MPP: a supersystem for satellite image processing

by KENNETH E. BATCHER
Goodyear Aerospace Corporation
Akron, Ohio

ABSTRACT

In 1971 NASA Goddard Space Flight Center initiated a program to develop high-speed image processing systems. These systems use thousands of processing elements (PE's) operating simultaneously to achieve their speed (massive parallelism). A typical satellite image contains millions of picture elements (pixels) that can generally be processed in parallel. In 1979 a contract was awarded to construct a massively parallel processor (MPP) to be delivered in 1982. The processor has 16,896 PE's arranged in a 128-row by 132-column rectangular array. The PE's are in the array unit (Figure 1). Other major blocks in the massively parallel processor are the array control unit, the staging memory, the program and data management unit, and the interface to a host computer.

ARRAY UNIT

Logically, the array unit contains 16,384 PE's arranged in a 128-row by 128-column square array. Physically, the array unit contains an extra 128-row by 4-column rectangle of PE's for redundancy. Each PE communicates with its four nearest neighbors: north, south, east, and west. Each PE is a bit-serial processor. With a ten-megahertz clock rate and 16,384 PE's operating in parallel the system has a very high processing rate. Each PE can read two 16-bit integers, generate their sum, and store the 17-bit sum in 49 clock cycles, so 16,384 additions are performed in less than five microseconds (more than 3000 MOPS). Floating-point operations are performed at a fast rate even though they are not particularly suited for bit-serial processing. Many different floating-point formats are possible. With a 32-bit format (1-bit sign, 7-bit base-16 exponent and 24-bit fraction) floating-point addition is better than 400 MOPS and multiplication is better than 200 MOPS.

Array Topology

The major application of the massively parallel processor is image processing. Since most of the processing is conducted between neighboring pixels it is natural to connect the thousands of PE's together in a square array with each PE communicating with its nearest neighbors. We investigated the use of other interconnection networks like the multistage SIMD interconnection networks,[1] but with over 16,000 PE's they become unwieldy. The layout of a square array is very simple with no long runs to slow down the transfer rate.

Certain image processing operations like the Fast Fourier Transform (FFT) require communication between pixels or points located far apart in the image. If we store one point in each PE, then the routing time would be severe in a square-array topology. But this is not the best way to do FFT's on the MPP. Each PE can store several points in its random access memory, so the number of PE's required to do an FFT can be reduced to a small compact subarray of the array unit. The processing power of the other PE's is not wasted, since when we want to do one FFT we usually want to do many FFT's so we can divide the array unit up into many compact subarrays, each performing one FFT. For example, suppose we want to do many 5120-point FFT's. Ten points can be packed into each PE, so each FFT can be performed in a 16-row by 32-column subarray of the array unit. Thirty-two such subarrays can perform 32 FFT's in parallel. The longest communication path in each FFT is half the width of the subarray (16 columns), so the routing time can be reduced to a fraction of the computation time.

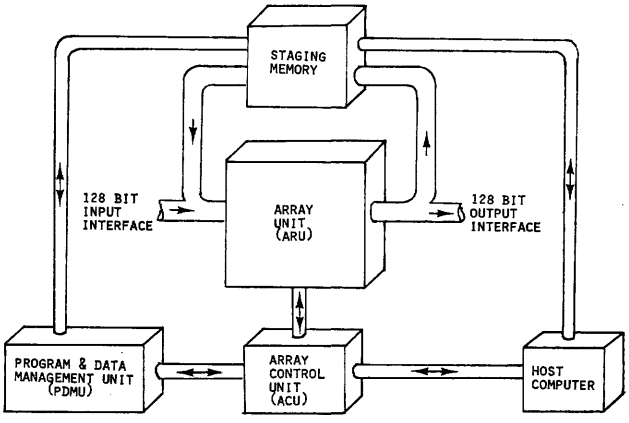

Figure 1—Block diagram of the massively parallel processor

One may ask the question of how the data can be input and output effectively, especially when it has a peculiar layout as in the FFT example. A 5120-point FFT is most easily performed by combining 1024 five-point FFT's with five 1024-point FFT's where the position of any point is a function of its index modulo 5 and 1024. The 5120 points of one FFT have a scrambled layout. The permutations required are akin to the permutations required to change a data array from an item format to a bit-slice format. Some kind of buffer memory will be required in the array unit I/O path to convert data arrays to and from the bit-slice format; if it is properly designed the same buffer memory could perform other permutations as well, such as those required by the 5120-point FFT example.

