ARRAY AND PIPELINED PROCESSORS

The concept of overlapped computation has been used extensively in computing field. One of the oldest and most powerful way of such computation is employing pipelining, wherein a functional unit is partitioned in to a number of independent steps, with buffers in between two stages of a hardware unit. Such an arrangement allows simultaneous execution of instructions, just like an assembly line. An example of pipelined hardware system is illustrated in Fig. 1. At any given time, each step performs only one computation and partially processed results are passed on from one stage to another. Such back-to-back simultaneous processing of operands does not change the total time, but drastically enhances the rate at which a sequence of inputs can be processed. This increase in the throughput rate implies reduced processing time and is especially useful for vector processing. If there are m-stages and n set of operands are processed, then the total computation time, T, and the speedup, S_p, due to pipelining is given by:

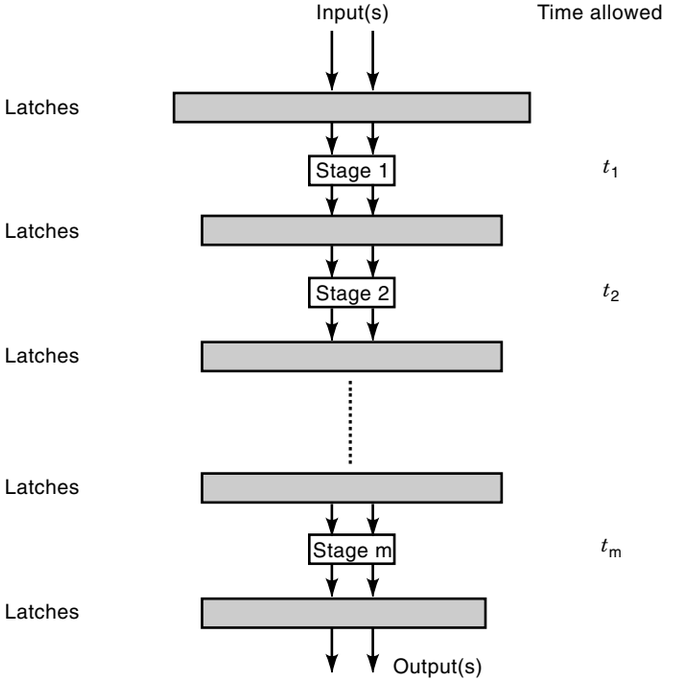

Figure 1. Pipelined hardware system.

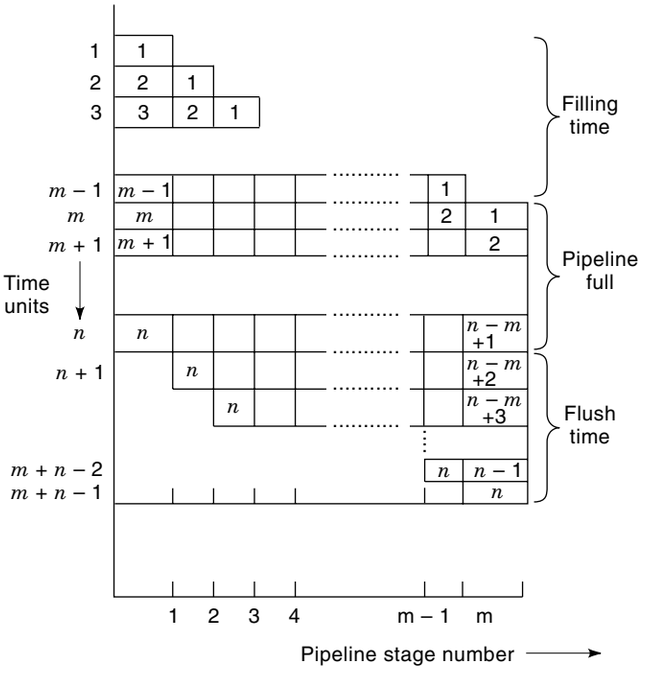

Figure 2. Use of m-stage pipeline for n-successive inputs ($n \geq m$) (number inside each block represents a specific input being processed by a stage).

