A NOVEL COMPUTATIONAL PARADIGM: MUCH MORE EFFICIENT THAN VON NEUMANN PRINCIPLES

R.W. Hartenstein, H. Reinig, M. Riedmüller, K. Schmidt

Universitaet Kaiserslautern, Fachbereich Informatik, Bau 12

Postfach 3049, D-W-6750 Kaiserslautern, Germany

phone: (+49-631) 205-2606, Fax: (+49-631) 205-3200

Abstract

Computers (based on von Neumann principles) are extremely inefficient. That's why this paper introduces a novel computational paradigm based on new hardware machine principles. Such machines, called “xputers" avoid most of the bottleneckknown from (von Neumann) computers, so that a hardware efficiency is obtained which is higher by several orders of magnitude. By means of a few algorithm examples the new paradigm will be introduced as a new programming paradigm, which is data-procedural (which is more direct than the control-procedural von Neumann paradigm). Finally the paper gives a survey on the novel application development environments needed for xputers and their advantages over such tools for computers. Such application support for xputers includes two alternative source levels: high level programs, or very high level algorithm specifications.

The Machine Paradigm of Xputers

In programming and coding von Neumann machines their basic paradigm causes massive overhead (control flow overhead, addressing overhead and other kinds of overhead [1]}, which eats up most of the processor's throughput and its primary memory bandwidth. Due to the control-driven operation principles by multiplexer-based instruction sequencing the individual processor permits only very limited intra-ALU parallelism [1]. Parallel computer systems suffer from high hardware cost and from massive interprocessor communication overhead [2]. In addition to that most of the processors are idling (a very few applications are exceptions from that). Data flow machines suffer from massive token flow overhead needed for arbitration and from other kinds of overhead [2] and debugging is extremely difficult.

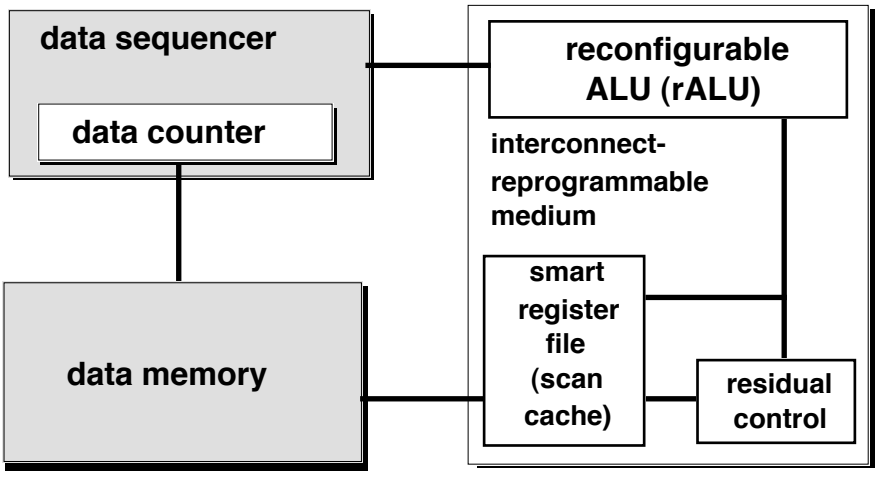

fig. 1: basic blocks of xputers

Xputers use a data sequencer (instead of an instruction sequencer) including a powerful address generator to avoid addressing overhead and control flow overhead whenever possible (see also fig. 1). Instead of an ALU driven by an instruction sequencer, xputers use a reconfigurable ALU (called rALU), such that powerful compound operators exhibiting intra-ALU parallelism may be defined at compile time. More hardware details have been published elsewhere [1-3]. A smart register file (called *scan cache*: being a size adjustable

window to memory space) including an intelligent data memory interface supports new compile time optimization strategies being more efficient than those which can be mapped on von Neumann hardware only. Algorithm execution examples within other sections of this paper will give a flavour of the high efficiency of this novel machine organization.
For computers the RAM (random-access memory) is the *central technology platform* which provides flexibility and universality. For xputers, however, the flexibility and universality is derived from using *interconnect-reprogrammable media* (compare fig. 1), also called *field-programmable media*, which are commercially available from a billion US-dollar niche of the integrated circuit market (1990: world-wide).

