An Eight-Processor Chip for a Massively Parallel Machine

David Elliot Shaw
Theodore M. Sabety

Department of Computer Science
Columbia University
New York, NY 10027
212-280-8100

Abstract

This paper describes a VLSI chip that serves as the basis for a massively parallel tree machine called NON-VON 3. The chip, which is implemented in 3-micron nMOS technology, contains eight 8-bit processing elements (PE's), each embodying 64 bytes of static RAM. Significant features of the design include: an unusually high processor density; a novel I/O switch that allows the machine to dynamically reconfigure to realize several logical communication topologies; logic supporting the pipelining of instructions, both within and among the individual PE's; a shared partial instruction decoder that reduces pinout and area; and a parallel self-testing, dynamically reconfigurable, fault-tolerant RAM that significantly increases both yield and reliability. The design and operation of the chip are discussed, along with its speed, area, and power dissipation characteristics.

I. Introduction

Decreasing integrated circuit device dimensions, coupled with the introduction of simplified, highly structured VLSI design methods, have made feasible the implementation of a single chip containing a number of simple processing elements (PE's). Such chips may be used to construct massively parallel machines containing very large numbers of PE's, and offering considerable computational power in certain applications. Several existing machines are based on such an approach. The ICL DAP [1], for example, contains 4K 1-bit processing elements, while

Technical Report, Department of Computer Science, Columbia University, August 1984

Goodyear's MPP [2] comprises 16K single-bit PE's. Machines containing as many as 10^5 to 10^6 processing elements [3], [4], [5] have recently been proposed for implementation using currently available MOS VLSI technology.

In this paper, we describe the organization and operation of an nMOS chip that was designed at Columbia University as the basis for a massively parallel tree-structured machine called NON-VON 3. A simplified version of the NON-VON 3 chip, incorporating most, but not all of the enhancements to be incorporated in the final design, has recently been fabricated in VLSI.

Among the novel features of the chip are:

1. An unusually *high processor density*. Eight 8-bit PE's are incorporated in the experimental version that has just been fabricated, each of which embodies 64 bytes of RAM. A production version of the design should offer the advantages of an "intelligent" RAM with only a modest increase in area over the cost of an equivalent amount of ordinary static RAM.

2. Logic supporting *dynamic reconfiguration of the machine* to implement several inter-PE communication topologies. In particular, the design provides efficient support for tree-structured, linear neighbor, and global communication. In addition, special features are provided for associative multiple-match resolution, and for the sequential enumeration of matching records.

3. Facilities supporting both *intra- and inter-PE pipelining of instruction execution*. Specifically, the design allows nearly all instructions to be pipelined through successive levels of the tree, yielding an execution speed that is both reasonably fast in absolute terms and independent of the size of the tree. The design also provides for the overlapped fetching and execution of instructions within each PE.

4. A *single, shared "pre-decoder" on each chip* that partially decodes instructions as they enter the chip. This allows instructions to be encoded in a highly vertical manner for transmission between chips and then translated to a more horizontal instruction format within the chips. The number of pins required for off-chip communication is thus minimized, while the area cost of a substantial portion of the decoding logic is amortized over all eight PE's.

5. A *parallel self-testing, dynamically reconfigurable, fault-tolerant RAM*, capable surviving both initial fabrication defects and subsequently

appearing failures. This facility, which would not be practical in the absence of massively parallel testing at "power-up" time, should greatly increase both yield (because of the proportion of silicon area expended on local RAM) and reliability (because of the ability to recover from errors appearing in the course of operation).

In order to provide a motivating context for these features, Section II provides a brief overview of the organization of that portion of the NON-VON machine with which we will be concerned in this paper. Section III describes the structure of the chip and the manner in which it is connected to other chips of the same type, while Section IV examines the organization of its constituent PE's. Section V describes the operation of the PPS chip, with particular emphasis on functions supported by novel aspects of the chip architecture. In Section VI, we describe the manner in which the chip was designed and fabricated, and offer a brief discussion of its speed, area, and power dissipation characteristics. The essential import of our investigations is briefly summarized in Section VII.

II. Organization of the NON-VON Primary Processing Subsystem

While the complete NON-VON architecture includes several other modules, this paper is concerned only with that portion which is known as the *Primary Processing Subsystem* (PPS). Although physically structured as a binary tree, the NON-VON PPS can be dynamically reconfigured to efficiently support communication patterns characteristic of two other topologies, as we shall see in Section V.