$$T = mt + (m+1)\tau$$

$$S_p = \frac{mnt + (m+1)n\tau}{(m+n-1)t + (m+1)\tau}$$

where t is time required for each stage, and τ is the delay in each latch circuit. Usually, τ is much smaller than t and can be ignored for $\tau \ll t$. For large m, speed up of n could be reached asymptotically. Another important parameter that characterizes the effectiveness of a pipelined system is its efficiency, which basically represents the average time all the pipeline stages are busy in computation. Figure 2 shows the use of various stages for m-successive inputs assuming $m \geq n$. Each time unit (cycle) represents time taken by each pipeline stage, including the latch circuit. The percentage utilization is also impacted by m and n and variation of speed up and efficiency is shown in Fig. 3.

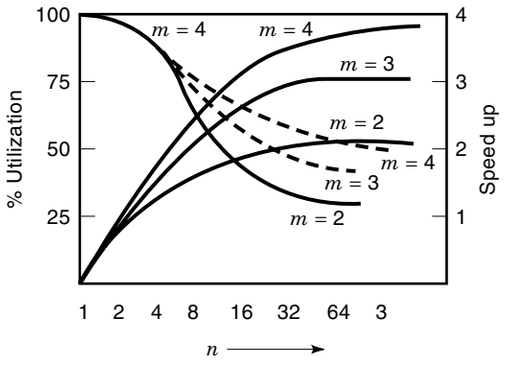

Figure 3. Variation of speedup and efficiency of m-stage pipelined system with n-successive inputs.

Another way of overlapping is to do spatial redundancy by duplicating each arithmetic and logic unit (ALU) to process multidimensional quantities, such as 2-D arrays. In such a configuration, all ALUs are controlled by a single control unit in single instruction multiple data (SIMD) computation mode. Again, with m processors, a theoretical speed up of m could be achieved. Of course, each ALU of the array could itself be pipelined. If one has to choose between pipelined versus array processing, one needs to carefully examine whether only one-dimensional vector operations are needed, or whether a lot of matrix manipulations needs to be done. In this article, numerous systems employing pipelining and array processing are considered, and then their relative advantages are examined in detail.

PIPELINED PROCESSORS

Introduction

Pipeline computers refer to those digital machines that provide overlapped data processing in the central processing unit (CPU), in the I/O processor, and in the memory hierarchy. Pipelining is practiced not only in program execution but also in program loading and data fetching operations. The performance of a pipeline processor may be significantly degraded by the data dependency holdup problem. The evolution of the CDC 6000/7000 series has contributed to the development of hardware/software mechanisms to overcome this difficulty. The CDC 6000 series uses a status check board to indicate the availabilities of various resources in the computer required to execute various stages of subsequent instructions. Resource conflicts are recorded in the check board. Instructions being interrupted are temporarily queued for deferred executions.

Vector Supercomputers

A supercomputer is characterized by its high computational speed, fast and large main and secondary memory, and the extensive use of parallel structured software. In terms of speed, current supercomputers should be able to operate at speeds of 100 MFlops or more. The first generation of vector supercomputers include TI-ASC which can handle up to three-dimensional vector computations in pipeline mode. Vector processors entered the second generation with the development of Cray-1, the Cyber-200 series and the Fujitsu VP-200. The maximum CPU rate of Cray-1 is 160 MFlops if all the resources are kept fully utilized.

TI-ASC Architecture

The central processor of ASC is incorporated with a high degree of pipelining at both instruction and arithmetic levels. The basic components of the ASC system are shown in Fig. 4. The central processor is used for its high speed to process a large array of data. The peripheral processing unit is used by the operating system. Disk and tape channels support the storage units. The memory banks and an optional memory extension are managed by the memory control unit. The main memory has eight interleaved modules, each with a cycle time of 160 ns and a word length of 32 bits. The micro controller unit (MCU) is an interface between eight independent processor ports and nine memory buses. Each processor port has full access to all memories.

