ARCHITECTURE OF A MASSIVELY PARALLEL PROCESSOR*

Dr. Kenneth E. Batcher
Digital Technology Department

Goodyear Aerospace Corporation
Akron, Ohio 44315

ABSTRACT

The massively parallel processor (MPP) system is designed to process satellite imagery at high rates. A large number (16,384) of processing elements (PE's) are configured in a square array. For optimum performance on operands of arbitrary length, processing is performed in a bit-serial manner. On 8-bit integer data, addition can occur at 6553 million operations per second (MOPS) and multiplication at 1861 MOPS. On 32-bit floating-point data, addition can occur at 430 MOPS and multiplication at 216 MOPS.

INTRODUCTION

In this decade, NASA will orbit imaging sensors that can generate data at rates up to 10^{13} bits per day. A variety of image processing tasks such as geometric correction, correlation, image registration, feature selection, multispectral classification, and area measurement are required to extract useful information from this mass of data. The expected workload is between 10^9 and 10^{10} operations per second.

In 1971, NASA Goddard Space Flight Center initiated a program to develop ultra high-speed image processing systems capable of processing this workload. These systems use thousands of processing elements operating simultaneously (massive parallelism) to achieve their speed. They exploit the fact that the typical satellite image contains millions of picture elements (pixels) that can generally be processed at the same time.

In December 1979, NASA Goddard awarded a contract to Goodyear Aerospace to construct a massively parallel processor (MPP) to be delivered in the first quarter of 1982. The basic elements of the MPP architecture were developed at NASA Goddard. This paper presents the architecture of the MPP system. The major components are shown in Figure 1. The array unit (ARU) processes arrays of data at high speed and is controlled by the array control unit (ACU), which also performs scalar arithmetic. The program and data management unit (PDMU) controls the overall flow of data and programs through the system and handles certain ancillary tasks such as program development and diagnostics. The PDMU-ARU I/O registers buffer and reorder data between the ARU, PDMU, and external (host) computer.

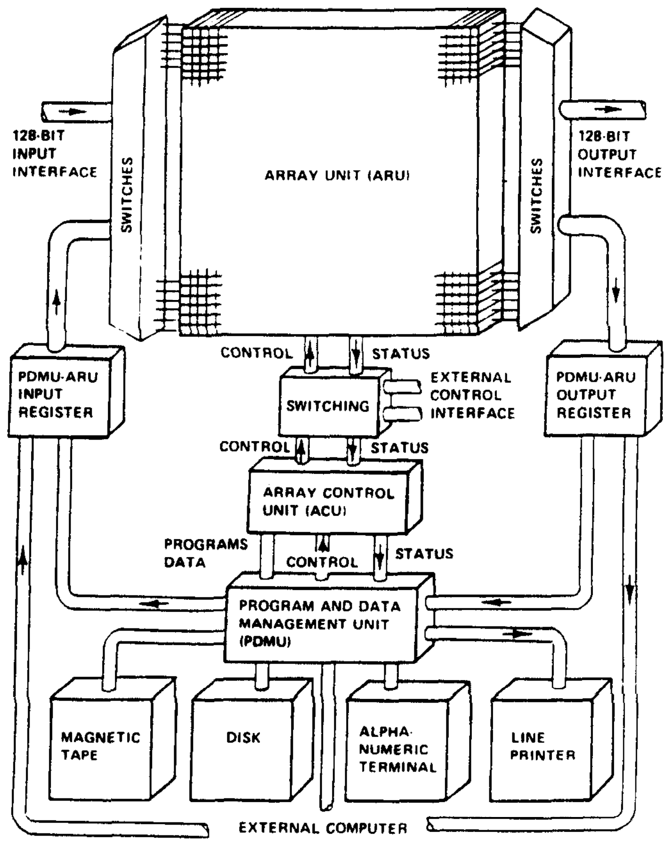

Figure 1 - Block Diagram of Massively Parallel Processor (MPP)

*This work was partially funded by NASA Goddard Space Flight Center under Contracts NAS 5-25392 and NAS 5-25942.