Thus, in the massively parallel processor we use a simple square array topology in the array unit and insert a buffer memory (the staging memory) in its I/O path to perform the permutations required by particular application programs. The staging memory transforms the bit-serial format of the array unit to the item format of the outside world.

Given a square array with 128 rows and 128 columns what do we do around the edges? Some application programs would like to see a planar topology. Other programs would like to see a cylindrical topology where the PE's on the north edge connect to PE's on the south edge. Also, some programs would rather have the 16,384 PE's connected in one long linear string rather than in a 128 by 128 plane. Thus, the edge connections should be a programmable function.

A topology register is included in the array control unit to allow programming of the edge connections. Between the north and south edges of the array unit one can either stitch them together to make the array look like a cylinder or separate them to make the array look like a plane. Similarly, the east and west edges can independently be stitched together or separated (if both pairs of edges are stitched together the

array looks like a torus). When the east and west edges are stitched together, one can either stitch corresponding rows together or slide the stitching by one row so the west PE of row i communicates with the east PE of row $i + 1$. If one slides the stitching, the rows are connected together in spiral fashion so the array of PE's looks like a long linear string.

Redundancy

One advantage of the rectangular connection network is the easy way it allows faulty PE's to be bypassed. When a faulty PE is discovered, one bypasses all the PE's in its column (or row) so the topology is not disturbed. To add redundancy to the array unit we implement more than 128 columns and insert bypass gates in the east-west routing paths. The array is reduced to 128 columns logically by bypassing some columns. If a faulty PE is discovered we bypass its column and use one of the spare columns instead. Logically, the array will still have 128 columns. Of course, the physical position of many data items will be shifted when the bypassed columns are shifted; but this presents no problem if we don't try to save the data when a fault is discovered. Since the discovery of a fault usually implies the presence of faulty data in the faulty PE and/or its neighbors we should not try to save the data anyway. After the array unit is reconfigured, recovery is accomplished by restarting the application program from the last checkpoint.

We could just add one redundant column of PE's and bypass the 129 columns individually. Instead we divide the array up into 32 four-column groups and add a redundant four-column group so only 33 sets of bypass gates are required instead of 129. When a faulty PE is discovered we bypass all PE's in its four-column group. All outputs from the group are disabled and the east-west routing paths of its two neighboring groups are stitched together. The redundancy of 3 percent is a small price to pay for the ability to reconfigure around any single faulty PE. The scheme does not handle the case of multiple PE's failing, but the probability of this event within a reasonable service interval is miniscule.

Processing Elements

Each PE is a bit-serial element. Initially the PE's had downshifting binary counters for arithmetic.[2, 3] The PE design was modified to use a full adder and a shift register for arithmetic. Each PE has six 1-bit registers (A, B, C, G, P, and S), a shift register with programmable length, a random-access memory, a data bus (D), a full adder, and some combinatorial logic (Fig. 2). The nominal clock rate of the PE's is 10 megahertz. In each clock cycle all PE's perform the same operations on their respective data streams (except where masked). The basic PE operations are microsteps of the array instruction set. The control signals come from the PE control unit of the array control unit, which reads the microcode from a writable control store. As long as there are no conflicts many PE operations can be combined into one clock cycle.

