Staran

The STARAN architecture and its application to image processing and pattern recognition algorithms

by J. L. POTTER

Goodyear Aerospace Corporation
Akron, Ohio

INTRODUCTION

The STARAN E is a general purpose parallel computer. However, certain aspects of the STARAN E's architecture, specifically, a single instruction stream-multiple data stream organization, high speed I/O, a flip (permutation) network and a conditional operation capability on each parallel processing element, are particularly pertinent to the areas of image processing and pattern recognition.

Some of the characteristics of image processing and pattern recognition which provide a good fit to the STARAN's architecture are repetitiveness, spatial dependencies, complex parallel decision making and high I/O to computation ratios. These characteristics and their corresponding STARAN E architectural accommodations are described in detail. The other aspects of STARAN's architecture are described in only enough detail to provide a basis for discussion.*

BACKGROUND

The STARAN E consists of an associative processor (AP) control module and a number (1 to 32) of associative arrays. The AP module consists of the AP control circuitry itself and bulk core. Figure 1 shows the basic STARAN architecture.

The arrays can be thought of as consisting of high speed memory, low speed memory and a bank of processing elements. Each array is organized into 256 "words." Associated with each "word" is a processing element (PE). Each word is 9K bits long with 1024 bits of high speed memory and 8192 bits of lower speed memory. Figure 2 illustrates the conceptual array organization and a general purpose layout for a 512×512 8 bit/pixel image.

Arithmetic operations are performed in parallel on every (enabled) word of memory, one bit at a time. That is, the least significant bit of every word in a field (a bit column vector) is added (multiplied, etc.) to the corresponding least significant bit of every word in a second field and stored in the least significant bits of the sum (product, etc.) field. The carry bits are saved and added into the second bit slices of the arguments. This process is repeated until the entire fields have been processed.

ARCHITECTURE—ALGORITHM PARINGS

Certain characteristics of image processing and pattern recognition mesh perfectly with some of the architectural features of the STARAN E. In the following paragraphs,

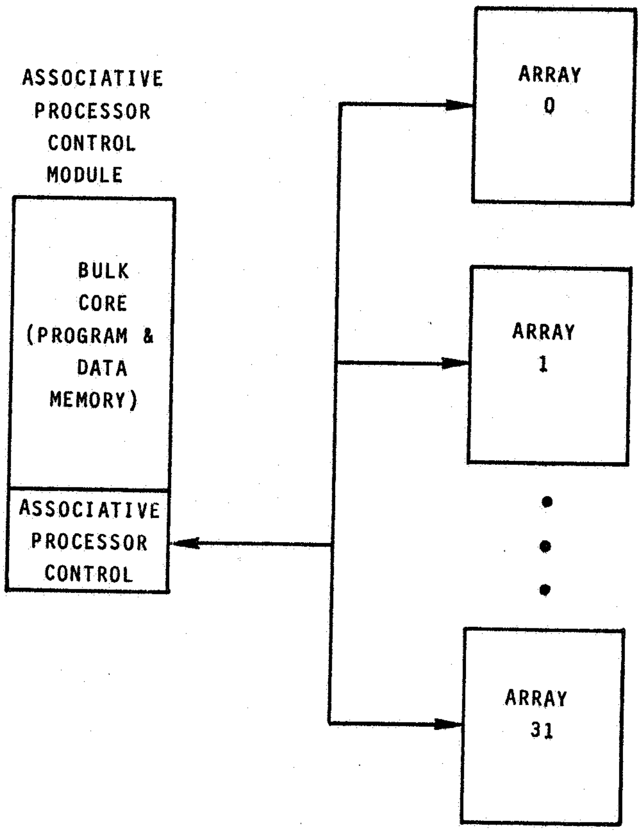
ASSOCIATIVE PROCESSOR CONTROL MODULE
BULK CORE (PROGRAM & DATA MEMORY)
ASSOCIATIVE PROCESSOR CONTROL
ARRAY 0
ARRAY 1
ARRAY 31

Figure 1—Basic STARAN architecture

* For a detailed description of the STARAN line of computers see References 1, 2 and 3.

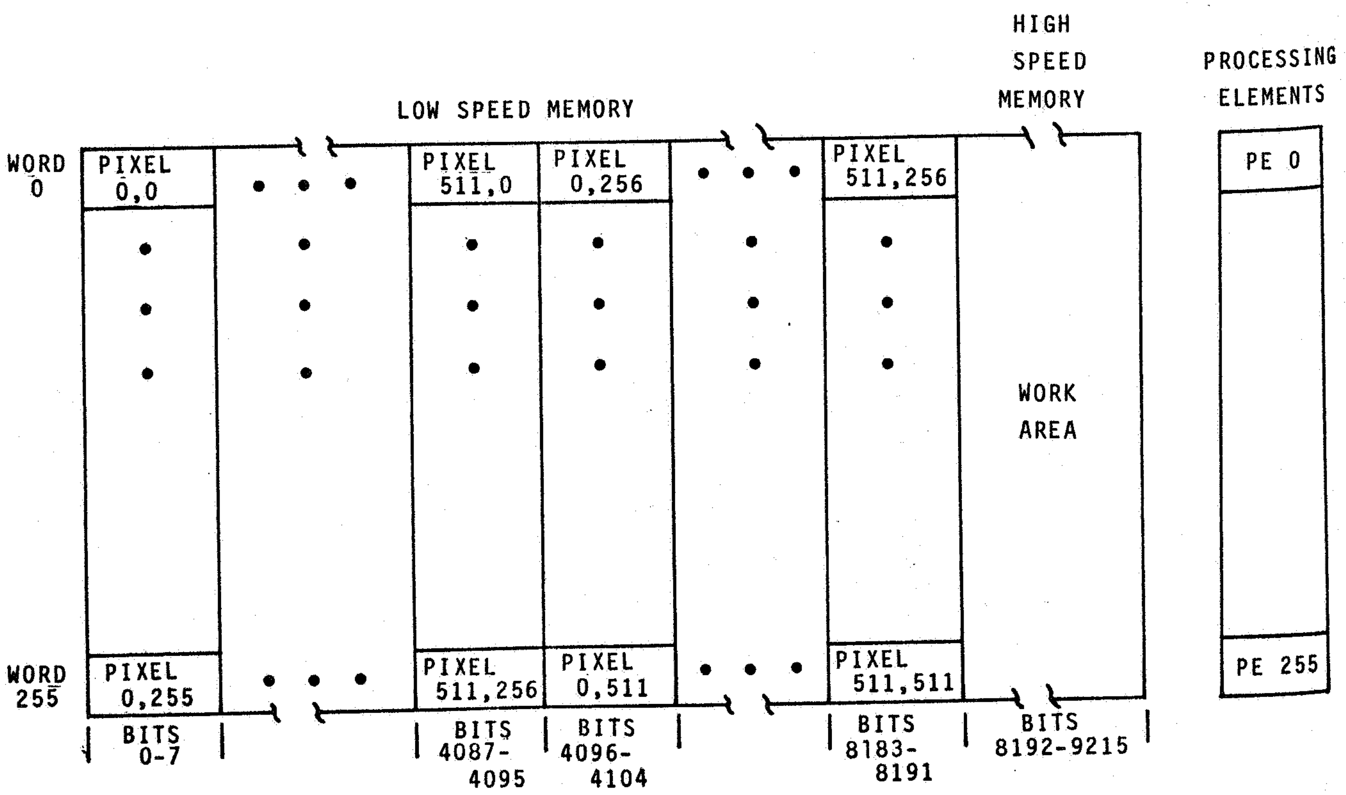

Figure 2—General purpose array organization

some characteristics of image processing and pattern recognition algorithms will be identified and described. Then the corresponding architectural feature of the STARAN E will be discussed and shown how it facilitates implementation of the algorithms.