The processing elements do not store and independently execute their own programs, but rather simultaneously execute instructions that are broadcast to them by a single *control processor* (CP), which is attached to the root of the PPS tree. Details of the mechanisms for instruction broadcast will be presented in Section VI. While the complete NON-VON architecture supports a number of independent and asynchronous instruction streams, NON-VON 3 executes in a strictly synchronous, *single instruction stream, multiple data stream* (SIMD) [7] manner.

III. Organization and Interconnection of the PPS Chips

The tree-structured NON-VON 3 PPS is implemented using a single type of custom nMOS chip, called the PPS chip, which contains eight PE's and a single, shared instruction pre-decoder. Our use of a single chip type is made possible by the adoption of a tree-partitioning scheme first suggested by Leiserson [8]. This approach embeds on each chip both a complete subtree (containing 2^c-1 constituent PE's, for some c depending on device dimensions) and a single internal tree node, which we will call the *solitary* node. Four nine-bit busses (eight bits for data, and one for control) enter the chip. One, called the T connection, leads to the root of the chip's subtree, while the other three, called the F, L and R connections, attach the single solitary node to its father, left child, and right child, respectively, within the tree.

A simple recursive procedure allows the construction of a complete binary tree of arbitrary size using only chips of this type. This construction is illustrated for the case of two chips in Figure 1. Note that the resulting circuit consists of a larger complete binary subtree (in this case rooted by the solitary node of the chip on the left side of Figure 1), together with a single unconnected solitary node (the solitary node of the chip on the right). This circuit has the same four external connections -- T, F, L and R -- as did a single chip. This procedure may be repeated recursively to construct a PPS comprising 2^b-1 PE's, for arbitrarily large b, and leaves only one solitary PE unused, independent of the value of b.

Insert Figure 1 (Interconnection of Two Leiserson Chips) here.

An annotated microphotograph of the actual PPS chip is shown in Figure 2. Since all PE's have the same top-level floor plan (differing only in certain lower-level details, as will be discussed shortly), the internal PE structure has been shown only for the solitary PE.

Insert Figure 2 (Microphotograph of the NON-VON 3 PPS Chip) here.

The shared instruction pre-decoder translates the 8-bit vertical format used to

transmit instructions between chips into a somewhat more horizontal 11-bit format which is used within the chip. Because this function need be performed only once per chip, and not once per PE, the technique of pre-decoding results in a reduction in the amount of area required for local PLA's without an increase in pinout. Pre-decoding is performed, however, only for the PE's of the on-chip subtree, and not for those of the solitary PE, since NON-VON's tree pipelining scheme in general causes a different instruction to be executed at each level of the tree at any given point in time. Since the root of the subtree is always at a different level than the solitary PE, the instructions to be executed by these two PE's at any given point in time may differ, and must thus be separately decoded.

Instructions can not be fully decoded by the shared pre-decoder, since different control lines are excited to execute a given instruction depend on whether the PE is a leaf or an internal node of the tree, and on whether it is a left or right child of its parent. By "factoring out" part of the decoding function that does not depend on the position of the PE within the tree, however, pre-decoding results in a reduction of both pinout and area.

IV. Organization of the Processing Element

The NON-VON PE includes a 64 word X 9-bit *random access memory*; a set of five 8-bit *byte registers*, called A8, B8, C8, IO8, and IMAR; a set of five 1-bit *flag registers*, called A1, B1, C1, IO1, and EN1; an eight-bit *arithmetic logical unit* (ALU), a *programmable logic array* (PLA) used for instruction decoding and other control functions, and two special combinational networks, called the I/O switch and the RESOLVE circuit. A top-level block diagram of the PE is presented in Figure 3.

Insert Figure 3 (Block Diagram of the Processing Element) here.

With the exception of IMAR and EN1, all of the byte and flag registers may be used as general-purpose registers. In addition, however, all ten registers have special functions (in some cases supported by special hardware), which will become clear in the context of our description of the instruction set. It will sometimes be

convenient to refer to the pair of registers A8 and A1 as if they were one compound register, to which we will apply the unsuffixed name A; the register pairs B, C, and IO are defined similarly.

EN1 is distinguished as the *enable flag*. This flag is used to activate and deactivate individual PE's within the PPS. In general terms, only those PE's whose enable flags are asserted will respond to instructions broadcast by the CP. If EN1 is set to 0 in a particular PE, all instructions except one (the ENABLE instruction, discussed below) will be ignored. A number of tricky issues arise in considering the behavior of enabled and disabled PE's, particularly in the case of inter-PE communication operations. The precise semantics of communication between enabled and disabled PE's will be described in Section V.

