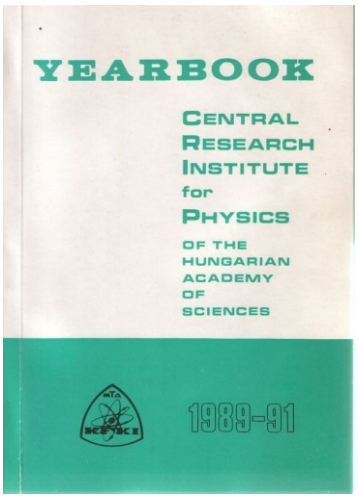

TPAXP-1 16-processor, Shared Memory MIMD Computer

György Matakovics

An usual method for Increasing the efficiency of a computer is to increase the performance of the processor; however, this is limited by current technology. Another possible solution is to disclose parallelism within a given task and to solve the task parallelly on more than one processor, or to run several application programs simultaneously on more than one processor in a multiuser operating system. The above said explains why multiprocessor computers are gaining more and more ground. This way of development is shown by large computer firms (IBM, DEC, Cray) as well as manufacturers of special computers of smaller series (Elxsi, Thinking Machines, BBN, Ncube, Encore, Convex, FPS).

During the last five years, production of such computers increased yearly by an average of 26% while only by 5% in the mainframe category. Almost half of all installed mini/mainframe computer systems have parallel architecture by the end of 1991. This means that several processors work on one or more programs parallelly, thus increasing the system's throughput. Parallel systems are suitable for geophysical and molecular biology research, in the field of chemistry, circuit and other simulations, signal and image processing. Multiprocessor systems are also used in financial modelling and data base management**.**

Parallel architectures have many advantages:

modularity - using processor elements of the same type a wide range of performances can be covered, and, If the user's demands Increase, further processors can be added to the configuration,

reliability - the system remains operable even if certain processor elements break down;

super performance - this can be achieved by arranging a system parallelly: when the fastest possible processor Is made the next step is to Interface two of them, in other words to arrange a multiprocessor architecture; In the case of high-performance multiprocessor systems made from relatively cheap microprocessors a better *price/performance* ratio is obtained - except for the low-end category.

Some problems of parallel processing systems

The present multiprocessor systems can be classified by memory architecture Into three groups:

Tightly coupled sytems - in such systems all processors share the same physical memory the whole of which is available for all processors; peripherals are available for all processors (occasionally indirectly). In tightly coupled systems the operating system (code and data structures) exists only in a single copy In the memory, which can be accessed by any other processor. In such *shared-memory* systems there Is no need for separate communication between processors because common variables are available for any processor by a memory reference Instruction realizing an implicit communication method.

In *snugly coupled* systems (it is also called Nonuniform Memory Architecture) several processor cards (any of them can be a tightly coupled multiprocessor system) are Interfaced by a common bus; any processor has its own memory, which is not available for other processors; the global memory, which is accessible from each processor, is on a separate memory card.

In *loosely coupled* systems several computers (any of them can be either a snugly or a loosely coupled multiprocessor system) are interfaced by a local network (e.g. Ethernet, hyper cube); it has not any common memory; peripherals Interfaced to a computer are usually available for any other ones. Any computer of this type has its own operating system but there is no global memory.

In such *local memory* systems information exchange takes place explicitly, through a local network. If any processor needs calculation results from any other one, it should be mailed. The duration of communication is in the millisecond range, which is remarkably slower than that in a tightly coupled system. Here, each member of the system has its own operating system, which communicates through a common memory. Communication can take place either through a common data base (implicit) or in the form of message passing (explicit).

From the viewpoint of programming, *shared memory systems* are advantageous because of implicit communication. Programmers do not have to pay extra attention to send results from processor to processor, they should only be aware that there will be a result by the time it is needed. In the present shared memory systems the main problem is caused by the access time of the memory being slower than the cycle time of the processor and thus in the case of the same memory unit there is no opportunity for two or three processors to operate concurrently. Namely, if there is a great number of processors a fast performance decrease happens as more and more processors want to reach the same memory unit *(memory conflict)* even if their traffic Is minimised by cache techniques.

A further problem may be caused by the bandwidth of the interconnection between processors and memories because - depending on Its arrangement - a situation of collision can occur *(interconnect conflict)* which leads to further performance decrease.

