The Design of Highly-Parallel Image Processing Systems Using Nanoelectronic Devices

T J Fountain
Image Processing Group
Department of Physics and Astronomy
University College London

ABSTRACT

As minimum device dimensions are reduced from a few hundred nm to a few nm the number of devices on a single small chip will rise from one million to ten billion. However, as dimensions are reduced below approximately 100 nm, device characteristics will all differ from those of current devices. The anticipated packing density (and performance) of such nanoelectronic devices could be usefully applied in the achievement of highly-parallel, highly-compact, computer systems but, because of the changes anticipated in device characteristics, the designs of such systems need to be reevaluated. This paper describes the re-evaluation of the data-parallel SIMD type of system in the light of a perceived problem concerning the difficulty of conveying signals over long distances on nanoscale wires. To overcome this problem, a novel architecture, the Propagated Instruction Processor, has been developed which incorporates design elements from SIMD arrays, pipelines and systolic architectures. Examples of circuit elements, suitable for incorporation in such an architecture, implemented in QCA components are presented together with the results of simulations which demonstrate the potential packing density and performance of such systems.[1]

1 Introduction

The main aim of the work reported here was to evaluate methods by which evolving nanoelectronic components could be effectively assembled into the type of highly-parallel computing structures which are suitable for low-level vision algorithms. A re-evaluation of the relationship between the requirements of parallel systems and the characteristics of devices was necessary because of the enormous changes of scale predicted from nanoelectronic device development, and because of the qualitatively different nature of the component characteristics themselves. For the first time, these advances offer the possibility of removing the low-level vision computing bottleneck entirely. Table 1 illustrates some of these differences.

Parameter	Micro scale	Nano scale
Minimum dimension	500-1000 nm	5-10 nm
Power consumption	1 μW	1 pW
Gain	100	≅1
Stable states	2	many
Clock rate (GHz)	1	1000

Table 1 A comparison between the main parameters of micro-scale devices and the predicted parameters of nano-scale devices

As minimum device dimensions are reduced from a few hundred nm to a few nm, a simple calculation shows that the number of devices on a single chip of moderate size will rise from one million to ten billion. However, because the physical principles on which current devices depend, such as the impenetrability of insulation layers, will not hold as dimensions are reduced below approximately 100 nm, it is predicted (and has been demonstrated in, for example, [1]) that device characteristics such as power consumption, drive capability and transfer characteristic will all differ from those of current devices, and from one class of nano-device to another. This work was therefore undertaken with the intention of providing both a valid applications field for the new devices and a feedback mechanism to guide their detailed development.

Long before the commencement of the development of nanoelectronic devices, many alternative types of

[1] This work was funded under the ARPA ULTRA program, contract number N00014-93-1-1087. Part of the work was undertaken in collaboration with the Special Architectures Group of the Mayo Foundation and with the Department of Electronic Engineering at Notre Dame University.

0-8186-7987-5/97 $10.00 © 1997 IEEE

highly-parallel computer architecture had been proposed and implemented. The main qualitatitive differences between three distinct types of architecture - data-parallel; function-parallel and cognitive - are shown in Table 2.

Parameter	Data	Function	Cognition
Degree of Parallelism	High	Low	High
Processor Complexity	Low	High	Medium
Interconnect Complexity	Low	High	High
Amount of Interfacing	Low	High	Low
Extensibility	High	Low	Low

Table 2 A qualitative comparison between the parameters of the three main classes of parallel electronic computers

connected, peripherally-interfaced data-parallel architectures provided an almost ideal match to the likely properties of many nano-devices, and should therefore be selected as our first candidate architecture.

The selection of the SIMD type of data-parallel system on which the work reported here was concentrated was based on three factors. First, SIMD systems have been shown to be the most widely-applicable within this class; second, such systems are commercially available, if not widely then at least from more than one manufacturer [2,3]; third, the majority of the systems which the authors have constructed and investigated were of this type.

The basic design of the SIMD architecture is well known and documented [4,5]. The acronym SIMD was coined by Flynn in 1965 as part of a taxonomy of computer architectures [6], and stands for Single Instruction stream, Multiple Data stream computer. The basic idea of the system is illustrated in Figure 1. The main part of the system comprises an array of small processors (usually called processing elements, or PEs).

