The Propagated Instruction Processor

T J Fountain & C D Tomlinson
Dept. of Physics and Astronomy
University College London

This work was funded by the ARPA ULTRA programme, grant No. N00014-93-1-1087

Abstract

As the dimensions of electronic devices become smaller, they enter the realm of nano-electronics, where the device dimensions lie between the inter-atomic spacing (1nm) and the electron wavelength (10nm). This offers greatly-enhanced packing density and speed of operation, but introduces the difficulty of maintaining signal quality over large distances. The paper introduces an architectural solution to this problem which is appropriate for the class of highly data-parallel computers [1,2,3]. In the propagated instruction architecture proposed here, instructions sweep across the array (and hence across the data set) in bands, one instruction following another closely. The paper examines the architectural implications of this idea, and assesses the benefits in the application area of image processing.

1 Introduction

Although most current Image Processing is carried out on serial computers, this represents a serious constraint on performance and hence on the applicability of the programs used. This situation can only get worse as typical image sizes rise from 512 x 512 (a quarter of a million) pixels to 4000 x 4000 (sixteen million) pixels, since the potential clock rates of serial systems are likely to be limited to about an order of magnitude improvement (from approx. 100 MHz to 1GHz).

Thus, the requirement for highly-parallel systems, which has always existed for the largest problems, is likely to become more widespread. However, the potential size of parallel systems implemented in conventional technology is limited by practical constraints such as volume and cost.

At present, partly in response to this problem, an evolution, which is threatening to become a revolution, is occurring in computer technology. The evolution arises from the continuing effort to achieve smaller and smaller dimensions for the electronic devices from which integrated circuits (and therefore computers) are made. Such devices are characterised by (amongst other things) a typical linear dimension which determines the smallest feature size of the integrated circuit. At present, a respectable value for this dimension is about 1000 nm (although leading-edge development technology utilises about 100 nm). However, it is apparent that, if dimensions continue to be reduced, the time will soon come where a revolution will be required in the way devices are designed and analysed, because each device will contain only a few atoms (perhaps as few as one in the limit). In this *nano-electronic* area, the quantum mechanical properties of atoms become significant.

Although the engineering developments which will underpin this revolution are still at an early stage, there is considerable work being carried

BMVC 1995 doi:10.5244/C.9.56

out world-wide in attempts both to achieve the technology and to understand how such devices will work and can be used [4]. In our programme, we are attempting to understand how the requirements of highly-parallel computers and the expected properties of nano-electronic devices can be matched [5]. The ideas put forward in this paper have not yet been subjected to complete and detailed analysis, but we regard the figures presented here as being correct in broad outline although subject to considerable refinement in detail.

2 Background

There is a natural match between the (anticipated) properties of nano-electronic devices and the architectural requirements of the class of highly-parallel computers variously known as *data-parallel, SIMD, MPPs* and *processor arrays*. A typical architecture for such a system includes the following aspects:

- A large number of low complexity PEs connected by a regular structure
- A single program sequencer from which control is simultaneously distributed to all PEs
- Local memory for each PE, which holds only data
- Special arrangements for input and output of data are provided

It is also important to note that, in such systems, the number of arrays of data which are input or output from the system is usually much smaller than the number of elements of the control program. A typical program would consist of the following elements:

(a) Input initial data set
(b) Read data from local memory - process data - store result in local memory
. "
Repeat for many different functions
. "
(n) Output resultant data set

In this sequence, operations (a) and (n) usually take much longer than other functions because data I/O occurs at the side of the array and data must be passed from PE to PE across the whole width of the array. However, because there are thousands of computations to be executed within the array, this can be tolerated. In some cases data input/output operations are overlapped with computation to reduce this overhead. It is also important to note that a single image processing operation may itself require on the order of 1000 clock cycles for execution on a typical bit-serial array. Thus, there can be many thousands of clock cycles between functions (a) and (n) in the above list [6,7,8].

The work of developing a technology to support the reliable manufacture of devices with feature dimensions between 10 nm and 1 nm is at a very early stage, and it is anticipated that many years will pass before success is achieved. The problem is that conventional transistors will certainly not work in the same way when they comprise only a few tens or hundreds of atoms. It has therefore been necessary to propose alternative structures whose operation is based on the quantum mechanical properties of lattice structures. In this area, the properties which can be expected to be of importance include the specific energy states in which an atom can exist, the probabilistic action known as *tunnelling*, the existence of an electron as a distributed *wave function* and the phenomenon of *resonance*. These, and many other quantum mechanical properties, are considered in detail in, for example, [9].

