The highly-parallel supercomputers: definitions, applications and predictions

by HUBERT H. LOVE, JR.
Radar Systems Group, Hughes Aircraft Company
Lincoln, Nebraska

1.0 INTRODUCTION

As computer processing power has increased over the past three decades, so have demands on computer performance. In the race of computer technology to keep abreast of these demands, more and more attention has been given to parallel hardware organizational techniques. Instruction and data pipelining, instruction overlapping and distributed processing are examples of such techniques.

More recently, a particular class of parallel architectures, which we shall term "supercomputers," has been receiving special attention. These are the very large, highly parallel reconfigurable array processors. In such a machine, the processing task is distributed among a large number of identical processors which are organized into an array configuration by means of a communication network. With proper algorithms and hardware implementation, these computers can achieve massive throughput rates in a number of useful applications. These include radar signal processing, short-term weather prediction, complex query/response systems and high-speed text processing.

There are several reasons for this increased interest. One, of course, is the importance of the aforementioned applications. Another is the possibility for massive computing capability inherent in the emerging VLSI technologies, and the particular suitability of highly-parallel computer organizations to these technologies. Still another is the increased importance of fault tolerance in computer systems designed for advanced, real-time applications. Many highly-parallel architectures have an inherent fault-tolerant capability.

The body of the present paper describes some of the highly parallel computer organizations and the applications suitable for them. General criteria for applicability are also given. The implications of VLSI with respect to these supercomputers and to fault-tolerant capability are explained. Finally, some predictions and suggestions are made as to the future course of highly parallel supercomputer development and the nature of the applications for which they will be intended.

2.0 GENERAL DESIGN CHARACTERISTICS

A reconfigurable array parallel system has several well-defined characteristics by which it can be identified. These are:

Most parallel array systems contain many identical processors (called "processing elements" or "PE's") capable of simultaneous arithmetic and logical operations. Their complexity varies from a few hundred gates to many thousands or tens of thousands. Their capability ranges from basic bit-serial logical and arithmetic operations on a single pair of given operands to full-scale bit-parallel processing on operands selected from large memories. The PE's themselves may contain their own internal memories or may use multiple shared memories, or may have combinations of both. During execution, various PE's will generally process similar data, using a common algorithm. In most systems, the PE's do not contain their own programs, but obtain their common control signals from external control units.

Figure 2-1 is a simplified block diagram of a typical reconfigurable array parallel system. The system shown contains sixteen processing elements.

Communication channels for control and data

All reconfigurable array parallel systems contain communication channels linking the processing elements. In almost all systems the PE's are linked to one another by these channels. In a few, they are simply connected to the common controller processor which determines their operation. The organization and capability of the system's communication channel network is critical to the system's applicability and performance. These channels pass data status and control information from one PE to another or from one to many PE's. The channels range in complexity from bit-serial to bit-parallel. Each PE may have one or many channels, each of which may connect to one or more of the other PE's or to a central control processor.

In general, each PE exercises some internal control over one or more of the following:

a) the internal source of the data to be output on the channel,
b) the particular channel to be involved, if there is more than one,
c) the intended destination of the data, whether a selected set of PE's or a central control unit,
d) the internal destination (register, memory location, etc.) of the data received on a channel.

**Control of execution: central or distributed**

Control of PE and communication channel operation is generally shared between the individual PE's and some central control unit (CU). The control unit consists of one or several serial processors with special interfaces for monitoring PE status and outputting common data and control information to the PE's. In general, the CU contains the system's programs and the system instruction-decoding logic as well. The individual PE's determine their operating states by a combination of the control signals from the CU and their own internal states, the latter being usually dependent on the results of the processing of the individual PE's particular data.

In Figure 2-1, the control signal paths from the CU to the processing elements are shown as dotted lines. The data paths between the PE's are shown as solid lines, as are the data paths between CU and the PE's. Some communication of data between the CU and the array is required, but will be minimal in a well-designed system.