REPETITIVENESS

Images consist of large volumes of data. A standard TV monitor size image contains over a quarter of a million pixels (picture elements). The images obtained from satellites may contain many millions of pixels. Almost all image processing and pattern recognition algorithms consist of performing the same sequence of operations for every pixel in an image. This aspect of image processing fits perfectly with the single instruction stream-multiple data stream organization of the STARAN.

The associative processor (AP) control portion of the STARAN computer provides a single sequential instruction stream to the associative arrays. Each associative array contains 256 processing elements (PE's) and a fully equipped STARAN E may have up to 32 arrays. Thus the single instruction stream can control from 256 to 8192 PE's resulting in a processing capability of from 11 to 356 million 32-bit adds per second (MIPS). Figure 1 illustrates the single instruction stream-multiple data stream organization of STARAN.

"Inherently serial" algorithms such as classical Maximum Likelihood classification are easily implemented in parallel with the above organization. This is because the algorithm is still executed in serial for every pixel, but from 256 to 8192 pixels can be handled with one pass of the algorithm. With the large number of pixels that need to be processed in a typical image, parallel application of the algorithm is the only practical answer.

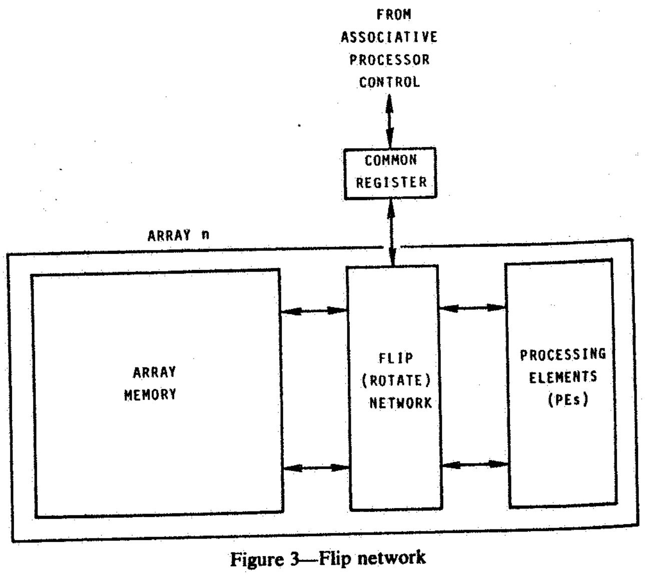

Figure 3—Flip network

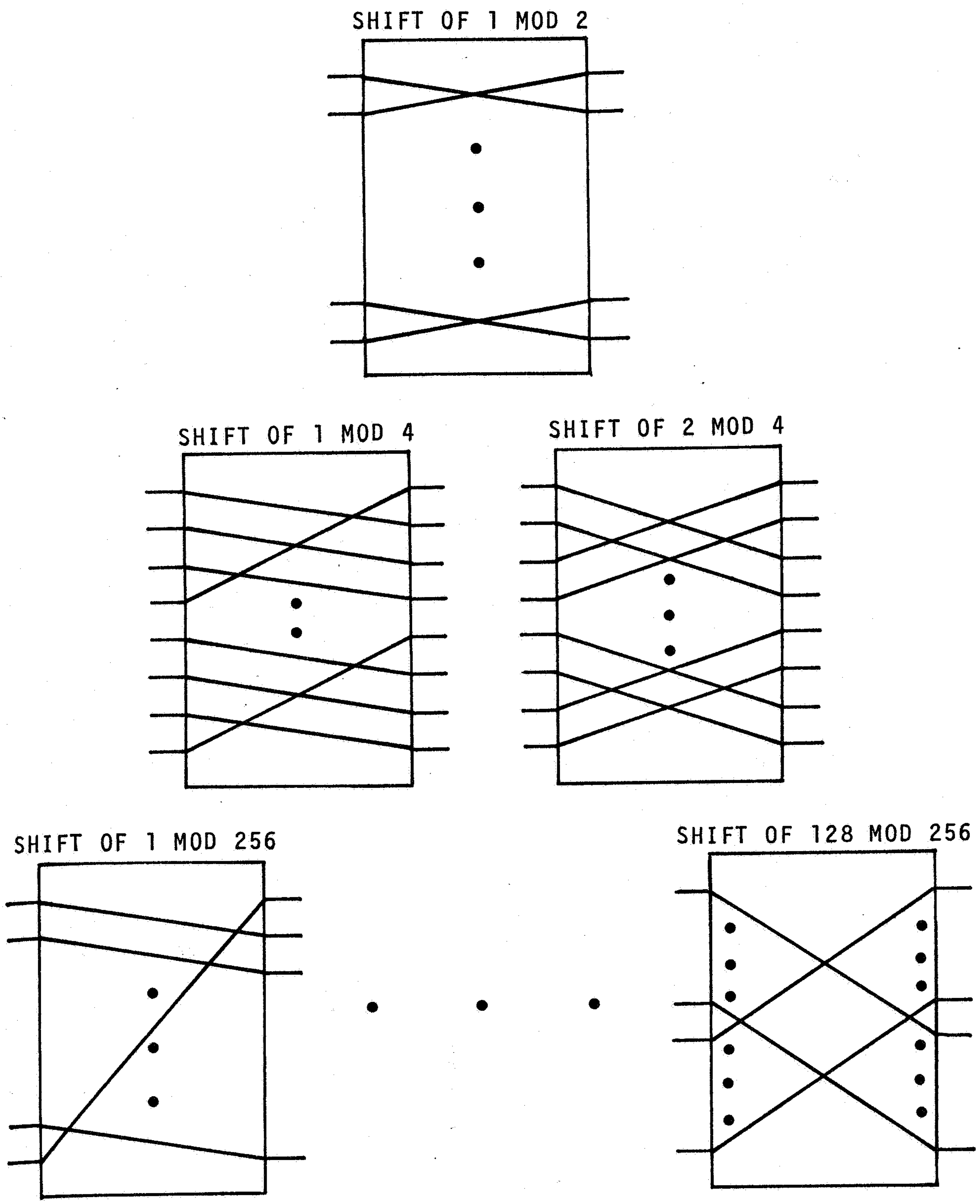

Figure 4—Flip network rotations

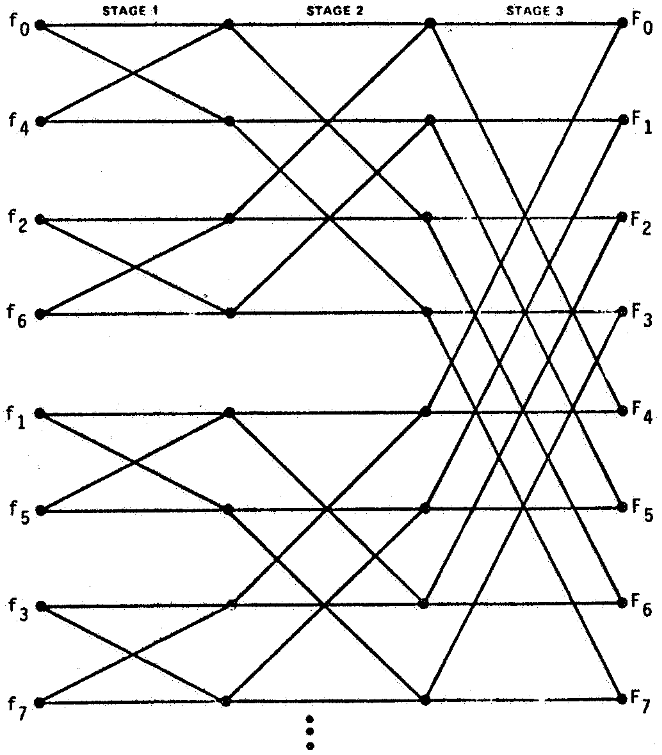

Figure 5—Butterfly diagram for FFT

SPATIAL DEPENDENCIES

Image data is inherently two dimensional and processing algorithms often must deal with this "built in" organization. In general, there are two different types of relationships. The neighborhood relationship requires that pixel neighbors in the two dimensional plane be readily accessible. The second type of relationship is normally spatially more extensive and is typified by the Fourier Transform where the pixel and its neighbors at power of two intervals for an entire row or column are related.

The STARAN associative arrays are well suited for accommodating both types of dependencies. Each array, as shown in Figure 3, has a flip network located between the memory and PE portions. This network performs as a specialized shift register and provides great flexibility in accessing pixels which are spatially related.

In particular, the flip network can accomplish power of two rotates within power of two sized fields with no time penalty. That is, a 1 bit rotate in 128 2 bit fields; 1 and 2 bit rotates in 64 4 bit fields; 1, 2 and 4 bit rotates in 32 8 bit fields; on up to 1, 2, 4, 8, 16, 32, 64 and 128 bit rotates in one 256 bit field. (See Figure 4) This means that inter-word operations at these intervals can be performed in parallel at the same rate as intra-word operations. Moreover, all inter-word operations can be performed with only a slight penalty.

The flip network then provides a very efficient method of implementing algorithms such as the Fast Fourier transform which utilize the spatial power of two interrelationships between pixels. This power of two spatial relationships is frequently expressed in the butterfly diagram shown in Figure 5. A detailed discussion of how the FFT can be implemented in the STARAN in log N steps of 1 add, 1 subtract, 2 real multiples and 2 exchanges has been published elsewhere.[4]

Template matching and spatial convolution are two algorithms which require the neighborhood type of pixel access. These processes can be implemented with little or no time penalty for the required spatial relationships. In particular, 2×2, 3×3, 5×5, 9×9 and 17×17 displacements require no time penalty. Other displacement amounts up to 16×16 require at most one extra shift except for 12×12 and 14×14 which require two shifts. In general, the required shifting for processing *any* window or template size within the 256 word array (i.e. 255×255 or less) is insignificant in overall algorithm time.

PARALLEL DECISION MAKING

An important aspect of image processing and pattern recognition is decision making. Many algorithms perform different operations as a function of the data. The complexity of the process is typified best perhaps by scene analysis techniques. In these situations it is essential to be able to record the exact state of each individual datum in the image.

Each STARAN array contains a special register which can be set as the result of searches (LE, GT, etc.) and arithmetic operations. This mask (or M) register can be used to select a subset of the words in an array to participate in subsequent operations. The results of tests, conditions and states can be stored, retrieved and operated on to achieve any desired logical combination of tests. Figure 6 indicates the physical location of this register.

Two examples of how the M register can be used are template matching and hierarchical structuring. To start a 3×3 template match, the M registers are set to all ones so

