where fastest von Neumann processors are too slow (e. g. where real time requirements are very high)
- where Xputer's new paradigms bring substantial benefits (where it is easy to map algorithms into Xputer-specific programming style)

(1) It turns out, that digital signal processing, image processing, bit map graphics and all application areas for systolic arrays, wavefront arrays, and cellular arrays are such areas. The volume of the market in digital signal processing, image processing, and in bit map graphics is by far large enough to keep the investment risk sufficiently low.

(2) Evolution is more easy to achieve than revolution. People should not be forced to give up their familiar environment to get involved in Xputer application. That's why application should support convenient embedding of Xputer use into traditional environments. It should:

- provide standard host interfaces
- provide minimized Xputer/host interaction (not to loose Xputer performance for overhead at the environment side)

The MoM Laboratory at Kaiserslautern has shown, that these requirements have been met. So the time has come to develop industrial precompetitive prototypes and pilote applications using the know how having been aquired from the MoM Lab University environment.

(3) Also the problem of user qualification should be solved evolutionarily. For most users no special skills should be required and only a minimum training effort should be needed. This may be achieved the following way:

- provide application development support which minimizes the need for retraining
- use host for modern human interface techniques (pop-up menus, windows, graphics, program execution visualization etc.)

The MoM Laboratory at Kaiserslautern has shown, that also these requirements have been met. The source languages accepted by MoM-DE support programming the MoM from a conventional

programmer background. Also the innovative paradigms are expressed in a procedural way, so that the familiarity of the notation supports comprehensibility such, that no special skills are required. Using conventional computers as hosts and user interface devices to Xputers make conveniently accessible all modern user interface products available from the market - now and in the future.

7. Conclusions

Xputer principles are a promising alternative to von Neumann: the narrow-bandwidth standard ALU (its computation depends on fine granularity sequential code having been laid down in a program memory) is replaced by a very wide bandwidth hardware resources (where computation depends on compiled interconnect definition). In many important application areas Xputer use yields dramatic acceleration factors, which has been shown by a number of demo applications: design rule check and circuit extraction support [18, 19], Lee Routing [22], image preprocessing [20, 50] and others (see section 4.4). Most benefit by Xputer use is achieved in the areas digital signal processing and image processing, bit map graphics, systolic array emulation and others.

The MoM xputer architecture has been developed and implemented at Kaiserslautern, along with a number of demo applications. The experimental set-up has been implemented using an *eltec 68k System* [67, 68, 69] and an *IBM PC-AT* as a user interface [66]. Also an EDIF interface is available [70]. For the MoM essential standard hardware parts in nMOS technology have been designed, fabricated and tested [55, 56, 57, 61, 62, 63, 65, 72]. Many other Xputer architectures are feasible. Also the MoM architecture example combines von Neumann's 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.

We have shown, that new theories and methodologies needed for Xputer application support have been easily derived from the rapidly growing powerful theories of systolic processing and digital signal processing as well as from the methodology of integrated circuit design. Also an experimental application development environment (MOM-DE) has been developed and implemented at Kaiserslautern [53, 71, 77, 78, 79, 80, 82, 83]. Its MoPL language minimizes the need for retraining: by its use the Xputer paradigm is smoothly embedded into conventional procedural programming language environments. The state of the art already to-day is ready for industrial development of at least precompetitive prototypes. For most applications no special skills are required from Xputer programmers. Already to-day it will not be difficult to create a market for Xputer hardware and software, if the above strategy will be applied.

The data-dependence-driven Xputer paradigm is also an excellent subject of modern visualization techniques [60]. This data-driven paradigm and is typical way of memory space mapping may be more easily supported by visual presentations, than the more string-oriented von Neumann world of sequential processes. Xputer principles have a high potential for future improvements and continuos product innovations and thus for dynamic long range marketing strategies based on growing product families.

By introducing Xputer innovative computational principles a new academic discipline in computer science research and engineering is

supported, much of the theoretical and practical background of which is readily available to be adapted from the areas of digital signal processing, systolic arrays, wavefront arrays, as well as ASIC development and VLSI design. Also a new direction of research in compilers, programming languages, and its architectural support, which stresses low level parallelism (e. g. see [13, 17, 84]) and data dependence analysis (e. g. see [85 - 87]) is efficiently supported by Xputer principles. So also progress in conventional computing will benefit from Xputer research.