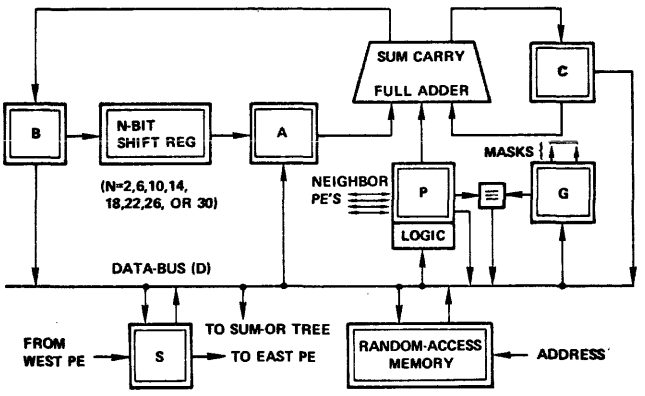

Figure 2—One processing element

Data-bus source selection

The source of the data bus can be the state of the B-, C-, P-, or S-register, the state of a selected bit from the random-access memory, or the output of the equivalence function between P and G ($P \equiv G$ equals 1 if and only if P and G are in the same state). The data-bus state (D) feeds a number of other parts of the PE.

Logic and routing

The P-register is used for logic and routing operations. A logic operation combines the state of the P-register with the state of the data bus (D) to form the new state of the P-register. Any of the 16 Boolean functions of P and D can be selected. A routing operation reads the state of the P-register in a neighboring PE (north, south, east, or west) and stores the state in the P-register. When routing occurs, the 128 by 128 plane of P-registers is shifted synchronously in any of the four cardinal directions.

A logic or routing operation can be unmasked or masked. An unmasked operation is performed in all 16,384 PE's. A masked operation is performed in only those PE's where $G = 1$—the P-register is not disturbed in those PE's where $G = 0$.

Arithmetic

The full adder, the shift register and registers A, B, and C are used for bit-serial arithmetic operations. A full-add operation sums the bits in the A-, P-, and C- registers to form a 2-bit sum that is placed in the C- and B- registers. A half-add operation is similar except that only the bits in registers A and C contribute to the sum.

The shift register has a programmable length. Its length can be set to 2, 6, 10, 14, 18, 22, 26, or 30 bits. A shift operation shifts the register one place to the right, with the state of the B-register entering at the left end of the shift register. If register A is shifted simultaneously, it reads the rightmost bit in the shift register. An operand of length $4n$, where n is an

integer from 1 to 8, can be recirculated around the path formed by register B, the shift register, register A, and the full adder; the shift register length is set to $4n$-2.

Register A has three operations: clear A, load A with the data-bus state D, or load A with the rightmost bit in the shift register (shift A). Register C receives the carry bit in full-add and half-add operations and has two other operations: clear C and set C.

These micro arithmetic operations are combined to perform the array arithmetic instruction set. Addition of two arrays of n-bit integers is performed with each PE treating one pair of integers. Corresponding bits of the integers are fed to the P- and A- registers, respectively, starting with the lowest order bits. They are added with full-add operations with the carry bits recirculating through register C and the sum bits being formed in register B and stored back in the random access memory. It requires $3n + 1$ cycles to read the two n-bit integers from memory and store the $(n + 1)$-bit sum back into memory. Subtraction is performed similarly except that the 1's complement of the subtrahend is loaded into the P-register in place of its true value. Two's-complement subtraction is done by initializing the C-register to 1 instead of 0.

The result of an arithmetic operation can be sent to the shift register instead of storing it to memory. Multiplication is performed by recirculating the partial product through the shift register and adding the multiplicand to it with appropriate shifts. A multiplier bit in the G-register controls the loading of the P-register. Multiplication of an array of n-bit integers by corresponding elements of an array of m-bit integers to produce an array of $(m + n)$-bit integers requires $(m - 1)p + 2(m + n)$ cycles where p is a multiple of 4 equal to n, $n + 1$, $n + 2$, or $n + 3$.

Division uses a nonrestoring algorithm where the partial dividend is recirculated through the shift register and the divisor or its complement is added for each quotient bit.

Floating-point multiplication is an addition of the exponents and a rounded multiplication of the fractions. Floating-point addition is a comparison of the values followed by an alignment of the fractions, addition of the fractions and then a normalization of the result.

Other PE operations

Other PE operations include loading the G-register from the data bus, writing the data bus to a selected bit of the random access memory, loading the S-register from the data bus, feeding the sum-or tree from the data bus, and clearing the memory parity error indicator.

The sum-or tree is a tree of inclusive-or gates with inputs from all 16,384 PE's. The output is fed to the array control unit, which can test and store the result. The sum-or tree is used in maximum value and minimum value searches and in other operations where it is necessary to get a global result from the set of PE's.