In a few system designs, there is essentially no central control unit. Instead, control is distributed among the PE's themselves in multiple-instruction streams which execute in parallel. The Holland Machine [3] is an example of this.

**Input and output**

Most parallel array systems have multiple input and output channels, the reason being that they are expected to have a high data throughput in most applications. In many systems, there is an input and output channel for each processing element. In the system shown in Figure 2-1, there are input and output channels only for the processing elements on one edge of the array, with the communication network providing for the remainder of the input and output operations.

**Fault-tolerant capability**

All parallel array designs share the capability to some degree, even if not actually implemented, to provide graceful degradation of performance in case of hardware failure, and usually the potential capability for software assisted fault isolation and recovery as well. This is inherent in the parallelism of the architecture. It is enhanced by the fact that the majority of the system logic is replaceable in small units, and thus cannot cause single-point failures.

**3.0 THE APPLICATIONS AND THEIR CHARACTERISTICS**

Applications suitable for reconfigurable array processors are not as numerous as for conventional serial processors, but are more common than may at first be supposed.

Present and potential applications include:

1. data base management and query processing, particularly when the data base is complex and dynamically varying, when the queries are very complicated, and when fast response time is a requirement;
2. real-time text processing, again when the processing is complex and fast response is essential;
3. applications dealing with large matrices; weather prediction, determination of neutron flux densities in reactors;
4. real-time radar data processing; track-while-scan, radar pulse deinterleaving;
5. real-time data compression involving very high data rates;
6. image processing, both real-time and non-real time.

For some input and throughput requirements, many of these tasks are quite beyond the capability of serial processors, and also the fast pipeline processors, even with the most advanced gate technologies presently being developed.

**Determining the suitability of an application**

There are a number of reasonably well-defined criteria which can determine the suitability of a prospective application for implementation on a reconfigurable array parallel system. These are:
Parallelism in the algorithms

There are several ways in which parallelism in the application algorithms can be present.

First of all, some applications have a characteristic which shall be termed "block oriented." Block-oriented applications are those which deal with a number of similar "objects" in the data base or in the outside world, generally a varying number, and in which the identical general process is performed on all of them. In such applications, a block of memory or a processing element in the processor will be assigned to each such "object," and will contain the parameters and working space for that object. The "objects" may be:

1. targets or threats being tracked by radar;
2. blocks of raw English text being searched or modified;
3. records in a file being searched or modified;
4. pixels or sets of pixels in an image being processed.

Depending on the particular architecture, separate PE's will be assigned to each block, a single PE will process several blocks, or a number of PE's will be assigned to each block. The determination of the assignment is made dynamically during processing in some systems.

Complexity of the process

The existence of parallelism in the application is not sufficient justification for the use of parallel hardware. It is a requirement that the amount of processing on each piece or each set of data be sufficient to justify the time required to load the data. If this is not the case, the complicated parallel hardware is not being effectively used, and the processing could perhaps be better performed by other means.

An example for which the use of parallel systems can be well justified is the inversion of a matrix. The matrix need be loaded only once, and the number of operations performed to invert the matrix is very large indeed. Another example is dictionary lookup using an associative memory (which will be defined later), when the same dictionary resides in the memory during a large number of lookups and when the dictionary entries are of variable size.

Input considerations

Parallel systems are most efficiently used when the amount of processing is high in proportion to the quantity of input data. The actual input data rate which can be accommodated varies considerably with the system design. Some systems, having very powerful bit-parallel processors as PE's, can handle correspondingly high input rates. Other systems, such as bit-serial associative processors, which may have processing capability as great as the former, may not be able to handle such high input rates. This capability is of major importance in choosing a parallel system architecture.

Output considerations

There is a point of considerable importance regarding output rates for parallel systems in real-time applications which deserves special mention. Parallel systems have very high internal processing capability and can accommodate very high input data rates with the reservations just mentioned. However, they are sometimes criticized for having relatively low output data rates and thus being handicapped in real-time applications. This is especially true with regard to the associative processors, which often are able to output no more than a single word slice or bit slice with each clock cycle. (No matter if the word be very long, the rate is still slow in comparison with the internal processing speed.)

