MOM - MAP-ORIENTED MACHINE

a partly custom-designed architecture compared to standard hardware

R.W. Hartenstein, A.G.Hirschbiel, M.Weber
University of Kaiserslautern, PB 3049, D-6750 Kaiserslautern, FRG

Abstract: The von Neumann principle has 2 bottlenecks: program accessing and data accessing. An innovative non-von-Neumann principle, having been introduced at Kaiserslautern, eliminates one of them. Its new processor, the Map-oriented Machine (MoM), is compared with the von Neumann concept. The MoM is the key resource to a completely new philosophy of data processing which we call "map-oriented processing". It does not use sequential programs, since it has no program sequencer. The way how it is programmed we call 'combinational programming'. For surprisingly many applications it provides acceleration factors of up to several orders of magnitude, compared to von-Neumann-type processing. Existing computer application support tools (assemblers, compilers, operating systems, etc.) cannot be used for the MoM since they produce sequential code. That's why a new programming theory and new application support tools are introduced such, that the MoM is based on a marriage between standard IC use and ASIC techniques.

1. Introduction

The standard hardware approach: von-Neumann [1] architectures are based on standard hardware. The "Harvard machine" [2] is divided into five parts: data memory, data sequencer, RALU, program sequencer and program store (fig. 1). Well-known are its two bottlenecks: both, data and program side are accessed sequentially. That's why its hardware is slow, but cheap (using standard hardware, if it is not a Cray). For many applications it is too slow. But this approach is highly flexible and the hardware is universal.

The specialized hardware approach: If the von-Neumann-based approach is too slow, specialized machines may be used for certain problem classes [3]. This approach is expensive and offers only very little flexibility. If even higher acceleration factors are needed (e.g. by millions as in [4]), a fully customized ASIC approach may be used, which is even more expensive, but has no flexibility at all.

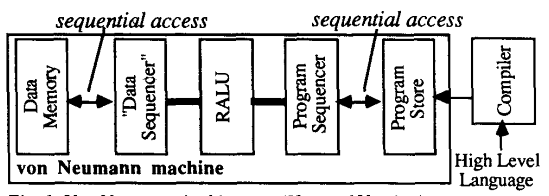

Fig. 1: Von Neumann Architecture (Harvard Version)

The gap in-between: Between these two approaches is a remarkably wide gap, which is filled by the ***Map-oriented Machine*** (MoM) [5, 6]. Its basic acceleration idea is to replace program sequencer and program store by combinational hardware (while data access remains sequential, compare fig. 3 and [5, 6]). Compared to a von Neumann machine use it offers higher speed, but still has high flexibility. It is substantially cheaper than fully customized hardware, but still offers high speed. The MoM is a general purpose accelerator, being a compromise, even a marriage, between ASIC solutions and standard hardware, combining advantages of both.

Many of the tools needed for MoM programming are adopted from the field of ASIC design (also see section 5). Also PLDs [7] and other electrically programmable hardware [8] are part of the ASIC scene. That's why the MoM concept is not just a combination of, but a marriage (figure 2) between, standard circuit use (the data side of the MoM) and ASIC techniques (the combinational program side). Therefore we call it a "partly custom-designed" machine concept.

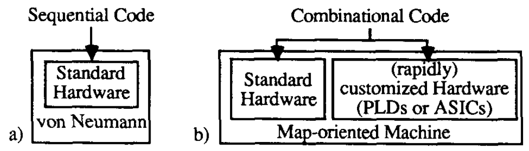

Fig. 2: The MoM - a technology alternative (b) to von Neumann (a)

2. Architecture Principles of the MoM

The Map-oriented Machine consists of four parts (figure 3), the map-oriented data memory, the data cache, the move control unit and the programmable part, the problem oriented logic units. The MoM is connected via a high speed bus to a workstation serving as a host.

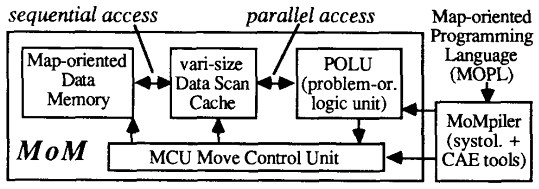
sequential access
parallel access
Map-oriented Programming Language (MOPL)
Map-oriented Data Memory
vari-size Data Scan Cache
POLU (problem-or. logic unit)
MoMpiler (systol. + CAE tools)
MoM
MCU Move Control Unit

Fig. 3: Illustration of MOM Architecture Principles

The programmable part of the MOM consists of one or more ***problem-oriented logic units*** (POLUs, fig. 4), which do not hold sequential programs, but are combinational hardware generated automatically, and, being faster than sequential programs. They replace program store, sequencer and RALU of a von Neumann machine. A POLU does not have a hard-wired instruction set. It only holds user-defined functions, programmed into PLDs, RAMs or electrically programmable gate arrays [8]. Due to the capability to store many terms, the POLU operations can be very powerful. For example, there can be hundreds of "if-clauses" coded be executed in parallel.