ARRAY UNIT

Logically, the array unit (ARU) contains 16,384 processing elements (PE's) organized as a 128 by 128 square. Physically, the ARU has an extra 128 by 4 rectangle of PE's that is used to reconfigure the ARU when a PE fault is detected. The PE's are bit-serial processors for efficiently processing operands of any length. The basic clock rate is 10 megahertz. With 16,384 PE's operating in parallel, the ARU has a very high processing speed (see Table I). Despite the bit-serial nature of the PE's, even the floating-point speeds compare favorably with several fast number crunchers.

TABLE I - SPEED OF TYPICAL OPERATIONS

Operations	Speed (MOPS)*
Addition of arrays	
8-bit integers (9-bit sum)	6553
12-bit integers (13-bit sum)	4428
32-bit floating-point numbers	430
Multiplication of arrays (element by element)	
8-bit integers (16-bit product)	1861
12-bit integers (24-bit product)	910
32-bit floating-point numbers	216
Multiplication of array by scalar	
8-bit integers (16-bit product)	2340
12-bit integers (24-bit product)	1260
32-bit floating-point numbers	373

* Million operations per second.

Routing Topology

Each PE in the 128 by 128 square communicates with its nearest neighbor; up, down, right, and left - the same routing topology used in ILLIAC IV and some other array processors. Alternative routing topologies such as the flip network[1] or one of its equivalents[2] were investigated. They have the ability to shift data over many PE's in one step and allow data to be accessed in many different directions[3]. Certain paths in the alternative topologies have long runs that complicate their layout and limit their cycle rate. When the number of PE's interconnected is only 256 as in the STARAN* computer, this is no problem; when 16,384 PE's are interconnected, it is a severe problem.

Most of the expected workload does not use the routing flexibility of the alternative topologies. The ability to access data in different directions is important when arrays of data are input and output; it can be used to reorient the arrays between the bit-plane format of the ARU and the pixel format of the outside world.

These considerations lead to the conclusion that the ARU should have a two-dimensional nearest-neighbor routing topology such as ILLIAC IV since it is easy to implement and matches the two-dimensional nature of satellite imagery. The problem of reformatting I/O data is best handled in I/O buffers that may employ the alternative routing topologies.

Around the edges of the 128 by 128 array of PE's, the edges can be left open (e.g., a row of zeros can be entered along the left edge when routing data to the right) or the opposite edges connected. Cases were found where open edges were preferred and other cases where connected edges were preferred. It was decided to make edge-connectivity a programmable function. A topology-register in the array control unit defines the connections between opposite edges of the PE array. The top and bottom edges can either be connected or left open. The connectivity between the left and right edges has four states: open (no connection); cylindrical (connect the left PE of each row to the right PE of the same row); open spiral (for $1 \leq n \leq 127$, connect the left PE of row n to the right PE of row n-1); and closed spiral (like the open spiral, but also connect the left PE of row 0 to the right PE of row 127). The spiral modes connect the 16,384 PE's together in one long linear array.

Redundancy

The ARU includes some redundancy so that a faulty PE can be bypassed. Redundancy can be added to a two-dimensional array of PE's by adding an extra column (or row) of PE's and inserting bypass gates in the routing network. When a faulty PE is discovered, one disables the whole column containing the faulty PE and joins the columns on either side of it with the bypass gates.

The PE's in the ARU are implemented with two-row by four-column VLSI chips; thus, it is more convenient to add four redundant columns of PE's and bypass four columns at a time. The PE array has 128 rows and 132 columns. It is divided into 33 groups, with each group containing 128 rows and four columns of PE's. Each group has an independent group-disable control line from the ACU. When a group is disabled, all its outputs are disabled and the groups on either side of it are joined together with 128 bypass gates in the routing network.