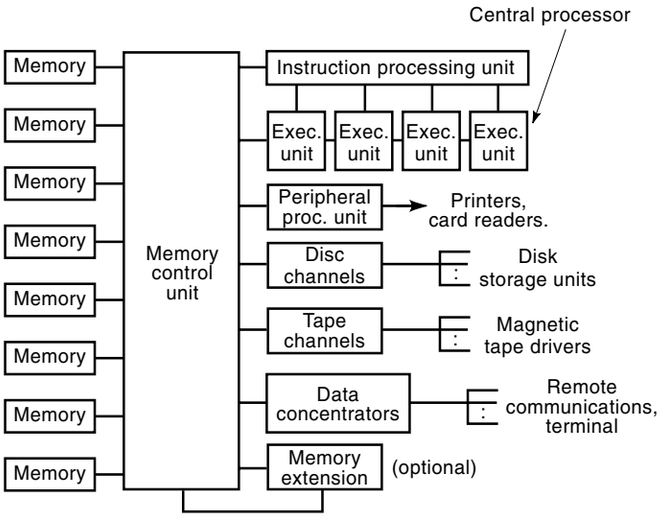

Figure 4. TI-ASC systems configuration.

The central processor can execute both scalar and vector instructions. Figure 5 illustrates the functional pipelines in the central processor. The processor includes the instruction processing unit (IPU), the memory buffer unit (MBU), and the arithmetic unit (AU). Up to four arithmetic pipelines (MBU-AU pairs) can be built into the central processor. The primary function of the IPU is to supply the rest of the central processor with a continuous stream of instructions. Internally, the IPU is a multisegment pipeline which has 48 program-addressable registers for fetching and decoding instructions and generating the operand address. Instructions are first fetched in bytes from memory into the instruction buffers of 16 registers. Then, the IPU performs assignment of instruc-

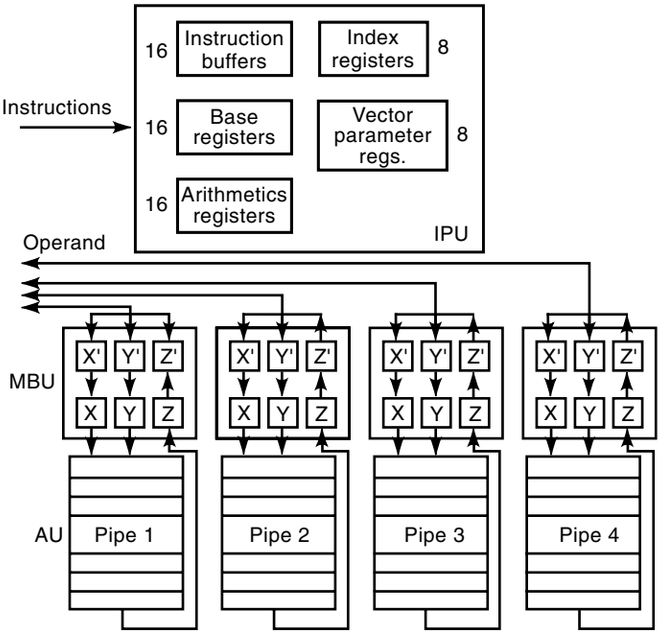

Figure 5. Central Processor of the TI-ASC with four arithmetic logic pipelines.

tions to the MBU-AU pairs to achieve optimal use of arithmetic pipelines. The MBU is an interface between main memory and the arithmetic pipelines. Its primary function is to support the arithmetic units with continuous streams of operands. The MBU has three double buffers, with each buffer having eight registers: X and Y buffers are used for inputs, and a Z buffer is used for output. The fetch and store of data are made in overlapped mode, using eight-word increments. The AU has a pipeline structure to enable efficient arithmetic computations.