Let's explain this parallelism with a pattern matching application example (e.g. in image preprocessing): the POLU contains a set of reference patterns, which are matched in the analyzer with the cache's contents in parallel within a single machine cycle. As an interpretation of this pattern match a different (result) pattern can be written back into the cache to cause the change of data in the memory.

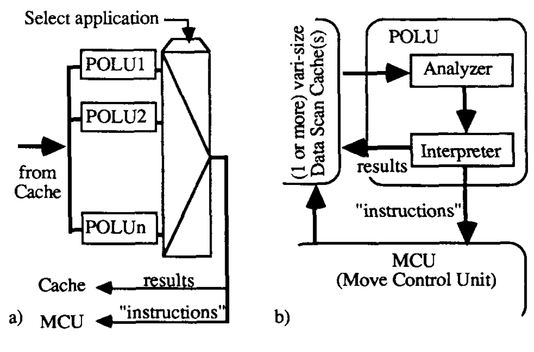

Fig. 4: POLU examples: a) multi tasking b) parallel pattern matching

CH2704-5/89/0000/5007/$01.00 © 1989 IEEE

Even numeric functions can be implemented as a set of quasi-reference patterns and result patterns derived from the corresponding function table. As a second result feedback to the move control unit may be generated. This feedback is used for data-dependent cache movements (as e.g. in curve following).

In contrast to the 1-dimensional von Neumann memory space, the ***map-oriented data memory*** is primarily 2-dimensional. At run-time the dimensionality of the memory can be switched to 1-D, 3-D, 4-D or more dimensions [9]. The MoM data word format is highly flexible, since it is not bound to instruction formats. Its word size may be easily increased by adding more memory planes. This is one major difference to conventional computers, which have a fixed word length. 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). Modern user interfaces encourage a graphical presentation of such memory maps.

The vari-size ***data scan cache*** is a window to scan the memory space [5, 6]. It holds and updates copies of a few neighbour words from mem- ory for read/modify/write access. All words in the cache can be ac- cessed in parallel by the POLU. The cache size is adaptable to different applications and may be reconfigured at run-time. So not only two or three words like in a von Neumann ALU are accessible at a time. By a 4-by-4 cache format, for example, 16 operands are directly connected to the POLU.

The ***move control unit*** (MCU) hardware provides accessing sequences for a controlled cache 'movement' over the memory space. So this unit serves as the data sequencing part of the MoM. Two major cache movement strategies are available and may be combined:

- systematic 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 manager** [9] using a structured jump generator hardware. It generates addresses for single steps to one of the 8 nearest neighbours, for jumps in memory space, as well as step sequences for a 'video scan', shuffle sequence or other 'travel paths'. An incomplete summary of possible movements is shown in figure 5. A second part of the MCU, the **task manager**, controls the coordination of several move patterns [9], which may be linked together to a so-called "MoM-Application".

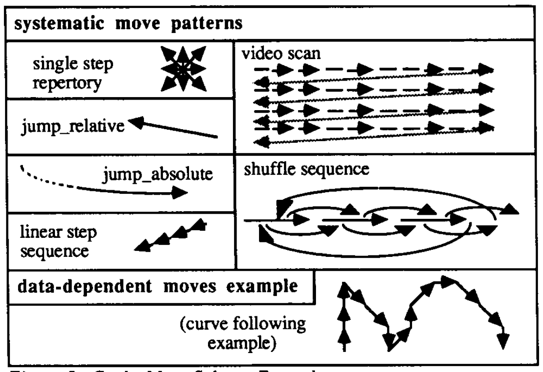

Figure 5: Cache Move Scheme Examples

3. A Few Application Examples

The MoM can execute almost any kind of data processing. Anyhow the MoM achieves very good performance in processing such problems, where the data can be efficiently mapped onto a two-dimensional memory space. Systolic arrays, for example, can be directly mapped onto the MoM memory space. Systolic data streams are converted into arrays stored at fixed memory location. Not the data, but a cache is moving such, that a POLU connected to it serves as an equivalent to a basic systolic processing element (here used in a time multiplex mode). Thus the systolic parallelism is sequentialized. But compared to a von-Neumann approach still a substantial acceleration is achieved, since the POLU itself is operating combinationally. Table 1 shows a few application area examples for MoM execution (see also [5, 6]). Table 2 shows a few acceleration factor examples.