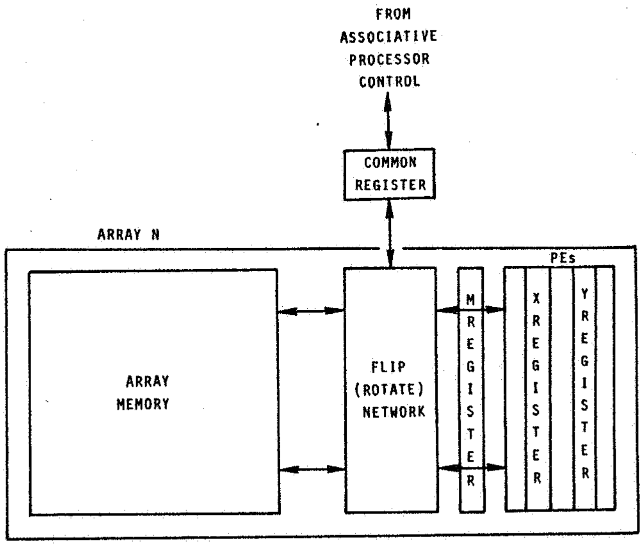

Figure 6—M register configuration

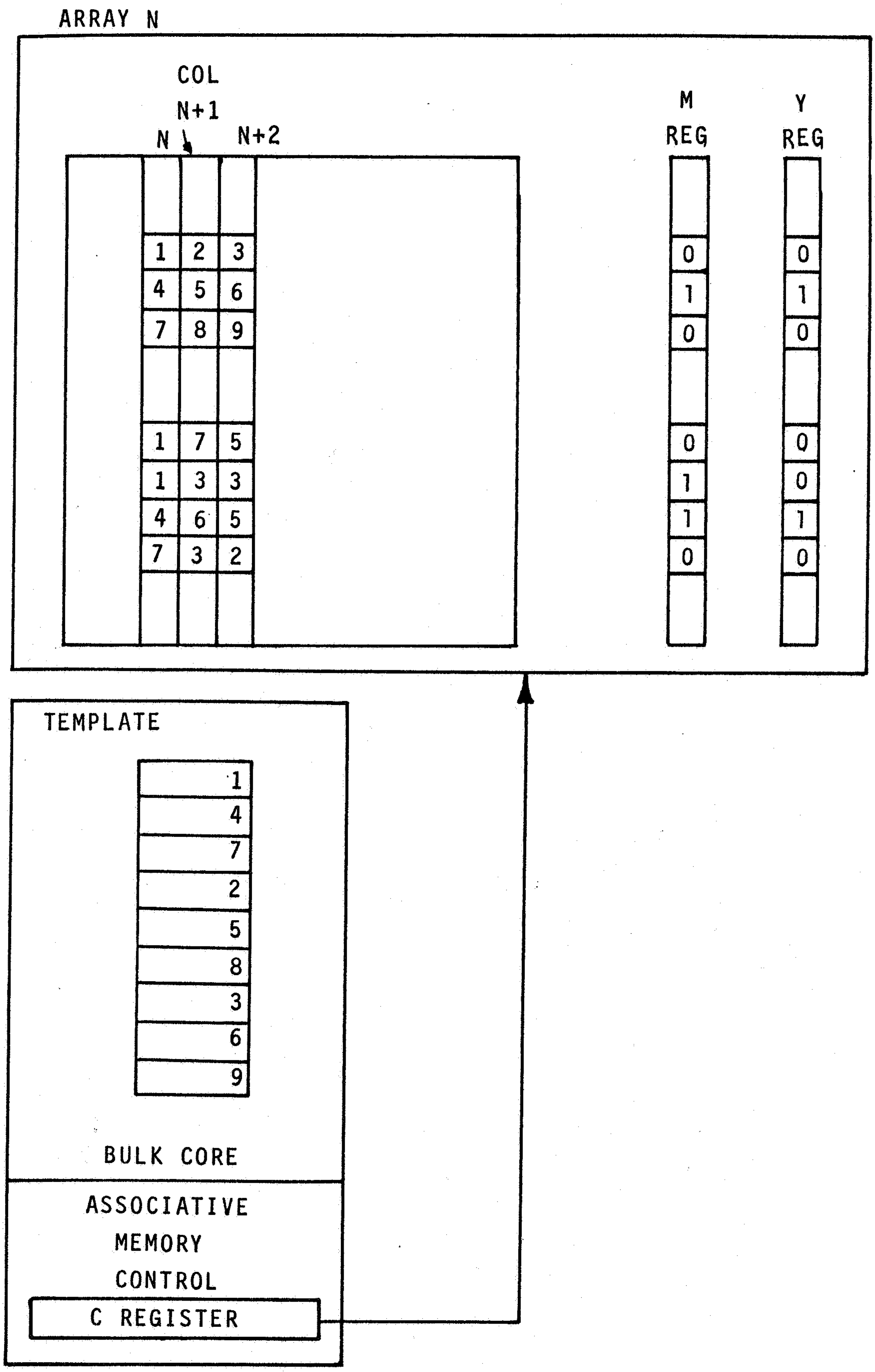

Figure 7—Template matching

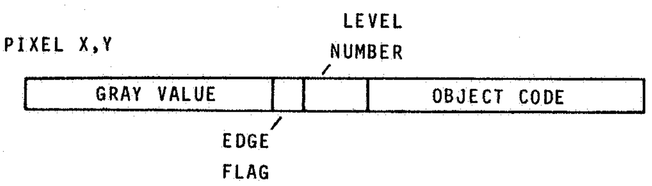

Figure 8—"Typical" scene analysis configuration

that every word participates in the search. The first value of the template is loaded into the common register and the first columns of all the arrays are matched against it. Every successful match is recorded in the Y register. The Y register is shifted down one and moved to the M register, the second template value is moved to the C register and the column is searched for a match of the second template value but on only those words with a corresponding one in the M register (i.e., only those which successfully matched the first value). Consequently at the end of the second search the Y register contains a one for every word and only those words which successfully matched both the first and second template values (the state shown in the M and Y registers in Figure 7). The procedure is repeated for each of the template values with an upward shift and column increment after searching for the third and sixth values. At the end of this process, the M register of all enabled arrays will contain a one (shifted down by two) corresponding to every successful template match covering columns 1, 2 and 3. The process is repeated for every set of columns to be searched.

Note that the columns need not be contiguous to be searched. Thus two situations can be easily handled. First, the columns can be organized in an order which facilitates another algorithm and/or input/output situations. Second, the template need not be contiguous but may cover essentially any size area in any desired manner. Both of these capabilities emphasize the flexibility of data processing in the STARAN arrays.

In a scene analysis application, each pixel might have a set of flags and auxiliary fields associated with it as shown in Figure 8. Then searches of the type "obtain the edges of object 5 on level 2" would be easily implemented in parallel by: first, searching all pixels in a column for object 5, then searching the matched words for level number 2 and then ANDing the edge flags with the Y register. The result is the answer to the search for the given pixel column. The process would then be repeated for each column under consideration. This example illustrates how easy it is to save the result of previous algorithms as flag vectors and codes in parallel and that this information is readily available for subsequent processing and analysis.

INPUT/OUTPUT VERSUS COMPUTATION

