Terminology

Address generator: extra hardware unit generating address sequences for *scan patterns* (see there) - improves processor performance by avoiding or reducing addressing overhead.

Addressing Overhead: the amount or percentage of computing capacity needed to compute addresses - can be avoided or reduced by xputer use (see: *address generator*)

ASAP: Application-Specific Array Processor (e.g. systolic array). **ASAP Synthesis:** methodology f. automatic synthesis of ASAPs by resource allocation (generating an array of PEs) and scheduling (generating a data stream scheme for a PE array)

ALU bottleneck: one of the von Neumann bottlenecks: the hardwired von Neumann ALU includes a large repertory of operators, but can execute only one at a time (also see *rALU*).

Code-compatible, Code compatibility: a gate array is said to be code compatible to a FPGA, if both accept the same personalization code. This technology feature permits (see:) *retargetting* of xputer machine code

Compound operator: complex data path within a rALU (e. g. implementation of an expression) - a source of high performance by low level parallelism (ultra micro parallelism): no storage of intermediate values.

Compound scan pattern: a compound of several *scan patterns* (see there) to be run in parallel (under a common clock) - needs multiple *scan caches* with multiple *address generator* (see there).

Computers: a class of processor architectures based on the *control-procedural* von Neumann machine paradigm.

Control overhead: %age of computing capacity consumed for control access (e.g.: dominance of control of the von Neumann paradigm) - avoided or reduced by *sparse control* (see there), in data-dependent scan patterns by hardwired short cut branching.

Data Flow Machines: non-von-Neumann processors being *data-driven by firing*, (execution order decided at run time).

Data Sequencer: xputers' driving unit including single or multiple address generators (used instead of an instruction sequencer: a data counter is used instead of a program counter - compare fig 1 versus fig 2).

DSP: digital signal processing

Field-programmable media: interconnect-programmable circuits, such as PALs, FPLAs, FPGAs (see there) etc. (billion-$ niche of the integrated circuit world market)

FPGA: field-programmable gate array

g-Algorithm: algorithm with dense *(locally and / or globally)* regular data dependencies. s-algorithms are a subclass.

Memory bottleneck: one of the von Neumann bottlenecks - limits communication bandwidth due to word-sequential transfer (conventional remedies: using cache buffer memory or interleaved primary memory) - xputers support program code optimization (by xpilers) leading to substantially reduced bandwidth requirements.

MoPL: Map-oriented Programming Language, (Pascal dialect) a data-procedural high-level programming language example

PE: processing element of an ASAP (see there)

rALU: .reconfigurable ALU (soft ALU) permitting intra-ALU parallelism, implemented by partially or completely using field-programmable media - personalized by xpiler-generated code.

Remapping: by projection methods systolic synthesis (see there) systematically derives resource allocation (array of PEs) and a schedule (data stream) from the data dependency graph of an algorithm. Remapping means the extension of such methods to derive xputer programs. The consequence is a generalization of projection methods from s-algorithms to g-algorithms.

Residual Control: (see: *sparse control*)

Retargetting: using xputer machine code for gate array fabrication (see *code compatibility*) - a technology feature permitting ASIC synthesis by xputer programming / debugging.

s-Algorithm: algorithm with dense *locally* regular data dependencies. s-algorithms are a subclass of g-algorithms.

Scan Cache or Window Scan Cache: xputers' smart register file with auto-apply feature (avoiding control overhead) and supporting optimizing xpiler to reduce memory traffic.

Scan Pattern or Scan Path: a (time) sequence of memory locations, in xputers created by an address generator (see there).

Sparse control: due to data-sequencing control flow is not the primary activator of xputer operation. Control is evoked only upon request occasionally (e.g. to link another scan pattern). That's why it is called *sparse control* or *residual control*. Only a very small F.S.M. is needed which is compiled into field-programmable media (physically: within the rALU - see fig.2).

