Optics Communications 100 (1993) 247–258
North-Holland

OPTICS
COMMUNICATIONS

Full length article

Comparison of local and global computation and its implications for the role of optical interconnections in future nanoelectronic systems

Haldun M. Ozaktas
Electrical Engineering, Bilkent University, 06533 Bilkent, Ankara, Turkey

and

Joseph W. Goodman
Electrical Engineering, Stanford University, Stanford, CA 94305, USA

Received 12 October 1992; revised manuscript received 9 February 1993

Various methods of simulating diffusion phenomena with parallel hardware are discussed. In particular methods are compared requiring local and global communication among the processors in terms of total computation time. Systolic convolution on a locally connected array is seen to exhibit an asymptotic advantage over Fourier methods on a globally connected array. Whereas this may translate into a numerical advantage for extremely large numbers of ultrafast devices for two-dimensional systems, this is unlikely for three-dimensional systems. Thus global Fourier methods will be advantageous for three-dimensional systems for foreseeable device speeds and system sizes. The fact that optical interconnections are potentially advantageous for implementing the longer connections of such globally connected systems suggests that they can be beneficially employed in future nanoelectronic computers. Heat removal considerations play an important role in our conclusions.

1. Locality, globality, physics and computation

Once the essential features of a physical phenomenon have been distilled down to a set of equations, computer implemented numerical methods may be used to predict the outcome of previously unobserved instances of that phenomenon [1].

In general, the state of the system we observe may be characterized by several quantities which are functions of the spatial and temporal coordinates (i.e. "fields"). The many ways we can solve the equations relating these quantities form a broad spectrum, of which we will concentrate on two extremes.

(i) Methods which involve only *local* operations in temporal and/or spatial coordinate space. The relaxation method for solving thermal boundary value problems is a simple example of a local method. Such methods are often *isomorphic* to the physical process they present, in the sense that the calculation mimics the actual physical process at a relatively primitive level. Of course, physical phenomena themselves unfold in time through local interactions, since no influence may propagate faster than the finite speed of light (there is no "action at a distance").

(ii) Methods involving *global* operations. Fourier spectral methods are the most widely used among such methods, because complex exponential functions are eigensolutions for linear equations [2]. The Fourier transform operation is itself global in the sense that the value of the Fourier transform of a function at any spectral point depends on the value of the function at every coordinate point. If we wait long enough, the state of any part of a physical system may potentially influence the state of any other part. Thus although physical processes unfold via local interactions, the field quantities at a given point may in general depend on those at any other point. Notice that spatial globality always comes hand in hand with temporal globality.

Our purpose is to find the optimal physical construct to solve a certain problem. In general, our figure of merit may involve measures of time, space and energy consumption. We will concentrate on solving the problem in the shortest possible TIME. Say we are given one local and one global algorithm to solve a certain problem. They are mathematically equivalent in the sense that they will produce the same result. The number of time steps (worst case or average over all possible initial states) required for the completion of the computation can then be determined. By their very design, it is almost always the case that global methods will require fewer time steps. However, this algorithmic comparison has little meaning, since the physical duration of a time step may be different when we actually build the machines that will execute these algorithms.

Just as the natural phenomena they are used to predict, computing systems must also obey the laws of physics. The mathematical theory of algorithms is meaningful to the extent that the underlying model physical computing system can be realized. The need for a physical theory of computation was stressed by Hillis [3,4].

Towards this end, it is necessary to characterize the information flow required for the solution of a problem on a distributed computing system. This is in general an intimidating task. There have been attempts to analyze the amount of information that must be exchanged between two parties in order to compute a certain function using combinatoric methods [5]. A general extension to many parties which also takes into account issues such as the communication delays among the elements, locality etc. seems exceedingly difficult. For this reason we will consider the simulation of a simple physical problem to illustrate certain principles. We will compare the time it takes to solve the diffusion equation using local and global methods. Though desirable, a more general treatment is beyond the scope of this paper.

The answer to whether and when global or local methods are preferable will have a significant impact on the usefulness of optical digital computing and interconnections. This is because optical interconnections are known to exhibit an advantage only for longer connections of global systems. If local systems are shown to be preferable, there would be little incentive to employ optics.

2. Implications for future nanoelectronic computing systems and optical interconnections

The direct implementation of global methods in parallel hardware requires a highly interconnected [#1] connection graph [#2] among the participating processing elements. Due to the space that must be allocated for communication, this results in large system size and propagation delays. Based on this and similar considerations, Hartman and Ullman [7] and Dally [8] have argued (in the context of a general purpose message passing parallel computer) that it is more beneficial to simulate such communication networks with a low-dimensional mesh, with only local connections. Likewise, based on the intimidating growth of wiring complexity of highly interconnected systems, Frazier has argued that it would be beneficial to implement future nanoelectronic systems based on quantum coupled locally connected cellular automata [9].