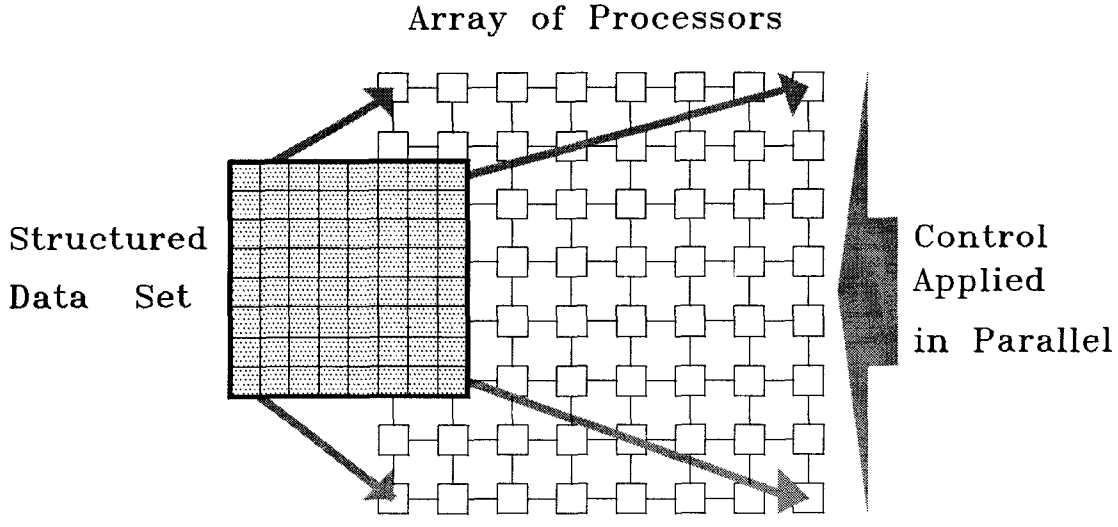

Figure 1 The SIMD paradigm maps a structured data set onto a similarly structured array of processing elements, all of which execute the same instruction at a given moment

2 Selection of the Computing Paradigm

Because we understood, both from the physics of proposed nano-devices and from early work undertaken in their development, that the low current drive capabilities of such devices would lead to problems with long-range connectivity and with interfacing to the outside world, we decided that highly-regular, locally-

The array usually forms a two-dimensional mesh, with each PE connected to its four or eight nearest neighbours, since the intention of the arrangement is to match the format of a two-dimensional data set by a similar format of PEs in the manner shown in Figure 1. The data set is usually one in which the physical relationships of the data items are important, examples being matrices, images, and data representing physical variables (such as stress) within an approximately planar material.

The functions executed by the array of PEs on the data are controlled by a single, central program store. Each control function is distributed to every PE simultaneously so that, at each moment in time, the whole array of PEs is executing the same instruction. A working program is likely to consist of many thousands of such instructions because of the complexity of the calculations involved.

Every PE must be supplied with a substantial amount of memory to which it has direct access, because typical programs create and use many intermediate results in a semi-random fashion, and many of these must be stored for intermediate periods. The high memory bandwidth which the SIMD arrangement provides is an important advantage of the architecture, although it does not easily support operations in which data must be exchanged between distant processors.

The final significant aspect of the architecture consists of the arrangements for inserting and extracting data from the array. This is usually carried out along rows or columns of the array in a semi-parallel manner which offers a reasonable compromise between interfacing complexity and speed of operation. Taken together, these factors mean that SIMD arrays have the following desirable properties for nano-implementation:

- The structure is regular and repetitive
- All connections between elements of the system are local
- All external connections are made to the periphery of the array
- The implementation of strategies for fault-tolerance is at least feasible

3 Performance Evaluation

The practice of benchmarking special purpose parallel computers is inherently difficult and a discussion of some of the previously devised benchmarks can be found in [7]. In this work we have adopted the low-level approach, in which a set of representative algorithms is used to evaluate likely performance. A test suite has been drawn up, consisting of tasks that are representative of the different types of operations involved in the early stages of image processing (see Table 3), and has been used to evaluate each of the architectures proposed in this program. The chosen algorithms are low level image processing tasks and are mainly iconic in nature (i.e. an input image is transformed into an output image of the same dimension), since such tasks dominate the programs of data-parallel arrays.