When there is no faulty PE, an arbitrary group is disabled so that the size of the logical array is always 128 by 128. Application programs are not aware of which group is disabled and need not be modified when the disabled group is changed. They always use the logical address of a PE to access PE-dependent data. The logical address of a PE is a pair of 7-bit numbers, X and Y, showing its position in the logical array of enabled PE's. A simple routine executed in 27 microseconds will load the memory of each PE with its logical address. No attempt to preserve PE data is made when a faulty PE is discovered since the data in the faulty PE cannot be trusted.

*Trademark, Goodyear Aerospace Corporation, Akron, Ohio.

Recovery is accomplished by restarting the application program from the last checkpoint or from the beginning.

Bit-Serial Processing

The elements in the arrays being processed have a wide variety of lengths. A spectral band of an input pixel may have a resolution of 6 to 12 bits. Intermediate results can have any length from 6 to more than 30 bits. Single-bit flag arrays are generated when pixels are classified. Some computations may be performed in floating-point. Thus, the PE's should be able to process operands of any length efficiently.

Conventional computers typically use bit-parallel arithmetic units with certain fixed-word lengths such as 8, 16, or 32 bits. Operands of odd lengths are extended to fit the standard word sizes of the machine. Some of the hardware in the memory and the arithmetic unit is wasted storing and processing the extensions.

Bit-serial processors process operands bit by bit and can handle operands of any length without any wasted hardware. Their slower speed can be counteracted by using a multitude of them and processing many operands in parallel. Because of the wide variety of operand lengths in the expected workload, bit-serial processors are more efficient in the use of hardware than bit-parallel processors.

Processing Elements

The initial MPP design had PE's using downshifting binary counters for arithmetic[4]. The PE design was modified to use a full-adder and shift-register combination for arithmetic. The modified design performs the basic arithmetic operations faster. Each of the 16,896 PE's has 6 one-bit registers (A,B,C,G,P, and S), a shift-register with a programmable length, a random-access memory, a data-bus (D), a full-adder, and some combinatorial logic (see Figure 2). The basic cycle time of the PE is 100 nanoseconds.

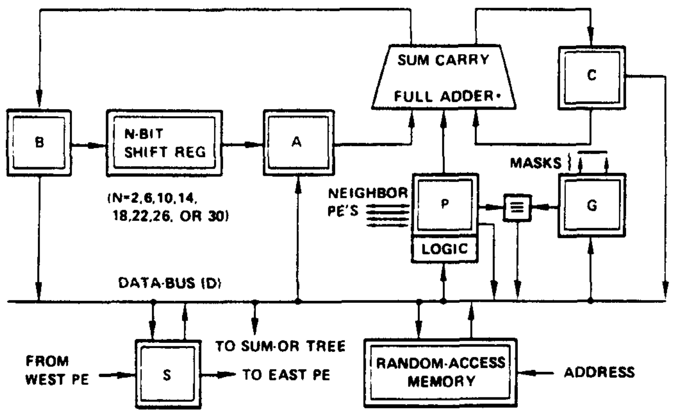

Figure 2 - One Processing Element

Logic and Routing. The P-register is used for logic and routing operations. A logic operation combines the state of the P-register and the state of the data-bus (D) to form the new state of the P-register. All 16 Boolean functions of the two variables, P and D, are implemented ($P \cdot D$, $P \vee D$, $P \oplus D$, $\bar{P} \cdot D$, $\bar{P}$, D, $\bar{D}$, etc.). A routing operation shifts the state of the P-register into the P-register of a neighboring PE (up, down, right, or left).

Arithmetic. The full adder; shift register; and registers A, B, and C are used for bit-serial arithmetic operations. To add two operands, the bits of one operand are put into the A-register, one at a time, least-significant-bit (LSB) first; corresponding bits of the other operand are put into the P-register; the full adder adds the bits in A and P to the carry bits in the C-register to form the sum and carry bits; each carry bit is stored in C to be added in the next cycle; and each sum bit is stored in B. The sum formed in B can be stored in the random-access memory and/or in the shift register. Two's-complement subtraction is performed by adding the one's-complement of the operand in P to the operand in A and setting the initial carry bit in C to 1 instead of 0.