Two internal buses, called the A bus and the IO bus, are used to transfer data within the PE. Both are capable of transferring either one- or eight-bit data, depending on the instruction being executed. The PE's dual-bus organization is required to support NON-VON's inter-PE communication instructions, which employ one bus to send data to a "neighboring" (in a sense that will soon be described more precisely) PE, and another to receive data concurrently from a different neighbor. The A bus connects all of the registers, the RAM, the ALU, and the I/O switch. The IO bus is also connected to the ALU and I/O switch, but is connected to only a subset of the register pairs -- specifically, A and IO -- which are dual-ported to allow transfers across both buses.

The I/O switch is a matrix of pass transistors that routes data between the two internal buses and the three inter-PE communication buses (parent, left child, and right child) in the course of executing inter-PE communication instructions. Depending on the particular instruction, these switches may be configured in such a way as to support one of several different logical communication topologies, as discussed in Section V. The RESOLVE circuit is another combinatorial network that is used to realize a particular instruction that selects a single PE out of a set of "marked" PE's, which in practice have typically been identified through a process of associative matching. The semantics of the RESOLVE instruction will also be described in Section V.

Finally, the local RAM comprises a 64 X 9-bit array of 6-transistor static RAM cells, together with control and decoding circuitry that allows access to one 8- or 1-bit location per instruction cycle. The instruction set supports up to 256 8-bit and 256 1-bit RAM locations. A production version of the chip would in fact include a 256-bit RAM; in order to minimize experimental fabrication costs by reducing die size, however, only 64 9-bit locations, implemented as two arrays of 32 locations each, were included in the current prototype chip.

In a production version containing 256 RAM locations, our measurements (presented in Section VI) indicate that just over half of the die area would be used for RAM cells. Using a simple Poisson model to estimate the incidence of defects, it was determined that the incorporation of fault-tolerant circuitry within the local RAM of each PE would very significantly increase the yield of working PPS chips. Moreover, the massively parallel structure of the NON-VON PPS offered a unique opportunity to recover not only from fabrication defects, but also from failures due to such phenomena as metal migration, which frequently occur in the course of circuit operation.

Our approach is based on the simultaneous testing and dynamic reconfiguration of the local RAM's of all PE's each time the machine is "powered up" or a memory error is detected. In addition to the advantages of dynamic failure recovery, this technique is implemented using a standard nMOS process, obviating the need for laser customization, the incorporation of fusible links, or other special processing of the sort used to manufacture most commercially-available fault-tolerant RAM's.

As in conventional fault-tolerant RAM's, each of the two 32-word RAM arrays is provided with a "spare" column, which may substituted for any single column which is found to be faulty. A maximum of 16 defective RAM columns may thus be tolerated within each PPS chip (providing, of course, that the faults are distributed perfectly across the RAM arrays -- an unlikely event in practice). If any of the RAM arrays embedded in a given chip contains more than one faulty column, however, the chip can not be reconfigured for correct operation, and must be discarded. In contrast with conventional RAM's, the column to be replaced is selected not by physically reconfiguring the hardware, but by setting an particular

flag according to the results of a parallel test program that concurrently tests a common 8-bit location in all PE's. (No fault-tolerance is provided for the 1-bit RAM array.) This process is repeated 64 times to test all locations.

It is worth noting that the practicality of the dynamic RAM configuration algorithm is critically dependent on the extensive, fine-grained parallelism offered by the NON-VON machine, and would not be practical in a sequential system embodying a comparable amount of RAM. By way of illustration, we consider the case of a production NON-VON machine containing a million PE's, each embodying 256 bytes of local RAM. We assume that 100 NON-VON instructions are required to test each byte of RAM (including tests of its functionality while different values are stored in neighboring cells). At 250 nanoseconds per instruction, approximately 6.4 milliseconds would be required to test the entire machine in parallel, independent of the number of PE's. By way of comparison, a sequential machine having an instruction cycle time of 250 nanoseconds would require nearly two hours to perform the equivalent tests on the same amount of RAM.

V. Operation of the PPS Chip

The operation of the NON-VON 3 PPS chip is best understood by reviewing the PE instruction set. The semantics of each instruction are described below, along with the set of permissable operands, where appropriate.