SIMD PROCESSORS

Array Processors are the SIMD class of architectures. Figure 6 shows the structure of a typical SIMD machine. There are n arithmetic-logic processors (P_1 through P_n), each with its own memory block (M_1 through M_n). The individual memory blocks combined constitute the system memory. A bus is used to transfer instructions and data to the control processor from the memory blocks. The control processor (CP) decodes instructions and sends control signals to processors P_1 through P_n.

The control processor is a full-fledged uniprocessor. It retrieves instructions from memory, sends ALU instructions to processors, and executes controls instructions (branch, stop etc.) itself. Processors P_1 through P_n execute the same instruction, each on its own data stream. Some computations on SIMD require that the data be exchanged between arithmetic processors. The processor interconnection network enables such data exchange. The most important characteristic of SIMDs is that the arithmetic processors are synchronized at the instruction level: that is, they execute programs in lockstep mode. SIMD machines are called *array processors* because computations involving arrays of data are natural targets for this class of architecture. The issues related to memory organization, the control processor, the arithmetic processors, and the interconnection networks used in SIMD machines are considered in the following sections.

Memory Organization for SIMD Machines

In a typical SIMD model, each processing element (PE) is connected to its own memory block. The conglomeration of all the memory blocks forms the system memory. Programs typically occupy parts of several memory blocks. For instance, if the memory addresses are interleaved between the blocks, instructions in a program would spread among all the blocks of the memory. Data are distributed among the memory blocks such that each PE can access the data it operates on. The control processor (CP) accesses the memory for instruction retrieval, and PEs access the memory for data retrieval and storage. This simultaneous access requirement might result in memory access bottleneck.

One way of minimizing the memory access bottleneck is to include a memory block at the CP in which the instructions reside. This organization partitions the system memory into data and program memories. In both of these organizations, each PE has its own local memory. This requires that the data be loaded into the memory such that the data elements required for computation by a PE are available in its local memory. Another possible variation is the system where the data memories are connected to the PEs with an (nxn) switch. This switch allows any data memory to be accessed by any PE. That is, the switch is a *data alignment* network. This allows for a flexible data structuring at the cost of switch complexity. Data structuring is an important consideration in the application of SIMD architectures.

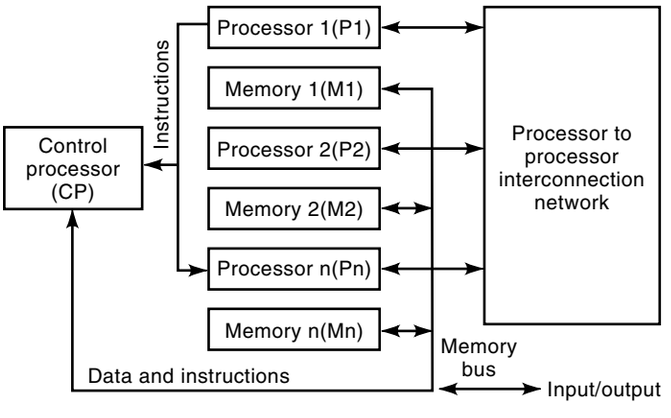

Figure 6. A genetic SIMD machine structure.

Control Processor for SIMD Machines

The control processor fetches instructions and decodes them. It transfers ALU instructions to PEs for execution. That is, the CP generates control signals corresponding to these instructions that are utilized by the PEs. Control instructions are executed by the CP itself. The CP performs all the address computations and might broadcast data elements to all the PEs as required.

Arithmetic/Logic Processors

The PEs perform the arithmetic and logical operations on the data. Thus, each PE corresponds to data paths and arithmetic/logic units of an SISD processor, capable of responding to control signals from the control unit.