At the present time, the devices which are attracting the most interest fall

into a relatively limited number of categories. These include resonant tunnelling transistors, quantum effect transistors, electron waveguide devices, quantum well modulation devices and lateral quantum devices. A survey of device types and their properties is given in [10] but it is apparent that, although the present states of development of the various families are very different, the differences between some of the target properties are less significant. We can assume that the limiting dimension for such devices is determined either by the electron wavelength (about 10nm) or by the atomic lattice spacing (1nm).

Minimum feature	Device size	No. per linear cm.	No. per square cm.
1 μm	10 μm x 10 μm	1000	1 000 000
100 nm	1 μm x 1μm	10 000	100 000 000
10 nm	100nm x 100nm	100 000	10 000 000 000
1 nm	10nm x 10nm	1 000 000	1 000 000 000 000

Table 1 Packing densities compared

Extrapolating from current figures [11], we conclude that this equates to a range of minimum device sizes between 100nm x 100nm and 10nm x 10nm. Table 1 illustrates the improvements in packing density which can be expected.

Table 1 does not tell the complete story, since five other factors must be taken into account. The first concerns functionality. Present micro-electronic transistors have a simple step transfer function. This is unlikely to be so for nano-devices, which can be analogue in operation, multi-level, or even probabilistic [9]. Any one of these characteristics might correspond to about one order of magnitude improvement in packing density.

The second factor concerns the implementation of memory. For commercial reasons, enormous effort has gone into optimising this at the present micro-electronic level. This is not yet true for nano-electronic circuits, implying that the balance of resources allocated to memory and processing components might be somewhat different for a nano-electronic processor array. This factor is not, as yet, susceptible to numerical analysis.

The third factor concerns fault-tolerance. This is already a serious problem in highly-parallel computers, and it is anticipated that dedicated methods of improving fault-tolerance will have to be built in to any nano-electronic array design. We estimate that this will decrease effective packing density by an order of magnitude.

Fourth, the problem of interfacing to other electronic systems (sensors and conventional computing circuits) must be considered. By definition, nano-electronic devices will operate with current levels corresponding to the flow of a small number of electrons. To interface with normal circuits, suitable amplification will be required. It is for this reason that the processor array, which requires
interfaces only at its periphery, is deemed an appropriate application area for nano-electronics. The silicon area required for interfacing is likely to reduce effective packing density by a factor of two.

The final factor concerns the propagation of signals across a system. Micro-electronic transistors are designed with built-in gain, so that signals can be periodically (and automatically) refreshed. The characteristic impedance

of metal strips at the micro-electronic scale is well characterised, signal loss being acceptable over typical distances of about 1 cm. Neither of these is true at the nano-electronic scale. These factors imply that long-range signal transmission cannot be implemented directly in a fully nano-electronic system. This involves no difficulty for the local inter-processor data connectivity required in an SIMD array, but is likely to pose serious problems for the distribution of control signals.

- The number of control lines per element is 35

To examine the implications of this design, we consider the relative areas of silicon required for each of the major components, making the assumptions set out in Table 2. In Table 3 we display the results of our calculations concerning the number of processing units which can be implemented on 1 sq. cm of silicon. Because the eventual dimensions of nano-devices are at present somewhat uncertain, we provide figures corresponding to minimum feature sizes of 10 nm and 1 nm.

Component	Type of devices	Number of devices
Computing element	Nano-transistors	200
Memory	Nano-transistors	20 000
Control distribution	Buffered nano-wires	35

Table 2 Components for a conventional array

3 Conventional SIMD systems

The final paragraphs of the previous section suggest that there may be difficulties in implementing a straightforward SIMD array processor in nano-electronic technology. It is sensible, therefore, to examine such an implementation in more detail so as to specify the magnitude of the likely problems. We take, as the starting-point for all the designs considered in this section, the following assumptions:

- The effects of increased functionality and increased fault-tolerance will balance out
- The implementation of the computing element requires 200 conventional transistors
- 1 kbyte of memory is needed per PE, implemented by 20,000 conventional transistors
- The array will comprise a single device to reduce connections to external devices
- 50% of total silicon area will be given over to external signal buffering

