USING XPUTERS AS INEXPENSIVE UNIVERSAL ACCELERATORS IN DIGITAL SIGNAL PROCESSING

A. AST, R. HARTENSTEIN, A. HIRSCHBIEL, M. RIEDM†LLER, K. SCHMIDT, M. WEBER
Universitaet Kaiserslautern, Informatik, Bau 12/4, Postfach 3049, D - 675 Kaiserslautern, F.R.G.

International Neural Network Conference, INNC 90, Paris, France, Juli 1990

Abstract. The paper introduces to xputer use to accelerate digital signal processing algorithms and other parallel algorithms within a wide variety application areas. (Xputers are a novel class of high performance processors.) The programming paradigm, which stems from the deterministically data-driven xputer machine paradigm, is illustrated by introducing a few xputer application examples in digital signal processing. The paper first briefly introduces the novel high performance machine principles. Finally the paper discusses, how the novel method may be also used for fast and cheap design of ASICs and highly flexible accelerators, and gives some throughput and hardware cost figures having been obtained experimentally.

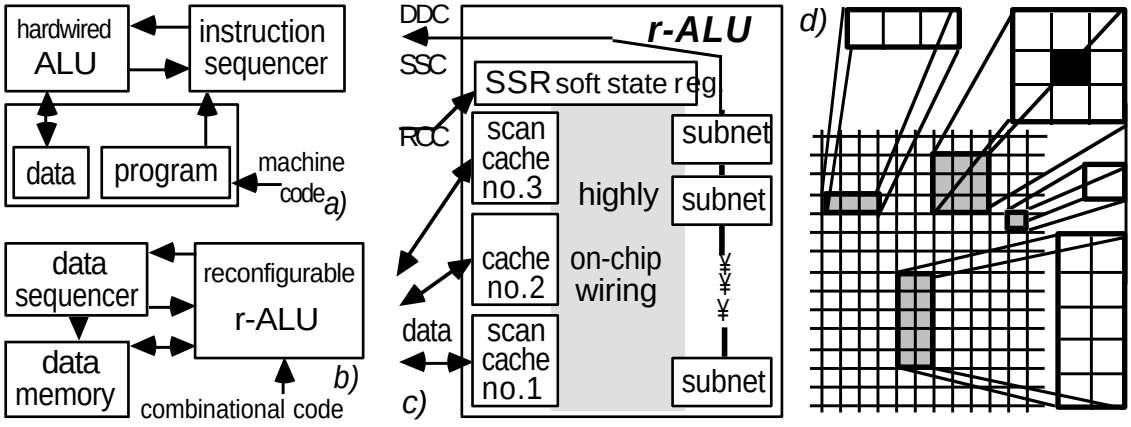
INTRODUCTION

Extremely high throughput (up to several kiloMIPS) is needed at very low hardware cost for quite a number of real-time applications, such as computer vision, computer graphics, signal processing and others, especially, when embedded in mass products. Such requirements cannot be met even by most advanced von-Neumann-type processors, nor by parallel or concurrent computer systems, because their communication mechanisms are not sufficiently powerful and too inflexible, so that most of the parallelism within parallel algorithms cannot be mapped onto the hardware [Mch90].

Even in monoprocessor applications the von Neumann principles includes many sources of overhead such that many extra memory cycles are needed. Examples of such overhead are: control flow overhead (control flow is much more extensive than would be really needed for decisions), address computation overhead for subscripted variables (which may consume up to 90% of the total throughput [HW90]), procedure call and parameter passing overhead and many others. Another well known von Neumann bottleneck is the fact, that the ALU can execute only a single simple operation at a time. An exhaustive overview on overhead (also indirectly) caused by von Neumann principles has never been published.

More sources of overhead are added to this by bundling von Neumann machines into parallel computer systems. The overhead (for communication, synchronization, multiplexing and scheduling) involved with distributing a task onto cooperating processors can be very significant [e.g.Py88]. Most existing parallel computer systems have not taken into account adequately this overhead issue, neither in the compiler, nor in the hardware [Py88], so that most of the parallelism within parallel algorithms cannot be mapped onto such von Neumann type hardware.