Interconnection Networks for SIMD Machines

The PE-to-PE interconnection network (IN) for an SIMD computer depends on the dataflow requirements of the application. An IN consists of several *nodes* interconnected by *links*. A node, in general, is either a PE or a memory block. A link is the hardware interconnect between two nodes. The two functional entities that form the interconnection structure are *paths* and *switches*. A path just transmits the message and does not alter it in any way. A switch may alter the message or route it to one of the number of alternative paths available. Some of the *static topologies* frequently used are the linear array and ring, two-dimensional mesh, star, binary trees, completely interconnected networks, and the hypercube.

A parallel computer system with *static interconnection topology* can be expected to do well on applications that can be partitioned into processes with predictable communication patterns consisting mostly of exchanges among neighboring PEs. *Dynamic networks* can be classified under the following categories: (1) bus networks, (2) crossbar networks, (3) switching networks.

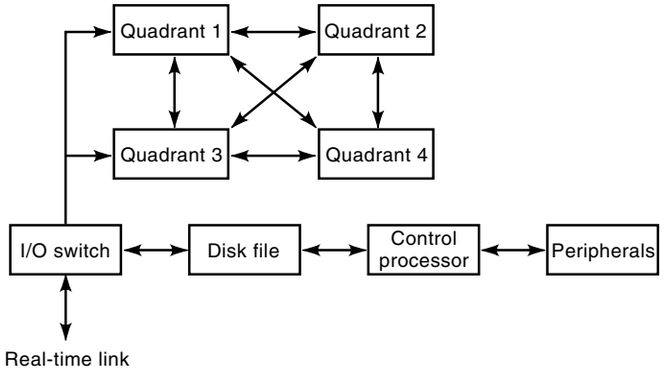

Figure 7. ILLIAC-IV structure.

Bus networks are simple to build. Several standard bus configurations have evolved with data path widths as high as 64 bits. A bus network provides the least cost among the three types of dynamic networks and also has the lowest performance. Bus networks are not suitable for PE interconnection in an SIMD system.

A crossbar network has the highest performance, highest cost alternative among the three dynamic network types. It allows any PE to be connected to any nonbusy PE at any time. Single-stage and multistage switching networks offer a cost-to-performance compromise between the two extremes of bus and crossbar networks. A majority of switching networks proposed are based on an interconnection network scheme known as the perfect shuffle or its equivalent.

EXAMPLE SIMD SYSTEMS

This section provides brief descriptions of the hardware, software, and application characteristics of a few SIMD systems.

ILLIAC-IV

Figure 7 shows the system structure. The system is controlled by a Burroughs B-6500 processor. This machine compiles the ILLIAC-IV programs, schedules array programs, controls array configurations, and manages the disk file transfers and peripherals.

Figure 8 shows the configuration of a quadrant. The control unit (CU) provides the control signals for all processing

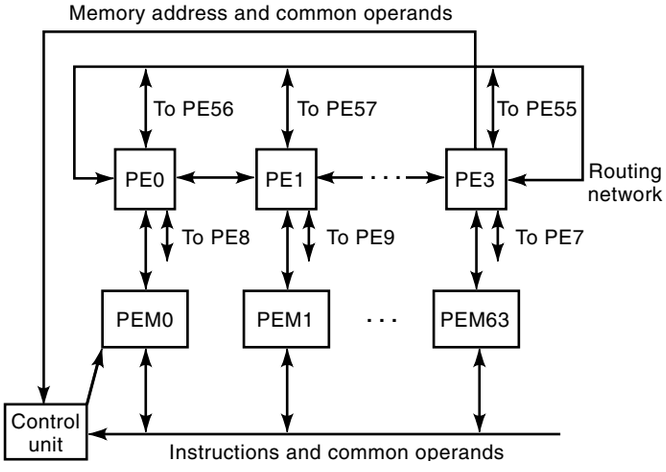

Figure 8. A quadrant of ILLIAC-IV.