The figures displayed in Table 3 seem satisfactory, but a serious drawback of this architecture exists in terms of performance. It would normally be assumed that, in an SIMD array, the distribution of control signals throughout the array of processors occupies negligible time compared to that taken by the actual computation. In the system outlined above, this would not be so. Because control information is distributed over buffered nano-wires, the time taken to distribute the signals is directly proportional to the number of processing elements in the linear dimension of the array. We present the implications of this in Table 3, based on the assumption of one clock delay per processing element. The performance penalty varies between a factor 50 and a factor 500, depending on the size of the array, which is obviously undesirable. Two conventional solutions can be proposed to address the problem.

Minimum feature (nm)	PE area (sq.μm)	Memory area (sq.μm)	Number of PEs	Time per binary function	Distribution time for control	Total execution time per function
10	2	200	500 x 500	10	500	510
1	0.02	2	5000 x 5000	10	5000	5010

Table 3 Performance penalty of buffered control distribution

The first conventional system is a hybrid of micro- and nano-technology, in which the control distribution wires are implemented in conventional micro-technology. In this work, we assume both a line width and a line spacing of 1 μm. Such a system recovers all of the performance loss of the nano-distribution system but there is a penalty in terms of packing density, as shown in Table 4. Comparison of these figures with those in Table 3 show a reduction in packing density by a factor ranging between 4 and 35. In particular, even the system based on the smallest possible feature size now has insufficient processors to achieve the desirable one-to-one mapping of data to processors for large data sets.

An alternative arrangement which might reduce the impact of the control distribution on packing density or performance is to distribute the control signals serially (along a single line) rather than in parallel, still using a micro-line to avoid propagation delays. This requires the same number of clocks as there are control bits - 35. Control signals are then parallelised at each processor by means of a suitable shift register. This arrangement has implications for both packing density and performance, illustrated in Table 4. The combination of properties seems much more encouraging (although a penalty over the ideal system by a combined factor of 7 is not negligible). However, it seemed possible to us that an alternative type of architecture (which we designate the propagated instruction processor) might offer an even better alternative.

Minimum feature (nm)	PE area (sq. μm)	Memory area (sq. μm)	Control line area (Serial) (sq. μm)	Control line area (Hybrid) (sq. μm)	Number of PEs (Serial)	Number of PEs (Hybrid)
10	2	200	20	700	500x500	230x230
1	0.02	2	2	70	3500x3500	825x825

Table 4 Packing densities for the conventional systems

4 The propagated instruction processor

A number of ideas contributed to the development of the propagated instruction processor. They include the basic SIMD processor array, the systolic array in which data is streamed across an array of fixed-function processors, the linear SIMD array which processes a square array of data row-by-row, the use of instruction pipelining which takes place in vector supercomputers, and the addition of local autonomy to each element of an SIMD system. Each of these is described in some detail in [12]. When combined in the manner described below, they result in the system whose

processor element is shown in Figure 1, a type of inverse systolic array in which successive instructions sweep across a static data array. The elements of the design are as follows.