Multiplication is a series of addition steps where the partial product is recirculated through the shift register and registers A and B. Appropriate multiples of the multiplicand are formed in P and added to the partial product as it recirculates. Division is performed with a nonrestoring division algorithm. The partial dividend is recirculated through the shift register and registers A and B while the divisor or its complement is formed in P and added to it.

Floating-point addition compares exponents; places the fraction of the operand with the least exponent in the shift register; shifts it to the right to align it with the other fraction; puts the sum of the fractions in the shift register; and normalizes the sum. Floating-point multiplication is a multiplication of the fractions, a normalization of the product, and an addition of the exponents.

Masking. The G-register can hold a mask bit that can control the activity of the PE. Unmasked logic, routing, and arithmetic operations are performed in all PE's. Masked operations are only performed in those PE's where the G-register equals 1.

Input/Output. The S-register is used to input and output ARU data. While the PE's are processing data in the random-access memories, columns of input data are shifted into the left side of the ARU (Figure 1) and through the S-registers (Figure 2) until a plane of 16,384 bits is loaded. The input plane is then stored in the random-access memories in one 100-nanosecond cycle by interrupting the processing momentarily in all PE's and moving the S-register values to the memory elements. Planes of data are output by moving them from the memory elements to the S-registers and then shifting them out column by column through the right side of the ARU. The shift rate is 10 megahertz; thus, up to 160 megabytes per second can be transferred through the ARU I/O ports. Processing is interrupted for 100 nanoseconds for each bit plane of 16,384 bits transferred - less than one percent of the time.

Storage. The random-access memory stores 1024 bits per PE. Standard RAM integrated circuits make it easy to expand storage as advances occur in solid-state memory technology.

Parity error detection is used to find memory faults. A parity bit is added to the eight data bits of each 2 by 4 subarray of PE's. Parity bits are generated and stored for each memory write cycle and checked when the memories are read. A parity error sets an error flip-flop associated with the 2 by 4 subarray. A tree of logic elements gives the ACU an inclusive-or of all error flip-flops (after some delay). By operating the group-disable control lines, the ACU can locate the group containing the error and disable it.

Sum-Or Tree. The data bus states of all 16,384 enabled PE's are combined in a tree of inclusive-or elements called the sum-or tree. The output of this tree is fed to the ACU and used in certain operations such as finding the maximum or minimum value of an array in the ARU.

Packaging

Standard 4 by 1024-bit RAM elements are used for the PE memories. All other components of a 2 by 4 subarray of PE's are packaged on a custom VLSI CMOS/SOS chip. The VLSI chip also contains the parity tree and the bypass gates for the subarray.

Each 8-1/2-inch by 14-inch printed circuit board contains 192 PE's in an 8 by 24 array. A board contains 24 VLSI chips, 54 memory elements, and some support circuitry. Sixteen boards make up an array slice of 128 by 24 PE's.

Five array slices (80 boards) make up the bulk of the ARU (128 by 120 PE's). The remaining 12 PE columns are packaged on 16 I/O-processor boards, which also contain the topology switches, I/O switches, and PDMU-ARU I/O registers. The 96 boards of the ARU are packaged in one cabinet (the leftmost cabinet in Figure 3). Forced-air cooling is used.

ARRAY CONTROL UNIT

Like the control units of other parallel processors, the array control unit (ACU) performs scalar arithmetic and controls the PE's. It has three sections that operate in parallel (see Figure 4): PE Control, I/O Control, and Main Control. PE Control performs all array arithmetic of the application program. I/O Control manages the flow of data in and out of the ARU. Main Control performs all scalar arithmetic of the application program. This arrangement allows array arithmetic, scalar arithmetic, and input/output to be overlapped for minimum execution time.