Higher parallelism is obtained from Application-specific Array Processors (ASAPs) which, however, cause external communication overhead (scrambling and unscrambling of data streams), cause high design cost (also very expensive, if programmable), and, support only algorithms with locally regular data dependencies (only systolic or systolizable algorithms). This paper first briefly summarizes xputer machine principles, which support a drastically more efficient hardware utilization. The paper then shows through example algorithms from digital signal processing the use of the xputer-based *data sequencing paradigm* (based on a *data map* - a data floor plan) for very high performance implementations of (also non-systolizable) parallel algorithms.



e)	(von Neumann) computers	xputers	data flow machines
the role of: data flow	derived (by pointers) from control flow	**primary activator** scheduled at compile time	**primary activator** determined at run time
the role of: control flow	**primary activator**	derived - only when needed: sparse control - residual	(not existing explicitely) control

Fig. 1

THE XPUTER MACHINE PARADIGM

This section summarizes the underlying novel hardware organization principles [MoM87]. In contrast to von Neumann (fig. 1 a) - xputer hardware (fig. 1 b) supports parallel algorithms drastically more efficiently due to avoiding most roots of overhead by a universal simple *data sequencer* hardware, by *(sparse) residual control,* by a very high degree of very fine granularity *intra-ALU parallelism* (by a PLD-based reconfigurable *r-ALU* permitting very fast extremely powerful *compound operators:* see *subnet* in fig. 1 c), and, by highly optimized deterministic *(data window) scan cache* use. Following paragraphs go more into detail. The basic partitioning scheme of computers (fig. 1 b) consists of: data memory, data sequencer, and r-ALU. Its underlying *xputer* machine paradigm [Mch90] is based on *data sequencing* - in contrast to von Neumann's *control flow sequencing* (fig. 1 a).

Fig. 1 e highlights the main differences between paradigms: xputers are (deterministically) data-driven, such that - in contrast to data flow machines - the order of data access may be scheduled in detail at run time. That's why xputers provide a very cheap, but much more flexible and much more compiler-friendly hardware, so that (also non-systolic !) parallel algorithms can be mapped onto it much more efficiently. Very high acceleration factors (fig. 6) have been achieved experimentally on the MoM hardware [CE89, LV90], hosted by a VMEbus-based *eltec* image processing computer system [eltec], with machine code generated by an *xpiler* (an innovative kind of silicon compiler [CE89]) from MoPL [We89] programming language sources, optimized by an optimizer (a version of a systolic array synthesizer [Lem89]). A second generation MoM hardware and development environment is currently being implemented at Kaiserslautern.

Scan caches. As an interface to the MoM's 2-D-organized data memory up to three 2-D *scan caches* are used (see fig. 1 c), being data windows onto memory space. Scan caches are resizable at run time (fig. 1 d) and have multi-directional on-cache shift paths and access mode tags per cache word (read-only, ignore, etc.) to minimize memory access time. Scan caches are connected with the r-ALU by massively parallel lowest level intra-chip interconnect (fig. 1 c), where a read / modify / write cycle takes only a few nanoseconds. In contrast to conventional caches permitting probabilistic strategies only, the xputer scan cache supports xpiler-defined *data-map-based* cache strategies which are fully deterministic, featuring Å100% hit rate already on very small caches. Also Å100% hit rates in memory interleaving are supported by this data map strategy (s.next section).

The data sequencer moves cache windows synchronously through primary data memory space. It is the primary activator of xputer operations: a kind of *data counter* replaces the program counter known from von Neumann machines. That's why xputers are *data-driven* (unlike computers, being *control-flow-driven*). The data sequencer provides a repertory of hardwired generic *scan patterns* (e. g. shuffle, butterfly, linear scan a others, for examples see fig. 2 - for universality also non-generic *list-driven scan patterns* are supported) to move a cache through a *data map* (a preorganized segment of data memory space).