In a real sense this criticism is largely unjustified. A purpose of a powerful real-time data processing system should be to reduce the amount of data from a high input rate to a relatively low output rate as a result of its internal processing. But suppose that it does not reduce the rate and that the output rate is as high in proportion to the processing power as for ordinary serial systems. It is difficult to see what could be done with this gargantuan stream of output. Humans cannot absorb it in real time, even if it could be printed fast enough. If the system output is to be directed into another computer for further processing, something is lacking in the system engineering. That is, the justification for the use of a parallel system is presumably its high processing rate, yet it is not doing the entire processing task. It should have been designed to accomplish the entire processing task in the first place.

If the output is to a high-speed telemetry channel, rather than directly to another processor, there is still little justification for massive output capability in general. Reconfigurable array parallel systems can do an effective job of compressing data internally, using one of a number of techniques. Such capability can generally be designed into the system, and the very high output rate (and the need for expensive wide-band telemetry) thus avoided.

Other parallel attributes

For some system designs, it is possible to organize the processing so that some time-consuming operations can be performed for several different (and even unrelated) processes in parallel by a single hardware operation. This is true for systems, such as most associative processors, which are bit-serial in operation and which depend for their high processing throughput on a high degree of parallelism, rather than on being able to perform individual operations rapidly. Multiplication, division and floating-point arithmetic are examples of operations which are slow on bit-serial associative processors, but whose number of executions can be reduced by making them multi-purpose.

Communication channel considerations

For many applications, the capability to communicate large amounts of data is critical to system performance, and
the communication scheme must be given special attention in the design. The result of deficient communication capability is an unbalanced system, in which the individual PE’s can operate independently with high throughput, but are greatly slowed down when they must communicate with one another. Communication problems are the most difficult to handle when one is trying to determine the suitability of an application for implementation on parallel systems.

Avoiding serial operations

In determining the suitability of a task for implementation on a parallel array system, it is not sufficient to demonstrate only that critical operations can be performed in parallel. It must also be shown that all sequences of operations can be performed in parallel fashion without the necessity of serial processing in between. If this is not possible, most of the advantage of the parallel-processing capability will be lost. It takes suprisingly few serial operations to seriously compromise the entire processing scheme. Associative processors are particularly subject to problems of this sort. In a typical case, after a parallel operation is performed, it may become necessary to reorganize the data so that the next operation can be performed in parallel. If the reorganization cannot itself be performed in parallel, the hardware and/or data organization must be redesigned. If this will not suffice, the parallel-processing approach may have to be rejected.

4.0 THE IMPLICATIONS OF VLSI FOR HIGHLY PARALLEL SYSTEMS

Array architectures are complicated and generally have a higher gate count in proportion to their processing throughput than serial architectures. This has hindered their accept- ance in the past. However, with the recent Very-Large-Scale-Integration (VLSI) technology, it appears that not only does the handicap of complexity for array systems tend to diminish, but that such systems are even preferable in many respects from the standpoint of fabrication as compared with conventional systems.

Amenability to on-chip fault repair

Well-designed array architectures consist largely of repeated logical elements. Depending on their complexity, a number of these elements can be fabricated on a single LSI or VLSI wafer. This can mean a higher yield if discretionary wiring or other techniques can be used to disconnect faulty elements during the manufacturing process. Moreover, with proper circuitry, it is often possible to implement software-controlled fault-isolation and fault repair on the wafer itself. This is of considerable benefit in some applications and will be of increasing benefit in all applications as growing system complexity increases the probability of run-time failure. The advantage of repeated circuit elements from the aspect of fault tolerance results from the fact that single failures on a wafer, or even multiple failures do not render the wafer worthless if the wafer contains spare elements which can be switched into the system under software control.

Suitability for very fast logic

The two principal characteristics of the gate technology that is presently being developed are extremely high speed (the order of 500 psec. gate delays) and very large numbers of gates per wafer (several hundred thousand). This sounds like very good news at first glance. However, when one looks at the problems being faced by the engineers trying to fabricate this circuitry and design systems using it, a quite different picture emerges.