If it is indeed the case that in the limit of ultrafast devices and very large numbers of elements, planar (or three-dimensional) mesh architectures offer better performance than highly interconnected ones (even with the optimal choice of interconnection technology), then we can conclude that in this limit the choice of interconnection technology (normal conductors, superconductors, optics) will be of little importance. This is because optical and superconducting interconnections tend to exhibit an advantage over normal conductors only when used to implement the longer connections of highly interconnected global systems [10–12].

In this limit, computing systems begin to resemble physical systems. We do not benefit from global connections because we can no longer assume constant delay along all wires. Ultimately, the transfer of information is limited by the speed of light and does

[#1] With this term, we refer to graph layouts with a large number of relatively far-reaching (global) interconnections, e.g. as in the perfect-shuffle or butterfly graphs. How this notion can be quantified is discussed in ref. [6].

[#2] The graph whose edges correspond to the interconnections between the processors.

not depend on how many hops the distance of travel is broken down to. A shift register begins to resemble a transmission line. A grid connected cellular automata can simulate any given computing system. However, we are presently very far from the device speeds and system sizes needed to reach this limit.

More importantly, we should note that the above arguments do not take into account the effects of heat removal. Although there is general consensus that non-dissipative computing does not contradict the laws of physics, we believe that for quite a time most digital computing systems will make use of dissipative elements. Even if non-dissipative computing becomes a reality, there is always the problem of getting rid of the so-called "garbage bits". Getting rid of these implies similar limitations as getting rid of the dissipated power, unless clever "recycling" schemes are developed.

Assuming constant power dissipation per element, heat removal implies that the linear extent of a system of N elements must grow as $\propto N^{1/2}$ [12,13]. This growth rate is equal to the worst case resulting from consideration of wiring density for bounded-degree graphs laid out in three dimensions [10,12]. Thus, since system size and propagation delays are set by heat removal, rather than finite wiring density; we might beneficially employ highly connected approaches without further increase in system size. On the other hand, for some applications, the amount of information emitted into the system (and thus the power dissipated) by each element may decrease with increasing N since the computational processes become bottlenecked by increasing propagation delays, resulting in a heat removal imposed bound on linear extent weaker than $\propto N^{1/2}$.

To us it is not yet clear whether/when highly connected approaches (such as neural networks) or locally connected approaches (such as cellular automata) or something in between is to be preferred and how this is related to the problem we wish to solve. Further research in this area will give us an idea of how future nanoelectronic processing systems should be contemplated, and of the role novel interconnection technologies such as superconductors and optics (which are known to be beneficial in highly connected systems) will play in such systems. We will merely try to illustrate certain considerations via example.

3. Solution of a problem with high information content

It is well known that many problems such as sorting, convolution, discrete Fourier transforms etc. have an *information content* proportional to N, where N is the problem size [#3] [14]. The information content is the amount of information that must pass through an imaginary boundary dividing the system into two roughly equal parts before the problem can be solved. The information content is also termed as the *communication complexity* [5] in a slightly different context.

For concreteness let us concentrate on a regular e-dimensional cartesian array of N very small processors [#4], $N^{1/e}$ along a side. Each processor contains an L bit precision number. Let it be necessary for NL bits of information to pass through the imaginary boundary mentioned above. Let there be $\mathcal{H}$ independent physical channels passing through this boundary. The total NL bits form trains of $(NL)/\mathcal{H}$ serial bits in passing through this boundary. If the bandwidth of each physical channel is $1/T$, this will take $(NL)T/\mathcal{H}$ time. If the transverse extent of a single physical channel is denoted by λ, the linear extent of the system is then $\mathcal{H}^{1/(e-1)}\lambda$ (since we need enough room for $\mathcal{H}$ channels to pass through the boundary dividing the system into two). Letting c denote the propagation velocity, a lower bound for the total computation TIME may be written as a sum of the propagation and serial contributions

$$\mathrm{TIME} = \mathcal{H}^{1/(e-1)}(\lambda/c) + NLT/\mathcal{H} . \qquad (1)$$

We have assumed that the information must ultimately traverse a distance comparable to the linear extent of the system before the problem is solved. The value of $\mathcal{H}$ minimizing the above is found to be

$$\mathcal{H} = \left(\frac{NLTc(e-1)}{\lambda}\right)^{(e-1)/e} \propto N^{(e-1)/e} . \qquad (2)$$

[#3] For the mentioned examples, the problem size is simply the number of items to be sorted, or the space–bandwidth product of the signal to be convolved or Fourier transformed. For a rigorous definition, see ref. [14].

[#4] We are simplifying by taking the number of processors to be equal to the problem size. In general, this need to be the case. Readers who wish to be concrete may imagine that the N processors contain N numbers which are to be sorted.

With this value of $\mathscr{H}$ we find

$$\mathrm{TIME} \approx (NLT)^{1/e}(\lambda/c)^{(e-1)/e} \propto N^{1/e} . \tag{3}$$

The above expression for TIME is a lower bound. There may be many additional bounds that must also be satisfied. In particular, the signals may need to go through several nodes, suffering additional delays.