It has been established that in general, image processing involves large volumes of data. The algorithms vary quite markedly however in the degree of computation involved. Simple grayscale remapping is such a useful function that special hardware circuitry is often contained within the display devices. In order to make such changes permanent, however, a computer must be used. Operations such as changing (remapping) the grayscale values of images require only one operation per pixel but must be performed on every pixel. These algorithms represent the high I/O-low computation end of the spectrum. At the other end are such algorithms as the domain transforms. Frequently these procedures require numerous arithmetic operations on a per pixel basis. For example, a two dimensional Fast Fourier Transform for a 512×512 pixel image requires 54 multiples and 90 adds per pixel (for a serial computer).

The STARAN has three I/O paths. The common register path (shown in Figure 1 and at the top in Figure 9) operates at between 12-15 million bits per second (MBPS). It is most useful for "broadcasting" data such as constants and parameters to all arrays in parallel. It is a 32 bit wide path.

The most useful path for data I/O is the 32 bit wide multiplexed I/O bus into and out of each array. This bus is capable of operating at between 80 and 640 MBPS. Thus an entire 512×512 8 bit per pixel image can be input or output in from 26.2 to 3.3 milliseconds. This data path is connected to a crossbar switch so that it can be used to transmit data between arrays as well as to peripheral storage or image display devices.

The fastest bus is the 256 bit parallel I/O bus which can operate from 512 to 2560 MBPS. The data transfer rate on this bus is such that special peripheral configurations are

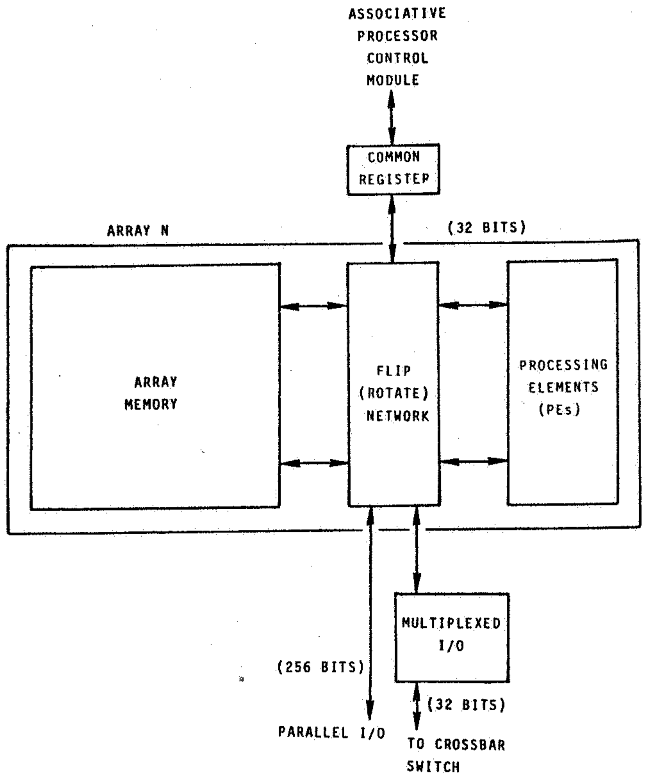

Figure 9—Array I/O paths

required. Consequently, it is best suited for special purpose applications.

Every array module is capable of being connected to either of two controllers. Thus in those applications where the equivalent of double buffering is desirable, some of the arrays can be switched to an I/O controller while the remainder are used for computation.

SUMMARY

The multiple data stream parallelism of the STARAN allows it to perform a "serial" algorithm on up to 8192 data elements simultaneously. This is important in image processing where large volumes of data are processed. Images are inherently two dimensional, but the large array size of the STARAN E, readily allows an entire 512×512 8 bit/pixel image to be stored in one array with essentially its two dimensional topography intact. The array addressing structure and the array flip network provides easy access to every pixel and its neighbors in a simple efficient manner. The mask (M) register operation in an array enables complex decision processes to be made on any subset of pixels in an array. All of the above aspects of the STARAN's architecture would not be valuable if it could not be efficiently used. The three I/O paths into every array provide the capability to efficiently broadcast parameters and constants as well as loading and unloading image data in an expeditious manner. Thus it is apparent that many of the architectural features of STARAN are ideally united for image processing and pattern recognition. This conclusion is confirmed by the execution times shown in Table I.