Input-output

The S-register in all PE's is used for input and output of array data. Columns of input data are shifted into the S-registers at the west edge of the array unit and shifted across the array until all 16,384 S-registers are loaded. Then the PE processing is interrupted for one machine cycle while the S-register plane is transferred to a selected plane of the random access memories. S-register shifting can run at a 10-megahertz rate, so data can be input at a rate of 160 megabytes per second (128 bits every 100 nanoseconds). Note that PE processing is only interrupted once every 128 columns, or less than 1 percent of the time.

Data output is similar. The PE processing is interrupted for one cycle and a plane of data is transferred from the random access memories to the S-registers. Processing resumes while the output plane is shifted across the array to the east edge, where it is output column by column. Each column is 128 bits long and can be shifted out at a 10-megahertz rate, so the output rate is also 160 megabytes per second. Note that while an output plane is being shifted out an input plane can be shifted into the array unit; so input and output can proceed simultaneously.

At first glance an I/O rate of 320-megabytes per second (160 in and 160 out) would seem to be more than adequate. But the processing rate is so high that some applications may still become I/O bound. When such an application arises (and when fast enough peripherals are available), the array unit I/O scheme can be modified to input and output data at several places in the array instead of just at the east and west edges.

Random access memories

Each PE has a random access memory storing 1024 bits. The address lines of all PE's are tied together so the memories are accessed by planes with one bit of a plane accessed by each PE. Four PE's share one 1K-by-4 RAM chip with an access time of about 50 nanoseconds. The address bus can be expanded up to 16 address lines, so PE memory can be expanded to 65,536 bits per PE or 128 megabytes total. The array unit clock system has enough flexibility to accommodate a wide variety of memory speeds.

Packaging

The PE random access memories use standard RAM integrated circuits. All other components of eight PE's are packaged on a custom VLSI chip. The chip holds a 2-row by 4-column subarray of PE's and 2,112 such chips are used in the array unit. The chip pinout is 52 pins and the complexity is about 8000 transistors.

A 16-row by 12-column subarray of 192 PE's is packaged on one 22-cm by 36-cm printed circuit board. The board contains 24 VLSI chips, 54 memory elements (48 for data plus 6 for parity) and some support circuitry. Eleven boards make up an array slice of 16 rows by 132 columns. Eight array slices (88 boards) make up the array unit and eight other boards hold the topology switches, the control signal fan-out and other support circuitry. The 96 boards are packaged in one cabinet and cooled by forced air.

ARRAY CONTROL UNIT

The array control unit has three subunits: the PE control unit to control processing in the array unit PE's, the I/O control unit to manage the flow of input/output data through the array unit, and the main control unit, which runs the application program, performs any necessary scalar processing, and controls the other two subunits (Fig. 3). This division of responsibility allows array processing, scalar processing and I/O to proceed simultaneously. A queue between the main control unit and the PE control unit can hold up to 16 calls to array processing routines.