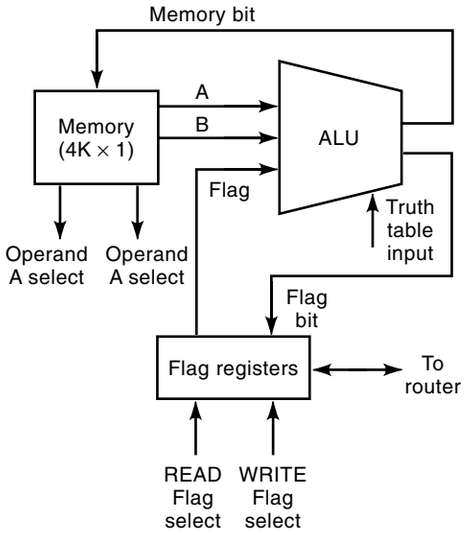

Figure 9. Connection machine processing element.

elements (PE_0–PE_{63}), which work in an instruction lock-step mode. The CU executes all program control instructions and transfers processing instructions to PEs. The CU and the PE array execute in parallel. In addition, the CU generates and broadcasts the addresses of operands that are common to all PEs, receives the status signals from PEs from the internal I/O operations and from the B-6500, and performs the appropriate control functions.

The PE-to-PE routing network connects each PE to four of its neighbors. The PE array is arranged as an 8×8, 2-dimensional (2-D) mesh or torus. Interprocessor data communication of arbitrary distances is accomplished by a sequence of routings over the routing network. Two high-level languages are used to program ILLIAC-IV:CFD, oriented toward computational fluid dynamics, and GLYPNIR, an ALGOL-like language. A compiler that extracts parallelism from conventional Fortran programs and converts them into parallel Fortran was also developed.

Thinking Machines Corporation's Connection Machine-2

The Connection Machine-2 (CM-2) is a massively parallel SIMD machine configured with up to 64 K single bit processors, providing a performance in the range of 2.5 Gflops.

Processor for CM-2. The processors are implemented using four different chip types: the processor chip, the memory chip, and two floating-point (FP) accelerator chips. The processor chip contains the ALU, flag registers, NEWS interface, router interface, and I/O interface for 16 processors. The 16 processors are connected by a 4×4, 2-D mesh allowing processors to communicate with all their four neighbors. The first of the floating-point chips contains an interface that passes the data on to the second chip, which is the floating-point execution unit. Thus, each section of the parallel processing unit (PPU) contains 1024 custom processor chips, 512 FP interface chips, 512 FP execution chips, and 128 Mbytes of RAM. Figure 9 shows a processing cell for CM-1. The ALU executes an instruction by taking three 1-bit inputs and producing two 1-bit outputs. Two of the inputs are taken from

that processor's memory, and the third bit is taken from one of the flag bits. The outputs are then calculated from one of 256 possible functions found in the memory/flag lookup table using 8-bit control bytes provided by the sequencer. One output bit is written back into the memory, and the other is written to one of the flag bits for subsequent processing. All processors receive the same instruction from the sequencer; however, individual processors can be masked off with a flag.

Network for CM-2. The processors are linked by a 12-dimensional hypercube router network. Message passing occurs in a data parallel mode. The communication hardware supports certain message combining operations. For instance, the processors to which multiple messages are sent receive the bitwise logical OR from the ALU output of all the messages, or the numerically largest, or the integer sum. The simplest mode of communication is broadcasting a value from the front-end system to all the processing elements (PEs). A context flag within each PE controls which PEs receive the broadcast value. When large amounts of data must be loaded into the CM memory, the I/O mechanism is generally faster.

I/O System for CM-2. Each 8K-processor section is connected to one of the eight I/O channels. Each I/O channel may be connected to either a high resolution graphics display frame-buffer module or an I/O controller. I/O transfers are initiated by the front-end computers, which cause direct, parallel transfers between the I/O devices and the data processors.

Software for CM-2. System software is based on the operating system or environment of the front-end computers with minimal software extensions. Thus, users can program using familiar languages and program constructs with all development tools provided by the front end. In addition to the assembly language Paris, the high level languages (HLLs) used for programming the CM are *LISP, CM-LISP, and C*.