One way of solving such problems is to use a highly interconnected graph, such as the butterfly graph [14]. Such a graph would only add a few node delays to the value of TIME in eq. (1). However, $\mathscr{H} \propto N$ for such a graph, which is inconsistent with the optimal value of $\mathscr{H}$ we found above.

A simple mesh architecture can be used to realize the optimal value of $\mathscr{H}$. A constant number of channels will be used to transfer information among neighboring processors. Information will have to traverse $\propto N^{1/e}$ nodes in the worst case, leading to an identical growth rate of the device contribution to total computation time as given by eq. (3). Thus the propagation, device and serial contributions to the delay are all balanced in this architecture. Although this argument still does not demonstrate that the problem can be actually solved in $\propto N^{1/e}$ time in general, it serves as an indicator of the well balanced nature of this architecture.

The above simplistic derivation does not take into account heat removal considerations. Also, the fact that the relatively large node (processing) delays may result in overall larger values of TIME even when the growth rate of TIME is optimal has not been taken into account. Under these conditions, the above derivation suggests that local methods may be superior to global methods. We will take a closer look at each method in the context of solving a particular physical problem.

4. Quantum diffusion as a prototype physical problem

Consider a regular cartesian $e(=2$ or $3)$-dimensional array of $N \gg 1$ cells with $N^{1/e}$ cells along each side. We will speak of N as the problem size. Without loss of generality, we assume $N^{1/e}$ to be an integer. Initially, there is a certain number $f_0[\boldsymbol{i}]$ of bosonic particles in each cell, where $\boldsymbol{i}$ is a vector of e integral indices which range from 0 to $N^{1/e}-1$. For simplicity we assume that the array of cells and the initial distribution of particles is replicated periodically through all space (toroidal boundary conditions). Thus indice values outside the interval $[0, N^{1/e}-1]$ may be interpreted modulo base $N^{1/e}$.

At a certain average rate of say once every 1 µs, each particle has an (equal) probability of jumping into any one of its 2^e nearest diagonal neighbors. We will assume that the number of particles per cell M/N is very large so that we may ignore the probabilistic aspect of the problem [#5]. Thus, we formulate the problem of determining the number of particles in each cell at time t as follows

$$f_n[\boldsymbol{i}] = \frac{1}{2^e} \sum_{j \in A_i} f_{n-1}[\boldsymbol{j}] , \tag{4}$$

where A_i denotes the set of 2^e cells with *all* indices differing from cell $\boldsymbol{i}$ by unity and $n = t/(1\ \mu\mathrm{s})$. The total number of particles is of course conserved. Thus, given $f_0[\boldsymbol{i}]$, we can calculate $f_n[\boldsymbol{i}]$ recursively. This process may be expressed as

$$f_n[\boldsymbol{i}] = h_1[\boldsymbol{i}] * f_{n-1}[\boldsymbol{i}] , \tag{5}$$

$$f_n[\boldsymbol{i}] = h_n[\boldsymbol{i}] * f_0[\boldsymbol{i}] , \tag{6}$$

where $*$ denotes the e-dimensional discrete convolution operator. $h_n[\boldsymbol{i}]$ is the n-fold self convolution of $h_1[\boldsymbol{i}]$. The value of $h_1[\boldsymbol{i}]$ is $1/2^e$ at the 2^e nearest diagonal neighbors of the origin and zero elsewhere. For instance, for $e=1$, $h_1[i] = (..., 0, 0.5, 0, 0.5, 0, ...)$.

The diffusion equation is separable in cartesian coordinates. This means that multidimensional impulse responses can be written as a product of the unidimensional impulse responses. For instance, $h_n[i, j] = h_n[i] h_n[j]$, where $h_n[i]$ and $h_n[j]$ should be interpreted as two-dimensional functions while taking their product. (It is also interesting to note that these functions have the property that $h_n[i, j] = (h_n[i]\delta[j] * h_n[j]\delta[i])$.)

Our purpose is to calculate the state of the system at a particular final time n_f. As a special case, we will

[#5] The problem is deliberately designed in this manner so as to yield a simple formulation. It would be more straightforward to state the difference eq. (4) directly as the problem under consideration and dismiss the "quantum diffusion" interpretation. We however felt that the concrete interpretation would aid visualization of the process.

be interested in the steady state solution. Of course, in our simple example, if the total number of particles M is known to us beforehand, the steady state solution is also known to be a uniform distribution of M/N particles over all cells. However, we must not forget that this information is not initially available on the processors, which are only aware of the number of particles they contain.

Notice that whereas convolving $f_{n-1}[i]$ with $h_1[i]$ requires only local communication, convolving $f_0[j]$ with $h_n[i]$ for large n requires global communication.

In the following sections, we will consider several methods of constructing a machine that will calculate the state of the system at time n_f.