8. Acknowledgements

Early versions of parts of the MoM-DE have been developed within the German multi university E.I.S. project, having been funded by the German Federal Ministry of Research and Technology and Siemens-AG, Munich, F.R.G., and having been coordinated by the GMD. This support is gracefully appreciated: without it the Kaiserslautern research on non-von- Neumann computing principles would not have been started, so that the Xputer principles would never have been developed. We also would like to acknowledge the various kinds of personal support we have received from Elfriede Abel, Herwig Heckl, Gustl Kaesser and Klaus Woelcken at GMD, Schloß Birlinghoven. We also appreciate valuable discussions with Klaus Singer from eltec GmbH at Mainz, F.R.G.

9. Literature

[1] R.W. Hartenstein, A.G. Hirschbiel, M.Weber: MoM - a partly custom-designed architecture compared to standard hardware, (*eds.:) W.E. Proebster, H. Reiner: Proceedings Comp Euro '89,* IEEE Press, Washington, DC et al. 1989.

[2] R.W. Hartenstein, A.G. Hirschbiel, M.Weber: MoM - Map Oriented Machine; (*eds.) E. Chiricozzi, A. D'Amico: Parallel Processing and Applications;* North Holland / Elsevier, Amsterdam / New York 1988.

[3] R.W. Hartenstein, A.G. Hirschbiel, M.Weber: MoM - Map-oriented Machine; *eds.: 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] J. P. Gray, T. A. Kean: "Configurable Hardware: A New Paradigm for Computation"; *Proc. Decennial Caltech Conference on VLSI;* Caltech, Pasadena, CA 1989

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

[8] J.B. Dennis: Data Flow Supercomputers, IEEE Transaction on Computers, 13, 11 (Nov. 1980)

[9] J. B. Dennis, D. P. Misunas: A Preliminary Architecture for a Basic Data Flow Processor, *Proc. 2nd Ann. Symp. on Computer Architecture, Houston, Tex., Jan. 1975,* ACM New York 1975, p. 126-132.

[10] H. W. Lawson, B. Magnhagn: Advantages of Structured Hardware; *Proc. 2nd Ann. Symp. on Computer Architecture (Houston, TX, 1975),* IEEE New York 1975

[11] R. Hartenstein, G. Koch: "The Universal Bus Considered Harmful"; in [12];

[12] R.Hartenstein, R.Zaks: *Microarchitecture of Computer Systems;* North Holland, Amsterdam / New York 1975

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

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

[15] D. A. Patterson: Reduced Instruction Set Computers; *Comm. ACM, 28, 1 (Jan. 1985)*

[16] R. S. Nikhil, Arvind: Can Dataflow Subsume von Neumann Computing?; *Proc. 16th ISCA (Jerusalem 1989);* IEEE Computer Society Press, Washington, DC 1989

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

[18] W. Nebel, *CAD-Entwurfskontrolle in der Mikroelektronik.* Teubner, Braunschweig, F. R. G., 1985.

[19] 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 (Santa Clara, CA);* IEEE Computer Society Press, Los Angeles et al. 1984

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

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

[22] I. Velten: Implementierung des Lee-Algorithmus mit der MoM-Entwicklungsumgebung; Proj.-Arb., Un. Kaiserslautern 1986

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

[24] K. Bromley, S.Y. Kung, E. Swartzlander jr.: *Proc. 2nd Int'l Conf. on Systolic Arrays (San Diego, CA, 1988);* IEEE Computer Society Press, Los Angeles et al. 1988

[25] J. G. McCanney, J. McWhirter, E. Swartzlander jr.: *Proc. 3rd Int'l Conf. on Systolic Arrays (Kilarney, Ireland, 1989);* Academic Press, London et al. 1988

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

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

[28] S. Y. Kung, H. J. Whitehouse,T. Kailath: *VLSI and Modern Signal Processing;* Prentice-Hall, Englewood Cliffs 1985

[29] K. A. Collins, J. B. G. Roberts: Stereo Matching of Satellite Images with Transputers; in [24], p. 175 - 182

[30] T. J. Fountain: The Use of Linear Arrays for Image Processing; in: [24], p. 183 - 192

[31] N. Charachorlo, et al.: A Million Transistor Systolic Array Graphics Machine; in: [24] p. 193 - 202

[32] K. A. Wen et al.: Implementation of Array Structured Maximum Likelihood Decoders; in: [24], p. 227 - 236

[33] S. Y. Kung: Parallel Architectures for Artificial Neural Nets; in [24], p. 163 - 174

[34] F, Blayo, P. Marchal: Generic Systolic VLSI Chip for Cellular Automata; in [25], p. 655 - 664

[35] National Bureau of Standards: *"Data Encryption Standard";* Fed. Inf. Process. Standard Publ. 46, January 1977

[36] National Bureau of Standards: *"DES Modes of Operation";* Fed. Inf. Process. Standard Publ. 81, January 1980