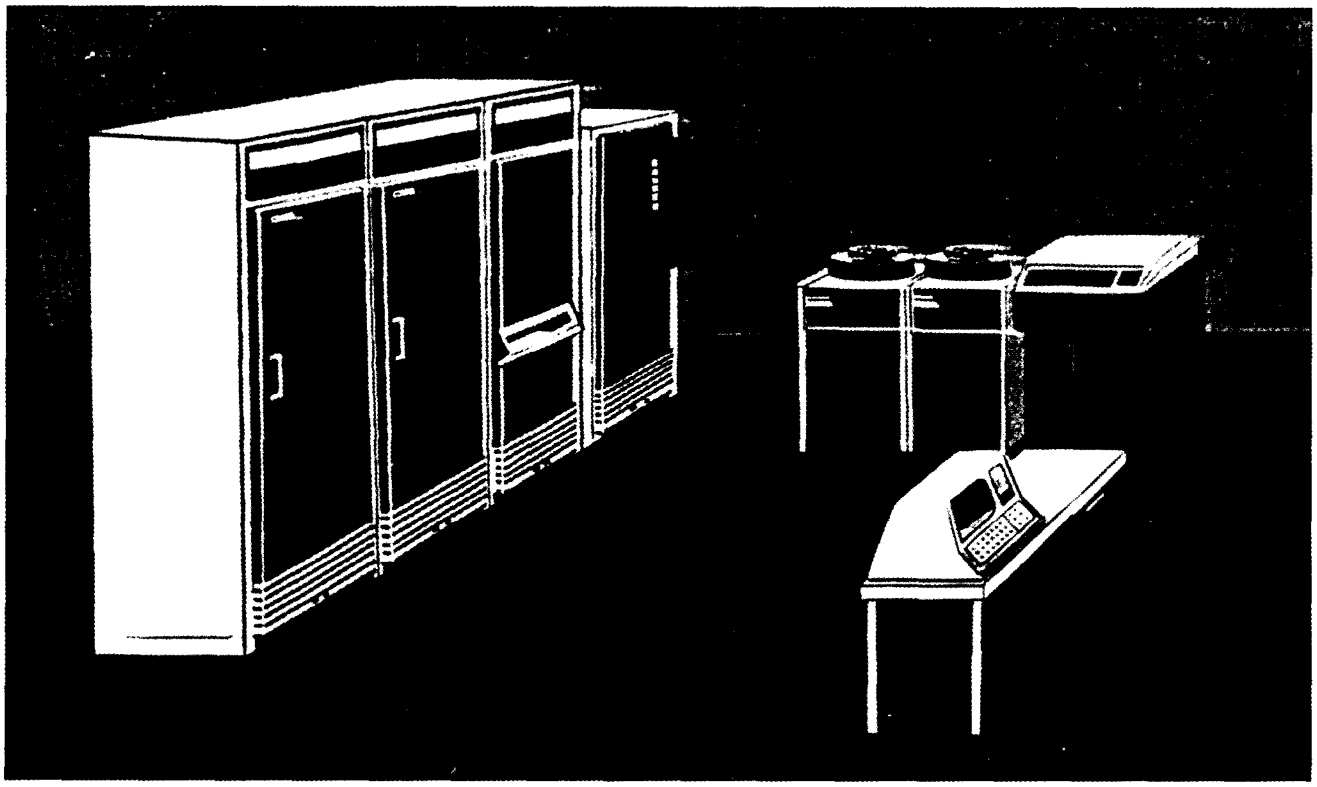

Figure 3 - MPP System Configuration

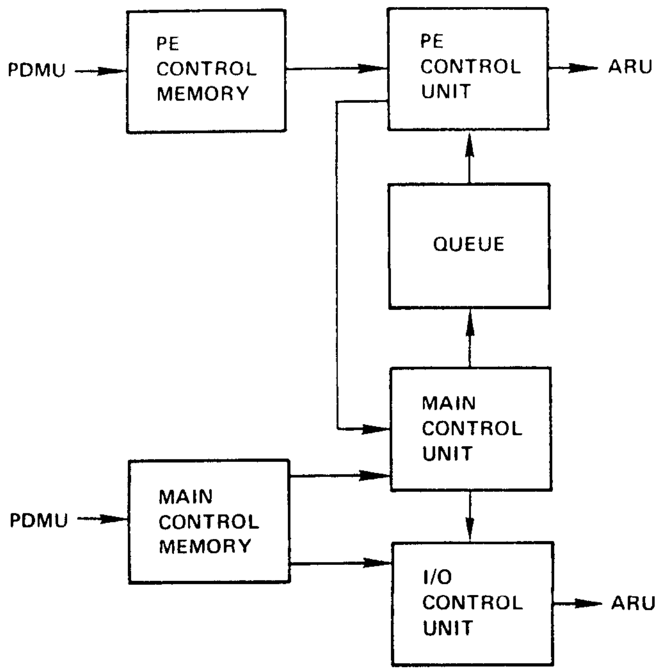

Figure 4 - Block Diagram of Array Control Unit (ACU)

PE Control

PE Control generates all ARU control signals except those associated with I/O. It contains a 64-bit common register to hold scalars and eight 16-bit index registers to hold the addresses of bit planes in the PE memory elements, to count loop executions, and to hold the index of a bit in the common register. PE Control reads 64-bit-wide microinstructions from PE Control memory. Most instructions are read and executed in 100 nanoseconds. One instruction can perform several PE operations, manipulate any number of index registers, and branch conditionally. This reduces overhead significantly so that little, if any, PE processing power is wasted.

PE Control memory contains a number of system routines and user-written routines to operate on arrays of data in the ARU. The routines include both array-to-array and scalar-to-array arithmetic operations. A queue between PE Control and Main Control queues up to 7 calls to the PE Control routines. Each call contains up to 8 initial index-register values and up to 64 bits of scalar information. Some routines extract scalar information from the ARU (such as a maximum value) and return it to Main Control.

I/O Control

I/O Control shifts the ARU S-registers, manages the flow of information in and out of the ARU ports, and interrupts PE Control momentarily to transfer data between the S-registers and buffer areas in the PE memory elements. Once initiated by Main Control or the PDMU, I/O Control can chain through a number of I/O commands. It reads the commands from Main Control memory.

Main Control