Before continuing however, we should clarify a common misconception. The parallel implementation of finite element calculations on an array of processors is often noted as an example of an application which requires only local communication among the processors [15]. However, since the state of the system at any point can eventually influence that at any other point, there are cases when it is beneficial for a processor to "look ahead" beyond its nearest neighbors. A boundary value problem requires a considerable amount of global transfer of information, since the resultant field distribution at any point depends on the values of the boundary at every point. Although it may depend more weakly on the values of more distant points, this does not weaken our argument. Consider that the field value at the boundary most distant from a given point is very large compared to anywhere else. Clearly this value will lead to a much different result than when it is a small value. Thus this information must somehow be conveyed to the other end.

Of course, there are examples of applications which are truly local. An extreme example of a problem which is infinitely local (which requires no communication) is that of independently adding a large number of pairs of numbers.

5. Solution by isomorphic simulation on a locally connected array of processors

We may compute the state of the system at any time step by using an array of simple processing elements, one for each cell, arrayed in identical fashion as the array of cells. Each processing element is connected to its nearest diagonal neighbors via $\chi \geqslant 1$ communication channels each of cross sectional area (or width) λ^{e-1}. Let the minimum pulse repetition interval and propagation velocity along these channels be denoted by T and c, respectively. Let each processor be a small cube (or square) with a linear dimension which is at least d_d. (It may have to be larger so that it can accomodate the wires coming out of it.) Each processor is capable of storing an L bit precision number representing the number of particles in the cell to which it corresponds and can update this value by averaging the values stored in its 2^e neighbors. Let this update take time τ_d. The minimum interelement spacing is given by $d \approx \max(d_d, \chi^{1/(e-1)}\lambda, d_Q)$, where d_Q is the interelement spacing required by heat removal considerations, and the second term accounts for the fact that there must be enough space between the elements for the passage of χ wires [#6]. In general, each iteration will take $\mathcal{T} = \max(\tau_d, d/c, LT/\chi)$ time. Here the third term accounts for the fact that it will take LT/χ time for L bits to the transferred over χ channels. For simplicity, let us assume that τ_d is defined inclusive of d_d/c and that λ and LT are small enough that the above expression reduces to $\mathcal{T} = \max(\tau_d, d_Q/c)$ with appropriate choice of χ. (This is readily verified for the parameters we choose for the numerical example in sect. 8 and $LT = 1$ ns.)

Let us now calculate d_Q. Let E_d denote the energy dissipation associated with each update on each processor, inclusive of the energy involved in transmitting information to its neighbors. (This quantity is constant and does not grow with N.) Q will denote the amount of power we can remove per unit cross section of our system. (This essentially means that we can remove QW^2 of power from a square or cube of edge length W. This heat removal model is derived in detail in ref. [13].) The average system power dissipation is $NE_d/\mathcal{T}$ and the edge length of our system is $N^{1/e}d$, so that the minimum value of d required by heat removal considerations (which we are denoting by d_Q) must satisfy

[#6] Throughout this paper, we will not make the distinction between the sum and maximum of a small number of positive numbers. Likewise, we will ignore numerical factors of the order of unity.

$$Q(N^{1/e}d_Q)^2 = NE_d/\mathcal{T} . \tag{7}$$

Solving for d_Q and substituting in $\mathcal{T}=\max(\tau_d, d_Q/c)$ we obtain

$$\mathcal{T}=\max(\tau_d, \tau_Q N^{1/3-2/3e}) , \tag{8}$$

where $\tau_Q=(E_d/Qc^2)^{1/3}$.

Now, the time its takes to compute the state of the system at time n_f may be expressed as TIME$=n_f\mathcal{T}$.

With increasing n, the state of the system will relax towards its steady state. Of course, in general, the system will never *exactly* reach steady state, (except in special cases where the solution falls in precisely to the steady state and stays there due to the discrete nature of our model).

We would not expect the system to reach steady state before $\sim N^{1/e}$ time steps, since this many steps are necessary for influences to propagate across the extent of the system. Exactly how long we must wait also depends on the error ϵ we are willing to tolerate. Let ϵ be defined as the worst case fractional deviation from steady state over all cells.

One way of determining the number of time steps necessary for given ϵ would be to carry out simulations for a variety of initial conditions. Instead, we will estimate the number of time steps n_ϵ it takes for an impulse of strength M to diffuse into a steady state of M/N particles per cell with fractional error ϵ.

The analysis is presented in the appendix, where it is found that for the one-dimensional case $n_\epsilon \sim N^2$. We could have guessed this result beforehand. The root mean square deviation of a random walk in any dimensional space is proportional to $n^{1/2}$. Thus we might consider that we have reached steady state when $n^{1/2}$ is comparable to the linear extent N of our system. This result easily generalizes to e dimensions for which the linear extent of the system is $N^{1/e}$. Thus in e dimensions

$$n_\epsilon = N^{2/e} . \tag{9}$$

The total time it takes to find the state of the system at time n_f and at steady state using isomorphic grid simulation are given in table 1.

Conventional VLSI complexity theory would predict the same growth rate of computation time, apart from the fact that heat removal is often not considered. These calculations would take $\propto n_f N$ time steps on a single processor machine.