Many of the biggest problems with such high-speed logic are caused by the “off-chip” delays for signals passing from wafer to wafer. Particularly damaging are the cases involving “round-trip” signals. A round-trip signal is one which passes one way between two chips, causing a signal to be generated at the receiving chip which then passes the other way between the chips. Time and phase lags occur because of such signals, requiring that the clock rate for the system be greatly reduced.

Many parallel-array system designs appear to offer advantages with respect to these problems as compared with more conventional systems. For one thing, inputs to the individual PE’s are often very orderly, consisting mostly of control signals originating in logically similar registers within the controller. Since almost all of the processing for the system takes place within the array of processors, there is very little communication of data from the PE’s to the controller, thus reducing the occurrence of round-trip delays.

5.0 SURVEY OF PROCESSOR DESIGNS

In this section, two types of general reconfigurable array parallel systems will be described and analyzed. For each type, a general description of the system, its operation and applications will be given and comparisons with other system approaches made.

The system types to be covered are:

1. parallel network processors;
2. associative processors.

Parallel network processors

The term “parallel network processor” shall refer to a reconfigurable array parallel processor in which

1. the processing elements themselves are full-fledged serial processors with sizable local memories;
2. the communication network connects each PE to several other PE’s in a regular fashion;
3. one or several sets of PE’s each execute essentially the same program on different sets of data simultaneously

From the collection of the Computer History Museum (www.computerhistory.org)
under central control. Generally, there is only one set and one program.

The concept of the network processor arises from the nature of the applications which are involved. These applications deal with “objects” in the real world which interact in the application in some orderly fashion. The communication network in the network processor interconnects the PE’s in the same fashion, under software control, and each PE corresponds to one object.

Applications well suited to parallel network processors include:

1. the solution of sets of partial difference equations (by the relaxation technique, for example);
2. general matrix operations;
3. many image-processing applications; each PE corresponds to a pixel or a set of pixels;
4. radar data processing; each PE corresponds to a target or threat being tracked.

General organization and operation

The processor organization to be described is essentially the same as that for the SOLOMON I computer, which is the earliest of the well-known parallel network architectures. The ILLIAC IV, probably the most powerful supercomputer yet built, is another example.

The processor array

Figure 5-1 shows the organization of the array of processing elements (PE’s) for the system. Only 16 PE’s are shown. The SOLOMON I design has 1024 PE’s, organized into 32 rows and 32 columns. The array contains about 90 percent of the logic of the entire system. Each PE is a serial processor in every respect except for the absence of instruction-sequencing and decoding logic. The array hardware interconnects the PE’s in a two-dimensional array configuration. In the original SOLOMON design, the PE’s were bit-serial processors, and the communication channels were also bit-serial. In newer designs, because of the advances in LSI fabrication techniques, the PE’s can be bit-parallel. Therefore, in order to maintain system balance, the communication channels are made bit-parallel also.

The operation of the communication channels during program execution is under software control. The control is largely central with respect to the general interconnection modes. These modes permit inter-element communication by one or more of the following schemes simultaneously:

1. Each PE is connected to its two neighbors in the same row. The first PE in each row is connected end-around to the last PE in the row, resulting in a “horizontal” cylindrical configuration.
2. Each PE is connected to its two neighbors in the same column, with the connection end-around as before, resulting in a “vertical” cylindrical configuration. Both this option and the first can be used together.
3. The PE’s can all be connected in a single one-dimensional array, end-around if desired.

In addition to the array interconnections, all of the PE’s are connected to common buses through which they can receive common data items simultaneously. There are common buses for each row and each column of the array. With this scheme, many common data items can be input to the array simultaneously, a different one for the PE’s in each row or each column.