Programming and Compilation for Xputers

The Xputer machine paradigm is also visible from a programmer's point of view. Figures 2 and 3 show two simple algorithm examples, which will be used to illustrate programming for xputers, execution of programs on xputers, as well as the compilation for xputers. Fig. 2a shows the textual notation of a simple recurrent 8 step loop. The right side of fig. 2b shows its equivalent graphic notation: a signal flow graph (SFG) which reveals the regular data dependencies. For programming an xputer from such a specification the following code elements are needed (to be generated by a compiler): a *rALU subnet* (*compound operators*: saving memory cycles, since intermediate results are not stored) has to be derived (e.g. see fig. 2b: by just picking an iteration from the SFG); ad data map has to be derived (2-dim. data map example in fig. 2c: mapping the SFG onto a grid); the scan cache size has to be selected (e.g. 1-by-4 words: see fig. 2d); and an address sequence (called *scan pattern*: see fig. 2d) has to be selected.
Fig. 2d also illustrates xputer operation in executing the algorithm example from figures 2a and 2b. First the data sequencer makes the scan cache jump onto the leftmost column of the data map. Fig. 2d also illustrates the auto-apply mode and the auto-copy mode. This means that, whenever the scan cache is placed onto a particular memory location, it automatically (i.e. without needing a controller, thus avoiding control overhead) invokes operation of the rALU subnet currently selected, as well as a cache/memory communication cycle.Since a register has been added to the rALU subnet to save c[i], only four memory read cycles are needed (see left side of fig. 2d). Due to the scan pattern in our example this is repeated 8 times until finally the scan cache arrives at the rightmost position of the data map. The tagged control word (TCW) found there tells where to store the final result and provides the linkage code to select the next scan pattern etc. So only 1 out of 34 memory semi cycles has been used for control, what we call *sparse control* or *residual control* (also compare fig. 1).
Fig. 3 shows another simple algorithm example: a 2-dimensional filtering example where a simple video scan is used as a scan pattern (fig. 3b) to scan a 2-dim. data map (fig. 3b) by a 3-by-3 word scan cache (fig. 3a). Fig. 3c shows the very powerful compound operator, which needs relatively few hardware, since smaller data path width (8 bits) could be traded for more operators. This illustrates the high flexibility of xputers. By code analysis it has been found, that in the von Neumann version of the same algorithm (on Vax-11/750) 93% of computation time is used for address computa-

tion. The MoM [2] xputer version avoids this addressing overhead by its hardwired address generator [3] running concurrently (MoM stands for Map-oriented Machine), where the 2-dim. memory organization permits even better performance. A spreadsheet representation of the data map [4] supports a highly intuitive user interface for xputer applications.