Main Control is a fast scalar processor. It reads and executes the application program in the Main Control memory. It performs all scalar arithmetic itself and places all array arithmetic operations on the PE Control call queue. It contains 16 general-purpose registers, three registers to control the ARU group-disable lines, 13 registers associated with the call queue, 12 registers to receive scalars from PE Control, and four registers to monitor and control the status of PE Control and I/O Control.

PROGRAM AND DATA MANAGEMENT UNIT

The program and data management unit (PDMU) controls the overall flow of programs and data in the system (Figure 1). Control is from an alphanumeric terminal or from an external (host) computer. The PDMU is a minicomputer (DEC PDP-11) with custom interfaces to the ACU control memories and registers and to the wide I/O ports of the ARU. The operating system is DEC's RSX-11M real-time multiprogramming system.

The PDMU also executes the MPP program-development software package. The package includes a PE control assembler to develop routines for PE Control; a main assembler to develop application programs executing in Main Control; a linker to form load modules for the ACU; and a control and debug module that loads programs into the ACU, controls their execution, and facilitates debugging.

PDMU-ARU I/O REGISTERS

The PDMU-ARU I/O registers reside between the wide I/O ports of the ARU and the PDMU. They also have a port to an external (host) computer. Besides acting as buffers for ARU data being input and output, the registers reorder arrays of data.

Satellite imagery is normally stored in pixel order in the PDMU and other conventional computers. That is, line 1 pixel 1 followed by line 1 pixel 2, etc., followed by the pixels of line 2, line 3, etc. The imagery might be band-interleaved (all spectral bands of a pixel stored together) or band-sequential (band 1 of all pixels followed by band 2 of all pixels, etc.).