	Point	**Local Neighbour**	**Global**
Binary	Logical AND	Dilation	Skeletonisation
			Labelling
Integer	Threshold	Median Filter	Adaptive Edge Extraction
	Multiply	Edge Detect	Matrix Multiplication
			Rotation
			Magnification
			Histogramming
			Hough Transform
Real	FP Division		Fourier Transform

Table 3 Programs developed using the simulation system can be categorised by their bit-resolution and by their degree of locality

In order to provide a broadly-based evaluation of the properties of possible nanoelectronic implementations of SIMD systems, a number of alternative architectures were simulated. The evolved systems were developed principally in an attempt to overcome one anticipated problem, that of the distribution of control signals. Although all data connections in an SIMD array are short, the control paradigm assumes that a given control function is applied simultaneously over the whole array. In a conventional system, this is achieved by a tree of buffering stages applied to each signal. This causes no difficulty in conventional systems, since the number of buffering stages is small, as is the delay incurred at each, so no undue slowing of the effective control clock rate ensues. In a nano-electronic system of the scale proposed, this is unlikely to be the case. A 10nm wire, crossing a 1cm die, has an aspect ratio on the order of one million to one. Even if no problems accrue from the (uncertain) signal conveying properties of such narrow conductors, the electrical impedance of such a wire is likely to cause sufficient signal degradation to impair the working of any circuit over a relatively short distance. This, in turn, implies that a large number of buffering stages will be required. The major part of the process of architecture evolution, therefore, concerned methods of dealing with this problem.

First, a purely conventional SIMD system was simulated. When combined with the best currently available figures for conventional microelectronic devices, this provided a baseline against which the more advanced proposals could be evaluated. The

fundamental parameters of conventional devices give rise to a clock rate of 500 MHz and a maximum of about 1600 processing units on a 5mm square die.

Second, the same system was simulated on the basis of anticipated nanoelectronic device properties. An SIMD array chip constructed from resonant tunnelling diodes and Schottky diodes would have parameters including a clock rate of 100 GHz and the capability to implement a quarter of a million processors on a single 5mm square die.

Third, a number of alternative architectural techniques were explored, aimed at overcoming the perceived problem of control-line distribution. Finally, a novel development of the SIMD array, termed the Propagated Instruction Processor, was implemented and evaluated. The development of this architecture, which provided a feasible vehicle for the utilisation of massive numbers of nanoelectronic devices, is one of the major results of this work.

The gross advantages of the SIMD system are obvious - improved packing density by a factor of 150, and increased clock speed by a factor of 200. However, the majority of the semiconductor area is taken up not with active devices but with the distribution of control signals, which is obviously undesirable. We therefore examined the consequences of utilising two alternative architectural approaches to the distribution of control:

1. Use nano-scale wires, but buffer the signals to overcome the likely signal degradation. The amount of buffering is, as yet, unknown, so the number of buffering stages is regarded as a variable in our simulations. The delay incurred by each stage of buffering is assumed to be one clock cycle.
2. Serialisation of control. This involves transmitting all 35 control bits along a single wire of appropriate robustness, whilst accepting the commensurate loss of performance.

Parameter	Thick wires	Thin wires	Single wire
Millions of PEs	0.25	1	1
Clock (GHz)	100	<10	3

Table 4 The three conventional SIMD systems described in the text would have significantly different properties. No alternative has the optimum combination

The results of carrying out this exercise are shown in Table 4. The 'thick-wires' system, which uses conventional scale lines to distribute control, retains the full clock speed but sacrifices packing density. The 'thin-wires' system uses nano-scale lines for control distribution, and therefore regains optimum packing density, but sacrifices effective clock rate because of the delays caused by buffering the control lines. The 'single-wire' system transmits control serially along one wire, so that a control word requires 35 clocks to arrive at a processor, thus slowing the effective clock rate by that factor whilst maintaining full packing density. Each of these arrangements offers a compromise between the best available packing density and the best available clock rate. In the next section we describe an architecture which combines the best of both.

4 The Propagated Instruction Processor

The design of the propagated instruction processor (PIP) arose from a synthesis of the ideas behind a number of highly parallel architectures - the 2-D SIMD array; the 1-D SIMD array; the pipeline processor and the systolic array [7]. We reasoned that, in vision applications of the type likely to be addressed, a fixed data structure existed, to which were applied a long sequence of control functions (analysis showed the typical program to be many thousands of microinstructions long).

When using a 1-D processor array to deal with 2-D data, one sweeps the array across the data. We reasoned that it should be possible to sweep the long sequences of instructions across the data, each instruction following closely on the heels of the one before - almost exactly the reverse of the situation obtaining in a data-systolic array. In systolic architectures, the main problem is the latency of the system, that is, the additional time which accrues from sweeping the data through a series of processors. We calculated, from a preliminary analysis of the lengths of instruction sequences involved in typical applications, that this effect would be negligible (~1%) in the type of design envisaged.