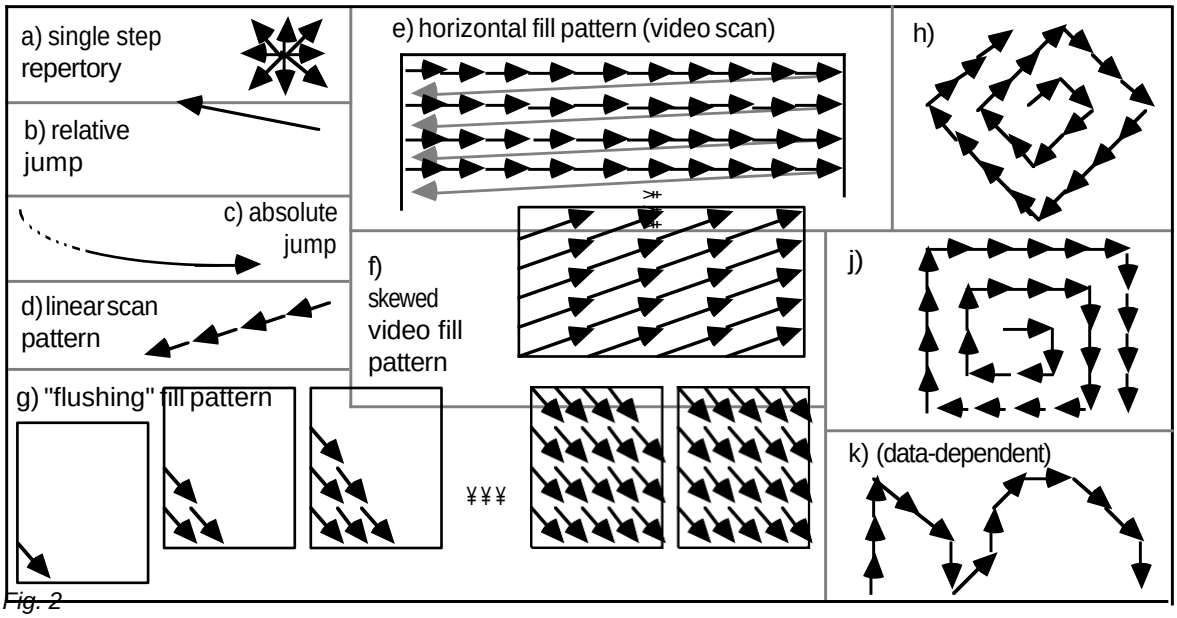


Fig. 2

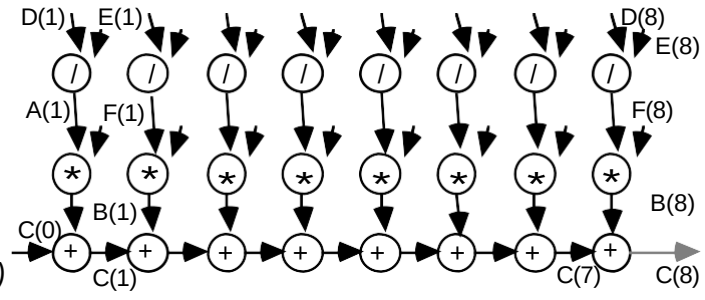
Residual Control: *tagged control words* (TWs) are inserted into data only where needed (compare fig. 4 c), what we call *sparse control* . TWs become effective after having been loaded into, or, decoded into soft state registers (SSRs), respectively (see fig. 1 c). We use the term of *residual control,* since SSR contents is changed only when needed (e. g. in fig. 4 d: only 3% of the code). Due to the softness of SSRs and decoders a much wider variety of optimization strategies is accepted by xputer hardware than known from any other kind of computational machines.

