Proceedings of the 1985 Winter Simulation Conference
D. Gantz, G. Blais, S. Solomon (eds.)

SIMULATION OF A PYRAMID PROCESSOR

Duane R. Ball
Gerard C. Blais
Federal Computer Performance
Evaluation and Simulation Center
Alexandria, VA 22310

David Schaefer
Gregory Wilcox
George Mason University
Fairfax, VA 22030

R. Neil Wagoner
Scott AFB, IL 36114

This paper describes a simulation model of the MPP Pyramid [1] that has been constructed at the George Mason University. This model has been designed to facilitate experiments in processor control and in instruction set design.

I. SOME CONCEPTS OF COMPUTER ARCHITECTURE

The architecture of computer systems has remained virtually unchanged since von Neumann developed the stored program model in the 1940's. This may, however, change dramatically in the near future. Most advances in computing have been the result of reducing the cycle time of the switching devices which comprise digital computers. Over the past forty years, cycle times for digital computers have been reduced by five orders of magnitude from hundreds of microseconds to approximately ten nanoseconds. The practical application of artificial intelligence techniques, the next anticipated advance in computing, will require even faster computers.

Further decreases in cycle time may be very difficult to obtain. An electrical pulse travels about one foot (the length of this page) in a nanosecond. To decrease cycle time to one nanosecond, an additional order of magnitude, will require a computer with a maximum distance between components of less than one foot to permit pulses to travel between components within a clock cycle. Because of the high degree of circuit connectivity within a digital computer, the entire computer, thus, must be contained within a one foot diameter sphere. To decrease the cycle time to one-tenth of a nanosecond, a second order of magnitude, will require that the computer be built in an volume delimited by a 1.2 inch sphere. This increasing miniaturization will be accompanied by increases in heat. Within a logic family, an increase in speed is always accompanied by an increase in the amount of heat generated by each component. Preventing very fast computers from melting will itself be a major engineering challenge.

A second technique for speeding up computation is to have a number of processing units operating simultaneously. The traditional architecture for digital computers, developed by von Neumann, involves a "data fetch", "operate", and "store result" cycle. This type of architecture is called single instruction single data (SISD). Because fetching data and storing results takes a disproportionate amount of time in the SISD computing cycle, architectures which operate on more than one unit of data simultaneously have been developed. This permits multiple data fetch and result storing operations to be overlapped.

To develop a new architecture, four major design issues must be addressed: (1) the number and power of the processing elements; (2) the interconnection of the processing elements; (3) the scope of memory (shared or local); and (4) the method of control.

One class of architectures, called "spatially parallel", combines a large number of identical processing elements [2]. Until recently, most spatially parallel computers were interconnected as two dimensional arrays of computing elements. These arrays of computing elements are generally called array processors. For problems like numerical solution of partial differential equations, this method of interconnection is generally satisfactory. The nature of these problems requires only very localized communications between processing elements. However, for problems in artificial intelligence (e.g., image recognition, expert systems), distant processors must be able to communicate easily.

The pyramid architecture is an attempt to combine a large number of processors in a way so that information can be transferred between them quickly. A pyramid is composed of a number of layers. The top layer consists of a single processor. Each successive layer contains k-times as many processing elements as the one above it. A processing element within a layer can directly communicate with the other processing elements immediately adjacent to it on the same layer. Thus, a pyramid processor can be visualized as a three dimensional structure containing interconnected layers of two dimensional processing arrays.

The scope of memory for multiprocessor systems can be local, global, or mixed. If the memory has local scope then each processor has a private memory and communicates with other processors by passing data as messages via the interprocessor buses. If the memory has global scope then all processors share a common memory. In some multiprocessor systems, part of the memory may be local while other elements of memory are global. This is called mixed scope.

A computer's control mechanism determines the sequence of instruction execution. Two of the most popular control mechanisms are control flow and data flow. In control flow systems, the instruction to be executed is pointed to by an instruction address register. Normally the instruction address register is incremented after each instruction is executed. This causes the instruction in the next memory address to be executed. A branch instruction simply puts a new memory address in the instruction address register causing the flow of control to "jump" to the new address. Another popular control mechanism is data flow. In data flow systems, an instruction is executed when all of its operands become available. Control flow is an example of synchronous control while data flow is an example of asynchronous control.