We therefore developed the system shown in Figure 2. The data on which instructions will be executed enters the array in the normal way. Each row of the array is supplied with a pipeline of register bits, one bit per instruction bit and one complete set per processor element. The first instruction enters at the left and is executed by the first column of processors. It is then passed to the next column, at the same time as a new instruction word enters at the left. The first instruction is then executed by the second column and the second instruction by the first column. This somewhat complex procedure, fully described in [8], continues until the last instruction in the stream enters at the left

of the array. No further instructions enter, but the existing ones continue to sweep across the array until all have been executed by all columns of processors.

permits simplified simulations of the circuits to be fully representative.

However, when a memory element is implemented

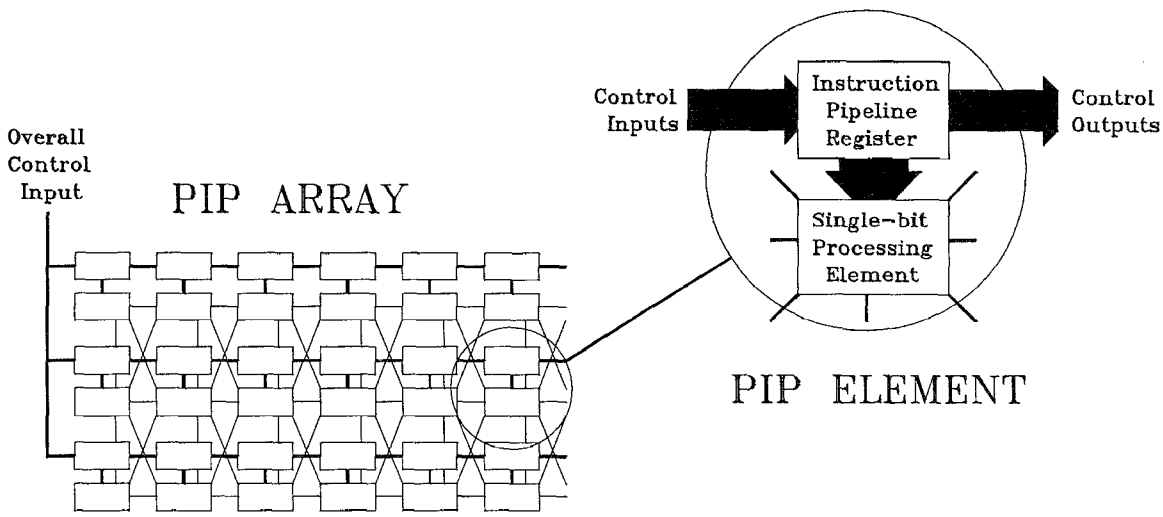

Figure 2 The propagated instruction processor adds a pipeline element to each of the normal processor array units. Instructions are inserted into the array through these pipelines.

5 Quantum Cellular Automata

One of the main types of nanoelectronic device under consideration in this program is the Quantum Cellular Automaton (QCA). The fundamental idea behind the operation of QCAs is that the energy state of a suitable assembly of electrons, initially in a specific ground state, will alter as a result of changed boundary conditions. If the change in energy states is carried out correctly, the final result will be a new ground state which is only dependent upon the new boundary conditions of the assembly. This can be thought of as equivalent to the result of a computational circuit element being dependent upon the states of its control and data inputs [9].

In order to formalise this idea somewhat, QCAs are presently defined in the form of basic cells, each consisting of five quantum dots, as illustrated in Figure 3. Each cell is populated by two electrons and has the two preferred states of polarisation shown in the diagram, the two configurations corresponding to the '1' and '0' states of a binary circuit element. Circuit elements, such as the majority gate also shown in Figure 3, are built up of assemblies of the basic cells. This representation of QCA-based circuits allows the normal rules of Boolean logic to be applied to their design, and as an assembly of QCAs, its operation embodies a contradiction. Before loading the memory a ground state corresponding to a stored zero exists for a given set of boundary conditions, whereas after the operation a different ground state, corresponding to a stored one, exists for the same set of boundary conditions. Considering the quantum mechanics of the array of electrons involved, this implies two equally probable (ground state) solutions to the time-dependent Schroedinger equation for the same set of boundary conditions. Whilst this is not impossible, it is unlikely that the two solutions would be separable, in which case the stored state would be a mixture of the two solutions, i.e. neither zero nor one. Thus the state of the memory cell would be undefined after any sequence of operations.