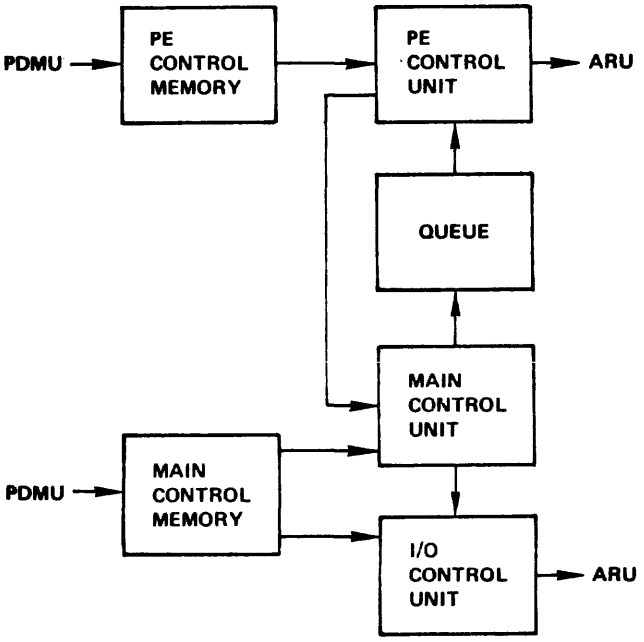

Figure 3—Block diagram of array control unit

PE Control Unit

The PE control unit generates all PE control signals except those associated with I/O and the *S*-registers. The control unit reads 64-bit-wide microinstructions from the PE control memory. The PE control memory holds the standard library of array processing routines plus any routines generated by users, so it is like the writable control store in other computers. When the PE control unit receives a call from the queue, it reads the calling parameters and jumps to the entry point of the called array processing routine. After executing the routine, the PE control unit then processes the next call from the queue.

The PE control unit contains a 64-bit-wide common register to hold the scalar values required by routines that combine scalars with arrays, that search arrays for values, or that generate a scalar from an array.

There are eight 16-bit index registers in the PE control unit. One index register holds the index of a selected bit in the common register. Since array processing is bit-serial, the common register scalar is also usually treated bit by bit. The selected common-register bit *(W)* can be tested by branch instructions, used to select a *P*-register logic function in all PE's, and loaded by the sum-or tree output. Note that using the common-register bit *(W)* to select a *P*-register logic function allows one to select any of the 256 logic functions of three variables—in every PE the selected function between register *P*, the data-bus state *(D)*, and the common-register bit *(W)* is stored in register *P*. This is the mechanism used to broadcast common-register values to all PE's.

The other seven index registers can hold the addresses of array bit-planes in the PE random access memories. Any of the eight index registers can be used to hold the length of an array. Many of the array processing routines are variable length, so they use an index register to hold a loop count and decrement it once for each bit-plane treated.

Other registers in PE control include the topology register to select the array unit topology, a program counter holding the location of the current microinstruction in the PE control memory and a subroutine return stack to facilitate using some array processing routines as subroutines to other routines.

The instruction register is 64 bits wide. Most instructions are executed at a nominal 10-megahertz rate. Several operations can be merged into one instruction; for example, several PE operations, modification of several index registers, and conditional branching. Merging allows most of the control unit overhead to be absorbed so that the PE's are doing useful work on every machine cycle.

I/O Control Unit

The I/O control unit shifts the PE *S*-registers, manages the flow of data in and out of the array unit, interrupts PE control to transfer data between the *S*-registers and the PE memory elements, and can also control the staging memory. Once initiated by main control or the program and data management unit, the I/O control unit chains through a sequence of I/O commands in main control memory.

Main Control Unit

Main control reads and executes the application program from the main control memory. It performs all scalar processing itself and sends all array processing calls to the queue for processing by the PE control unit. Input and output operations for the I/O control unit are either sent directly to the I/O control unit or sent to the program and data management unit for coordination with its peripheral transfers.

Main control has 16 general-purpose registers, some registers to enter calling parameters into the PE control unit queue, and other registers to receive scalars generated by certain array processing routines.

STAGING MEMORY

The staging memory is in the I/O path of the array unit. Besides acting as a buffer between the array unit and the

outside world, the staging memory reformats data so both the array unit and the outside world can transfer data in the optimum format. The array unit sees data in a bit-plane format (one bit from 16,384 different items), while the outside world sees data in an item format (all bits of one item). The staging memory can also rearrange data to match the scrambled layouts of some application programs. The 5120-point FFT example is one such program.

The staging memory comprises a main stager memory, an input sub-stager, and an output sub-stager (Fig. 4). The main stager memory can have 4, 8, 16, or 32 banks of storage with 16K, 64K, or 256K words per bank. Each word holds 64 data bits plus 8 error-correction bits. A fully implemented main stager would hold 67 megabytes of data. Each bank contains 72 dynamic MOS RAM elements. Initially, 16K bit elements are used. When repopulated with 64K bit elements, the storage in each bank is quadrupled. When repopulation with 256K bit elements is feasible the storage per bank can be quadrupled again. Each bank can accept a 64-bit word and present a 64-bit word every 1.6-microsecond cycle time (the cycle time also includes any memory refresh required), so each bank has a ten-megabyte-per-second I/O rate (5-megabyte-per-second input and 5-megabyte-per-second output). A 32-bank main stager can accept and present data at the 160-megabyte-per-second rate of the array unit I/O ports.