Systolic Synthesis: (also see *ASAP synthesis*) automatic synthesis of systolic arrays by resource allocation and scheduling (generating a data stream scheme).

Xpilers: a class of 'compilers', but targetted to (non-von-Neumann) xputers - due to relative 'softness' of xputer hardware permitting a much wider variety of optimization strategies then compilers for (von Neumann) computers - new research area, a first implementation: MoMpiler for MoM-2 (Kaiserslautern) - Xpiler use for ASIC design: see *Retargetting*.

Xputers (no transputers!): a class of processor architectures based on a non-von-Neumann *data-procedural* machine paradigm - as universal as (von Neumann) computers - for g-algorithms by orders of magnitude more efficient then computers. In contrast to data flow machines: xputers are *data-driven by schedule* (but not by firing). Xpilers push as many decisions as possible back to compile time (maximized scheduling).

Xputer data map: specifies storage locations for xputer run time data - a kind of schedule: important subject of performance optimization strategies based on the xputer paradigm and its supporting hardware features such as data sequencer (address generator), smart register file, and, smart memory interface.

industrial competitiveness by drastically more performance on much less hardware

Xputers

A New Machine Paradigm

a New R&D Area

- **xputer: a universal accelerator add-on within a workstation: the host becomes a supercomputer?**
- **a short cut on the way from algorithm to customized silicon**
- **new directions in hardware architecture and parallel computing**
- **new directions in programming languages and compilation**

Are interested in receiving more information on xputers? Are you interested in joining an ACM / IEEE special interest group on Xputers? - Contact:

Leni Müller c/o Universität Kaiserslautern, Bau 12/4, Postfach 3049, D - (W) - 6750 Kaiserslautern, Germany, phone: (++49-631) 205-2606, fax: (++49-631) 205-3200 e-mail (uucp net or cs net): abakus@informatik.uni-KL.de

The Significance of Xputers

The term of ***Xputers*** stands for a new computational paradigm opening up a wide design space for many Xputer architectures. Xputers (fig. 2) are by several orders of magnitude more efficient than (von Neumann) **com**puters (fig. 1) for a large class of algorithms (s-algorithms) being commercially important (signal processing, image processing, consumer electronics, aerospace, and others): Xputers offer drastically more performance with less hardware. W.r. to performance most of such algorithm implementations are competitive to (traditional) ASIC solutions. Due to retargetting capability xputers also offer a bypass on the way from algorithm to silicon: no expensive hardware design process is needed.

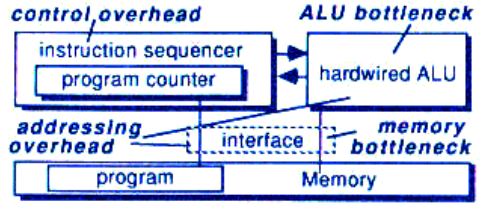

Fig. 1. computers and their von-Neumann bottlenecks

Resource Optimization by Xpilers

The programmable part of xputers is the ***rALU*** - a reconfigurable ALU permitting internal parallelism (fig. 2). rALUs don't have a hardwired instruction set. An xpiler compiles problem-oriented *compound operators* by combinational switching networks into the rALU (xputer standard parts like the data sequencer are added from a library). Utilizing this rALU softness only exactly those resources are generated, which are needed for a given problem: only the number and size (path width) needed. For very high throughput requirements, however, more resources will be created to achieve sufficiently high parallelism (using a larger rALU if needed).

Xpilers: a new kind of Silicon Compilers?

rALUs made up from FPGAs raise the question: are Xpilers a new kind of silicon compilers? The answer is: Xpilers are much more than just silicon compilers. Xpilers generate the *target hardware **and** the machine code* at the same time. Silicon compilers, however, only generate a target hardware, but not the machine code needed to run it.