INSTRUCTION	SEMANTICS
MOV8 <byte reg 1> <byte reg 2> **<byte reg> = {A8, B8, C8, MAR, IO8}**	**<byte reg 2> <- <byte reg 1>**
MOV1 <bit reg 1> <bit reg 2> **<bit reg> = {A1, B1, C1, EN1, IO1}**	**<bit reg 2> <- <bit reg 1>**

The MOV8 and MOV1 instructions transfer are used to transfer data between bit and byte registers within the PE.

READRAM8 <byte reg>	<byte reg> <- RAM8 (IMAR)
WRITERAM8 <byte reg>	RAM8 (MAR) <- <byte reg>
<byte reg> = {A8, B8, C8 or IO8}	
READRAM1 <bit reg>	<bit reg> <- RAM1 (IMAR)
WRITERAM1 <bit reg>	RAM1 (MAR) <- <bit reg>
<bit reg> = {A1, B1, C1 or IO1}	
INCREMENT	MAR <- MAR + 1

The READRAM and WRITERAM instructions are used to transfer data between a register and the RAM location whose address is stored in the "incrementing memory address register", IMAR. The INCREMENT instruction adds one to the address stored in the IMAR (The IMAR may also be "auto-incremented" by executing a string loading or string matching instruction, both of which will be discussed later in this section.)

ADD <byte reg>	C8 <- (<byte reg> + A8 + C1); C1 <- carry
SUB <byte reg>	C8 <- (A8 - <byte reg> - C1); C1 <- borrow
COMPARE <byte reg>	if <byte reg> = A8 then A1 <- 1 else A1 <- 0; if <byte reg> > A8 then B1 <- 1 else B1 <- 0

<byte reg> = {B8, IO8, MAR, or RAM}

The ADD, SUB and COMPARE instructions may be used to perform arithmetic and comparison operations on two 8-bit operands. The carry bit must be cleared before these instructions are initiated. The results of a COMPARE are stored in the A1 and B1 flags.

ROTRB	Rotate B right 1 bit
ROTLB	Rotate B left 1 bit

The B8 and B1 registers contain logic enabling them to function together as a 9-bit circular shift register. Specifically, ROTRB shifts all but the low-order bit of B8 into the next lowest bit position within B8, the low-order bit of B8 is moved into

B1, and the value previously stored in B1 is moved into the high-order bit of B8. ROTLB performs an analogous left circular shift. The B register can thus be used to transfer information between the 8- and 1-bit data paths.

```
LOGICAL8  <operation>          C8 <- (A8 <operation> B8)
LOGICAL1  <operation>          C1 <- (A1 <operation> B1)
```

where <operation> is a four-bit code specifying one of the sixteen possible boolean functions of two variables. LOGICAL8 applies the specified operation in a bitwise fashion to all eight bits of its operands. Special cases of the LOGICAL1 instruction include SET, CLEAR, NEGATE, AND, OR, XOR, EQU, NAND, NOR and NOP. LOGICAL1 may be used to combine the results of a COMPARE instruction to test all six possible arithmetic relational predicates (EQ, NE, GT, LT, GE and LE) on two 8-bit operands.

```
SEND8    <PE>    <byte reg>        IO8 (<PE>) <- <byte reg>
SEND1    <PE>    <bit reg>         IO1 (<PE>) <- <bit reg>
RECV8    <PE>    <byte reg>        <byte reg> <- IO8 (<PE>)
RECV1    <PE>    <bit reg>         <bit reg>  <- IO1 (<PE>)
<byte reg> = {A8, B8, C8, MAR}
<bit reg> = {A1, B1, C1, EN1}
<PE> = {LC, RC, LN, RN} for SEND instructions
       {LC, RC, LN, RN, PR} for RECV instructions
```

The SEND and RECV instructions are used to transfer data in parallel not only between PE's that are physically adjacent within the PPS, but also between two PE's that are adjacent in a particular linear sequence defined by an inorder traversal [9] of the nodes of the PPS tree. In both cases, data is transferred between some register in the PE in which the instruction is executed and the IO register within some neighboring PE. It is always possible to RECV data from a PE, regardless of whether it is enabled, but an attempt to SEND data to a disabled PE will not result in a transfer of data.