6. Solution by systolic grid convolution on a locally connected array of processors

We now discuss a method of directly performing the convolution of eq. (6) on a nearest neighbor connected array, like the one used in the isomorphic simulation. For instance, consider the two-dimensional case. The convolution is written out explicitly as

$$f_n[i,j] = \sum_l \sum_k f_0[k,l] h_n[i-k, j-l] , \tag{10}$$

$$f_n[i,j] = \sum_l h_n[j-l] \sum_k f_0[k,l] h_n[i-k] , \tag{11}$$

since $h_n[i,j]$ is separable. Thus, this amounts to first computing $g_n[i,l]=\Sigma_k f_0[k,l]h_n[i-k]$ for every l systolically in the i (also k) direction and then computing $\Sigma_l g_n[i,l]h_n[j-l]$ for every i in the j (also l) direction. What is termed "systolic" convolution can be performed by rotating the values of $f_0[k,l]$ (or $g_n[i,l]$) and accumulating the properly weighted sums [17–19]. This takes $2N^{1/2}\mathcal{T}$ time where $\mathcal{T}$ is the same as that during isomorphic simulation. (It takes $N^{1/2}$ steps to convolve sequences of length $N^{1/2}$ in each of the two dimensions [17].) In e dimensions, this takes $eN^{1/e} \approx N^{1/e}$ time steps regardless of n_f.

Once again, conventional VLSI analysis would yield the same predictions for the computation time, apart from heat removal considerations. A single processor machine would take $N^{1+1/e}$ time steps.

7. Solution by Fourier transform techniques on a globally connected array of processors

Equation (6) may be evaluated conveniently using Fourier domain techniques, since convolution in coordinate space corresponds to multiplication in Fourier space. We assume the Fourier transform of the impulse response is pretabulated and piped in proper synchronicity. After multiplication with the Fourier transform of the initial distribution, we inverse transform to get the desired result. Thus the

Table 1

Comparison of total computation time with the various methods for calculating the state of the system at time step n_f: I, isomorphic simulation; S, systolic convolution and F, Fourier transforming. Line A gives the growth rate of delay in e dimensions when heat removal is ignored. Line B is for $e=2$ dimensions and line C for $e=3$ dimensions. The steady state for the isomorphic simulation is obtained by setting $n_f=N^{2/e}$. The other methods calculate any time step n_f, as well as the steady state, equally quickly. $\tau_0=\lambda/c$, $\tau_Q=(E_d/Qc^2)^{1/3}$. The notation (x, y) is short for $\max(x, y)$.

	I	I_∞	S	F
A	$n_f\tau_d$	$N^{2/e}\tau_d$	$N^{1/e}\tau_d$	$(\tau_d, N^{1/(e-1)}\tau_0)$
B	$n_f(\tau_d, \tau_Q)$	$N(\tau_d, \tau_Q)$	$N^{1/2}(\tau_d, \tau_Q)$	$(\tau_d, N\tau_0, N^{1/3}\tau_Q)$
C	$n_f(\tau_d, N^{1/9}\tau_Q)$	$(N^{2/3}\tau_d, N^{7/9}\tau_Q)$	$(N^{1/3}\tau_d, N^{4/9}\tau_Q)$	$(\tau_d, N^{1/2}\tau_0, N^{1/3}\tau_Q)$

total time of computation is about equal [#7] to the time it takes to evaluate the transform of the input function $f_0[i, j]$, which is not in general separable. (If it was separable, the problem would reduce to a unidimensional problem.)

The relationship between N point one-dimensional Fourier transforms and $N^{1/2}\times N^{1/2}$ point two-dimensional Fourier transforms is well known in the context of raster scan-folded spectrum techniques [20], where the two-dimensional Fourier transforming capability of an optical lens is exploited for high time–bandwidth product spectral analysis of one-dimensional analog signals. The FFT decomposition shown in fig. 1 may be interpreted either as a 16 point one-dimensional transform or a 4×4 two-dimensional transform. The reader is referred to refs. [20–22] for analytic discussions. The essential idea is that regardless of its dimensionality, the value of the transform at each spectral point depends on the value of the original function at every coordinate point, so that both problems require the same pattern of information flow. A construct for solving either problem can be used for the other with often only trivial modification (such as the inclusion of additional phase shifts along a few paths, as discussed in ref. [21]). This discussion generalizes to $N^{1/3}\times N^{1/3}\times N^{1/3}$ point three-dimensional transforms. Thus we may speak of the complexity of evaluating an N point Fourier transform without reference to its dimensionality.

[#7] Again ignoring a factor of 2.

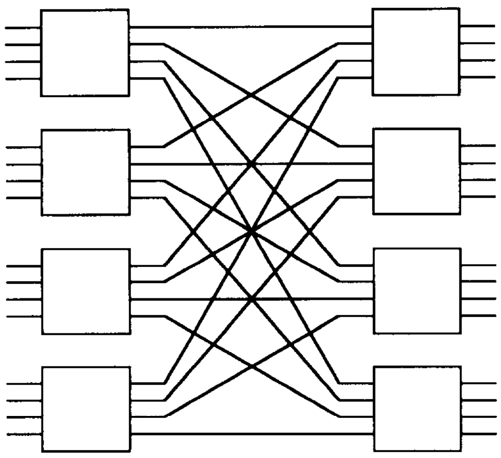


Fig. 1. Decomposition of an $N=16$ point FFT.