Why Xputers are so efficient

The xputer paradigm combines several measures to improve performance (fig. 2) by reducing or avoiding von Neumann bottlenecks (fig. 1): introducing (intra-ALU) low level parallelism, avoiding or reducing addressing overhead, control overhead, synchronization overhead, resulting in substantially reducing code size. This, smart register files, and smart memory organization support rich optimization strategies (by xpiler) to substantially reduce memory traffic.

Xputers for Real-Time Applications

The data-procedural machine paradigm of xputers strongly supports optimization strategies by xpilers which push as many decisions and overhead as possible from run time to compile time. Such strategies met the goal, that always the right data are found at the right time in the best possible place. This is another factor contributing to the extreme run time efficiency of xputers which is useful to meet extraordinary real-time requirements.

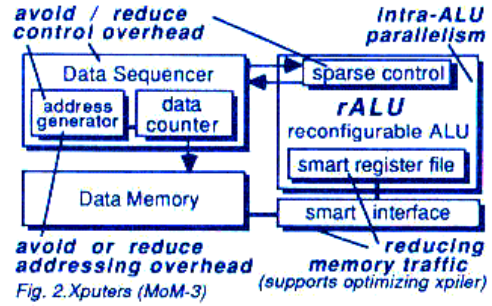

Fig. 2.Xputers (MoM-3)

The Technology Platform of Xputers

Von Neumann principles are based on sequential code, having been laid down in a RAM (memory) and being scanned by an instruction sequencer (fig. 1). Xputer machine code, however, is primarily non-sequential: for interconnect programming into field-programmable media (e.g.FPGAs: field-programmable gate arrays). In contrast to the RAM being the most important technology platform of **com**puters already for several decades, commercially available versions of field-programmable media just recently obtained properties needed in use for a universal machine paradigm: such as being electrically (re)alterable quickly, incrementally (re)programmable, and having sufficiently high integration density and switching speed. Already now (1990) field-programmable media market is a (worldwide) billion US-dollar niche of the integrated circuits market: more than 12 vendors from Europe, Japan, Korea and USA.

Bypass on the Way to ASIC Implementation

A very interesting property of FPGAs having been made commercially available recently is the ***retargetting*** capability: for particular FPGAs (e.g. Plessey) there are also "real" Gate Arrays (much higher integration density), which are ***code-compatible*** to these FPGAs. Due to such a technology platform xputers offer a drastically shortened path from algorithm to silicon: the machine code obtained by programming an xputer may be submitted for gate array fabrication. No expensive hardware design process is needed. Xputers feature sufficient throughput to be competitive to (traditional) ASICs.

Xputer Architecture Research

For a number of application areas it is area-inefficient to completely compile the rALU. For instance for very high performance number crunching it is much more efficient to use a repertory of predefined super operators which are embedded into a field-programmable framework. Also very long data word rALUs are not a trivial design problem. To implement pipelined compound operators based on interleaved memory the rALU architecture should support pipelining xpilers. In general it would be useful to develop rALU architectures being as flexible as possible for particular commercial application areas, such as e. g. digital signal processing, image processing, data base machines and others.

Parallel Xputer Systems

Xputers are sometimes called communication machines, since the rALU is a highly efficient communication channel between different register files. In multiple scan cache implementations each cache uses its own address generator and memory interface. Each such a cache/ sequencer/interface combination could be viewed as a processor. For instance in a 3 scan cache FFT implementation this would mean a system, where 3 processors communicate with each other through the rALU. So this is a shared rALU parallel xputer system. We could also have several shared rALUs in such a parallel xputer system.

New Directions in Programming Languages

Code generated by xpilers is fundamentally different from code for **com**puters: one difference is the use of field-programmable media. Another difference of the target hardware is it, that its operation principles are data-procedural (in contrast to control-procedural von Neumann hardware). This requires a new class of (data-procedural) programming languages and also a fundamentally new kind of compiler. Due to the softness of the rALU the variety of constructs to be generated is substantially larger than for traditional compilers. This is a chance for much more efficient optimization strategies, but also a challenge to the compiler implementer: a new direction of R&D.