In a multiprocessor system, a decision must be made regarding the scope of the control mechanism. In one scheme, all of the processors are controlled by a single control mechanism. Each processor executes the same instruction on the operands available to it. This is called single instruction multiple data (SIMD) control. Another scheme uses a separate controller for each processor or for a number of groups of processors. This is called multiple instruction multiple data (MIMD) control.

II. ARCHITECTURE OF THE MPP PYRAMID

The Massively Parallel Processor (MPP) [3], a historical predecessor of the MPP Pyramid, is a single-instruction-stream multiple-data-stream (SIMD) computer designed for the rapid and economical extraction of information from data, especially data in the form of images. The MPP was built by Goodyear Aerospace Corporation for the NASA Goddard Space Flight Center.

The logical building block of the MPP is the Processing Element (PE), a one-bit computer with local memory, and local connectivity. The MPP consists of a square (128 x 128) array of 16,384 PEs, each of which has nearest neighbor communication. Operating on eight bit integers, the MPP can achieve speeds measured in billions of operations per second. A conventional computer, currently a Digital Equipment Corporation VAX, provides supporting control and data management functions.

Custom integrated circuits, used in the MPP, and each containing eight PEs, were available to George Mason University, and hence were selected to be the basis of the MPP Pyramid. Each PE has six one-bit registers (labeled A, B, C, G, P, and S); a variable length (up to 30 bits) shift register; a full adder; a random access memory with at least 256 bits of data; and a data bus. The six one-bit registers are used as follows:

- A - addend register, added to P and C
- B - stores the sum of P, A, and C
- C - stores the carry of P, A, and C
- G - mask register, used to inhibit operations
- P - boolean logic or inter-PE data routing
- S - input/output register.

The shift register is connected from the B register to the A register. Its length may be set, under program control, to 2, 6, 10, 14, 18, 22, 26, or 30 bits, providing a shift path length of 4, 8, 12, 16, 20, 24, 28, or 32 bits. The adder adds the contents of the A, P, and C registers (a full-add), or just the A and C registers (a half-add), and stores the sum and carry bits in the B and C registers, respectively. The data bus is selected, under program control, to transfer data among the registers, the local memory, or the Boolean logic generator associated with the P register. A sum-or circuit forms the logical or of all eight PE's on a chip; circuitry provides the sum-or bit for each level of the pyramid to the control processor.

Table 1 summarizes seven sub-categories of an MPP instruction; one instruction, for example, could add the A, P, and C registers; shift the shift register one position; "and" the P register with the data bus; and load G from the data bus.

The MPP PE's were designed to facilitate two-dimensional (i.e., planar) data transmission. In order to implement a pyramid, a new circuit was needed to permit inter-level data transfers. In the MPP Pyramid, each PE (except the top and bottom) is connected to nine neighbors: one above, four on its plane (N, E, S, and W), and four below (called V, X, Y, Z).

Data Bus Source	Memory C, B, P=G, S, P
P Register Logic	15 Boolean operations on P, data bus
P Register Routing	N, E, S, W
Adder	Full Add, Half Add, Set C, Clear C
Shift Register	Shift Set length to 2, 6, 10, 14, 18, 22, 26, 30
A Register	Shift A from Shift Register Load A from Data Bus Clear A
Miscellaneous	Load G from Data Bus Write Data Bus to Memory Load S from Data Bus Feed Sum-or from Data Bus Clear Parity Error

Table 1: MPP Operations

A Level Transfer Control Unit (LTCU) controls the transfer of data between levels. A control computer, external to the pyramid, stores and transfers 64-bit instructions to the pyramid. Seventeen bits become the operation code for each PE in the pyramid. Ten bits are used by the LTCU to identify a global transfer direction; select levels that will transfer data; and identify PE's within a level that will transfer data. A parent PE may transmit data to four children PE's simultaneously; or the parent PE's on a level may all simultaneously receive data from one of their V, X, Y, and Z children.

III. SIMULATION

The authors decided to develop an instruction-level simulation, or emulation, of the three-layer pyramid to provide multiple, simultaneous program development facilities to coders; and to provide a basis for exploring architectural alternatives for the five-level pyramid. The simulation would be most useful if it was inexpensive, not too slow, could run on computers commonly available, and provided a clear view of the internal state of the pyramid as simulated execution progressed.

Apple IIe microcomputers with UCSD-PASCAL were available to the authors, and to the George Mason University community. Three programmers implemented an emulation of the three-level pyramid in about 1,500 lines of PASCAL over a two month period. Total development time was approximately one hundred hours. Figure 1 depicts the structure of the simulation.