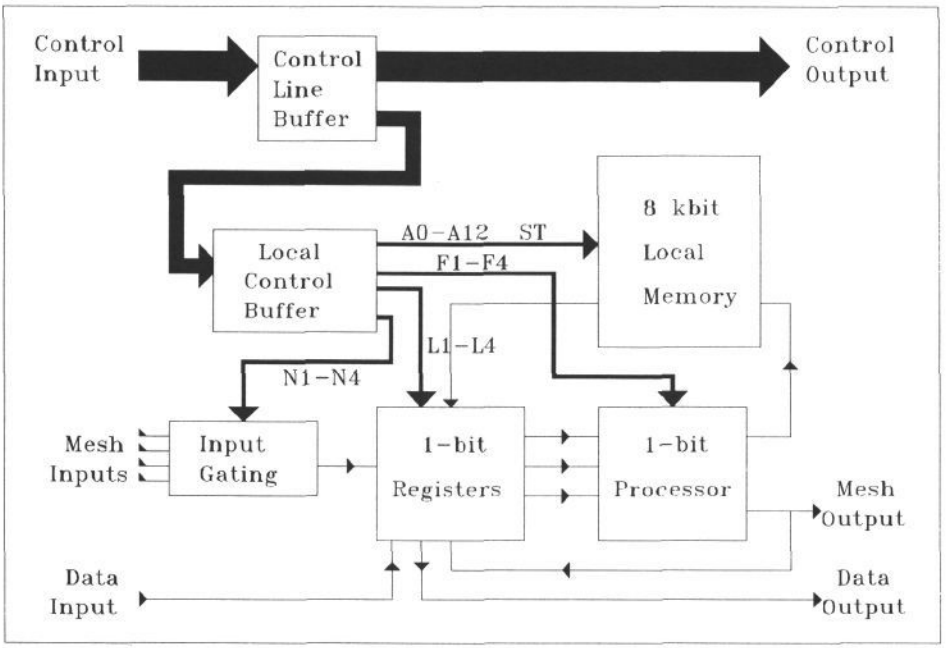

Figure 1 The propagated instruction processor element

- There is a near-neighbour square 2D mesh of bit-serial PEs
- Data is inserted into the array along shift registers by the usual method
- In addition to the normal elements, each PE contains an instruction pipeline
- Control lines, including address lines, enter each row of the array at one end of that row
- Control signals are buffered in each PE and propagated with one clock period of delay
- Control signals are driven into the array from an external pipeline

The most significant consequence of this design is that there are no signal lines (either control or data) within the array which are longer than the neighbour-to-neighbour element spacing, including that bearing the synchronising clock signal.

Such an array operates as follows. First, a program is generated which consists of the sequence of operations to be carried out on the array to accomplish the task. In this context, a single operation is of the form *detect the boundaries of binary objects.* In this analysis, a complete program commences with *input original image* and ends with *output final result.* For the purposes of illustration, we consider a segment of a complete program comprising noise removal by median filtering; Sobel edge detection; thresholding and skeletonisation. This sequence is stacked in an external instruction pipeline.

The result of passing these operations in sequence across an image is illustrated in Figure 2, which shows an intermediate stage in the process. The operation begins with the first instruction being applied to the first strip of the image

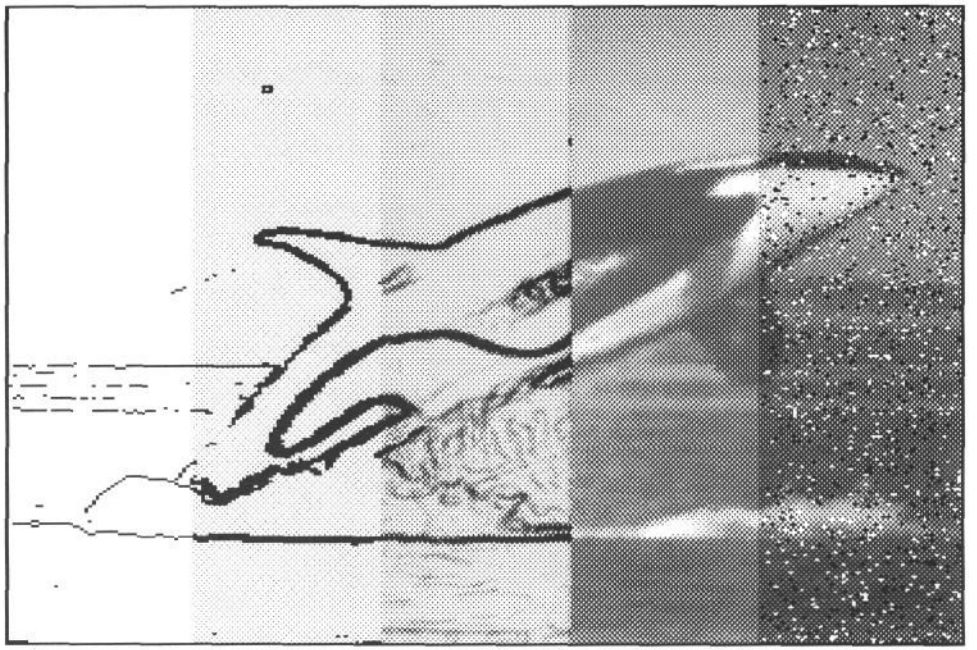

Figure 2 An illustration of propagated instuction processing

(this would normally be a single strip of pixels, but wider strips are shown in the image to illustrate the process more clearly). When the required computation has been executed, the first instruction is propagated to the next strip, whilst the second instruction is applied to the first strip. This second instruction (in general) acts upon the results of the first instruction. Once again, when the instructions have been executed, they are propagated onwards, with a new instruction being input (from the instruction pipeline) at the left of the image. The procedure continues until the final instruction has been propagated to, and executed over, the entire image. At this stage, further processing could occur, or output of the result could begin. We designate this complete process as *macro-pipelining*.