XPUTER IMPLEMENTATION OF SIGNAL PROCESSING ALGORITHMS

Also in signal processing there are two types of algorithms: *local communication algorithms* (systolizable algorithms, like convolution, filtering, or matrix and transform algorithms) and *global communication algorithms* (a typical example is FFT). The first algorithm example (fig. 3 a) is systolizable. Fig. 3 b shows its data dependence graph (DG). For xputer implementation a single iteration from the DG (see fig. 4 a) may be directly converted into a r-ALU subnet or *compound operator* (left side in fig. 4 b). Fig. 4 c shows the *data map*, which consists of cache size adjustments (here: single 3 by 2 cache, also see fig. 4 b), and a linear *scan pattern* (arrows in fig. 4 c). At each individual scan step the r-ALU subnet currently activated applies a read / modify / write cycle onto all caches currently active. The TW finally found (fig. 4 c) indicates the end of the scan pattern and through r-ALU specifies the link to next data sequence. Fig. 5 a shows the feasibility of high hit rate 4-phase memory cycle interleaving, controlled by the least significant bit of both, x and y address.

```
C(0) := 0;
for i from 1 to 8 do
A(i) := D(i) / E(i);
B(i) := A(i) * F(i);
C(i) := B(i) + C(i-1)
endfor
```

a)



b)

Fig. 3.

FFT Example. In contrast to parallel computer systems and ASAPs - xputer hardware directly also accepts data maps derived from global communication algorithms (non-systolizable algorithms: exhibiting ***global*** reg-

ular dependencies). Fig 5 b shows the r-ALU subnet and its 3 cache adjustment, derived (fig. 5 c) from a constant geometry FFT (fig. 5 e: DG of a 16 point example). Fig. 5 d shows, how the memory map may be directly derived from the DG. Boxes, overlaid in fig. 5 d indicate initial and final cache locations from the 2-level nested 3 cache scan sequence, where 3 caches move simultaneously (synchronously); inner loop: 8 steps down (cache1: SW (step width) = 2, cache2, cache3: SW=1); outer loop: 4 steps right (SW=2).

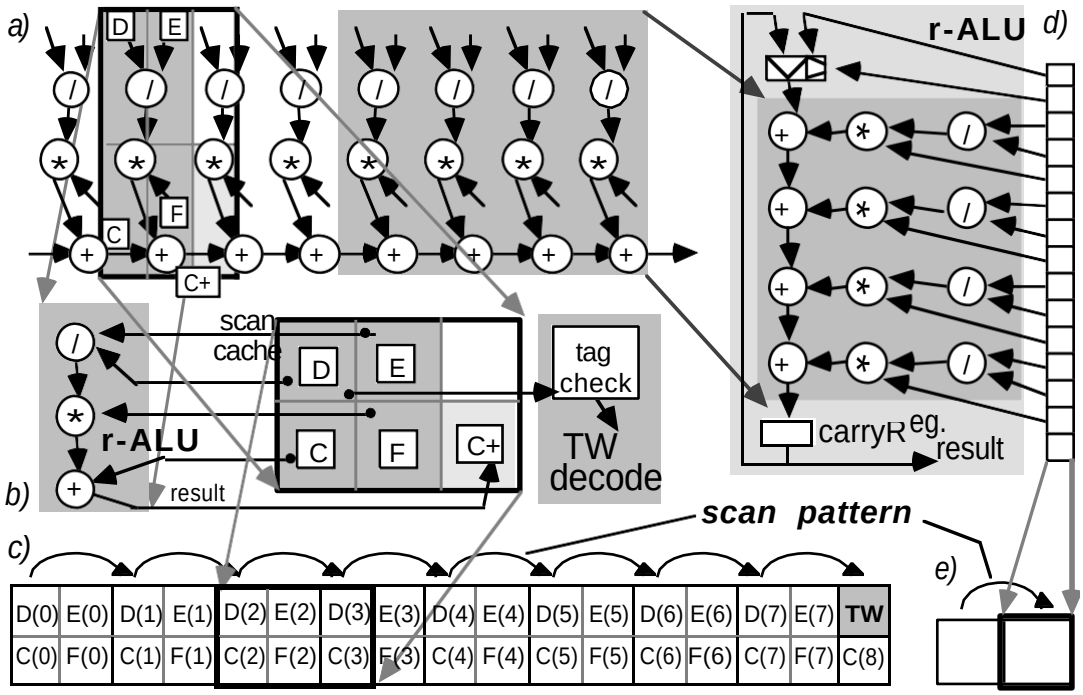


Fig. 4.