Two further factors concerning the operation of QCA arrays have led to proposals, reported elsewhere [10], which we utilise to solve the problem posed above. First, the time taken for an array to reach its ground state under the conditions described above is somewhat undetermined, being dependent on the exact configuration of the array, amongst other factors. This led to the idea of adiabatic switching of arrays, in which the energy barriers between quantum dots are first reduced to such an extent that the wave functions are almost completely unlocalised (the array is designated

as being 'mushed out' at this point). The boundary conditions are then established, and the inter-dot barriers are slowly increased at a rate which corresponds to adiabatic switching of the array. As the barriers are raised, the wave functions localise to determine the new ground state of the array. Under these circumstances, the required switching rate can be determined from the size of the array to be switched. This basic idea leads logically to the requirement for each sub-array of QCAs within a given circuit to be in one of four states:

- Inactive, in which the positions of electrons are undetermined
- Computing, in which the inter-dot barriers are being raised
- Locked, in which the array is impervious to external influences
- Relaxing, in which inter-dot barriers are being lowered

The second factor to be considered concerns the number of cells which can be implemented within a given array. For a specific set of parameters, there is a limiting number of cells which can be implemented if the ground state is to remain the lowest energy configuration. Above this number, thermodynamic considerations mean that non-ground level states can be preferred. This limiting number is between 10^3 and 10^4 basic cells for reasonable values of the relevant parameters. The whole system must therefore be split into separate small blocks, each of which is in one of the four states described above.

It is apparent that, in general, each block must cycle through all four stages in order to execute computation within the overall assembly. The control of the system can therefore be thought of in terms of the application of four clock phases. The implementation of such a system can be achieved by overlaying the computing assembly with suitably patterned metal layers which carry voltages corresponding to the clock phases.

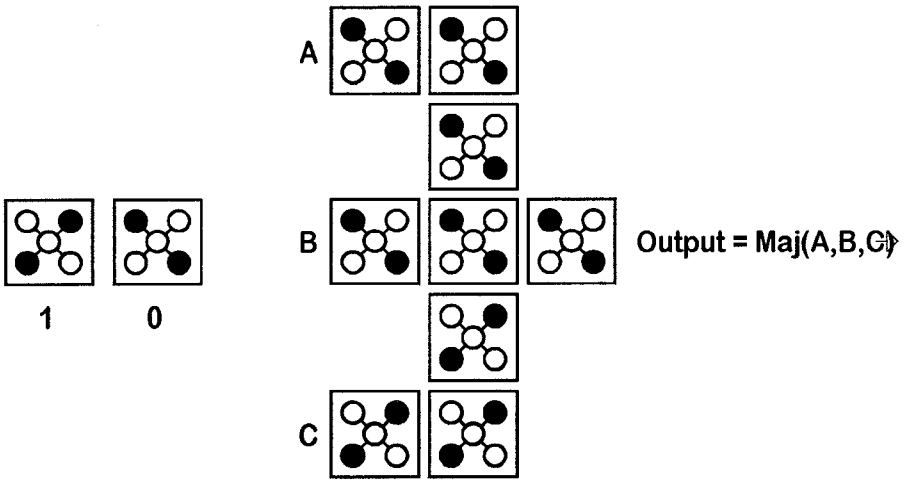

Figure 3 QCA elements can exist in two equally probable states, and can be configured into a majority logic gate as shown here

Careful thought must be given to the geometrical layout of the metal overlayers which carry the voltages corresponding to the four clock phases. In general, the problem is to supply the correct sequence of phases to any circuit, or sub-circuit, which involves feedback. Since this applies not only to each processor element as a whole, but to every memory element within each processor, complex arrangements are required. The optimisation of these arrangements, which constitutes a considerable problem for overall array design, is considered in a paper currently in preparation.

The factors involved in the design of a bit-serial SIMD processing element have been described above. In order to develop a realistic design, two simulation tools have been used. The first, described in [10], has been used to verify the operation of fundamental circuit blocks by solving the time-dependent Schrodinger equation for the relevant assemblies of quantum dots. This necessary, but time-consuming, process resulted in the definition of basic cell dimensions and clocking frequencies. The second simulation technique, used to

generate and verify the logical operation of the overall element design, utilises the EXCEL ® spreadsheet. This simulation, which operates in terms of the logical function of, and interactions between, QCA cells, is sufficiently fast and flexible to permit many possible configurations to be tested and evaluated.

spreadsheet corresponds to a single, five-well, QCA element. The positions of logic gates are indicated by cells marked with **M** (for majority gate), while crossovers, which are achieved in a planar fashion, are indicated by an **X**.