TABLE 1.—Sample Algorithm Processing Times

Function	Description	Image Size	Speed*
Magnification	2.5× cubic convolution interpolation	512×512 8 bit/pixel	588 milliseconds
Convolution	3×3 window	512×512 8 bit/pixel	700 milliseconds
FFT	1 Dimension	512 16 bit/pixel	2.7 milliseconds

* Measured times in the STARAN B machine exclusive of I/O. The STARAN E instruction execution time is approximately 20 percent faster.

REFERENCES

1. Batcher, K. E., "STARAN Parallel Processor System Hardware," *1974 Computer Conference,* AFIPS Conf. Proc., Vol. 43, pp. 405-410.
2. Batcher, K. E., "STARAN Series E," *1977 International Conference on Parallel Processing.*
3. Goodyear Aerospace Corporation, "The STARAN E System—An Overview," GAC Document Number AP-123226, 29 September 1976.
4. Goodyear Aerospace Corporation, "Application of STARAN to Fast Fourier Transform," GAC Document Number GER-16109, 31 May 1974.
5. Rohrbacher, D. and J. L. Potter, "Image Processing with the STARAN Parallel Computer," *Computer,* Vol. 10, No. 8, August, 1977, pp. 54-59.
6. Gambino, L. A. and B. L. Schrock, "An Experimental Digital Interactive Facility," *Computer,* Vol. 10, No. 8, August, 1977, pp. 22-28.

Bit-Serial Parallel Processing Systems

KENNETH E. BATCHER

***Abstract*—About a decade ago, a bit-serial parallel processing system STARAN®[1] was developed. It used standard integrated circuits that were available at that time. Now, with the availability of VLSI, a much greater processing capability can be packed in a unit volume. This has led to the recent development of two bit-serial parallel processing systems: an airborne associative processor and a ground based massively parallel processor.**

The airborne associative processor has about the same processing capability as a three cabinet STARAN system in a volume less than 0.5 cubic feet. The power required and weight are also reduced dramatically for the airborne environment.

The massively parallel processor has about 100 times the processing capability as the STARAN system in about the same volume. Floating point speeds are better than 100 MOPS (million operations per second). Integer arithmetic speeds depend on operand lengths—for 16 bit integers the speed is better than 3000 MOPS for addition and 450 MOPS for multiplication.

After presenting the basic rationale for bit-serial parallel processors, the organizations of the two recent systems are shown and some of their applications are outlined.

***Index Terms*—Airborne processors, bit-serial processors, custom VLSI chips, image processing, multidimensional access, parallel processors, radar processing.**

I. Introduction

Like other parallel processing systems, the systems we describe in this paper have a number of processing elements (PE's) operating in parallel on separate data streams under the control of a single control unit. The systems have a large number of processing elements (in the thousands) and each element is very simple—operating on its data stream bit by bit. Thus, we call them bit-serial parallel processing systems. Given an array of 1000 16 bit items, a conventional computer would process the array item by item in 1000 steps, whereas a bit-serial parallel processor would process the array broadside in 16 steps with each processing element treating a bit of one item on each step. The array is accessed by bit-slices instead of by items. Since the number of items in a typical array is much larger than the number of bits per item, processing is much faster.

A feature of bit-serial processing is its ability to handle data items of any length. There is no need to extend operands with filler bits to pack them into standard machine words as in a conventional computer. This raises the storage and processing efficiency. Suppose we have an array of M operands with N bits per operand. Any parallel processor suffers some loss in efficiency when M is less than the number of processing elements. A conventional computer experiences a similar loss in efficiency when N is less than its word length. Parallel processors with processing elements of a standard word length suffer from both losses in efficiency when M is less than the number of PE's, and when N is less than the word length of a PE. Bit-serial parallel processors, like conventional computers, experience only one inefficiency.

Another advantage of bit-serial parallel processors occurs when only a part of the operand needs treatment. For example, only one cycle is required to test the signs of all elements in an array. Associative processing where data items are addressed by content is easily performed by reading out the keys bit by bit and comparing their values to the bits of a comparand. Only the keys involved in the search need be accessed.

While a data array would be accessed broadside by the set of processing elements, the same array would be read and written in the orthogonal direction (item by item) by input-output channels. This is the normal way data arrays are generated, stored, output, and processed in conventional computers. We should accommodate the rest of the world rather than force it to transfer data arrays by bit-slices. Thus, we need a means of accessing data in two directions: item by item for input and output and bit-slice by bit-slice for the set of processing elements.

In STARAN® processors, data arrays are stored in multidimensional access memories [1], [2] so they can be accessed in either direction equally well. The memories use the same kind of random access memory integrated circuits as conventional solid-state stores. The locations of the data bits are scrambled a certain way so data arrays can be accessed many different ways including the orthogonal directions. The inclusion of a few EXCLUSIVE-OR gates in the memory address bus generates the necessary addresses for the memory elements. A network called the flip network [3] is included in the memory data bus to scramble data being written into storage, and to unscramble data being read from storage. This network is also used to route data between processing elements and is akin to a number of multistage interconnection networks [4].

The first STARAN was demonstrated in 1972. It used standard off-the-shelf integrated circuits that were available at that time. Now with the availability of VLSI, much greater packing densities can be obtained. This has led to the recent development of two bit-serial parallel processors: the airborne associative processor and the massively parallel processor.

Manuscript received June 4, 1981; revised December 10, 1981.

The author is with the Digital Technology Department, Goodyear Aerospace Corporation, Akron, OH 44315.

[1] Registered Trademark, Goodyear Aerospace Corporation, Akron, OH 44315.

0018-9340/82/0500-0377$00.75 © 1982 IEEE

II. Airborne Associative Processor

During 1978, studies to determine the feasibility of an airborne version of a STARAN processor were conducted under company reseach and development programs and under U.S. Navy contracts. The studies culminated in the design of an airborne associative processor using VLSI [5]. The architecture of the processor is shown in Fig. 1. The five major blocks are as follows:

1) the array unit containing over 2000 PE's and over 1 Mbyte of data storage,

2) the array control unit which supplies the control signals for the array unit,

3) the register and arithmetic section which controls the input and output of array unit data and generates array addresses,

4) the program execution control unit which executes the application program and drives the array control unit and the register and arithmetic section, and

5) the control memory which stores the application program and buffers data between the airborne associative processor and a host computer.

A. Array Unit

The array unit comprises 17 array modules, 16 for the application and a spare module to replace any other array module found to be in error. Each array module contains 128 PE's and four 32 × 4096 bit arrays of multidimensional access storage. Thus, the array unit contains 2048 PE's (plus spare) and one Mbyte of data storage (plus spare).

Fig. 2 depicts one array module. The module contains four custom design VLSI chips with each chip containing the registers for 32 PE's, a 32 line flip network, and a resolver (Fig. 3). The multidimensional access memories for one array module are packaged in eight hybrid packages with each hybrid containing 16 1K × 4 random access memory (RAM) chips. The 32 PE's of one VLSI chip access the storage of two memory hybrid circuits through the 32 line flip network on the VLSI chip. The arrangement is akin to one STARAN array module except that the flip network is only 32 lines wide instead of 256 lines wide [1], [3]. The multidimensional access memories allow the register and arithmetic section to input or output all bits of a 32 bit operand in one memory cycle and the set of PE's to access one bit-slice from all operands in one memory cycle.

