The Concept and Implementation of Data-Driven Processor Arrays

Israel Koren
Dept. of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003 (on leave from Technion, Haifa 32000, Israel)
Irit Peled
Dept. of Electrical Engineering, Technion, Haifa 32OOO, Israel

Various topologies and architectural designs for processor arrays have recently been proposed. These designs include systolic arrays and globally asynchronous wavefront arrays.[1] Both array types have a computation front that propagates according to a predetermined control sequence, and consequently these *control-driven arrays* have proven to be very effective for executing highly regular algorithms like vector and matrix operations. There are, however, many computationally demanding problems that do not exhibit high regularity and may therefore prove unsuitable for these control-driven arrays. Still, many of these problems have an inherent parallelism, and it should be possible to exploit this parallelism by means of processor arrays that can provide a high degree of pipelining.

In Koren and Silberman,[2] a new approach to array design was proposed—that of developing specialized array architectures that would be capable of executing any given algorithm. In this approach, the algorithm is first represented in the form of a *dataflow graph* (DFG) and is then mapped onto the array. The processing elements (PEs) in the array execute the operations included in the corresponding nodes (or subsets of nodes) of the DFG; regular interconnections of these PEs serve as edges of the graph.

In general, when an arbitrary algorithm is executed on an array there is no regular propagation of computation fronts. Hence, to speed up the execution of arbitrary algorithms, a more flexible array is needed. Such an array should make possible the generation of new computation fronts and their cancellation at a later time (the time depends on the arriving data operands). We therefore call these arrays *data-driven* arrays. The cell (that is, the PE) in these arrays should be capable of testing for the presence of its operands and executing only the instructions for which all the necessary operands have arrived. Thus, the order in which instructions are executed is data dependent, and the cell is truly a *data-driven* PE.

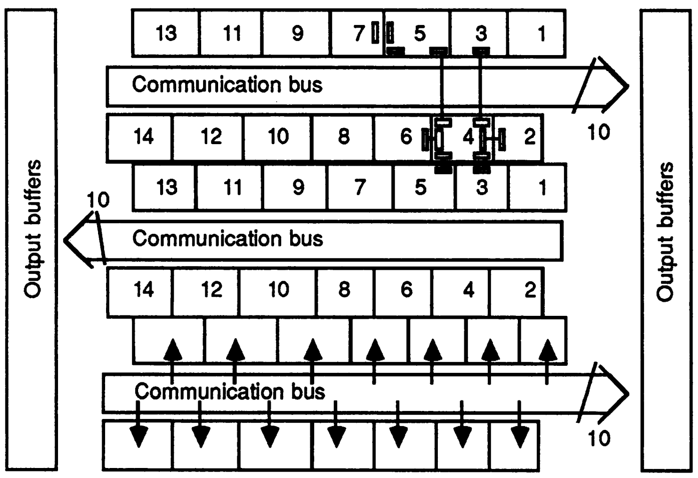

Figure 1. The floorplan of the processor-array chip.

Processor-array architecture and principles of its operation

The feasibility of designing control-driven arrays was never in question; several types of PEs for these arrays have already been designed (see Fisher et al.[3]). However, the degree of hardware complexity required to add the data-driven property was not clear to us. Therefore, we made a preliminary design of an appropriate processing element. The result of this design is very encouraging: The total hardware complexity of the cell—which is presented next—is less than 9000 transistors in NMOS technology. This low complexity should make possible the fabrication of a VLSI chip containing about 50 to 100 cells. The first phase of the design has already been completed and is reported in Peled.[4] A group of graduate students in the Dept. of Electrical and Computer Engineering at the University of Massachusetts in Amherst is now finalizing the detailed design and layout of the VLSI chip.

The proposed floorplan of the chip is shown in Figure 1. The floorplan contains data-driven cells arranged in rows in such a way that the typical cell has six immediate neighbors in a hexagonally connected processor array. The cells communicate with an external host computer through buses, as shown in Figure 1. In this host, the programs of the individual cells are prepared and are then distributed to the cells. The host also supplies the input operands and accumulates the final results. Unlike host-to-array communication in control-driven arrays, restricting the host-to-array communication to passing only through boundary cells can slow down the array's operation substantially. Consequently, we have decided to allow each cell to communicate directly with the host.

The functional blocks that make up the basic cell are depicted in Figure 2. There are six registers, R_1 through R_6, which are connected to a common internal bus; each of them is also directly connected to its corresponding register in one of the six neighboring cells (see Figure 2). The latter connection is under hardware control, since its timing is crucial. In our design, this data transfer takes a single clock and is done *in parallel* with all other operations in the cell.

The *instruction memory* contains six instructions that specify the cell's operations; one instruction per register (out of R_1 through R_6 registers). Having six instructions per cell increases the level of utilization of the cell and leads to a lower overall execution time.