An N point Fourier transform can be computed effectively in both two and three dimensions by using a one- or two-dimensional lens respectively [23]. In three dimensions, one may situate the elements on a $N^{1/2}\times N^{1/2}$ array and use the third dimension for communication. Such a setup implies a linear extent and propagation delay of $\sim N^{1/2}\lambda$ and $\sim N^{1/2}\lambda/c$, respectively, assuming an $f^{\#}\sim 1$ optical processor. Here λ is interpreted as the wavelength of light. In two dimensions, the same setup must be squashed onto the plane. The N independent spatial channels now imply a linear extent $\sim N\lambda$, since they have only one dimension to pass through. Those familiar with the butterfly graph [14] on which the FFT is performed will immediately recognize that these results are analogs of the results stating that the butterfly lays out in $\propto N^{1/2}$ linear extent in three dimensions and $\propto N$ linear extent in two dimensions, with identical growth rate of longest wire length and propagation delay. These results are a consequence of the inherent non-partitionability of the Fourier transform operation [24]. (The VLSI implementation of the butterfly graph results in $\log_2 N$ node delays in addition to the propagation delays. Th s is avoided in the optical implementation.)

In conclusion, the evaluation of an N point Fourier transform on a globally connected array in the manner described requires propagation delay $\sim N^{1/(e-1)}\tau_0$ where $\tau_0=\lambda/c$.

Heat removal requires that the system linear extent be at least $(NE_d/\mathrm{TIME}\,Q)^{1/2}$, assuming the total energy dissipation NE_d is spread over the total time of computation TIME. (It is easy to show that any other choice is suboptimal.) Thus the total computation TIME is given as

$$\mathrm{TIME}=\max(\tau_d, N^{1/(e-1)}\tau_0, N^{1/3}\tau_Q)\,, \qquad (12)$$

where τ_Q is as defined before. Heat removal will have less and less importance as N increases. The results are again summarized in the table.

Conventional VLSI complexity theory predicts a $\propto \log N$ growth rate of delay for parallel implementation of the FFT, since propagation delays are ignored. The single processor implementation takes about $N\log_2 N \sim N$ time steps.

8. Comparison

The results are summarized in table 1. Let us ignore heat removal for the moment ($\tau_Q=0$). First consider the limit where device delays dominate (τ_d is large or N is small). The direct Fourier method is clearly superior in this case unless n_f is very small. (One arrives at a similar conclusion based on a single processor model.) Which of isomorphic simulation and systolic convolution would be preferred depends on the values of n_f and N. When n_f is small and N is large, the first method is to be preferred. When n_f is large however, systolic convolution is preferred. In particular, consider the steady state which takes $N^{2/e}\tau_d$ time with isomorphic simulation in contrast to $N^{1/e}\tau_d$ time with systolic convolution.

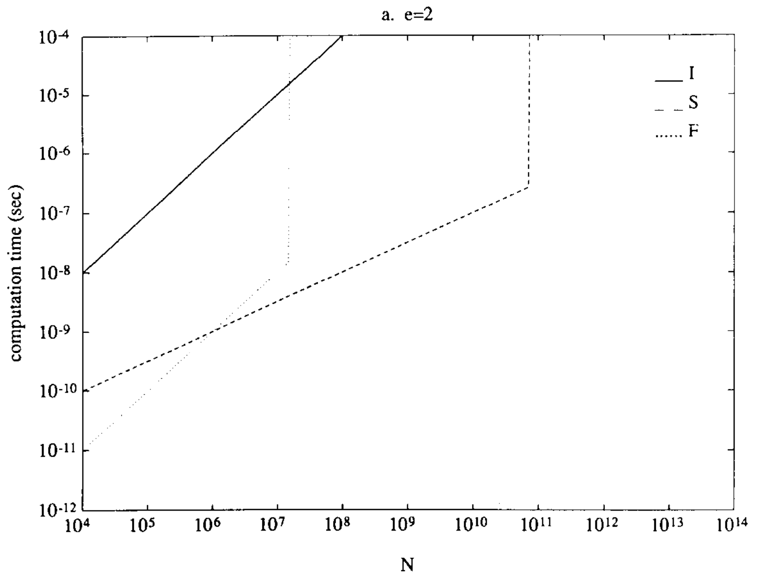
Now let us assume ultrafast devices and compare the growth rate of computation time as a function of N and n_f. Systolic convolution always exhibits a growth rate $N^{1/e(e-1)}$ better than Fourier transform methods. This result was anticipated in an earlier section where we discussed the solution of problems with high information content. In practice, $\tau_d \gg \tau_0$. Thus, systolic convolution will be preferred over Fourier methods when $N>(\tau_d/\tau_0)^{e(e-1)}$. For instance, with $\tau_d=1$ ps and $\tau_0=\lambda/c=1$ fs the condition is approximately $N>10^6$ in two dimensions and $N>10^{18}$ in three dimensions. Conversely, Fourier methods are preferred until $N=10^6$ in two dimensions and $N=10^{18}$ in three dimensions. We are not the first to speak of such a large number of very simple processors, Hillis has speculated about what might happen when we have a mole ($\sim 10^{23}$) of processors [4].