In the case of tree communication, the neighboring PE is chosen to be either the left child (LC), right child (RC), or parent (PR). (PR is disallowed as the operand

of a SEND instruction, however, since the semantics of this operation would be ill-defined in the case where both children simultaneously attempted to transfer information to their common parent.) In the case of linear neighbor communication, the neighboring PE is chosen to be either the left linear neighbor (LN) or right linear neighbor (RN), which respectively designate the PE's occuring immediately before and immediately after (in inorder traversal order) the PE executing the instruction.

```
BROADCAST8  <byte reg>  <byte>          <byte reg> <- <byte>
BROADCAST1  <bit reg>   <bit>           <bit  reg> <- <bit)
REPORT8     <byte reg>                  logical reg. GG8 (in CP) <- <byte reg>
REPORT1     <bit reg>                   logical reg. GG1 (in CP) <- <bit reg>
<byte reg> = {A8, B8, C8, MAR, IO8}
<bit reg> = {A1, B1, C1, EN1, IO1}
```

The BROADCAST instructions are used to transfer a single data value from the control processor into a specified destination register within all enabled PE's simultaneously. The REPORT instructions, on the other hand, are defined only when exactly one PE is enabled, and result in the transfer of data from the specified register within that PE to a particular "logical register" within the control processor, which is called GG.

```
RESOLVE                          A1 <- 0 in all PE's except
                                    'first' PE where A1 = 1;
                                 if no PE has A1 = 1 then
                                    logical register R1 (in CP) <- 0
                                    else R1 <- 1
```

In many NON-VON programs, a subset of the PE's, often identified by associative matching, are "marked" by storing a 1 in one of the flag registers or one-bit RAM locations. The RESOLVE instruction can be used to identify exactly one of the members of a set of PE's that have previously been marked using the A1 flag. After execution of a RESOLVE, the A1 register is reset to zero in all PE's except the one that occurs first in inorder traversal order. The RESOLVE instruction is frequently used in conjunction with REPORT to read data into the control processor from each one of a set of PE's in turn.

ENABLE — **EN1 <- 1 in all PE's, including those previously disabled**

ENABLE is the only instruction that is executed by all PE's, whether or not they are already enabled. It is used to set all of the the EN1 registers to 1, thus "awakening" all PE's in the PPS after some subset have been disabled.

STRINGBROADCAST <length> <string>
STRINGREPORT <length>
STRINGCOMPARE <length> <string>

The semantics of these three operations are described below

The three string operations use the auto-increment capability of the MAR to perform highly efficient loading, unloading, and matching operations on successive locations in RAM. The appropriate opcode is broadcast to the PPS only once per string. Simultaneously, the control processor raises a special control wire, called ISYNC, which causes all enabled PE's to enter a special state in which they are prepared to accept a string of data bytes without intervening opcodes, incrementing their respective MAR's after each byte. While the last byte is being broadcast, the control processor lowers ISYNC to cause all PE's to exit this special state.

The STRINGBROADCAST instruction transfers a common string from the control processor into the local RAM's of all enabled PE's, starting at the location specified by their respective MAR's. (In the case of applications involving records having variable-length fields, these starting addresses will not in general be the same in all PE's.) STRINGREPORT functions in a similar manner, but is used to transfer a string into the control processor from a single enabled PE. STRINGCOMPARE compares a string broadcast by the control processor in parallel against those stored in all enabled PE's. If a data byte is broadcast that fails to match the corresponding byte stored within some PE, the EN1 flag is set to 0 in that PE, disabling it for the remainder of the matching process. At the end of the STRINGCOMPARE instruction, only those PE's containing a matching string are left enabled.

Most of the instructions in the NON-VON instruction set are expected to require approximately 250 nanoseconds for execution, independent of the size of the PPS

tree. This rate is attained by pipelining the broadcast of all but a few of the instructions, latching each instruction at every level within the tree. Data broadcast to all PE's from the control processor is also pipelined down the tree. The linear neighbor communication instructions, however, along with RESOLVE and REPORT, make global use of the PPS, and are not pipelined. The pipeline must be "flushed" before these instructions are executed, requiring a longer instruction period. The exact length of this period will depend logarithmically on the number of PE's embodied in the particular NON-VON configuration at hand. In addition to NON-VON's level-by-level tree pipelining scheme, the design provides for the overlapped fetching and execution of instructions within each PE.

VI. Design and Fabrication

The datapath in each PE is identical, and was laid out manually using a graphical editing system to achieve the highest possible packing density. Four different variants of the control path were necessary, however, to accomodate the somewhat different behavior of leaves and internal nodes, and of left and right children. In order to produce these variants without introducing human error, and in order to shorten the development cycle to allow extensive experimentation with alternative instruction sets, a semi-automatic layout program called PLATO [10] was developed.