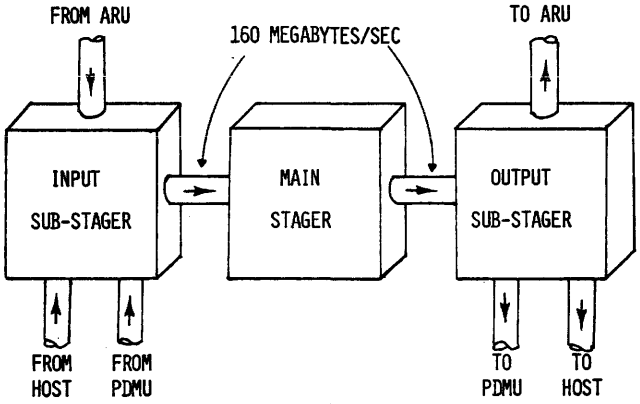

Figure 4—Staging memory

The sub-stagers are fast 128-bit by 1024-bit ECL multidimensional access memories.[4] The input sub-stager accepts data in the format of the source (array unit, program and data management unit, or the host) and rearranges the data to agree with the main stager format. The output sub-stager performs the complementary function of rearranging data from the main stager format to the format of the destination.

Many different main stager formats are possible—a main stager word may contain one bit of 64 different elements, two bits from 32 different elements, and so on. The main stager format is selected according to the data formats in the source and the destination. A software module in the program and data management unit can be used to select the main stager format and program the internal transfers of the staging memory.

PROGRAM AND DATA MANAGEMENT UNIT

The program and data management unit can control the overall flow of programs and data in and out of the massively parallel processor. It acts as a small-scale host when the normal host is not available. The program and data management unit is a DEC PDP-11 minicomputer with a number of terminals, a line printer, disk storage, and a tape unit operating under DEC's RSX-11M real-time multiprogramming system. Custom interfaces provide communication with the array control unit and the staging memory.

The program development software package for the massively parallel processor executes in the program and data management unit. The package includes an assembler for the PE control unit to facilitate developing array processing routines; an assembler for the main control unit to develop application programs; a linker to form load modules for the array control unit; and a control and debug module to load, execute, and debug programs. Much of the software development package is written in FORTRAN to ease the transfer of the package to the host computer.

HOST INTERFACE

The massively parallel processor to be delivered to NASA will connect to a DEC VAX-11/780. The staging memory is connected to a DEC DR-780 high-speed interface of the VAX that can transfer data at a rate of 6 megabytes per second. The staging memory interface is designed to accommodate other devices such as high-speed disks. To allow control of the massively parallel processor by the host, the array control interface can be switched from the program and data management unit to the host computer. Transfer of the software is simplified by writing much of it in FORTRAN.

CONCLUSIONS

The massively parallel processor is designed for high-speed processing of satellite imagery. The typical operations may include radiometric and geometric corrections and multispectral classification. Preliminary application studies indicate that the processor may also be useful for other image processing tasks, weather simulation, aerodynamic studies, radar processing, reactor diffusion analysis, and computer image generation.

The modular nature of the processor allows the number of processing elements and the capacities of its memories to be scaled up or down to match the requirements of the application.

REFERENCES

1. Siegel, H. J., and S. D. Smith, "Study of Multistage SIMD Interconnection Networks," *Proceedings of the 5th Annual Symposium on Computer Architecture,* Palo Alto, Calif., April 1978. IEEE Computer Society, Long Beach, Calif., pp. 223–229.
2. Fung, L. W., "A Massively Parallel Processing Computer," in *High-Speed Computer and Algorithm Organization,* D. J. Kuck et al. eds., New York: Academic Press, 1977, pp. 203–204.
3. _____ "MPPC: A Massively Parallel Processing Computer," GSFC Image Systems Section Report, Sept. 1976, Goddard Space Flight Center, Greenbelt, MD.
4. Batcher, K. E. "The Multidimensional Access Memory in STARAN." *IEEE Transactions on Computers,* C-26 (1977), pp. 174–177.