Now let us consider the effect of heat removal. This has little or no effect on our conclusions in two dimensions. In three dimensions, the leading terms will be $N^{4/9}\tau_Q$ for systolic convolution versus $N^{1/2}\tau_0$ for Fourier transforming. Letting $E_d=10$ fJ and [#8] $Q=1$ kW/cm^2 so that $\tau_Q\approx 0.1$ ps, we find that systolic grid convolution is preferred over the Fourier method only when $N>10^{36}$, an unreasonably large value by any standard.

To sum up, isomorphic simulation may be better when n_f is small, or when we want to track the whole evolution of the system up to n_f, rather than just its final state. (The problem of piping out this data from the system remains unsolved however.) Otherwise, especially when we want to calculate the steady state, this method is not very good. This is because of the inefficient way in which information transfer occurs. The averaging at every step results in loss of information for which communication resources have already been utilized, resulting (just like exchanging a developed piece in a game of chess) in inefficient resource utilization. Systolic convolution may be preferred over Fourier methods in two dimensions for large values of N. In three dimensions, the asymptotic superiority of systolic grid convolution over Fourier methods is so slight as not to be of any significance so that the latter is to be preferred.

Figure 2 illustrates the total computation time as a function of N for the three methods, anticipating nanoelectronic processors. The curves have been terminated when the system linear extent exceeds 10 m.

[#8] We choose Q more conservatively than derived in ref. [13], where it is shown that $Q=10$ kW/cm^2 is possible. In any event, since the relevant parameter $\tau_Q \propto Q^{-1/3}$, this has little effect on the results.



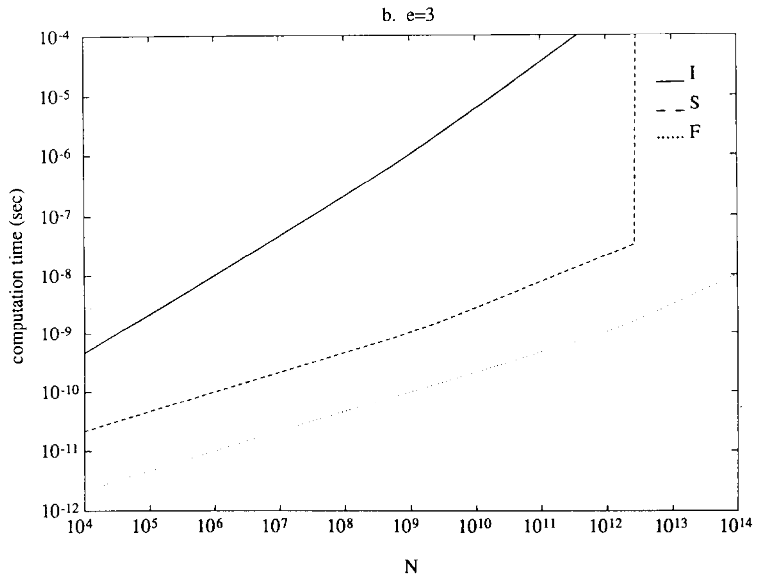


Fig. 2. Computation time for the three methods with $n_f = N^{2/e}$ for isomorphic simulation. $\tau_d = 1$ ps, $\tau_0 = 1$ fs, $\tau_Q = 0.1$ ps, $d_d = 1$ μm. The curves are terminated when the linear extent exceeds 10 m.

9. Discussion and extensions

Although it may seem that the diffusion process we have considered is a very special example, we stress that diffusion phenomena underly many natural occurrences. The diffusion and wave equations are often the starting point of a course in partial differential equations. In their steady state both reduce to Laplace's equation. The wave equation is often associated with coherent, orderly transfer of information, energy of particles; whereas the diffusion equation is associated with random transfer. The diffusion process is redundant in the sense that the same information is retransmitted several times only to be destroyed by the averaging process. We can do better than the isomorphic simulation of diffusion by carrying out the calculations in an orderly and efficient manner, rather than imitating the physical process itself. Similar arguments may be possible against other numerical methods which involve randomness and/or averaging, such as Monte Carlo or relaxation methods [#9].

Convolution is one of the basic operations in signal and image processing. Thus, we would expect extensions of our discussion to have implications in these areas. Of course, some applications require convolution with only a finite window function, and would probably the implemented using local methods. For other applications however, global methods may be preferable.

Similar analysis as we have presented can be carried out in different contexts. For instance, it is well known that the same logic function can be implemented with a fewer number of locally connected elements but with large overall logic depths, or with a larger number of globally connected elements with less logic depth [25]. The optimal implementation will lie somewhere in between.