[37] U. Frisch, B. Hasslacher, Y. Pomeau: Lattice Gas Hydrodynamics in Two and Three Dimensions"; *Complex Systems 1, 1987*, p 646-703

[38] T. Shimomura, G. D. Doolen, B. Hasslacher, C. Fu: "Calculations using Lattice Gas Techniques"; *Los Alamos Science, Special Issue 1987*

[39] K. Stüben, U. Trottenberg: Multigrid Methods: Fundamentals, Algorithms, Models, Problem Analysis and Applications; report, SFB 72, Univ. Bonn, F.R.G., Sept 1982

[40] R.W. Hartenstein, A.G. Hirschbiel, M.Weber: Mapping Systolic Arrays onto the Map-Oriented Machine (MoM); in: [25], p. 320 - 336

[41] J. Holzer: *Hardware Implementation of a Pattern Recognizer*, Projektarbeit, Univ. Kaiserslautern 1988.

[42] A. Burks, H. Goldstine, J. v. Neumann: "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument" in: *v. Neumann: Collected Works, Vol.V*, Pergamon Press, 1961.

[43] H. Aiken, G. Hopper: "The Automatic Sequence Controlled Calculator", Elec. Eng. *65*, Aug/Sept, Oct, Nov 1946.

[44] T. Blank: "A Survey of Hardware Accelerators used in Computer-Aided Design", *IEEE Design&Test*, August 1984.

[45] A. Fisher, H. Kung: "Special-Purpose VLSI Architectures: General Discussions and a Case Study", *S. Kung et al.: VLSI and Modern Signal Processing*, Prentice Hall, 1985.

[46] R. Newton, D. Ditzel et al.: (panel on) CAD Tool Needs for High Performance Systems; *Proc. 26th Design Automation Conference, June 1989, Las Vegas, Nevada*, IEEE Computer Society Press/ACM, 1989

[47] W. T. Lin, J. P. Hwang: A High Speed Shuffle Bus for VLSI Arrays; *IEEE J. Solid-State Circuits 1 (1)*41-68 (1983)

[48] R. W. Hartenstein, A. G. Hirschbiel, M. Weber: Mapping Systolic Arrays onto the Map-oriented Machine; in: [25], p. 320 - 336

[49] R.W. Hartenstein, A.G. Hirschbiel, M.Weber: *MoM - Map-oriented Machine, An Innovative Computing Architecture;* Interner Bericht, 181/88, Fachbereich Informatik, Universität Kaiserslautern, 1988.

[50] A.G. Hirschbiel: *PISA-Maschine: Eine spezielle Hardware für pixelorientierte Bildverarbeitung*, Diplomarbeit, Universität Kaiserslautern, 1985.

[51] A.G. Hirschbiel, M. Weber: *The Kaiserslautern MoM Architecture and its Implementation*, internal report, Univ. Kaiserslautern 1989

[52] R. Hartenstein, U. Welters: Higher Level Simulation and CHDLs; in: *(eds.:) W. Fichtner, M. Morf: VLSI CAD Tools and Applications;* Kluwer Acad. Publ., Boston, Mass, 1987

[53] M. Weber: Transforming Systolic Algorithms into MoPL Programs; internal report; Univ. Kaiserslautern 1989

[54] T. Becker: *PISA - CACHE als sequentielles Schieberegister - Beschreibung eines strukturierten Entwurfes*, Projektarbeit, Univ. Kaiserslautern 1986.

[55] R. Müller: *Multshift - eine Vollkundenschaltung für die Map-oriented Machine - Layout und Simulation*, Projektarbeit, Univ. Kaiserslautern 1987.

[56] H. Nicklaus: *Multshift - eine Vollkundenschaltung für die Map-oriented Machine - Spezifikation und Problemanalyse;* Projektarbeit, Univ. Kaiserslautern 1987.

[57] B. Mank: *Eine PLD Cache Implementierung für die MoM*, Projektarbeit, Univ. Kaiserslautern 1989.

[58] C. Mead, L. Conway: *Introduction to VLSI Systems.* Addison Wesley, 1980.

[59] N. N.: *CMOS Design Rules für das E.I.S.-Projekt;* Fraunhofer-Inst. f. Mikroel. Systeme; Duisburg, F.R.G. 1988

[60] M. H. Brown: Perspectives of Algorithm Animation; *(eds.) E. Soloway, D. Frye, S. B. Sheppard: CHI'88 Conference Proceedings (Washington DC) - Human Factors in Computing Systems;* Association for Computing Machinery, New York, 1988

[61] A. Schaffer: *MCU - Move Control Unit for the MoM;* Diplomarbeit, Univ. Kaiserslautern 1989.