PLATO accepts as input a set of instruction opcodes, together with associated control information, and produces as output a functionally correct, highly area-efficient set of PLA's. Among the novel aspects of the program are the use of a channel routing algorithm to generate a Weinberger Array layout for the PLA and the generation, from a single input description, of different PLA variants corresponding to processing elements serving different functions within the machine.

The chip was implemented using 3 micron nMOS design rules, and was fabricated through the MOSIS "silicon brokerage" system, which is operated by the Information Sciences Institute under contract with the Defense Advanced Research Projects Agency. The chip contains about 65,000 transistors, and has a functional pinout of 53. The die measures 8.1 X 7.4 mm, of which approximately 19% is occupied by RAM. A production version of the chip containing 256 bytes of local RAM per PE would thus require only about 2.1 times as much area as would be

occupied by an equivalent amount of ordinary static RAM. The chip dissipates approximately 3.4 watts of power, requiring the use of finned packages and forced air cooling.

VII. Conclusion

The NON-VON 3 PPS chip achieves a very high processor density while still attaining high execution speeds on those operations that appear to be most important in massively parallel SIMD tree algorithms. A production version of the chip would form the basis for an "intelligent memory" costing only about twice as much as an equivalent amount of ordinary static RAM, but incorporating an eight-bit processing element along with every 256 bits of storage. A number of novel architectural features are incorporated in the design, many of which should prove applicable to other massively parallel machines.

Acknowledgement

Many members of the Columbia's NON-VON Project contributed to the work described in this paper. The authors would like to give particular acknowledgement, however, to the major contributions of John K. Lai, Brian Mathies, and Robert Montay, who had principal responsibility for the NON-VON 3 PPS chip layout, and for the development of some of the software tools used in its generation.

This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-80-G-0132, and in part by the New York State Center for Advanced Technology in Computers and Information Systems at Columbia University.

References

[1] P. M. Flanders, D. J. Hunt, S. F. Reddaway, and D. Parkinson, "Efficient high-speed computing with the Distributed Array Processor", in *High Speed Computer and Algorithm Organization*. New York: Academic, 1977, pp. 113-128.

[2] J. L. Potter, "Image processing on the Massively Parallel Processor", *Computer*, vol. 16, no. 1, Jan. 1983

[3] D. E. Shaw, "The NON-VON Supercomputer", Technical Report, Columbia Computer Science Department, August, 1982.

[4] W. D. Hillis, "The Connection Machine", Technical Memo, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, September, 1981.

[5] T. Kondo, T. Nakashima, M. Aoki and T. Sudo, "An LSI adaptive array processor", *IEEE Journal of Solid-State Circuits*, vol. SC-18, no. 2, pp. 147-156, Aug. 1983.

[6] D. E. Shaw, "SIMD and MSIMD variants of the NON-VON supercomputer", Proc. COMPCON '84, San Francisco, California, Feb. 1984.

[7] M. Flynn, "Some Computer Organizations and their Effectiveness", in *IEEE Transactions on Computers*, vol. C-21, pp. 948-960, September, 1972.

[8] C. E. Leiserson, *Area-Efficient VLSI Computation*, Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon University, October 1981.

[9] D. E. Knuth, *The Art of Computer Programming, vol. 1: Fundamental Algorithms*, Addison-Wesley, 1969.

[10] T. M. Sabety, B. Mathies and D. E. Shaw, "The Semi-Automatic Generation of Processing Element Control Paths for Highly Parallel Machines", Proc. 21st ACM/IEEE Design Automation Conference, Albuquerque, New Mexico, June, 1984.

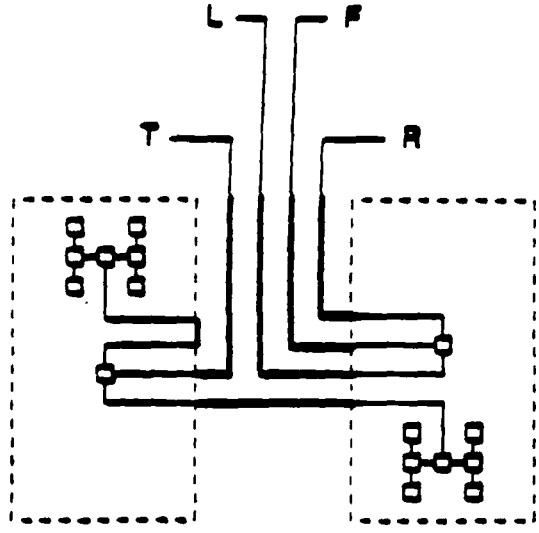

Figure 1. Interconnection of Two Leiserson Chips

D E Shaw and T M Sabety

PLEASE NOTE:

Our microphotograph of the eight-PE chip was not received in time to be submitted along with the review copy of this paper. In order to allow the paper to be reviewed without delay, we have substituted a microphotograph of the previous, four-PE version of the chip for now, and will replace it with the eight-PE microphotograph, which should arrive any day, if the paper is accepted for publication.