VLSI layout processing	design rule check, circuit extraction, compaction [10]
Image processing	pattern recognition, pattern matching, shrink, expand, contour following, segmentation, set operations [11]
Minimum-cost path	Lee algorithm [12]
Arithmetic	matrix operations, convolution ...
Signal processing	Fourier transformation, filtering [11]
Ape Systolic Systems	many applications [13, 14, 15]

Table 1: Application Area Examples

The speed benefit of the MoM varies from application to application. This is illustrated by a few physical demos having been set up at Kaiserslautern (table 2). Dramatic improvement has been achieved in design rule check, routing and image preprocessing applications. E.g. the check of a one million square lambda NMOS design with grid-based "Mead-and-Conway" design rules [16] takes 1 second, compared to minutes or hours using mini computers of the von Neumann type. The acceleration has several orders of magnitude.

Operation	MoM*	VAX 11/750		68000	
(512x512 sized memory)	msec	sec	accelerat'n factor	sec	accelerat'n factor
Grey-Image operations Binary-Image operations	60	7.8	>130	14.7	>240
Erosion, Dilation, Skeleton, Edge Detection	210	38	>180	64	>300
Design Rule Check (λ-based)	260	300	>1000	600**	>2000
Minimum-cost Path	150	20	>130	40**	>160
10x10 Matrix multiplication aping systolic array using 2 caches	16 1	0.04	(>2) 40	0.15	(>9) 150

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

Table 2: Some acceleration factor examples

4. Programmable MoM vs. Partly Customized MoM

A second dimension of flexibility - superior to the flexibility of the von Neumann principle - is the flexibility in the choice of techniques for the POLU implementation. Using PLDs or programmable gate arrays we get the *"programmable MoM"* with substantial acceleration factors (figure 6a-b), where customizing requires only seconds (fig. 6c). Taking normal gate arrays or even full-custom circuits may provide an even much faster solution, but needing a turn-around time of weeks or months. We call this the *"partly customized MoM"* solution ("partly", since data memory, cache, and MCU still use standard circuits). The tradeoffs are summarized in figure 6.

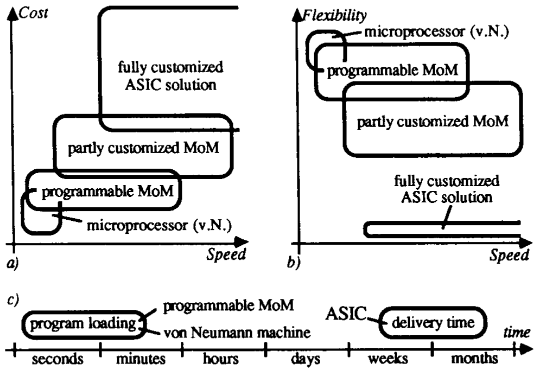

Figure 6: The MoM - filling a gap in algorithm implementation space

5. One Principle - Many Architectures

We have explained the innovative MoM principle (also see figure 2) by a particular example of a MoM architecture (e.g. figure 3). But there is room to develop many other MoM architectures. This is similar to the von-Neumann domain where also many architectures have been proposed and developed through the decades.

Considering the speed benefits and the range of applications to be covered by a specific MoM architecture, we could choose between different repertories of cache movement patterns (different "move instruction" sets), multiple caches (to emulate multiple data streams) or large caches. Also mixes between programmable MoM and partly customized MoM are feasible. 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 an ALU is added to the POLU (standard circuit or taken from a cell library) even a hybrid architecture may be created, which adds von-Neumann features to the MoM.