The processing element

Figure 5-2 shows the organization of each processing element in the array. Each PE contains a set of working registers and a local random-access memory for data only. There is a logic module for performing arithmetic, Boolean and comparison operations. There is also instruction-execution logic, but there the similarity to a conventional serial processor ends. The PE receives its instruction control from an external source. It has no internal instruction sequencing or instruction decoding logic. In addition, the PE has several unique devices with functions as follows:

1. Routing logic. This controls the source or destination of data and control information passed between the PE and other PE’s and common buses. During the execution of an instruction, the routing logic state permits the PE to send or receive information from the common row or column bus.
2. Mode register. This register (which is two bits long in the SOLOMON computer) is set from within the PE.
The instruction control information received by the PE for all instructions contains a specification as to which mode or modes the PE must be set in order that the PE execute the instruction. (All modes may be specified, in which case the execution is mandatory.) Through this device, each PE can determine in most cases whether or not it will execute an instruction. Each PE makes this determination as the result of the processing of its local data, and then sets its mode register accordingly.

The system organization

Figure 5-3 shows the general organization of a parallel network processor. This system consists of the parallel network processor array just described, plus a number of modules for control of the array, for operator interface and for input and output. These modules and their functions are as follows.

1. The Control Unit. This unit receives from the control memory each instruction executed by the system. It decodes the instruction and the addresses. For those instructions to be executed by the PE array, it sends the resulting control information and the row and column addresses to the PE array. These instructions will be executed by those PE's in the specified row(s) and column(s) whose internal mode register settings correspond to the mode settings specified in the instruction.

2. The Control Memory. This memory contains the system's programs, plus common data items and common working storage. The Control Unit has normal serial processing capability, which is used occasionally to calculate common data values and to determine the flow of the instruction execution sequence.

3. The Parallel Buffer. This buffer receives and sends data in parallel to and from all PE's in the array, or to and from all PE's in specified rows and columns. The buffer holds one data item for each row and one for each column. Data transfer between the Parallel Buffer and the remainder of the system is word-serial.

4. The Input/Output Control Unit. This device controls all system input and output for most applications. For some real-time applications, it is best to bypass such complex control units altogether and input data directly into the processor array.

ILLIAC IV and PEPE

The ILLIAC IV is perhaps the best known example of an array processor. It was designed with very demanding real-time applications in mind, such as real-time radar processing with a phased-array antenna system. The ILLIAC IV processing array consists of 256 large-scale processors having memory cycle times and add times of 240 nsec. (64-bit operands) and multiply times of 400 nsec. The array is essentially two-dimensional, like that of the SOLOMON.

The PEPE (Parallel Element Processing Ensemble) is a highly parallel organization lacking an inter-processor communication network. Instead, each processing element has the capability for independent decision, based upon its internal state, as to whether to participate in a particular operation or sequence of operations. It is also well suited to radar data processing applications, since such applications

From the collection of the Computer History Museum (www.computerhistory.org)
require little interchange of information between processors if the program is suitably organized.

**Associative processors**

The associative processor is one of the more unusual of the reconfigurable array systems, because its basic operating principle is in a sense the reverse of that for serial processors. The associative processor is based on a device known variously as the "associative memory" or "content-addressable memory" which, in turn, is an outgrowth of a simpler device known as a "search memory."

**The search memory and its operation**

The search memory can be said to function in reverse fashion with respect to a random-access memory. That is, a RAM accepts the address of the desired memory location as input and outputs the contents of that location in response. The search memory accepts the desired contents of the memory location or locations as inputs and outputs flag settings indicating the memory location or locations having those contents. To prevent the operation from being trivial, the content search is field specified.

The distinction between the search memory and a conventional memory is that the search is performed in the former at all locations or cells in the memory simultaneously. This capability is the result of having compare logic at every cell in the memory. (The term "cell" and "word" are sometimes used interchangeably in the discussion to follow, since the cell, which is a hardware device, contains a "word" of data in the usual sense.)