Each cell also normally bears a colour code which

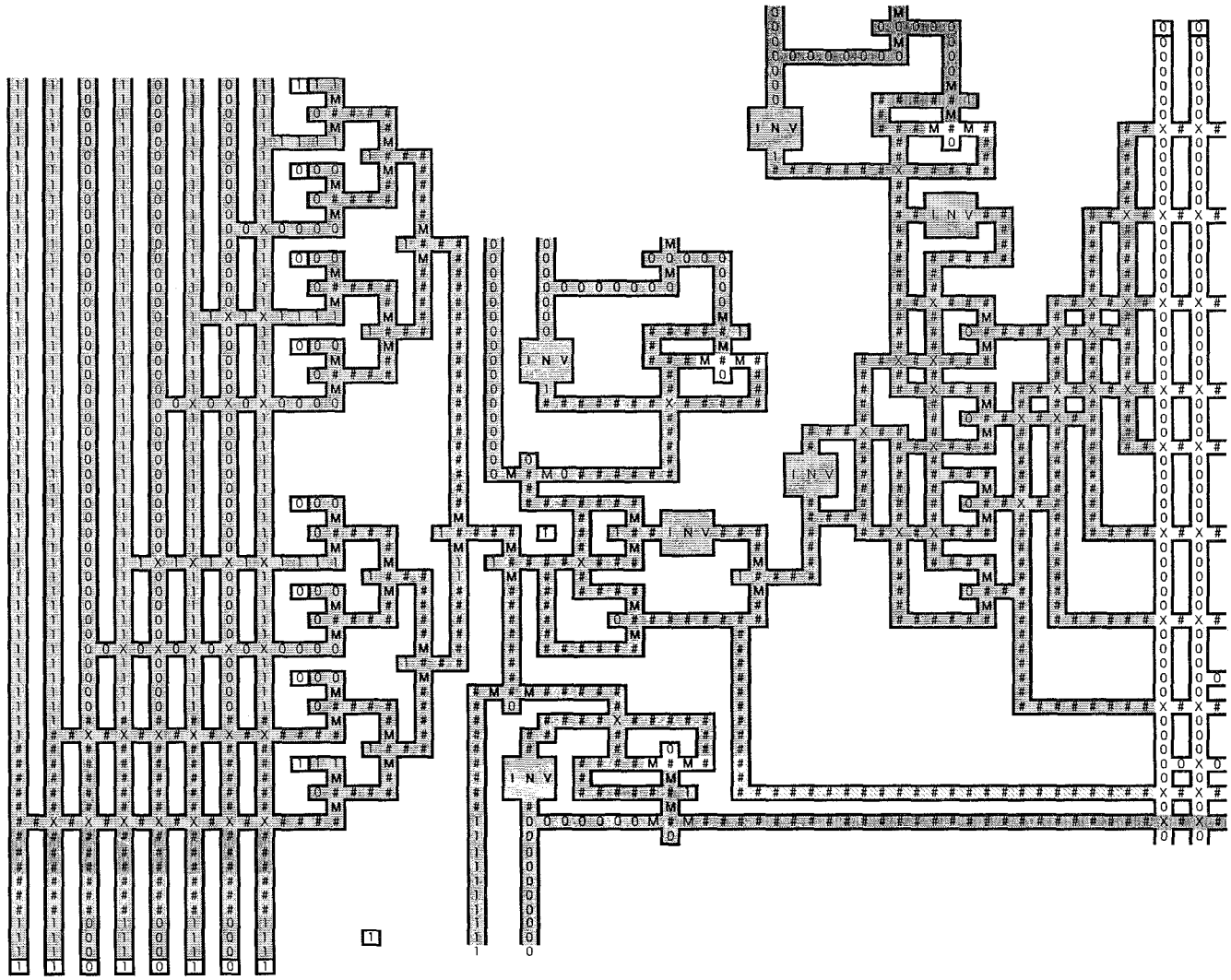

Figure 4 Part of the SIMD processing element as simulated using the EXCEL spreadsheet. Each cell of the spreadsheet corresponds to a single QCA, and the status of the array is iteratively updated

In addition to verifying the design, the EXCEL simulation allows an accurate assessment of packing density to be made. One of the outputs of the simulation, shown in Figure 4, corresponds to the physical layout of the circuit, and of the applied clock phases. In the diagram, each (square) cell of the indicates the clock phase which affects its operation. Certain areas are used for the distribution of clock-phase metallisation through the array. Crossovers of clock-phase metallisation are necessary, but are not permitted over active QCA areas.