The processing elements are much like the PE's of STARAN. Each PE contains a one bit X-register and a one bit Y-register which perform the bulk of the bit-serial arithmetic [6]. The one-bit mask register holds a mask bit which governs writing of PE data into the multidimensional access storage in masked-write operations. Each PE also has a one bit hold register for masked-write operations. This register was not needed in STARAN systems because they performed masked writes by selectively setting the write-enable pins of the memory chips. In the airborne associative processor the memory chips are four bits wide so individual masking is not possible at the memory chips. Masked write operations are performed by reading the memory data into the hold register, changing those data bits where the mask register is set, and then writing the data back into the multidimensional access memory.

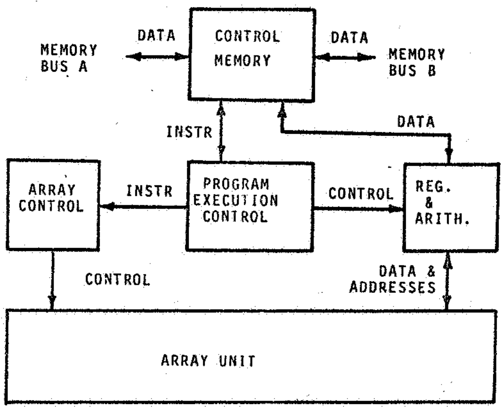

Fig. 1. Block diagram of the airborne associative processor.

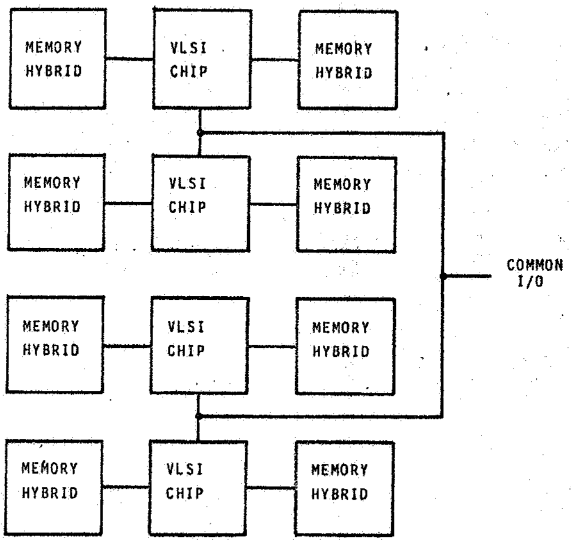

Fig. 2. One airborne associative processor array module.

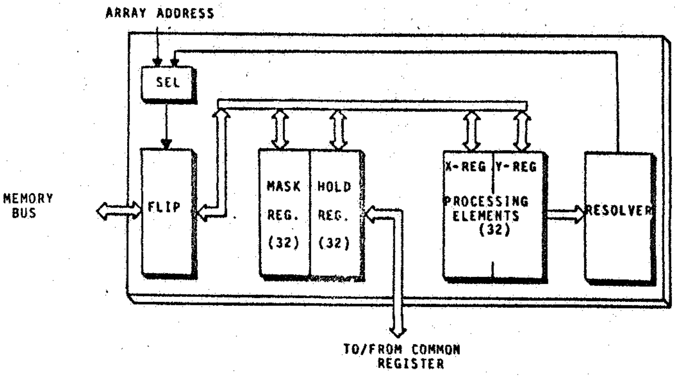

Fig. 3. Custom PE/flip network chip.

The resolver on each PE/Flip network chip tests the 32 bit data bus for the presence of one or more 1-states. The SUM-OR output is an INCLUSIVE-OR of the 32 data lines. The position of a 1-state is output on a 5 bit wide resolver address bus. The resolver is used after associative searches. After each PE has checked its associated storage for the presence or absence of data satisfying the search criteria the match bits are fed to the resolver which locates the position of one item matching the criteria. The resolver outputs of the PE chips are collected together in a super resolver to see if any of the 2048 PE's in the system have matching data and to generate the address of one match.

The PE/Flip network chip is a custom VLSI chip using CMOS/SOS technology. Cycle times are longer than the STARAN cycle times which used ECL technology. The set of 2048 PE's in the airborne associative processor has about the same processing power as the set of 1024 PE's in a four-array STARAN system.

B. Array Control Unit

The array control unit broadcasts control signals to all PE's in the array unit. Several basic array operations are possible using the PE registers, multidimensional access storage, or the common register in the control unit as operands:

1) the *X* and/or *Y* registers can be loaded or logically combined with data from the common register, the multidimensional access storage, or the PE registers;

2) the mask register can be loaded from the common register, the multidimensional access storage, or the PE registers;

3) the multidimensional access storage can be written (either masked or unmasked) with data from the common register or the PE registers; and

4) the common register can be loaded with data from the PE registers or the multidimensional access storage.

C. Register and Arithmetic Section

The register and arithmetic section generates the addresses for the array unit operations and controls the input and output of array unit data. It contains 24 16 bit registers, two 32 bit registers, an Arithmetic and Logic Unit (ALU), and a number of selection gates (Fig. 4).

D. Program Execution Control

The program execution control unit executes the application program. Conditional branching is provided to any instruction in the control memory. Instruction fetching is overlapped with execution for faster processing. Besides the program counter and the instruction register, the program control unit contains a stack to store up to 16 return addresses and logic to allow prioritized interrupts.

E. Control Memory

The control memory has three types of storage: program memory holding the application program, buffer memory to buffer data to and from the host computer, and a read-only memory to store routines for certain essential operations such as the data transfer program and the basic built-in test routines. The host computer is a dual processor and the buffer memory communicates with each host processor over its memory bus. It has two banks, each holding 8192 32 bit words. To the host computer, the buffer memory appears as one of its own memory banks (the whole airborne associative processor occupies a memory bank space in the host computer).

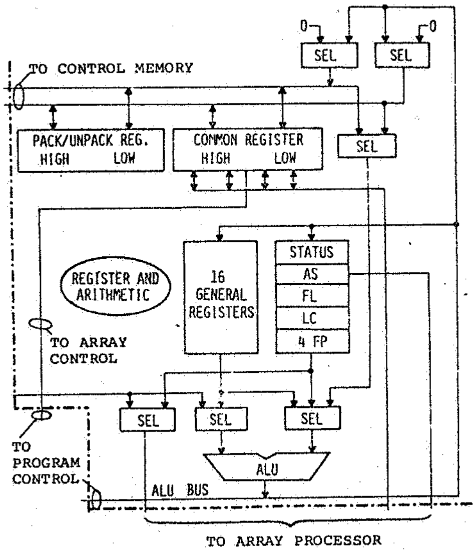

Fig. 4. Register and arithmetic section.

F. Applications

The airborne associative processor was designed to upgrade early warning radar surveillance and command and control processing aboard the Navy's E-2C aircraft. The E-2C is a carrier-based aircraft that receives target data from its own radar and track data from data links. Targets are processed and correlated to form track data which are output to three operator displays. The airborne associative processor fits inside the existing computer as an easily replaceable unit. The high-speed content addressability of the airborne associative processor is used to correlate radar reports with each other and with existing tracks. The software required to maintain the surveillance database is simplified as well.