To use the model, a researcher prepares a text file of 64-bit machine language instructions, in either decimal or hexadecimal. The routines on the left of

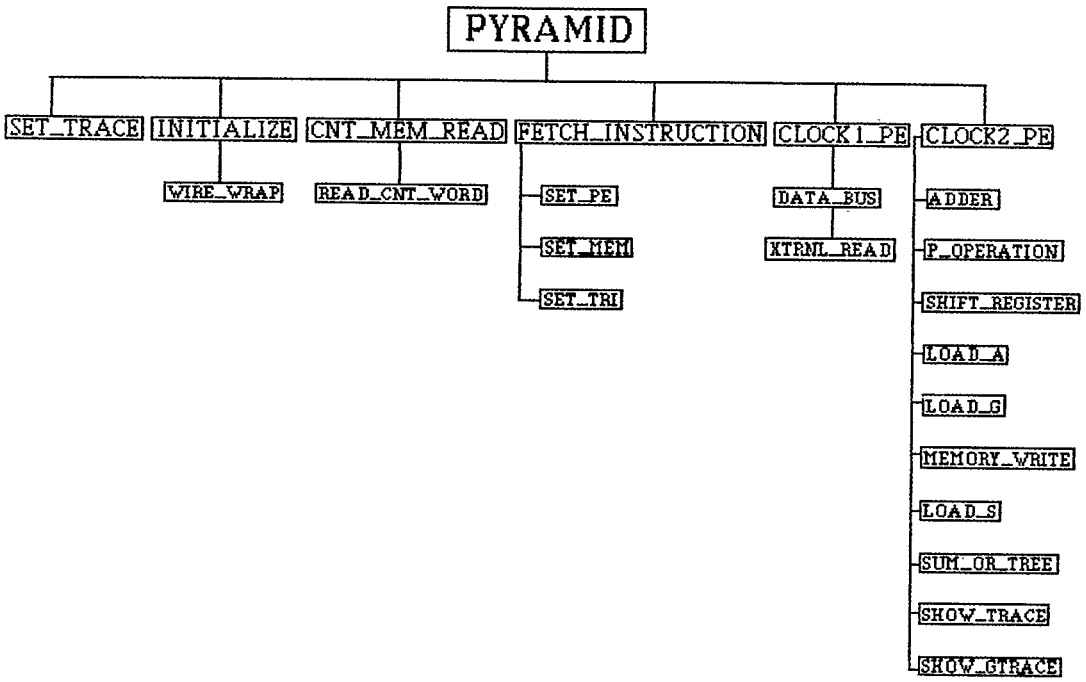

Figure 1

Figure 1 will read the pyramid program to be simulated (Cnt_Mem_Read and Read_Cnt_Word), zero the simulated registers and memories (Initialize), establish pyramidal connectivity among the simulated PE's (Wire_Wrap), and prompt the researcher to choose graphics or hard-copy trace, or both (Set_Trace).

The routines on the right of Figure 1 simulate a single instruction execution across all PE's. Fetch_Instruction increments the control processor program counter, and decodes the sixty-four bits which specify on-PE processing (Set_PE), which PE's will be masked (Set_Mem), and any inter-level transfers (Set_Tri). Each clock tick is processed by the simulation as a first half, and a second half. In the first half clock tick, Clock 1_PE initiates data bus transfers, and stores the values of the P and S registers for each PE.

Once Clock1_PE has looped through all of the PEs, Clock2_PE can simulate data transfers into each PE, and arithmetic and logical operations within the PE. The Adder routine simulates a full add (of registers A, P, and C) or a half add (registers A and C), and stores the results in the B and C registers. The P_Operation routine simulates any boolean operation on the P and D registers, and stores the result in P. The Shift Register routine simulates the variable length shift register described earlier. Load_A may clear the A register, or set it from either the D register or shift register. The Load_G routine can set the G register to the contents of the D register. The Memory_Write routine stores the D register in a local memory location. The Load_S routine may clear the S register, or set it to the contents of the D register, a neighbor's S register, or a bit in the instruction word. The Sum_Or_Tree routine extracts the contents of the D register to from an input to the logical sum-or of all PEs on a level. The Show_Trace and Show_Gtrace routines implement hard-copy and CRT traces of program executions, respectively.