Operation	Type	Time (μS)
AND of two images	1-bit pointwise	0.005
3x3 median filter	8-bit local	0.190
floating-point division	32-bit pointwise	1.900
matrix multiplication	8-bit global	17.60

Table 5 The figures given in the table are the times required to execute a number of different algorithms. In each case the data set is a square array of 1000 by 1000 elements

The parameters of a PIP array constructed from QCA elements would result in a basic 100 GHz clock rate and the implementation of a million processing elements on a single 5mm square die, thus fulfilling the requirement to combine optimum performance and packing density. However, the evaluation of the performance of such an array is complex. For purposes of clarity we present figures only for a representative subset of algorithms. The algorithms used are given in Table 5.

	1996	20??
Number of PEs	1 000	1 000 000
CMOS clock (MHz)	500	
QCA clock (MHz)		100 000

Table 6 The performance and packing density of the arrays considered in this paper (based on QCA devices) are greatly improved over those of currently available (CMOS) systems. It is, however, uncertain when this technology will become available

6 Discussion

The fundamental result of this work has been to develop a data-parallel architecture which can exploit the presently-predicted properties of proposed nano-devices. During the work we encountered three very significant, architecturally-related problems: the difficulty of long-range communication; the provision of compact memory; and the requirement for fault-tolerance.

From our early considerations of system design, we believe that the single most critical aspect for success will be the ability to incorporate fault tolerance at every level. The problem is simply illustrated by considering that an SIMD array of 1024 x 1024 processors would incorporate on the order of ten billion devices of the RTD type, or somewhat greater numbers of QCA elements. With a single atom of impurity being of significance for faulty operation, and fundamental impurity levels in silicon material at about 1 in one billion atoms, there are bound to be faulty elements in such systems. When the astonishing technical demands of manufacturing nano-devices are taken into account, it is obvious that fault-tolerance techniques and strategies will be absolutely necessary if the promise of nanotechnology is to be realised. The investigation of such methods forms a major part of our current program.

Overall, we conclude that the implementation of nano-scale circuitry offers the possibility to construct computing systems which will be qualitatively superior to their conventional equivalents, partly because of the huge quantitative leaps in computing power which should become available, but also because of the different approaches to computing which the new devices should offer. We are currently continuing our investigation of the mutual interaction of nano-device properties and the design of highly-parallel, high-performance computers in the areas of SIMD array implementation, evaluation of alternative architectures, and a study of the requirements for fault tolerance under various circumstances.

7 References

[1] Proceedings of the ULTRA Electronics Program Review Meeting, Boulder, Colorado, (1995)

[2] Commercial literature, MasPar Corp.

[3] Commercial literature, Cambridge Parallel Processors

[4] Fountain T J & Goetcherian V, CLIP4 parallel processing system, IEE Proc. 127E, pp. 219-224, (1980)

[5]Batcher K E, Design of a Massively Parallel Processor, IEEE Trans. C-29, 836-840, (1980)

[6] Flynn M J, Very high speed computing systems, Proc. IEEE 54, pp. 1901-1909, (1966)

[7] Fountain T J, Parallel Computing - principles and practice, CUP, pp. 300, (1994)

[8] Fountain T J & Tomlinson C, The propagated instruction processor, Proc. BMVC95, pp. 563-572, BMVA Press, (1995)

[9] Lent C S, Tougaw P D, Porod W and Bernstein G H, Quantum Cellular Automata, Nanotechnology, 4, pp. 49-57, (1993)

[10] Tougaw P D & Lent C S, Dynamic behaviour of quantum cellular automata, Journal of Applied Physics, 80, 8, pp. 4722-4736, (1996)