VLDW (Very Long Data Word) Architectures. Since there is no "hard" instruction format the hardwired part of xputers does not impose any constraints onto the choice of memory word formats. Thus extensible xputer architectures may be easily developed which can be upgraded by just inserting more PLD integrated circuits and more memory boards (more bit planes) to extend data word length [em90], and also to enhance the capacity of parallelism of the r-ALU.

Such a high flexibility of xputer hardware allows multiple choice of different cost/performance ratios. Figures 4 d/e show a high performance version of the example from fig. 4 b/c: this time 4 iterations are used to derive a more powerful r-ALU subnet. Fig. 4 d (right side) shows the VLDW format including 15 subwords accessed by a single memory cycle. The data map (fig. 4 e) has only 2 words accessed via single 1 by 1 scan cache. Only 2 memory read cycles are needed (compared to 32 read cycles and 8 write cycles, also see fig. 4 f), so that an acceleration factor of about 20 has been achieved.

Several encouragingly good performance results (fig. 6) have been obtained experimentally on the MoM xputer architecture [CE89, LV90]. It has been shown that in some important applications a single processor xputers may be competitive to ASIC solutions and may even outperform large parallel computer systems. Second column from the right in fig. 6 shows some hardware expense figures obtained experimentally on the MoM with code generated by the MoMpiler (xpiler of the MoM: in 1987). Rightmost column shows figures based on contemporary PLDs: even single PLD chip solutions (fig. 7 d) are feasible for most applications.

XPUTERS AS A NOVEL ASIC DESIGN APPROACH

Gate arrays are available commercially, being fully compatible to particular PLDs so, that after testing and debugging an xputer application the same machine code (generated by the xpiler) may be used to fabricate a gate array version (fig. 7 a / b). This is a fast way to obtain a cheap ASIC version from an xputer application, where, instead of simulation, direct design *execution* is used being by orders of magnitude more efficient. Redesign for latest technology PLD use would be trivial: just recompile onto new version r-ALU.

We strongly believe, that such a novel design environment could substantially reduce design time and design cost. We strongly believe, that surprisingly many xputer implementations obtained this way would have sufficient performance to compete with expensive ASICs having been designed the 'traditional way'. Such a novel ASIC design environment would even further decrease the chip count, if xputer-specific parts would be available within the cell library, so that the ALU together with all other xputer parts could be merged onto a single chip (fig. 7 c). For many embedded applications even the data memory could be merged onto the same chip, so that the total chip count could be reduced to 1 (fig. 7 d).

XPUTERS - COMPARED TO COMPUTERS

Fig. 9 summarizes part of the differences between xputers an computers. Instead of hardwired simple instructions the xputers allows very powerful compound functors to be defined and optimized by the compiler (optimized path width, optimized cost/performance trade-off etc.) Because not having sequential machine code the xputer is data-driven by a hardwired data sequencer avoiding address computation overhead. Xputers use parallelism at lower levels than that in computers: at gate level and/or at data path level. This lowest level parallelism is extremely efficient: for communication it mostly uses dedicated on-chip wires (within the r-ALU), which is much more efficient than (slow) intra-chip buses (also causing multiplexing overhead).