Careful sequencing is required to correctly simulate up or down inter-level transfers. During a transfer down, Clock2_PE simulates the on-chip processing for each PE starting at the top. Functions and transfers which require the value of the P or S register at the start of the clock tick access the old contents stored by Clock1_PE. This is all that is required to serialize the parallel processing of the pyramid. During a transfer up, Clock2_PE begins at the bottom, and processes PEs in reverse order.

Figure 2 depicts a graphical snapshot of simulated execution. In the figure, 0's and X's on the left represent the contents of the P-register for each PE in the pyramid. The line labeled 0 through F depicts sixteen bits of the current PE instruction, and "a b c g p s" depict the six registers for the current PE, in this case number 9. The eight numbers across the bottom are the entire sixty-four bit controller instruction, shown as decimal values for each eight bits.

IV. CONCLUSIONS

Currently, the model takes approximately fifty seconds to simulate a clock tick, approximately five hundred million times slower than the native speed of the twenty-one PEs in the three-level pyramid. Even though most algorithms written to date for the three-level pyramid are relatively short, under a hundred instructions, this speed is too slow for practical use. The authors are nontheless quite pleased that a simulation this complex could be done at all on a machine as small as an Apple IIe. Several possibilities for significant speed-up are readily available.

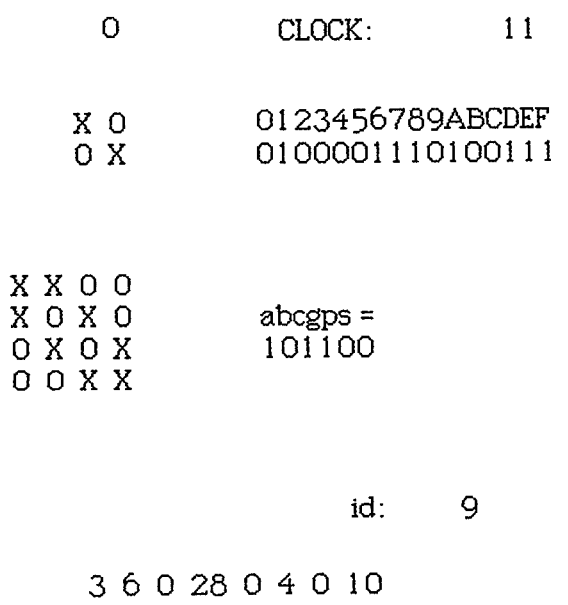

Figure 2: Pyramid Processor Simulation

Fidelity goals, and ease of use goals, were met. A text file of controller instructions for the simulator is identical to the text file for the same program that would be input to the pyramid controller. Several programs have been written and debugged on the simulator.

The relatively low investment in program development suggests that modification to explore MIMD, or different inter-level transfer control, could also be easily simulated.

Preliminary experiments with Turbo PASCAL and a Z-80 board for the Apple suggest a speed improvement of ten to twenty times is readily available. Two to three seconds to simulate a clock tick would probably make the simulator a useful tool. Simulation of the five-level, 341 PE pyramid will require another factor of ten improvement in speed; that may be achievable on a mini-computer, such as a VAX 750 or 780.

REFERENCES

1. Schaefer, David H., Wilcox, Gregory C., and Harris, Victor J., "A Pyramid of MPP Processing Elements - Experiences and Plans", Proceedings of the Eighteenth Annual Hawaii International Conference on System Sciences, 1985. Volume 1, pages 178-184.

2. Schaefer, David H., "Spatially Parallel Architectures: An Overview", Computer Design, August 1982, pages 117-124.

3. Potter, J.L. Editor, The Massively Parallel Processor, MIT Press, Cambridge, MA, 1985.

4. Wilcox, Gregory C., Pyramid Computer Systems. Master's Thesis, George Mason University, Fairfax, VA 1985.

5. "Functional Description of the MPP PE", internal publication of the Goodyear Aerospace Corporation, Akron, Ohio 44315, under contract NAS5-25392 to NASA/Goddard Space Flight Center. Document GER-16624, 1 December 1978.

6. Schafer, David H., Wilcox, Gregory C., and Harris, Victor J. "A Pyramid of MPP Processing Elements", Proceedings of the Workshop on Algorithm-Guided Parallel Architectures For Automatic Target Recognition, Leesburg, VA, July 1984. Prepared by C. Lee Giles, Naval Research Laboratory, Washinton, DC 20375 and Azriel Rosenfeld, University of Maryland, College Park, MD 20742, February 1985.