The *flag array* is a uniquely designed block that makes possible the data-driven operation of the cell. The instructions in the cell are not executed in any predetermined order. Instead, the arrival of all operands for a certain instruction enables the cell to execute set instructions. The flags monitor the movement of operands both within the array and in and out of the cell. For each register there is a flag indicating whether the register has an operand or whether it is empty and can receive a new operand. Only a single cycle is needed to test these flags to determine whether an instruction is ready to be executed.

The other functional units in the cell are self-explanatory.

In parallel with designing the array, a procedure for mapping DFGs onto data-driven arrays has been developed and programmed by Mendelson and Silberman.[5] In this procedure, the user's program (in VAL) is translated into a DFG and then mapped onto a finite array of PEs. The procedure allows us to take advantage of the data-driven cell's capability of performing up to six operations. A node in the original DFG includes only a single operation. Therefore, we may combine up to six simple neighboring nodes and map them onto a single PE. Finally, the array is partitioned into several subarray chips according to the technology-imposed limitations on the number of PEs per chip.

In summary, the idea of directly mapping an arbitrary algorithm onto a VLSI array has been shown to be feasible. Further research is now being carried out to prove the effectiveness and practicality of this approach.

References

1. S.Y. Kung, "On Supercomputing With Systolic/Wavefront Array Processors," *Proc. IEEE,* July 1984, pp. 867-884.
2. I. Koren and G.M. Silberman, "A Direct Mapping of Algorithms Onto VLSI Processor Arrays Based on the Data Flow Approach," *Proc. 1983 Int'l Conf. Parallel Processing,* Aug. 1983, pp. 335-337.
3. A.L. Fisher et al., "Design of the PSC: A Programmable Systolic Chip," *Proc. Third Caltech Conf. VLSI,* March 1983, California Institute of Technology, Pasadena, Calif., pp. 287-302.
4. I. Peled, "A Data-Driven Processing Element," master's thesis (in Hebrew), 1986, Dept. of Electrical Engineering, Technion, Haifa, Israel.
5. B. Mendelson and G.M. Silberman, "Mapping Data Flow Programs on a VLSI Array of Processors," *Proc. 1987 Int'l Conf. Computer Architecture,* June 1987. Also published as a technical report in 1987 by the Dept. of Computer Science, Technion, Haifa, Israel.

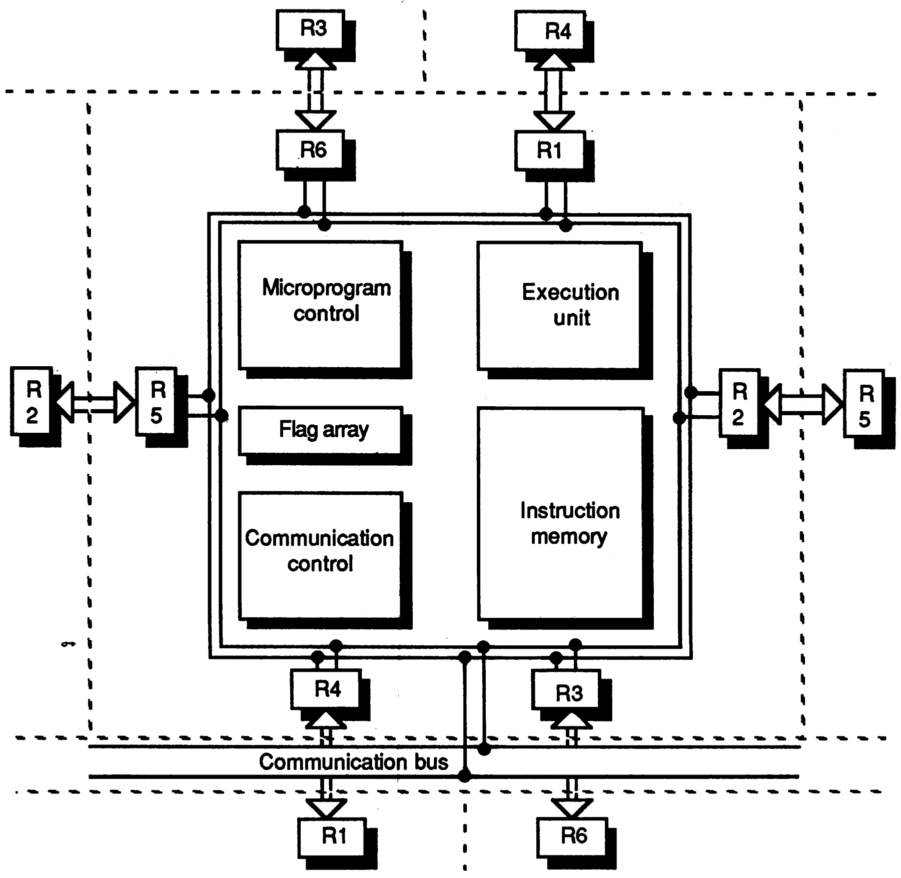

Figure 2. Functional blocks in the data-driven cell.