Applications for CM-2. The CM-2 has been used in the following application areas: (1) Document retrieval from large databases, analysis of English text, and memory-based reasoning; (2) circuit simulation and optimization of cell placement; (3) object recognition and image processing; (4) computer-generated graphics for animation and scientific visualization; and (5) neural net simulation, conceptual clustering, classifier systems, and genetic algorithms, and so on.

MasPar Corporation's MP Series

The MasPar MP-1 is a data parallel SIMD system with the basic configuration consisting of the data parallel unit (DPU) and a host workstation (DEC VAX station 3520). The DPU consists of between 1024 or 16,384 processing elements (PEs). The maximum performance ranges up to 26,000 MIPS and 1,300 Mflops. The peak I/O bandwidth to the external host for a fully configured system is 200 Mbytes/s. The programming environment is UNIX-based and fully supports network system access. Many programming languages are available on the host workstation, which may call functions on the DPU that are developed using MasPar Fortran (MPF) or MasPar Programming Language (MPL).

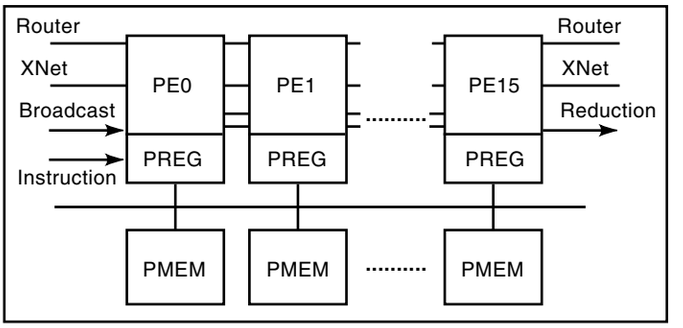

Figure 10. PE cluster for MP-1.

Hardware Architecture. The MP-1 DPU consists of a PE array and an array control unit (ACU). The ACU fetches and decodes instructions, computes address and scalar data values, and issues control signals to the PE array. It is a 14 MIPS scalar RISC processor. Most instructions can be executed in one 80 ns clock cycle. The PE array is configurable from 1 to 16 identical processor boards. Each processor board has 64 PE clusters (PECs) of 16 PEs per cluster. Each processor board thus contains 1024 processor elements. Each PEC (Fig. 10) is composed of 16 PEs and 16 processor memories (PMEM). Each PE has a large register set (PREG). Load and store instructions move data between PREG and PMEM. The ACU issues the same control signals to all PEs, but the operation of each PE is locally enabled by the E-bit in its flag unit.

Processor memory (PMEM) is implemented with 1 Mbit DRAMs that are arranged in the cluster so that each PE has 16 kbytes of data memory. A processor board has 16 Mbytes of memory, and a 16-board system has 256 Mbytes of memory. The PMEM can be directly or indirectly addressed. The X-net interconnect directly connects each PE with its eight nearest neighbors in a two-dimensional mesh. Each PE has four connections at its diagonal corners, forming an X-pattern. The multistage crossbar interconnection network provides global communications between all the PEs and forms the basis for the MP-1 I/O system. The multistage crossbar is well matched to the SIMD architecture because all communication paths are of equal length, and therefore, all communications arrive at their port targets simultaneously. The router chip connects 64 input ports to 64 output ports by partially decoding the target PE address. The array control unit can broadcast data to all the PEs. The I/O system consists of dedicated I/O RAM buffers arranged 16 Mbytes per board for a total of 256 Mbytes RAM. An equal number of I/O boards and PE boards are required. The I/O RAM is accessed by the PEs through the multistage global router. The external host workstation interface is the MasPar I/O Channel (MPIOC). This channel has a 64-bit data bus with a 200 Mbytes/s transfer rate.