[62] A. Schaffer: *Single Step Control Unit,* Projektarbeit, Univ. Kaiserslautern, 1989.

[63] T. Mayer: *RANGECOUNTER - eine Vollkundenschaltung für die MoM (Map-oriented Machine),* Projektarbeit, Univ. Kaiserslautern 1987.

[64] H. S. Stone: Parallel Computing with the Perfect Shuffle; *IEEE TC-20 (1971),* p. 153 - 161

[65] R. Müller: Task Manager - A Hardware Task Concept for the MoM; Diplomarbeit, Univ. Kaiserslautern 1989.

[66] J. Westphal: MOM-Interface, ein Host Interface für die Map Oriented Machine, Diplomarbeit, Univ. Kaiserslautern 1986.

[67] S. Burkhardt: *Implementierung eines MoM-Treiberprogramms für ein 68000-System unter OS-9;* Projektarbeit, Univ. Kaiserslautern 1989.

[68] N.N.: eltec-68k-system user manual; eltec GmbH, Mainz, F.R.G., 1986.

[69] T. Blüthner: VME-Bus Interface for the MoM; Diplomarbeit, Univ. Kaiserslautern 1989.

[70] S. Reibnegger: *EDIF Interface for the MoM;* Diplomarbeit, Univ. Kaiserslautern 1989.Univ. Kaiserslautern 1989.

[71] A.G. Hirschbiel: *Interface zum Anschluß einer Design-Rule-Checker-Maschine an die Siemens Rechenanlage 7.751/7.760,* Projektarbeit, Univ. Kaiserslautern 1983.

[72] T. Mayer: *POLU (Problem Oriented Logic Unit);* Diplomarbeit, Univ. Kaiserslautern 1989.

[73] C. Q. Zhu, P.-C. Yew: A Scheme to Enforce Data Dependence on Large Multiprocessor Systems; *IEEE Trans. on Software Engineering,* (June 1987) p. 726 - 739

[74] R. Hartenstein, K.Lemmert: SYS^3 - A CHDL-Based CAD System for the Synthesis of Systolic Architectures; *eds.: J. Darringer, F. Rammig: Hardware Description Languages and their Applications;* North Holland, Amsterd./New York 1989.

[75] K.Lemmert: *SYS^3 - A Systolic Synthesis System around KARL,* Ph. D. Thesis, Univ. Kaiserslautern 1989.

[76] N.N.: *LOG/iC User Manual;* ISDATA, Karlsruhe, 1988.

[77] J.Holzer: *Optimal Programming Code for the POLU (Problem-Oriented Logic Unit);* Dipl.-Arb., Univ. Kaiserslautern 1989.

[78] W. Müller: Implementierung des MoMpiler-Codegenerators für die Move Control Unit der MoM; Diplomarbeit, Univ. Kaiserslautern 1989.

[79] M. Weber: *Ein Patterngenerator zur automatischen Referenzmustererzeugung;* report, Univ. Kaiserslautern 1985.

[80] C. Münster: *Implementierung eines graphischen Editors zur Eingabe der POLU-Operationen der MoM;* Diplomarbeit, Univ. Kaiserslautern 1989.

[81] N. N.: PISA Reference Manual, Univ. Kaiserslautern 1984

[82] P. Dewes: *Implementierung eines syntaxgesteuerten MoPL-Editors für die MoM;* Diplomarb., Univ. Kaiserslautern 1989

[83] A.G. Hirschbiel, M. Weber: *Methodology of MoM application support*, internal report, Univ. Kaiserslautern, 198

[84] H.-M. Su, P.-C. Yew: On Data Synchronization for Multiprocessors; *Proc. 16th ISCA (Jerusalem 1989);* IEEE Computer Society Press, Washington, DC 1989

[85] 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

[86] D. Kuck, R. Kuhn, D. Padua, B. Leasure, M. Wolfe: Dependence Graphs and Compiler Optimizations; *ACM Conf. on Principles of Programming Languages (July 1981);*

[87] D. Kuck, A. Sansh, R. Cytron, A. Veidenbaum, C. Polychronopoulos, G. Lee, T. MacDaniel, B. Leasure, C. Beckman, J. Davies, C. Kruskal: The Effects of Program Restructuring, Algorithm Change, and Architecture Choice on Program Performance; *1984 Int'l Conf. on Parallel Processing;* (Aug 1984)

This paper may be split up into 3 shorter papers:

1) Xputer Principles: A non-von-Neumann Computational Approach

2) The MoM Xputer Architecture and its Implementation

3) MoM-DE: An Optimizing Xputer Development Environment