Figure 5-4 illustrates the search-by-content operation as performed by a search memory. In the figure, the memory is shown containing an indefinite number of cells, each containing seven characters (used instead of binary bits for clarity in the illustration). With each cell is shown a corresponding "match flip-flop" which is a flag indicating the success/fail status of the cell as a result of a content search. At the bottom of the figure are shown two registers. One of these, called the "compare register" or "comparand register," contains the data item that is the object of the search. The other register, called the "mask register," specifies by its contents the field or fields within the memory cells at which the content search will be made. In the figure, the 1st, 4th, 6th, and 7th character positions are masked out. The search will be conducted only at the remaining character positions.

It is seen that only the first and second cells contain the same characters in the unmasked character positions as the compare register. As a result of the content search, then, the match flip-flops will be set only at those two cells as shown. For most implementations of search memories, the cells' contents reside in a shift register, and bit-serial logic is used at each cell, rather than bit-parallel logic. The operation is still word-parallel, and the desired speed advantage will be achieved if there is enough data to be searched to require a large number of cells.

**The associative memory and its operation**

The associative memory is an extension of the search memory. Its operation is based on the same search-by-content function implemented in the search memory. The associative memory contains additional logic at each cell which permits it to perform word-parallel logical, arithmetic, input and output operations, generally in bit-serial fashion. The logical functions to be described are not common to all associative memories. However, they are quite typical, and give a good characterization of the associative memory and its general capabilities.

**The parallel-write operation**

This operation is that of simultaneously modifying selected bit positions at all cells in the associative memory or at a selected subset of the cells. The bit positions to be modified are determined by the contents of the mask register. The contents to be inserted into those bit positions are the settings of the corresponding bit positions in the compare register and are the same for all selected cells. The cells to be modified are selected by the settings of the match flip-flops.

It is the parallel-write operations and their extensions and variants which give the associative memory its capability to process data internally. This capability includes word-parallel arithmetic and logical operations as well as file searching and modification.

Parallel arithmetic operations are generally implemented through special arithmetic logic, including a full adder, at every cell. Only a single adder per cell is required for a bit-serial machine employing a shift register to contain the cell's data.

**Other operations**

It is in the nature of associative processors to have no conventional hardware addressing for randomly accessing
individual cells. Such serial processing as is necessary is usually limited to that of assessing a selected subset of cells in order (say, a set of cells whose contents have matched to a sequence of searches) so that their contents can be output a word at a time. All associative memories have serial-access logic for this purpose.

Other basic operations implemented in the typical associative memory are such auxiliary and support operations as setting or resetting all match flip-flops and ladder flip-flops, and the operation of outputting the contents of the (single) cell having its ladder flip-flop set.

The associative processor

Figure 5-5 shows the organization of a typical associative processor. An associative processor is a system which consists of an associative memory, together with a serial processor for controlling the operation of the associative memory, for controlling input and output, and for interfacing with an operator.

Reconfigurable associative processors

It will be noticed that the associative processor as just defined has no reconfigurable characteristics and no intercommunication network (except for the sequential-access logic). For this reason, it is very limited in its applicability. For example, it is impossible in the system just described to transfer data from one cell to another except by means of the word-serial input and output operations using the sequential-access logic or parallel-write logic. For some file-managing operations this is sufficient. However, in order that the associative processor have wider applicability, a communications network must be added so that data and also control states can be transferred between cells simultaneously (that is, between many pairs of cells or from one cell to many others). A number of techniques for accomplishing this through specialized hardware operating under software control have been devised. Two of these schemes are discussed in the remainder of this subsection.

The STARAN

More effort has been devoted to the design, fabrication and application of the Goodyear STARAN* than to any other associative processor. A number of full scale STARAN systems of various designs have been built, both for general-purpose application studies and for particular designated applications. Some 155 different application functions have been studied for implementation of STARAN, and some 75 of these actually programmed and demonstrated. These deal with such general applications as surveillance systems, sensor signal processing, general scientific applications, communications processing and data management.