With suitable modifications, the airborne associative processor could be adapted to other uses in airborne environments or to other applications where a large amount of specialized processing power must be packed into a small volume.

III. Massively Parallel Processor

In 1971, the NASA Goddard Space Flight Center initiated a program to develop high-speed image processing systems. They will be required to process the large amount of image data that will come from satellites that NASA will orbit during the 1980's. These systems use thousands of 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 to be delivered in 1982 [7]. The processor has 16 896 PE's arranged in a 128 row × 132 column rectangular array. The PE's are in the array unit (Fig. 5). 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.

A. Array Unit

Logically, the array unit contains 16 384 PE's arranged in a 128 row × 128 column square array. Physically, the array unit contains an extra 128 row × 4 column rectangle of PE's

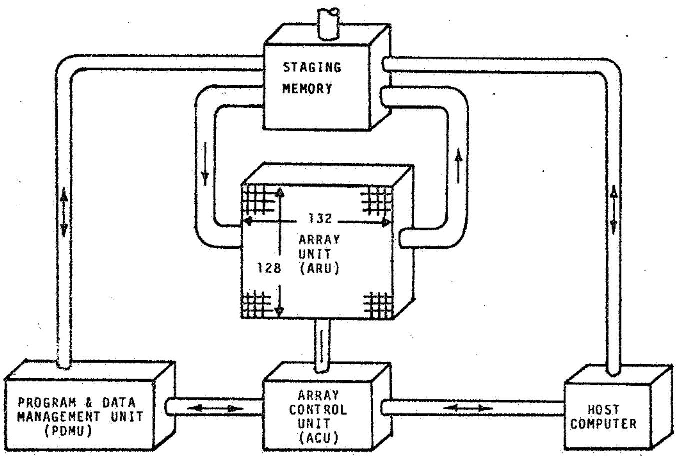

Fig. 5. Block diagram of the massively parallel processor.

for redundancy. Each PE communicates with its four nearest neighbors, north, south, east, and west. Each PE is a bit-serial processor. With a 10 MHz 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 that 16 384 additions are performed in less than 5 μs (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), the floating-point addition is better than 400 MOPS and multiplication is better than 200 MOPS.

1) 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 [4], 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 nearest neighbor 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 RAM so that 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 if we want to do one FFT, we will invariably want to do many FFT's so that we can divide the array unit up into many compact subarrays, each doing 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 × 32 column subarray of the array unit. Thirty-two such subarrays can do 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.

One may ask the question of how the data can be input and output effectively especially when they have a peculiar layout like 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 by item format to a bit-slice format. Some kind of buffer memory will be required in the array unit input–output 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 input–output 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 by item format of the outside world. With 16 000 PE's this is a better solution to the problem than the solution used in STARAN where a common multidimensional access memory was used for both PE random access storage and input–output transformation. Because of the planar nature of the array unit in the massively parallel processor we will refer to accesses as bit-plane accesses instead of bit-slice accesses.

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 where, for example, the PE's on the north edge see zero when data items are routed to the south. Other programs would like to see a cylindrical-topology where the PE's on the north edge see data from PE's on the south edge when data items are routed to the south. Also, some programs would rather have the 16 384 PE's connected in one long linear string rather than in a 128 × 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 (Fig. 6). 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 that the array of PE's looks like a long linear string.

2) Redundancy: One advantage of the rectangular nearest neighbor 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. We have found no similar technique for bypassing faults in a multistage SIMD interconnection network. To add redundancy to the array unit, we implement more than 128 columns and insert bypass gates in the east–west routing paths.

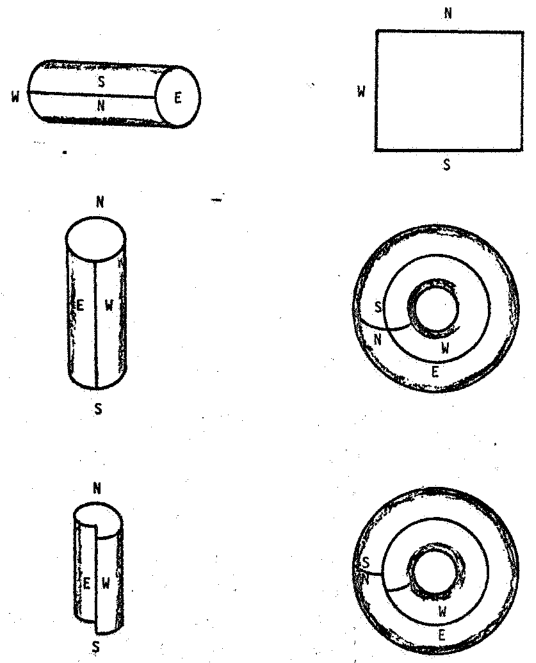

Fig. 6. MPP array topologies.

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 do not 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 (4/128) 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.

3) Processing Elements: Each PE is a bit-serial element. Initially, the PE's had down-shifting binary counters for arithmetic [8], [9]. The PE design was modified to use a full adder and a shift register for arithmetic. The modified design performs the basic arithmetic functions faster. Each PE has six 1 bit registers (A, B, C, G, P, and S), a shift register with programmable length, a RAM, a data-bus (D), a full-adder, and some combinatorial logic (see Fig. 7). The nominal clock rate of the PE's is 10 MHz. 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 100 ns clock cycle.

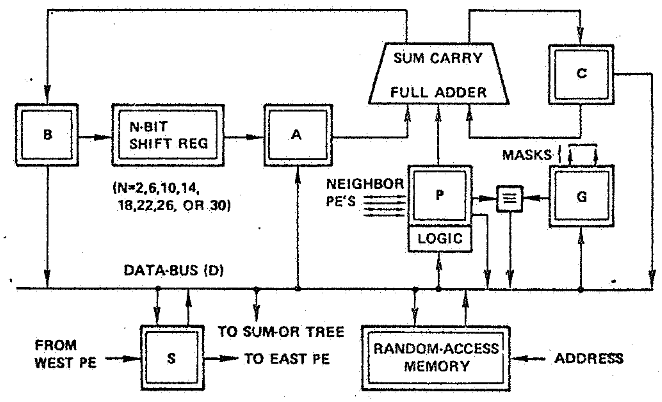

Fig. 7. One MPP processing element.

a) 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 RAM, 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.

b) 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 × 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 $G = 0$.

c) 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 which is placed in the B and C registers with B receiving the least significant bit and C receiving the most significant bit. 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 then 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 microarithmetic operations are combined to perform the array arithmetic instruction set. The 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 least significant 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 RAM. 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 instead of its true value. Two's complement subtraction is done by initializing the C-register to 1 instead of 0. Note that the add and subtract operations we described read two operands from storage, and put the result back in storage so they are equivalent to a sequence of three instructions (load accumulator, add or subtract, and store accumulator) executed 16 384 times.

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.

d) 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 RAM, 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.