In *local memory* systems there is no memory conflict and thus they are allowed to consist of a great number of processors. However, they may also have some disadvantageous characteristics: none of the processors is able to communicate directly with any of the others. Complexity of the communication network is quadratically proportional to the number of processors. In order to achieve lower complexity the whole communication network is not usually realized Thus communication between processors becomes indirect, namely the data reach their destination through intermediate processors increasing thereby

the length of communication. A further problem involved is that *explicit* communication increases software complexity thus decreasing performance.

TPA XP-1 multiprocessor

TPA XP-1 is a *shared memory, tightly coupled* multiprocessor system. Its processor hardware is of general purpose, it is not specialized for any use. Its flexible architecture enables all accessible processors to work together even in the case of a single application.

Its architecture consists of 16 processors. The processors are completed by a floating-point coprocessor, which - by virtue of its high performance - may substitute the vectorprocessor. 64 Mbyte memory corresponds to each processor which is part of the global memory and is accessible for each processor. The large and high-performance crossbar memory system can be extended up to 1 Gbyte.

The console subsystem of the computer provides customary console functions, serves diagnostic purposes, and performs error logging for the operating system. With its help the computer can be put into a remote diagnostic network in order to increase availability.

Hardware

Processor/memory interconnect

In shared memory multiprocessor systems data traffic between processor and memory units takes place through an interconnect network. Several types of interconnects exist, e.g. bus, multiple bus, Multistage IN, Crossbar IN. Conflicts arising from the use of interconnect significantly decrease the resulting performance of the multiprocessor system. This shows why the type of interconnect should be considered thoroughly and optimized to data traffic and the number of processors.

Crossbar Interconnect

TPA XP-1 interfaces 16 processors and 16*64 Mbyte memory through two one-way crossbar networks. The whole memory is physically shared among processor elements, in order to simplify the use of the cache technique. The

whole memory Is accessible for any processor, however, processors reach their own memory not through the crossbar interconnect, but in a local way. The processor-cache has an effect only on local accesses. The architecture consists of two one-way crossbar interconnects, one from the processor to the memories and another from the memories to the processors. Crossbar data paths are each 40 bits wide in each direction with 32 data, parity and hand-shake lines.

Processor

MIPS R3000/R3010 type microprocessor is one of the best microprocessors due to its relatively low frequency (25-33 MHz), high scalar (20-27 MIPS), and very high floating-point DP Linpack performance (3.9 MFLOPS), its large cache memory size and multiprocessor support. The R3000 chip consists of a central processing unit, 32 registers each of 32 bits, a memory management unit, cache and memory controller logic, and manages 4 Gigabyte virtual address space. An R3010 tightly coupled floating coprocessor extends the R3000 by floating-point instructions and operates with single (32 bit) and double (64 bit) precision data format in accordance with IEEE standard 754. The processor unit of TPA XP-1 is built on R3000 which operates on 40 nsec cycle time. The 128 Kbyte cache memory (64 Kbyte instructions, 64 Kbyte Data) operating by physical addresses consists of SRAMs of 20 nsec access time. The two caches work so that they overlap in time. This ensures 200 Mbyte/sec bandwidth of the cache, which is needed for the processor to work without waiting state. The cache memory is write-through, directly mapped. Automatic consistency is ensured only at data-cache. Instruction-cache consistency is ensured by software (cache flush). In order to increase the hit rate of the cache there is a 4-word, so called "look-ahead[1]' (block refill). "Instruction streaming" mode that enables the instruction to be performed at the time of l-cache refill.

The connection between the processor and the memory is a 100 Mbyte bandwidth, synchronous local bus. The crossbar interface and the optional I/O interface are connected there. Total transfer bandwidth of interconnect is 1280 Mbyte/sec by 16 node. With the help of the optional I/O interface the processor is

able to manage peripherals through any standard I/O bus. At present VME, the most widely known I/O bus is used.

The CPU card contains a 512 Kbyte Monitor program, which is burnt in EPROM. It contains power-up diagnostics and enables the CPU to operate under control (memory write/read, loading operation system, loading and starting standalone programs, etc.)

Memory

One memory module of XP-1 operating synchronously with the CPU is of 64 Mbyte capacity by using 4 Mbyte dynamic memory units (DRAM). Using 25 MHz system clock frequency the VLSI programmable DRAM controller ensures a 160 ns access time.

A memory module consists of four independent 16 Mbyte units. It is interleaved four times, namely there is an opportunity to read four words of 32 bit each from sequential addresses.