This extensive effort has resulted in considerable refinement of the original STARAN concept, and particularly in the addition of a communication network, called the "flip network." The flip network provides the means for performing any desired permutation on the contents of the cells in a 256-cell STARAN module. The STARAN is especially adept in operations in which this fast permutation capability can be effectively utilized, for example, in performing the Fast Fourier Transform (FFT). Other applications include the processing of image data, in which the STARAN, operating as a peripheral device, performs high-speed correls for a serial processor.

The ALAP

The Associative Linear Array Processor (ALAP) [6] uses a different means for obtaining intercell communications from that used by STARAN. The ALAP has a multi-use communication channel, called the chaining channel. Each cell in the ALAP memory array is connected to the next and previous cells in the array by this channel, over which both data and status flags can be transferred. The chaining channel thus organizes the cells into a linear array. Parallel pairwise arithmetic operations can be implemented using the chaining channel.

Applications for which the ALAP has been programmed include radar signal sorting, several radar track-while-scan tasks, and generalized data-retrieval from a structured data base. The ALAP is particularly suited to tasks which are block-oriented in the sense previously defined, and in tasks in which many diverse parallel operations take place simultaneously within the same memory array.

The massively parallel processor (MPP)

A very powerful associative processor with an array of 128 x 128 cells is a recent design of Goodyear Aerospace

* TM, Goodyear Aerospace Corporation, Akron, Ohio.
Corporation. The cells in the MPP are organized into a two-dimensional array by the communication network, with each cell communicating directly with its four nearest neighbors. Each cell can perform arithmetic and logical operations simultaneously with all others, with one operand resident in the cell, and the other either resident in the cell or input from another cell or a common bus. Each cell, in addition to its arithmetic register and arithmetic unit, contains a 1024-bit local memory. All operations within the cell are performed bit-serially.

In the construction of the MPP, almost all components of a sub-array of 2x2 cells are fabricated on a single VLSI wafer, using CMOS/SOS technology. The local memories for the cells are constructed from standard 4 x 1024-bit RAM chips. The clock rate of the array is 10 MHz. Software-controlled fault isolation is provided for each sub-array of 128 x 128 cells.

The square array organization is well suited to the intended application of the MPP, which is the processing of satellite image data. From 100 to 10,000 operations per pixel are required for the processing. This satisfies one of the most important of the basic requirements for applicability for parallel processing previously discussed. The two-dimensional communication network will give the MPP an order-of-magnitude increase in processing speed for matrix operations as compared with the STARAN or ALAP. The latter two machines are limited to an order of n-to-l increase in speed for such operations as compared to serial processors, where \( n \) is the dimension of the (square) matrix. The MPP will have up to an \( n^2 \)-to-1 increase.

6.0 FUTURE TRENDS IN RECONFIGURABLE ARRAY DESIGN

To assess the future of very large, fast reconfigurable arrays as practical data-processing systems, one must examine:

1. the expected technology, its advantages, the desirable characteristics of the circuitry that is to be implemented with it, especially with regard to VLSI;
2. the expected applications that will require very powerful array systems;
3. the limitations on the performance of present array processors with respect to throughput, fault tolerance, amenability to VLSI implementation and suitability to the expected applications.

The VLSI technology

The problems caused by the use of several interconnected wafers in a system when the very fast, new logic technologies are used have been mentioned in Section 4. There is another set of problems which have to do with the nature of VLSI itself.

With regard to VLSI, the problem of most concern to system architects is that of taking advantage of the capability to put so many gates on a single wafer. Conventional serial processors are not altogether suitable for VLSI implementation. One of the main reasons is that it is difficult to factor the logic so that the interfaces between wafers in the system are "clean." This means relatively few pins (not easy when the wafer contains so many gates), similarity in the signal paths to other wafers, and so on. When VLSI fabrication is combined with the very fast gates, the problems multiply. Even interfacing with a conventional RAM when it is on a different wafer from that containing the CPU can be a major problem with 250 MHz clock speeds.

Some parallel array designs have a strong appeal to the VLSI designers because the number of pins on the wafer is low, and is independent of the number of cells on the wafer, and because almost all of the pins are for bit-serial inputs from identical external registers.