The memory parity error indicator senses a parity error in the RAM and latches in the 1-state until cleared. The state of all indicators feeds the SUM-OR tree when the tree is not being used for a SUM-OR operation so the control unit will sense the presence of an error in any PE memory and take appropriate action.

e) 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 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 bit-plane of the RAM's. S-register shifting can run at a 10 MHz rate so data can be input at a rate of 160 Mbytes/s (128 bits every 100 ns). 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 bit-plane of data is transferred from the RAM's 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 MHz rate so that the output rate is also 160 Mbytes/s. 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 Mbytes/s (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. One can see an indication of this from the fact that running I/O at a full rate slows the processing down by only a few percent. 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.

e) Random Access Memories (RAM's): Each PE has a RAM storing 1024 bits. The address lines of all PE's are tied together so the memories are accessed by bit-planes with one bit of a bit-plane accessed by each PE. Conventional RAM integrated circuits are used to make it easy to expand storage when advances in solid-state memory technology allow it. Four PE's share one 1K × 4 RAM chip with an access time of about 50 ns. 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 Mbytes total. The array unit clock system has enough flexibility to accommodate a wide variety of memory speeds so the massively parallel processor can be tailored to other applications which may require more memory at a slower speed.

4) Packaging: The PE RAM's use standard RAM integrated-circuits. All other components of eight PE's are put on a custom VLSI chip. The chip holds a 2 row × 4 column subarray of PE's and 2112 such chips are used in the array unit. The chip pinout is 52 pins and the complexity is about 8000 transistors.

A 16 row × 12 column subarray of 192 PE's is packaged on one 22 cm × 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 × 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.

B. 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. 8). This division of

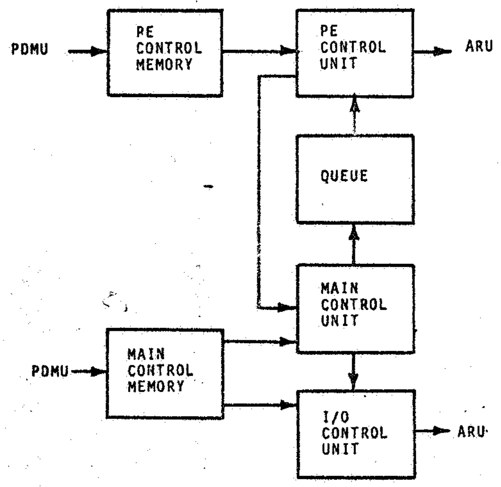

Fig. 8. Block diagram of the MPP array control unit.

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.

1) 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 user-generated routines 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 (e.g., add scalar to array), search arrays for values (e.g., find all the elements of an array greater than a scalar), or generate a scalar from an array (e.g., find the minimum value in 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 value to all PE's.

The other seven index registers can hold the addresses of array bit-planes in the PE RAM's. An array is usually processed by stepping through its bit-planes either from the most significant bit to least significant bit or vice versa. Any of the eight index registers can be used to hold the length of an array. Many of the array processing routines are of 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 MHz rate. Several operations can be merged into one instruction, e.g., several PE operations, modification of several index registers, and conditional branching. Merging allows most of the control unit overhead to be absorbed so the PE's are doing useful work on every machine cycle.

2) 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.

3) 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.

The 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.

C. 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 that 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 (Section III-A.1) is one such program.

The staging memory comprises a main stager memory, an input substager, and an output substager (Fig. 9). 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 for single error correction and double error detection. A fully implemented main stager would hold 67 Mbytes of data. Each bank contains 72 dynamic MOS RAM elements. Initially, 16K bit elements are used. When 64K bit elements are readily available, the storage in each bank can be quadrupled to 64K words, and when 256K bit elements are 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 μs cycle time (the cycle time also includes any memory refresh required), so that each bank has a 10 Mbyte/s I/O rate (5 Mbytes/s input and 5 Mbytes/s output). A 32 bank main stager can accept and present data at the 160 Mbyte/s rate of the array unit I/O ports.

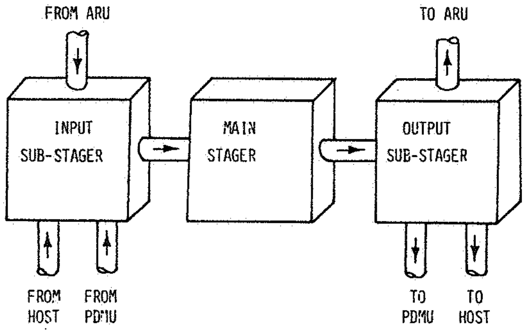

Fig. 9. Block diagram of the staging memory.

The substagers are fast 128 bit × 1024 bit ECL multidimensional access memories [1]. The input substager 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 substager 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, etc. The main stager format is selected based on 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.

D. 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 units 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 is executed in the program and data managment 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 using a Ratfor preprocessor to ease the transporting of the package to the host computer.

E. Host Interface

The massively parallel processor to be delivered to NASA will use a DEC VAX-11/780 for a host computer. The staging memory is connected to a DEC DR-780 high-speed interface of the VAX which can transfer data at a rate of 6 Mbytes/s. 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.

F. Applications

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.

IV. Conclusions

Bit-serial parallel processors can perform certain tasks much faster than other architectures. The use of VLSI allows a large amount of processing power to be packed in a small volume. The airborne associative processor illustrates the use of bit-serial parallel processors in an airborne environment, while the massively parallel processor is an illustration of a ground-based system.

References

[1] K. E. Batcher, "The multidimensional access memory in STARAN," *IEEE Trans. Comput.*, vol. C-26, pp. 174–177, Feb. 1977.

[2] ——, "STARAN series E," in *Proc. 1977 Int. Conf. Parallel Processing*, Aug. 1977, pp. 140–143.

[3] ——, "The Flip network in STARAN," in *Proc. 1976 Int. Conf. Parallel Processing*, Aug. 1976, pp. 65–71.

[4] H. J. Siegel and S. D. Smith, "Study of multistage SIMD interconnection networks," in *Proc. 5th Annu. Symp. Comput. Architecture*, Apr. 1978, pp. 223–229.

[5] T. DiGiacinto, "Airborne associative processor (ASPRO)," in *Proc. AIAA Comput. in Aerosp. III Conf.*, Oct. 1981, pp. 202–205.

[6] K. E. Batcher, "STARAN parallel processor system hardware," in *Proc. 1974 Nat. Comput. Conf.*, May 1974, pp. 405–410.

[7] ——, "Design of a massively parallel processor," *IEEE Trans. Comput.*, vol. C-29, pp. 836–840, Sept. 1980.

[8] L. W. Fung, "A massively parallel processing computer," in *High-Speed Computer and Algorithm Organization*, D. J. Kuck *et al.*, Eds. New York: Academic, 1977, pp. 203–204.

[9] ——, "MPPC: A massively parallel processing computer," Goddard Space Flight Cen., Greenbelt, MD, GSFC Image Syst. Section Rep., Sept. 1976.

Kenneth E. Batcher received the B.S. degree in electrical engineering from Iowa State University, Ames, in 1957, and the M.S. and Ph.D. degrees from the University of Illinois, Urbana, in 1962 and 1964, respectively.

In 1957 he was a trainee at Goodyear Atomic Corporation, Portsmouth, OH, and in 1958 he joined the Goodyear Aerospace Corporation, Akron, OH, where he is now an Engineer, Principal in the Digitial Technology Department. His main field of interest is parallel processing. He developed the odd-even and bitonic sorting networks, and was the chief architect on the STARAN and STARAN-E parallel processors. Currently, he is the chief architect on the Massively Parallel Processor.

Dr. Batcher is a member of the Association for Computing Machinery, Phi Eta Sigma, Phi Kappa Phi, and Sigma Xi.