Arrays of data are transferred through the ARU ports in bit-sequential order. That is, the most- (or least-) significant bit of 16,384 elements followed by the next bit of 16,384 elements, etc. Reordering is required to fit the normal order of satellite imagery in the PDMU or the host. Thus, the PDMU-ARU I/O registers are given a reordering capability.

The registers use a common 2^{19} bit multi-dimensional-access (MDA) memory[3] to buffer and reorder arrays of data. Each of the three memory ports (ARU, PDMU, and external computer) can read and write data in the MDA memory in a different direction. Data arrays up to 2^{19} bits in size can be reordered by writing them into the MDA memory in one direction and reading them out in a different direction.

The ARU ports transfer data to and from the I/O registers at a 160-megabyte-per-second rate (128 bits every 100 nanoseconds). The external computer ports transfer data at rates up to six megabytes per second. The PDMU transfer rate is limited by the PDMU peripherals. The fastest peripheral is the disk system, which has a 806-kilobyte-per-second rate.

EXTERNAL COMPUTER INTERFACE

The MPP, to be delivered to NASA, will use a DEC VAX-11/780 computer as a host. The interface to the host has two links: a high-speed data link and a control link.

The high-speed data link connects the PDMU-ARU I/O registers of the MPP to a DR-780 high-speed user interface of the VAX-11/780. Data can be transferred at the rate of the DR-780 (six megabytes per second).

The control link is DEC's standard DECNET link between a PDP-11 and a VAX-11/780. The DECNET hardware and software let VAX users transfer programs and program requests to the MPP.

CONCLUSION

The massively parallel processor is a ultra high-speed processor designed to perform a variety of image processing tasks on satellite imagery. It may be used in other applications where large arrays of data must be processed quickly. The MPP uses a large number of simple PE's to process arrays of data in a bit-plane or bit-slice mode. A number of papers in the literature describe other systems using the same processing mode[5-10].

REFERENCES

1. Batcher, K.E.: The Flip Network in Staran; 1976 International Conference on Parallel Processing, pp. 65-71.
2. Siegel, H.J., and Smith, S.D.: Study of Multistage SIMD Interconnection Networks; Fifth Annual Symposium on Computer Architecture, pp. 223-229, April 1978.
3. Batcher, K.E.: The Multi-Dimensional-Access Memory in STARAN; IEEE Transactions on Computers, vol. C-26, pp. 174-177, February 1977.
4. Fung, L-W: A Massively Parallel Processing Computer; High-Speed Computer and Algorithm Organization (D.J. Kuck, et al, ed.), pp. 203-204, Academic Press, 1977.
5. Slotnick, D.L.; Borck, W.C.; and McReynolds, R.C.: The SOLOMON Computer; 1962 Fall Joint Computer Conference, pp. 97-107.
6. Shooman, W.: Orthogonal Processing; Parallel Processor Systems, Technologies, and Applications (L.C. Hobbs, et al, ed.) pp. 297-308, Spartan Books, 1970.
7. Handler, W.: A Unified Associative and Von-Neumann Processor - EGPP and EGPP Array; Lecture Notes in Computer Science, vol. 24 - Parallel Processing, pp. 97-99, Springer-Verlag, 1975.
8. Reddaway, S.F.: DAP - A Distributed Array Processor; First Annual Symposium on Computer Architecture, December 1973.
9. Flanders, P.M.; Hunt, D.J.; Reddaway, S.F.; and Parkinson, D.: Efficient High-Speed Computing with the Distributed Array Processor; High-Speed Computer and Algorithm Organization (D.J. Kuck, et al, ed.), pp. 113-128, Academic Press, 1977.
10. Batcher, K.E.: STARAN Parallel Processor System Hardware; 1974 National Computer Conference, AFIPS Conference Proceedings, vol. 43, pp. 405-410.