We have attached an overlay that should be printed in white over the microphotograph itself to identify different portions of the chip. Although the two images have not yet been superimposed, it should not be difficult for the referees to discern the structure of the chip using this separate overlay.

We apologize for any inconvenience resulting from the incomplete state of this figure.

- D. E. Shaw and T. M. Sabety

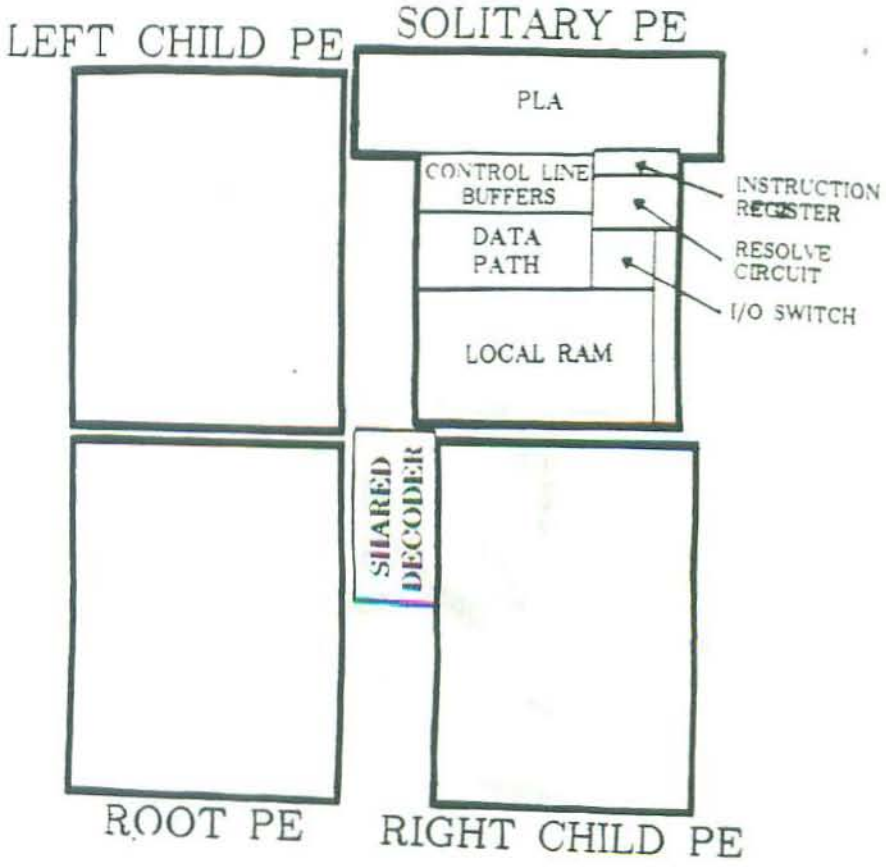

Figure 2. Microphotograph of the NON-VON 3 PPS Chip (Overlay)