For some types of operation, this level of explanation is sufficient. For example, if two images are being added together, then each column of the image pair can be treated independently as the instruction 'sweeps' past it. This is not true, however, for all image processing operations. In particular, it is important to consider the class of near-neighbour operations, such as the majority of the operations included in the example above, since these are fundamental in image processing.

When a near-neighbour operation, in which data from a three by three area of the data set is required, is to be programmed on a propagated instruction array, the following factors have to be considered.

- The same instruction may be applied to more than one contiguous column simultaneously
- Only one column will, in general, produce a correct result at each moment
- A correct result will already have been computed for previous columns; overwriting this must be prevented

In general some modified procedure will be appropriate for this situation. The necessary modification can be achieved by *micro-pipelining*. In this process, the higher-level operation is split up into the necessary sequence of nano-instructions (such as loading a register) and these are appropriately interleaved to achieve the desired results. The process is illustrated below for the case of a binary edge finding operation. The relevant nano-instructions are:

- Set the appropriate memory address to retrieve the data (A_1).
- Load the data from the memory to a local register. Perform a function on the data so that it will become available at the neighbour output $(L_1{:}F_1)$.
- Perform the same function again (F_1).
- Accept local neighbour inputs and perform the same function a third time $(N{:}F_1)$.
- Load the neighbour inputs to a local register. Calculate the edge pixels from the values of both local registers. Set a new memory address $(L_2{:}A_2{:}F_2)$.
- Store the result (ST).

Next, the necessary sequence of these events for each column of PEs has to be determined. The required sequence is illustrated in Table 5, showing which of the control lines illustrated in Figure 2 is active for each column of processors at each clock cycle.

Cycle #	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
1	A_1					
2	L_1: F_1	A_1				
3	F_1	L_1: F_1	A_1			
4	N:F_1	F_1	L_1: F_1	A_1		
5	L_2:A_2: F_2	N:F_1	F_1	L_1: F_1	A_1	
6	ST	L_2 A_2: F_2	N:F_1	F_1	L_1: F_1	A_1
7		ST	L_2 A_2: F_2	N:F_1	F_1	L_1: F_1
8			ST	L_2 A_2: F_2	N:F_1	F_1
9				ST	L_2 A_2: F_2	N:F_1
10					ST	L_2 A_2: F_2
11						ST

Table 5 PIP nano-instructions required for a local neighbourhood operation

The table shows that the nano-components of the micro-instruction are distributed over a number of columns of the array. Thus, at cycle 6, for example, column 6 is performing the first element of the micro-instruction whilst column 1 is performing the last.

This micro-pipelining naturally becomes more complex for operations which address more of the neighbours of a PE. Although at the present stage of this work it is not possible to define a full set of efficient algorithms, it is already apparent that algorithms specifically designed to take advantage of the architecture will need to be developed. Because of the semi-pipelined nature of the system, it is possible these may include the Recursive Neighbourhood Operations described by Komen [13].

The PIP system proposed above is conceptually more complex than the SIMD arrangement from which it is derived. In order to evaluate the impact of this complexity on the performance of

a system, we make the following estimates concerning the orders of magnitude of the system and program parameters:

- An array consists of 1000 x 1000 elements
- All signals, both data and control, must be originally input at the edge of the array
- The same time (one clock period) is needed to execute each of the lowest level of array operations (designated nano-instructions)
- A micro-instruction (a complete 1-bit operation) comprises 10 nano-instructions
- A macro-instruction, such as an 8-bit median filter operation or the addition of two 16-bit integers, comprises 100 micro instructions
- The typical processing sequence consists of 100 macro instructions
- The processing sequence commences with the input of one 8-bit array of data and ends with the output of a similar array

Table 6 shows the components of the performance.

Because the PIP system overlaps the propagation of control data with function execution continuously, the result is a total program time of 117 000 clock cycles, determined effectively by the execution time of the PEs. The latency of the pipeline, which might be thought an important factor, is a negligible part of this processing time.