6. Theory of Operation and Application Support

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

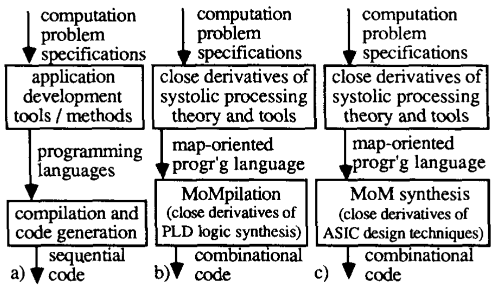
computation problem specifications
application development tools / methods
programming languages
compilation and code generation
sequential code
a)
computation problem specifications
close derivatives of systolic processing theory and tools
map-oriented progr'g language
MoMpilation (close derivatives of PLD logic synthesis)
combinational code
b)
computation problem specifications
close derivatives of systolic processing theory and tools
map-oriented progr'g language
MoM synthesis (close derivatives of ASIC design techniques)
combinational code
c)

Fig. 7: Application support: programmable MoM (b) and taylored MoM (c) versus von Neumann (a)

For a surprisingly wide variety of application areas [13] 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 the MoM, these results have mainly to be remapped into a more refined time step scale. This mapping is very simple, so that the derivation of the theory of MoM 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 [17], which also will serve as part of our MoM development environment (MoM-DE), currently being implemented (compare fig. 8).

For application development a special high level language (MOPL [9]) is available. But conventional compilers cannot be used, since they generate sequential code. That's why the MoM-DE includes the 'MoMpiler', which provides CAE tools like PLD 'programming' tools needed to code desired operations as POLU personalizations.

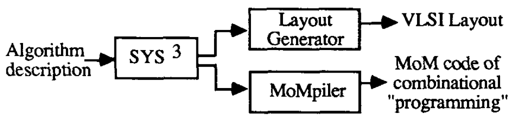

Fig. 8: SYS^3 as part of MoM development environment (MoM-DE)

Cache movements may be specified in a procedural way to be translated by MoMpiler into MoM-executable task register sets (more details are found in [9]). 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.

7. Conclusions

The MoM has been developed at Kaiserslautern, where a demo set-up has been personalized with design rule check, Lee routing, image preprocessing and other applications. For the Kaiserslautern MoM essential standard hardware parts in nMOS technology have been designed, fabricated and tested. Many other MoM architectures are feasible. The speed benefit in using MoM varies from application to application. A dramatic acceleration factor has been demonstrated experimentally for several application examples.

The Map-oriented Machine principles have been compared to the von Neumann concept. The MoM combines von Neumann's flexibility with the speed advantages of specialized hardware solutions. It is slower, but cheaper and more universal than fully customized hardware. However, for many applications it is substantially faster than a von Neumann machine, but only slightly less universal. We have shown, that new theories and methodologies needed for application support can be easily derived from the rapidly growing rich theory of systolic processing. An example application development environment (MOM-DE) is currently being implemented.

8. Literature

[1] 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.
[2] H. Aiken, G. Hopper, "The Automatic Sequence Controlled Calculator," Elec. Eng. 65, Aug/Sept, Oct, Nov 1946.
[3] T. Blank, "A Survey of Hardware Accelerators used in Computer-Aided Design," IEEE Design&Test, August 1984.
[4] K. Bastian et al., "VLSI-Algorithmen: Innovative Schaltungstechnik statt Software - Shuffle Sort," VDI-Berichte 550, 1985.
[5] R. Hartenstein, A. Hirschbiel, M. Weber, "A Flexible Architecture for Image Processing," Microprocessing and Microprogramming 21, 1987.
[6] R. Hartenstein, A. Hirschbiel, M. Weber, "Map-oriented Machine," in: Ambler, Agrawal, Moore: Hardware Accelerators for Electrical CAD, Adam Hilger, 1988.
[7] S. Muroga, VLSI System Design. Wiley, New York, 1982.
[8] R. Freeman, "User-programmable Gate Arrays," IEEE Spectrum, pp. 32-35, December 1988.
[9] A. Hirschbiel, M. Weber, The Kaiserslautern MoM Architecture and its Implementation, report, Kaiserslautern, 1989.
[10] W. Nebel, CAD-Entwurfskontrolle in der Mikroelektronik. Teubner, 1985.
[11] A. Rosenfeld, A. Kak, Digital Picture Processing. Academic Press, 1976.
[12] C. Lee, "An Algorithm For Path Connections And Its Applications," IEEE Trans. EC-10, September, 1961.
[13] 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.
[14] H.T. Kung, "Let's Design Algorithms for VLSI," Proceedings of the Caltech Conference on VLSI, 1979.
[15] H.T. Kung, "Why Systolic Architectures?," Computer 15/1, January 1982.
[16] C. Mead, L. Conway, Introduction to VLSI Systems. Addison Wesley, 1980.
[17] R.Hartenstein, K.Lemmert, "SYS3 - A CHDL-Based CAD System for the Synthesis of Systolic Architectures," IFIP CHDL '89, North Holland, 1989.