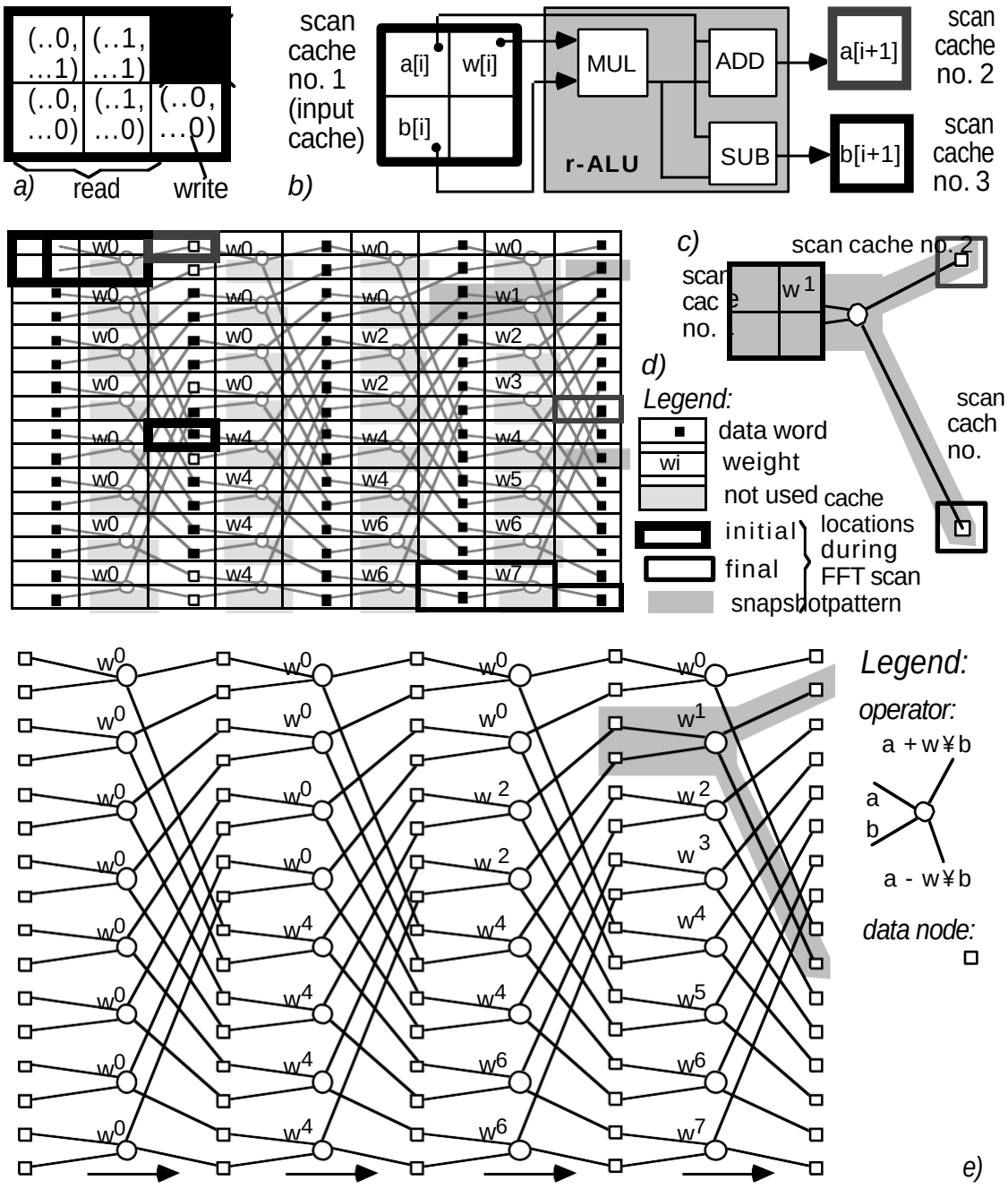


Fig. 5

Since xputers do not have a hardwired instruction format, data memory word size is much more flexible than in computers. Even highly flexible alterable word size memory architectures are feasible for xputers. Due to xputers' map-oriented data sequencing paradigm the compiler may optimize data maps for very high hit rates in interleaved access memory organizations. Also the concept of xputers' scan cache, being somewhat similar to a small cache, may be subject of very high hit rate optimization. In total there are many differences between xputer and computer such, that here is no room to mention all of them.

application area	#	algorithm example	acceleration factor	r-ALU size *	r-ALU size **
VLSI design automation	1	CMOS design rule check by pattern matching	>2000	45	4
	2	electrical rules check support	>300	5	0.5
	3	Lee routing	>160	5	0.5
digital signal and image processing	4 a	vector-matrix multiplication ¥ single scan cache	>9	7	0.5
	4 b	• double scan cache	150	7	0.5
	5	two-dimensional filtering (3 by 3 window)	>300	<0.5	0.02

*) number of PLD chip(s) needed with Altera MAX EPM 5128 **) w. Plessey ERA 60400 (40,000 gates)

Fig. 6.