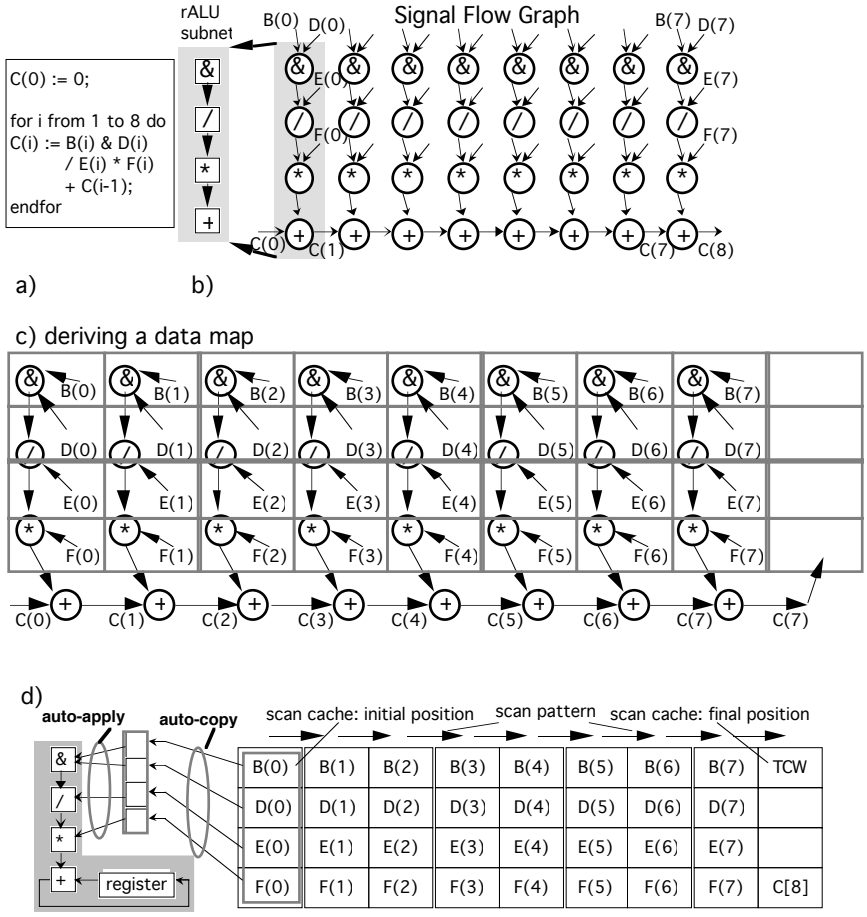

fig. 2: A simple algorithm example: a) textual, b) graphic specification, c) deriving a data map, d) xputer operation illustration

High Performance Applications

Fig. 4 shows a number of acceleration factors having been obtained experimentally from a number of algorithm implementations on the MoM-1 and MoM-2 xputers, compared to implementations on a Vax-11/750. These results show, that for a number of application areas (Digital Signal Processing, image processing, computer graphics, uniform equation systems, etc.) xputers are competitive to many ASIC solutions, to implementations on special processors (e.g. digital signal processors), or to implementations on supercomputers. A so called *retargetting* feature, being available commercially, permits a direct path from an implementation on a programmable xputer to a very cheap customized silicon solution: machine code can be directly submitted for gate array fabrication, such that no expensive ASIC design process is needed [2].

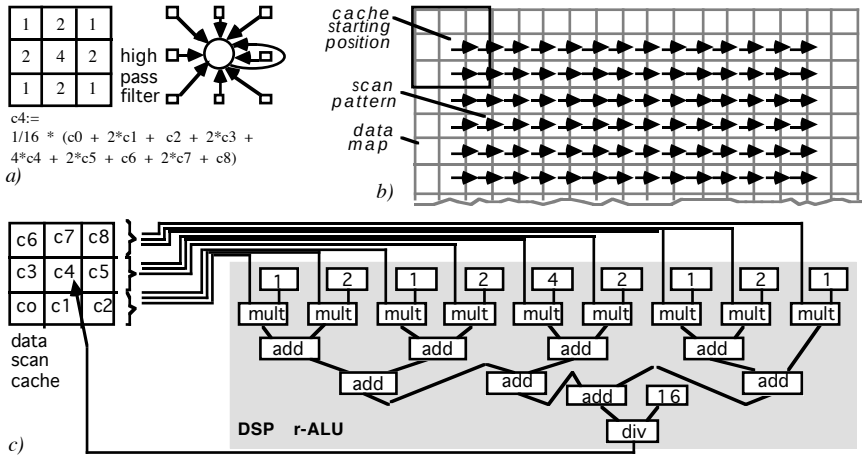

fig. 3: A 2-dimensional filtering example

Conclusions

A novel high performance machine paradigm of xputers has been introduced, which for a very large class of algorithms is by several orders of magnitude more efficient than the von Neumann paradigm. The data-procedural principles of operation, which are more intuitive and more direct than traditional principles, have been illustrated by 2 examples. two experimental xputer architecture examples (the MoM-1 and the MoM-2) have been implemented by breadboard techniques. Highly promising acceleration factors for a number of algorithms have been obtained experimentally by comparing MoM implementations versus implementations on Vax-11/750. The results show, that xputers are competitive to many ASIC solutions as well as to many algorithm implementations on supercomputers. A third xputer architecture (the MoM-3) stressing flexibility and universality is currently being implemented at Kaiserslautern.

application example	acceleration factor
CMOS design rule check	>2000
sup. electrical rules check	>300
Lee routing without obstacles	>160
with obstacles	>800
2-dimensional filtering (3 by 3)	>300
vector-matrix multiplication	150

fig. 4: acceleration factors obtained ecperimentally (xputer imple-

mentation versus optimized Vax-11/750 implementation

References

[1] R. Hartenstein et al: A Novel Paradigm of Parallel Computation and its Use to Implement Simple High Performance Hardware; Intl. Conference on Information Technology, Japan, Oct. 1990

[2] R. Hartenstein et al: Xputers: An Open Family of Non-von Neumann Architectures; internal report; University of Kaiserslautern, 1989

[3] A. Hirschbiel: A Novel Processor Architecture Based on Auto Data Sequencing and Low Level Parallelism; Ph. D. Thesis; University of Kaiserslautern, March 1991

[4] M. Weber: An Application Development Method for Xputers; Ph. D. Thesis; University of Kaiserslautern, Dec. 1990