On the negative side with regard to the use of array architectures as candidates for VLSI implementation is the fact that these processors are not yet as general-purpose in their applicability as serial processors. It is evident from the standpoint of economics that wide applicability is a prime requirement for a VLSI wafer which will be very expensive to design. Array processors must therefore be designed with such versatility in mind, and an equal effort must be expended on applications analysis to discover new, highly parallel procedures for many common applications. One promising aspect of some parallel array organizations is that the controllers for the arrays can operate at much lower clock rates than the very fast rates for the arrays, since they have much less to do. This means that serial controllers, with their relatively irregular logic organizations, will not have the off-chip delay problems or the other problems associated with the extremely fast logic needed for the arrays.

Future applications for array processors

The problem of versatility with regard to the applicability of highly parallel array processors has been mentioned. Consider the application areas for which the arrays are suitable or superior. These include parallel file-processing tasks, such as radar data processing. They also include real-time text processing, although suitable systems will be very special-purpose designs. Advanced image-processing tasks, which are important as well as challenging, are high on the list. Very advanced file-retrieval applications, such as question-answering tasks involving deductive and inductive inference, are even more promising tasks, now that suitable algorithms and data structures are being developed.

The author's favorite application field is the aforementioned advanced question-answering systems. Such systems may open the door to such capabilities as the translation and interpretation of context-sensitive languages, and thus to the simulation of human reasoning processes. To accomplish even a very small step in such a direction is an objective of immense appeal. The context-sensitivity of the query-answering process seems to be a key factor. The interpretation of context must require, among many other things, a very large data base of possible contexts which can be searched
very frequently and rapidly with complex search criteria. The potential applicability of very powerful associative processors to such a process is evident.

**Limitations in the performance of present designs**

**Throughput limitations**

The $n$-to-$1$ limitation of one-dimensional processor array organizations on matrix operations has already been discussed. The MPP, which has a two-dimensional array, overcomes this limitation for many classes of problems. When the structure of the application (for example, the structure of the real-world system being modeled by the computer) does not match the regular array structure, however, such machines suffer a great loss in efficiency of operation.

**Fault-tolerance limitations**

In most parallel processor arrays having a regular communication network, there is a tradeoff between the order of parallel operation and the level at which fault tolerance can be implemented. With one-dimensional arrays, fault tolerance can be implemented at the level of the single cell. With the MPP, an entire column of processing elements must be disabled if a single cell fails. This is not appealing in concept as a remedy, and is a most difficult problem with regard to effective fault tolerance.

**Limitations with regard to VLSI**

The dimensionality of the processor array is of great importance with regard to amenability to VLSI fabrication. The PEPE, STARAN and ALAP are well suited in this regard. The number of pins on a wafer is independent, or nearly so, for such zero and one-dimensional arrays. For two-dimensional arrays in general, this is not the case. If many cells are fabricated on a wafer, the number of pins will increase as the square root of the number of cells if the array is two-dimensional. All of the cells on the edge of the wafer must have external connections, else processing throughput will suffer because of bottlenecks in the interwafer communication.

A possible solution to this problem is to construct three-dimensional arrays by putting a set of "two-dimensional" wafers together in a stack like pancakes, with vertical communication channels fabricated directly on pads on the wafers. This may actually be practical at some time in the future, when new technology can be devised, but does not appear to be a near-term solution.

**Recommendations, predictions and suggestions**

For a solution to many of the problems just addressed, one is tempted to look at the Holland machine concept. This machine has a two-dimensional array organization. However, in this concept, the communication paths taken by the data in operation are data-determined, rather than location-determined. The Holland machine has serious drawbacks as a practical system. Nevertheless, it does not suffer from some of the limitations with regard to fault tolerance and pin count if it is properly implemented. Perhaps the user should not expect optimal efficiency in the utilization of the hardware. There can be cost/efficiency tradeoffs which can achieve equal results in throughput, while still retaining some of the advantages of the multi-dimensional array. It is these advantages which should be a prime objective of future efforts in parallel array design.

**REFERENCES**