Many differences between performance-relevant features are in favor of the xputer. This explains the fact, that, although being based on a (sequential) machine paradigm, xputers very often may outperform parallel computers, super computers, and even ASICs. Currently we cannot completely explain the high acceleration factors of xputers over computers, having been obtained experimentally. A more exhaustive analysis of overhead phenomena in computers would be needed, than that available from current literature.

CONCLUSIONS

An alternative high level synthesis approach has been introduced being based on a data sequencing paradigm. In contrast to traditional control-driven paradigms it achieves dramatically much better performance for a large class of parallel algorithms, also including non-systolizable algorithms, being very important also for digital signal processing. This novel machine paradigm for parallel algorithms has been introduced and it has been shown that it is comprehensible and achieves a drastically improved hardware utilization. The new paradigm is a highly promising alternative to currently dominating universal programmable computational devices, at least to von-Neumann-based ones. Here fig. 8 summarizes most of the reasons of xputers' superiority. The approach having been presented also permits, that in an innovative ASIC design environment logic and functional simulation may be replaced by execution, which substantially accelerates the design process. With xputers the race to catch most advanced technology is no more a major problem (in contrast to computers needing expensive redesigns): the memory uses catalog circuits, data sequencer speed is not a bottle neck, and newest r-ALU technology is available from stock from a (1990) billion US-$ world market of 23 vendors.

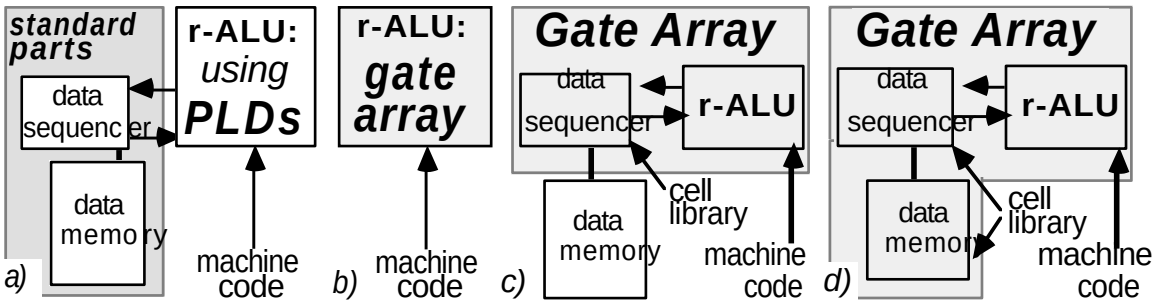


Fig. 7.

With the MoM-I a first generation xputer architecture example has been built at Kaiserslautern including an xpiler running on a VAXstation, along with tools supporting pattern matching applications by automatic generation of reference patterns [LV90]. Acceleration factors of >100 seem to be typical and occasionally up to >2000 have been obtained experimentally. A second generation MoM-II hardware along with a second generation application development environment is currently being implemented at Kaiserslautern.

With the MoM an extremely flexible xputer architecture has been implemented which avoids most of the roots of overhead by achieving, that the massively parallel r-ALU directly reaches the right data at the right time at the right place. Xputers are extremely powerful universal accelerators, which also may run in a stand-alone mode. Due to simplicity and flexibility its hardware principles support a very wide variety of choices in cost / performance tradeoffs and performance enhancement strategies. Cheap mass production single-chip ASICs may be easily derived directly from xputer machine code without extra design effort.