Programming Environment. The MasPar programming environment (MPPE) is a comprehensive set of development tools and provides a single context for software development integrating myriad steps into a fluid, graphical process. The programming environment allows a range of languages to be used on the host; however, the DPU must be accessed from these languages from vendor-supplied libraries or user-devel-

Table 1. Comparison of Pipelining versus Array Processing

Criteria	Pipelining	Array Processing
Operation	Vector operation	Multidimensional vector (array)
Processing	Element by element	Element by element of partitioned array
Hardware	Single ALU with latches between stages	Multiple ALU
	MISD	SIMD
Control	Single control	Single control
Data	Placed with ALU	Data need to be distributed among different ALUs
Results	Single result sent out or fed back	Results with different processors, need to combine data
Ideal speed up	No. of stages	No. of ALUs
Use	Extensive use in current microprocessors	Primarily useful for special purpose applications
Programming	Easy	Difficult

oped code using MasPar Fortran (MPF) or MasPar Programming Language (MPL). MPL is based on C and is minimally extended with new keywords, statements, and library functions that operate on parallel data.

Applications. The important characteristics of the MP-1 include: (1) distributed and isolated local processor memory for high-memory bandwidth, (2) local memory indirect addressing, which provides indexed data access for optimizing PE loads, and (3) massive parallel data communications between processors via the X-net and global router connections. Some applications for which solutions are implemented on the MP-1 are: (1) maximum entropy method applied to deblurring images, (2) Mandelbrot set, and (3) large fast Fourier transforms.

CONCLUDING REMARKS

This article provided an overview of some of the recent pipelined and array processors. The first section introduced the concept of pipelining, and the second section discussed the details of the TI-ASC, a vector supercomputer and the overall system configuration and some information of its central processor. The third section developed the basic concepts used in the organization of SIMD machines, memory issues in SIMD machines, the control processor design, the ALU design for SIMD machines, and appropriate interconnection networks. The last section discussed some example systems, in particular, ILLIAC IV, Thinking Machines Corp's Connection Machine, and MasPar Corp's MP Series. Finally, a relative comparison between pipelined and array processors is given in Table 1.

BIBLIOGRAPHY

1. G. B. Adams, D. P. Agrawal, and H. J. Siegel, A Survey and comparison of fault-tolerant multistage interconnection networks, Los Alamitos, CA: *IEEE Comput.,* **20** (6): 14–27, 1987.
2. D. P. Agrawal, *Advanced Computer Architecture* (Tutorial textbook), Washington, DC: IEEE Computer Society Press, 1986.
3. M. Cosnard and D. Trystram, *Parallel Algorithms and Architectures,* London: International Thomson Computer Press, 1995.
4. R. Cypher, *The SIMD Model for Parallel Computation,* Berlin: Springer-Verlag, 1994.
5. M. J. Flynn, *Computer Architecture Pipelined and Parallel Processor Design,* Boston: Jones and Barlett, 1995.
6. J. P. Hayes, *Computer Architecture and Organization,* 2nd ed., New York: McGraw-Hill, 1988.
7. K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability,* New York: McGraw-Hill, 1993.
8. R. Y. Kain, *Advanced Computer Architecture,* Englewood Cliffs, NJ: Prentice-Hall, 1996.
9. Sajjan Shiva, *Computer Design and Architecture,* 2nd ed., New York: HarperCollins, 1991.
10. H. S. Stone, *High-Performance Computer Architecture,* 3rd ed., Reading, MA: Addison-Wesley Series, 1993.
11. M. Tchuente, *Parallel Computation on Regular Arrays,* Manchester: Manchester University Press, 1991.
12. D. A. Patterson, *Computer Architecture: A Quantitative Approach,* San Mateo, CA: Morgan Kaufmann, 1995.

UNDARU RAGHAVENDRA
Intel Corporation

DHARMA P. AGRAWAL
University of Cincinnati, Ohio

ARRAY-BASED LOGIC. See LOGIC ARRAYS.
ARRAY FEEDS. See MULTIBEAM ANTENNAS.
ARRAYS. See FOCAL PLANES.
ARRESTER, SURGE. See SURGE PROTECTION.