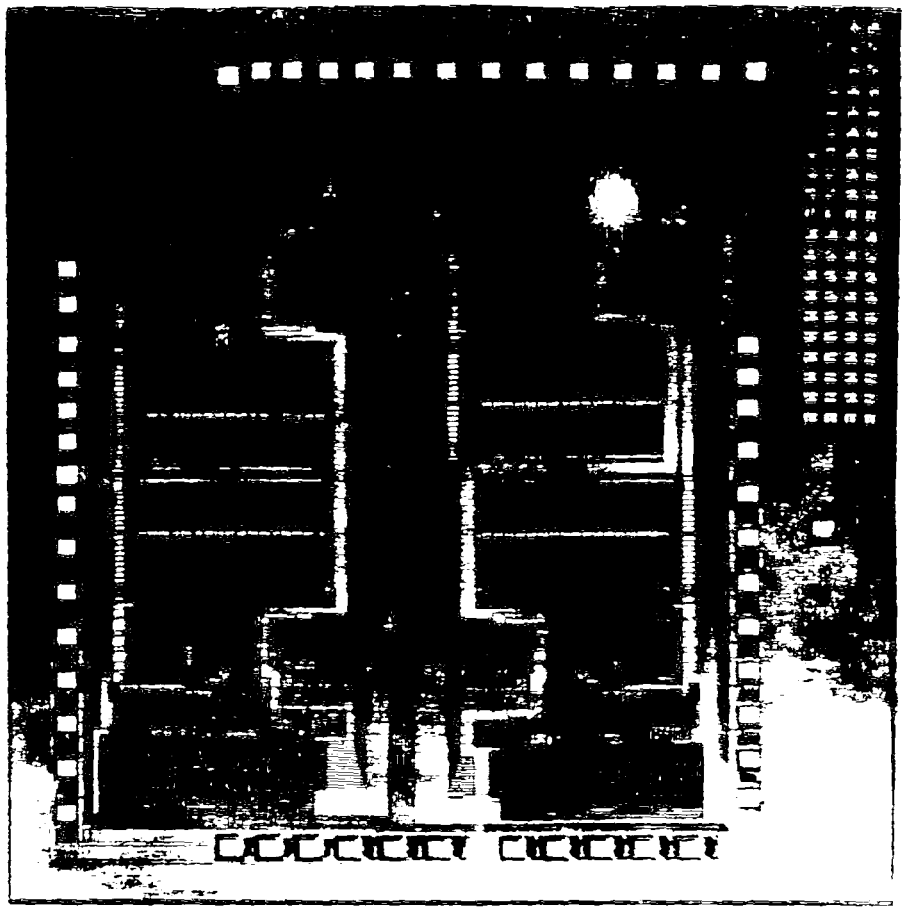

Figure 1. Microphotograph of the NON-VON 3 PPS Chip (Photo: D. E. Shaw and T. M. Sabety

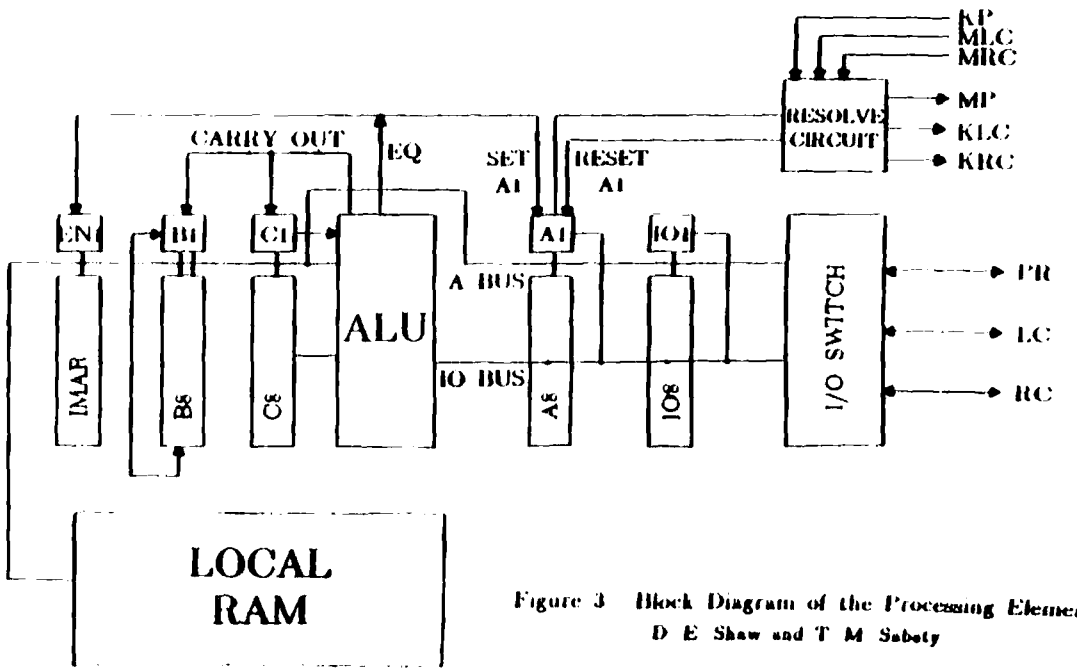

Figure 3 Block Diagram of the Processing Element
D E Shaw and T M Sabety