XPUTERS - AN EMERGING DISCIPLINE IN R&D

The highly promising results having been reported above have been obtained mainly by brute force approach-

es. We expect even much better results from more advanced optimization strategies and much better xputer architectures [Mch90]. Although for xputers the creation of a completely new theory is not needed, since most of the fundamentals have been adopted from mature other areas (like ASAP compilation, interconnect structures, parallel algorithms, fast signal processing algorithms, VLSI design, high level synthesis, logic synthesis and others), a lot of research is needed. Since xputer principles open up a wide variety of architectures exhibiting a drastically increased acceptance for optimization and parallelization strategies, a very exciting new research area has been opened up - at a time where progress in traditional parallel computing principles and practices seem to have piled up in a dead road. The new area is not yet crowded, in an early phase of immaturity, where the rate of innovation is very high.

	(von Neumann) computers	**xputers**
instructions	hardwired, simple, fixed repertory	tailored: configured at compile time: also very powerful compound functors
machine code	sequential (-> program store)	combinational (-> configure PLDs)
sequencer	instruction sequencer	data sequencer
parallelism at	process level or (VLIW) instruction level	gate level and/or data path level
compilation techniques	traditional	derived from VLSI design techniques (data-driven high level synthesis, ASAP compilation, logic synthesis, fast signal processing algorithms,...)
data format	dependant of instruction format	independent - highly flexible adaptable VLDW (very long data word) is feasible
interleaved data memory	low hit rate	very high hit rate
'cache' use	probabilistic - low hit rate	deterministic - very high hit rate
addressing overhead	high - subscripted var.: very high	low parallel algorithms: very low
other overhead	high	low

Fig. 9

ACKNOWLEDGEMENTS

Early versions of MoM concepts result from the multi university E.I.S. project, having been jointly funded by the German Federal Ministry of Research and Technology and the Siemens-AG, Munich, under coordination by the GMD, Schlo§ Birlinghoven. We also acknowledge the various kinds of personal support received from Elfriede Abel, Herwig Heckl, Gustl Kaesser, and, Klaus Woelcken (now: Commission of the European Communities) on leave from GMD. We appreciate valuable ideas from Klaus Singer at *eltec* GmbH, Mainz, and Karin Lemmert at BASF Ludwigshafen. We also appreciate contributions of our students: T. BlŸthner, S. Burkhardt, P. Dewes, J. Holzer, B. Mank, T. Mayer, R. MŸller, W. MŸller, C. MŸnster, H. Nicklaus, S. Reibnegger, H. Reinig, A. Schaffer, K. Schmidt, and J. Westphal.

2880

LITERATURE

[CE89] R. Hartenstein, A. Hirschbiel, M. Weber: *MOM - a partly custom-designed architecture compared to standard hardware;* Proc. IEEE Comp Euro '89, Hamburg, F.R.G.

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

[em90] R.Hartenstein..: *Electrically Alterable Word Length Memory & Applications;* subm.f. publication

[HW90] R.Hartenstein,M.Weber: *Some observations on von Neumann overhead;* rep., Kaiserslautern 1990

[Lem89] R.Hartenstein, K.Lemmert: SYS^3 - *CHDL-based System for Synthesis of Systolic Architectures;* Proc. IFIP CHDL'89, Washington, DC, June 1989, North Holland, Amsterdam/NY, 1989

[LV90] R. Harten-

stein, A. Hirschbiel, M. Weber: *Acceleration of Layout Verification and Routing Achieved by Xputer Use;* submitted for publication

[Mch90] R.Hartenstein,A.Hirschbiel,M.Weber:*Xputers: an open Family of non-von-Neumann Architectures;* Proc. ITG/GI Conf. Architecture of Computing Systems, Munich, 1990, VDE-Verlag, Berlin 1990

[MoM87] R. W. Hartenstein, A. G. Hirschbiel, M. Weber: *MOM - Map Oriented Machine;* in: Ambler et al.: (Prepr.Int'l Workshop on) Hardware Accelerators (Oxford, 1987), Adam Hilger, Bristol 1988.

[Py88] C. D. Polychronopoulos: *Parallel Programs and Compilers;* Kluwer Ak. Publishers, Boston 1988

[We89] M.Weber: *MoPL - Map-oriented Programming Language,* report, Univ. Kaiserslautern, 1989