At this stage it becomes possible to evaluate the performance and packing density of the proposed PIP architecture against those of the systems described in section 3, at least in terms of the operations suggested above. Table 7 shows how the number of PEs and the time to execute the above program vary, for five different possible architectures, including an (impossible) ideal SIMD array. The figures given have been calculated on the basis of a minimum feature size of 1 nm and an overall die size of 4 sq. mm.

Function	Clocks per function	No. of functions	Total clocks
Input/output 8-bit image	8000	2	16 000
Insert parallel control information	1	100 000	100 000
Pipeline latency	1000	1	1000
Execute macro-operation	1000	100	100 000

Table 6 Performance figures for the PIP array

Architecture	Number of PEs	Execution time
Ideal array	1000x1000	116 000
All nano-technology	1000x1000	10,000,000
Hybrid technology	200x200	116 000
Serial instructions	500x500	366 000
Propagated instructions	1000x1000	117 000

Table 7 Comparison between alternative architectures

Using these figures, we can compute the time taken to execute our supposed task on the PIP architecture.

5 Conclusions

In this paper we have shown that at least one serious problem arises when

attempting to implement SIMD computers using nano-electronic devices. Either there is a performance penalty caused by delays in long-range signal distribution through nano-wires, or there is a penalty in terms of packing density occasioned by the use of micro-wires. However, we have also demonstrated that an architectural solution, namely the propagated instruction processor, is available which recovers all of the potential lost performance whilst retaining the optimum packing density. This architecture, developed by combining the characteristics of a number of previously described systems, is closest in operation to the instruction systolic array architecture described in [14], but demonstrates a number of additional properties.

It must be emphasised that the figures presented in this paper are based on a series of calculations concerning the structure and use of nano-electronic arrays which, whilst being derived from the best current ideas, are certainly open to further refinement. At present the development of the PIP architecture is still at the conceptual stage and any claims about its capabilities are made tentatively.

6 References

[1] Tucker L.W. and Robertson G.G., **Architecture and Applications of the Connection Machine,** *Computer* pp. 26-38, August 1988

[2] Nickolls J.R., **The design of the MasPar MP-1,** *Proc. of CompCon, San Francisco, CA,* pp. 25-28, Spring 1990

[3] Schmitt L.A. and Wilson S.S., **The AIS-5000 parallel processor,** *IEEE trans. on PAMI,* V 10 No. 3, pp. 320-330, May 1988

[4] Alexander J., **Opening Remarks,** *Ultra Electronics Programme Review - Presentation Summaries,* Santa Fe, New Mexico, October 1993

[5] Duff M.J.B. and Fountain T.J., **Novel Processor Arrays,** Int. Rep. 92/3, Dept. of Physics and Astronomy, University College London, 1992

[6] Hunt D.J., **The ICL DAP and its application to image processing,** *in Languages and Architectures for Image Processing,* Eds. Duff and Levialdi, pp. 275-282, Academic Press

[7] Potter D.J., **Colony counting and analysis,** *in Cellular Logic Image Processing,* Eds. Duff and Fountain , pp. 111-140, Academic Press, 1986

[8] Strong J.P., **Computations on the Massively Parallel Processor,** *Proc. of IEEE,* Vol. 79, No. 4, pp. 548-557, April 1991

[9] Reed M. A. and Kirk W. P., Eds., **Nanostructure Physics and Fabrication**, Proc. of the Int. Symp., College Station, Texas, 1989

[10] Moffat C. D., **A Survey of Nanoelectronics**, Int. Rep. No. 94/2, Image Processing Group, Dept. of Physics and Astronomy, University College London, 1994

[11] Denboea A., **Inside Today's Leading Edge Microprocessors,** *in Semiconductor International,* pp. 64-66, February 1994

[12] Fountain T.J., **Parallel Computing: Principles and Practice,** Cambridge University Press,1994

[13] Komen E.R. **Low-level Image Processing Architectures - Compared for Some Non-Linear Recursive Neighbourhood Operations**, PhD Thesis, Technical University Delft, 1990

[14] Kunde M. *et al.*, **The Instruction Systolic Array and its Relation to Other Models of Parallel Computers**, *in Proc. Int. Conf. on Parallel Computing 85,* North Holland, 1985