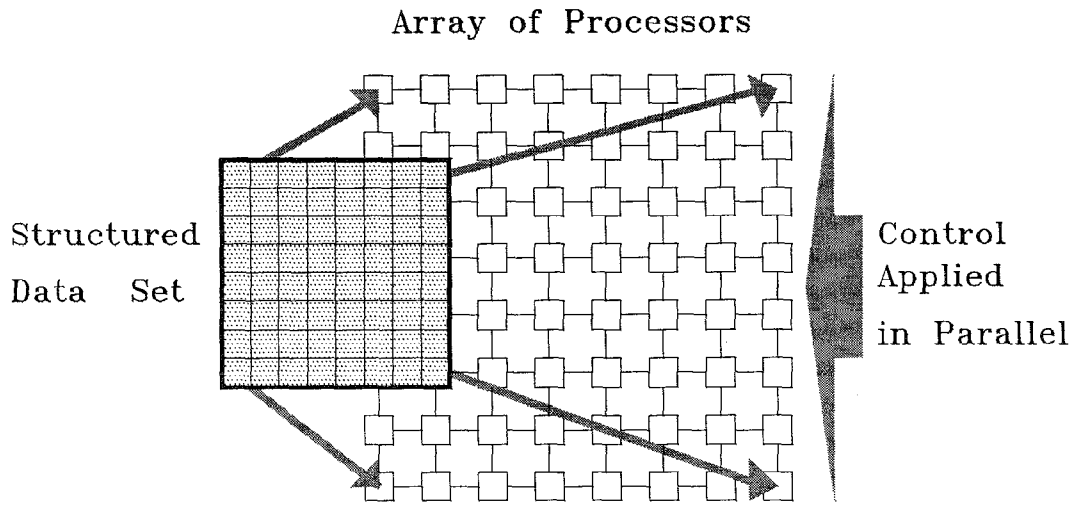

Figure 1 The SIMD paradigm maps a structured data set onto a similarly structured array of processing elements, all of which execute the same instruction at a given moment

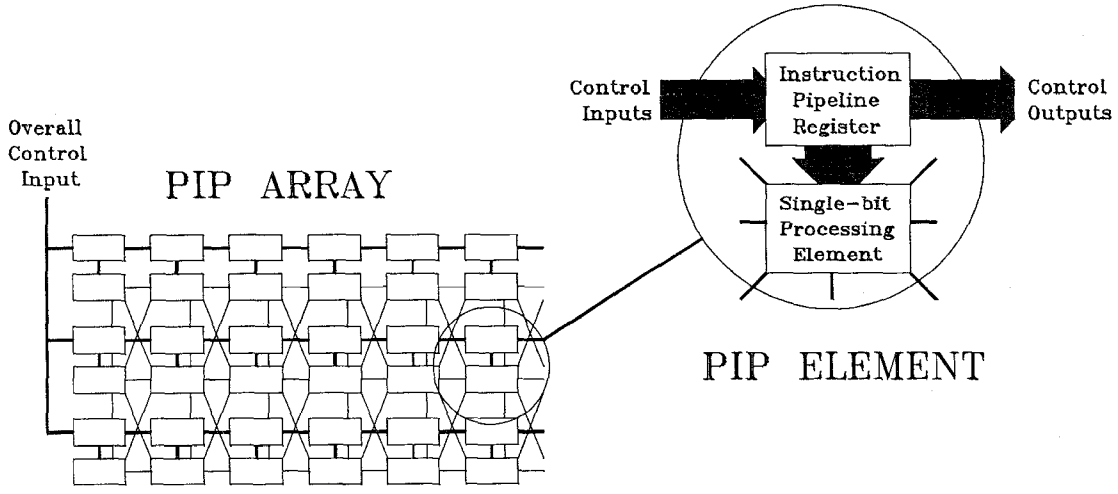

Figure 2 The propagated instruction processor adds a pipeline element to each of the normal processor array units. Instructions are inserted into the array through these pipelines.

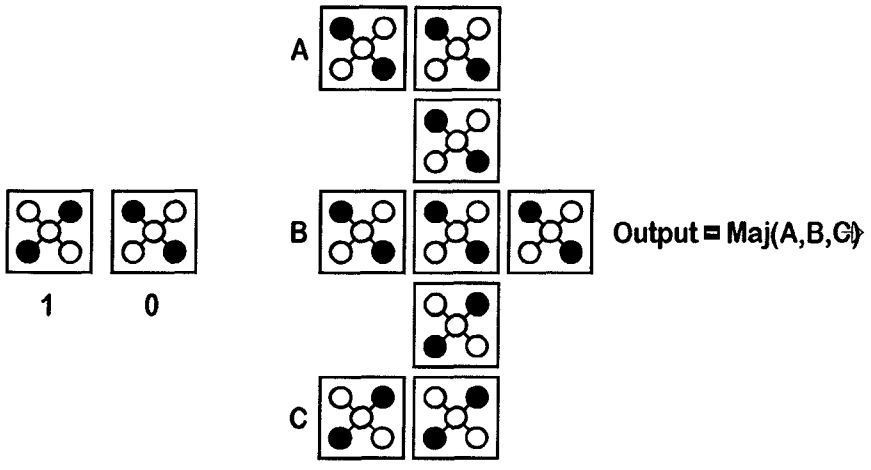

Figure 3 QCA elements can exist in two equally probable states, and can be configured into a majority logic gate as shown here