Error control is ensured by a 14/20 nsec EDAC (Error Detection And Correction) circuit - which is a basic safety requirement. It corrects single errors and signals multiple ones.

Supervisory system

The supervisory system monitors environmental conditions of the operation of TPA XP-1 multiprocessor and intervenes if necessary. It measures environmental temperature, the temperature difference between in- and outflowing air, the ventilation, the voltage level of the power supply units, the quality of the grounding system (ground current).

Technology

Circuit elements of TPA XP-1 multiprocessor are High-speed CMOS based. The 64 Mbyte memory and the processor module are mounted on a 12-layer printed circuit board. Their current supply is ensured by 48V/5V 30A DC/DC converters mounted on them. The back panel is assembled from both sides and

has 14 layers. Interconnect cards are of 6 layers with a current supply ensured by the processor units.

Software

The operating system of the XP-1 computer is the multiprocessor version of the UNIX operation system. "Version" does not mean another (non-compatible) UNIX system, merely an adaptation of an existing (AT&T System V Release 4) system to the multiprocessor TPA XP-1 configuration.

User requirements of adaptation:

— *Portability of programs*

Existing programs (application programs, compilers, libraries, etc.) should be able to be used in the adapted system without any modification. Adaptation should "hide" TPA XP-1 multiprocessor architecture from these programs.

— *Supporting of multiprocessor applications*

Adaptation should enable applications that explicity require a multiprocessor environment. Simple portability cannot be expected - as there are no standards -and System V adaptations tailored to other multiprocessor systems are regarded as a norm.

— *Hardware requirements of adaptation*

TPA XP-1 architecture is situated between tightly and loosely coupled systems, because the whole physical memory is available for each processor, but **a** part of the memory corresponds to any processor which seems to be local (from the point of view of access time). TPA XP**-1** architecture Is symmetric: any of the processors can perform Input/output tasks. However, in practice more than one, occasionally two, input/output processors are needed. Hence XP**-1** architecture becomes asymmetrical - at least from the point of view of the operating system.

The operating system has thus to be prepared for the following cases:

— Configuration is fully symmetric, each processor performs both input and output tasks.

— Configuration is fully asymmetric, all input/output tasks are performed by single processor. It can be a "dedicated" input/output processor, which deals only with input/output operations.
— Any quasisymmetric configuration between the two extreme cases. This means that the operating system should be dynamically reconfigurable. It requires the functional separating of the operating system kernel functions between processors.

Kernel functions

1. System calls

System calls determine the surface of the operating system "visible" from the side of the application programs. Consequently, system calls determine the operating system (e.g. System V, its version, etc.).

Serving the majority of system calls requires services from other functional areas (e.g. input/output, scheduling, etc.), however, system call definitions fully belong here. Bearing in mind the wide utilization of UNIX systems, the "natural" place of algorithms serving system calls is the local memory of each processor. This means that these algorithms exist In as many copies as there are processors.

2. System services

System services that are invisible to the user (or, which should be invisible, but are independent of the interface) belong here.

System administration has both processor-dependent and independent functions. An example of the latter is the serving of hardware interrupts (excluding input/output Interrupts which are managed as separate functions).

The case of processor-dependent functions is fairly unambiguous: interrupts can be software or hardware errors. Software errors (usually fatal ones) can arise either from the operating system (system crash) or application programs (program abort), their service is in no way time critical. The size of the service code determines whether it is worth storing it in a single copy (anywhere)

or whether there is "enough space" for it in more than one copy, in the local memories of each processor.

However, the serving of hardware errors is connected with the local memory: an interconnect error cannot be managed by a procedure whose code Is In a "remote" memory, and It would be reached only through the (faulty) Interconnect.

The case of processor-Independent functions is not so simple and It Is Impossible to come to a final decision without performance investigations. A major difference from single-processor systems Is that resource management Is not restricted to one processor; balanced loading of processors should be ensured. Process migration with all its problems is inevitable.

3. Input/output services

Input/output procedures belong here; they are independent of the type of peripheral (class drivers). For example, in the case of each disc peripheral there Is a need for buffer management, but its algorithm is independent of the type of disc unit. These management functions also have their natural place: the local memory of processors to which the given type peripheral is interfaced.

4. Peripheral management

Here belong the procedures operating the peripherals directly (port drivers). The natural place of these procedures is the local memory of the processors to which the peripheral is Interfaced.