Such studies may also be used to evaluate the usefulness of neural networks. We would not be surprised if similar the usefulness of highly connected systems are reached.

We only concentrated on extreme locality and extreme globality. Intermediate approaches are possible and may offer the optimum performance. For instance, given a family of algorithms for solving the diffusion equation on multidimensional meshes of every dimension, we may pick the optimum dimension. Also, we considered only nearest neighbor communication in the grid method. In serial computation often a bounded grid of higher order neighbors is used, leading to faster convergence. In parallel hardware, this might lead to a small amount of dilation, leading to a tradeoff. In recent years, a number of (serial) numerical methods involving higher order interpolation polynomials and combinations of spectral and finite elements methods have emerged [26], evidence of the fact that the optimum lies somewhere in between the two extremes of global and local methods. There seems much that is unexplored as far as the use of parallel hardware is concerned.

[#9] Redundant systems also have a significant advantage: reliability. However, one probably does not need as much redundancy as in the diffusion process for this purpose.

10. Conclusion

We discussed various methods of simulating a particular physical process, that of particulate diffusion, using *realistic* (in the sense of ref. [7]) hardware models. We considered isomorphic simulation of the physical process, systolic convolution and Fourier transform methods.

Systolic convolution is asymptotically superior to Fourier methods. However, for three-dimensional systems, the growth rate of delay is only slightly less (because of the effects of heat removal), so that for reasonable system sizes Fourier methods result in smaller total time of computation. Optical interconnections, which are ideally suited for such highly interconnected approaches, will thus find use in future nanoelectronic systems. This is despite the fact that the "width" of an optical channel ($\sim\lambda$) is very large compared to contemplated nanoelectronic dimensions [#10].

Acknowledgements

We greatly benefited from discussions with Hakan Oksuzoglu of Stanford University. We are grateful to Brian Wherrett of Heriot-Watt University for reading the manuscript and making many suggestions which greatly improved the paper, and also for bringing to our attention their related work [27].

The first author would like to extend thanks to Adolf W. Lohmann of the University of Erlangen-Nürnberg for teaching the concepts of locality–globality and isomorphic–nonisomorphic solutions, in a seminar he gave many years ago. The first author also acknowledges support of the Alexander von Humboldt Foundation through a Post Doctoral Research Fellowship.

[#10] The reader is also referred to ref. [10] where the limitations of normally and superconducting interconnections for *arbitrary small* linewidths are discussed.

This work originally appeared as part of a PhD thesis ([12], ch. 14).

Appendix

For simplicity we consider the one-dimensional case (i.e. $e=1$). A one-shot impulse of unit strength diffuses in the following Pascal's triangle-like manner:

$$\begin{array}{ccccccccc}
0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.500 & 0.000 & 0.500 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.250 & 0.000 & 0.500 & 0.000 & 0.250 & 0.000 & 0.000 \\
0.000 & 0.125 & 0.000 & 0.375 & 0.000 & 0.375 & 0.000 & 0.125 & 0.000
\end{array} \tag{13}$$

and so on. Thus for an impulse of strength M located at the origin ($i=0$) at $n=0$, the number of particles in location i after n steps is

$$f_n[i]=\frac{M}{2^n}\operatorname{com}\left(n,\frac{i+n}{2}\right), \quad -n\leqslant i\leqslant n\,, \tag{14}$$

for even $(n+i)$ and 0 for odd $(n+i)$. Since N is large, n_ϵ will also be large. Thus, using the DeMoivre–Laplace theorem [16] the above expression may be approximated as

$$\frac{M}{\sqrt{2\pi n}}\exp\left(-\frac{i^2}{2n}\right), \tag{15}$$

where we have included an additional factor of $\frac{1}{2}$ so as to ensure proper normalization over all values of i, rather than just odd or even values depending on whether n is even or odd. Now, remembering that we are employing cylindrical (cyclic) boundary conditions, the number of particles at the midpoint between the origins at $i=0$ and $i=N$ (which will be the latest to reach the steady state of M/N particles) is

$$\sum_{j=-\infty}^{\infty}\frac{M}{\sqrt{2\pi n}}\exp\left(-\frac{(N/2+jN)^2}{2n}\right). \tag{16}$$

Using some algebra involving Fourier transform techniques [2], this summation may be expressed in the equivalent form

$$\frac{M}{N}\left\{1+2\sum_{j=1}^{\infty}(-1)^j\exp\left[-\left(\frac{\pi\sqrt{2n}}{N}\right)^2 j^2\right]\right\}. \tag{17}$$

The second term in square brackets is simply the fractional error we do not want to be greater than ϵ. Since we are confronted with an alternating series, we can ensure the error to be less than ϵ by ensuring that

$$n\geqslant N^2\left(\frac{\ln(2/\epsilon)}{2\pi^2}\right)$$
$$\approx N^2[0.035+0.117\log_{10}(1/\epsilon)]\,. \tag{18}$$

Because of the weak dependence on ϵ, we are justified in writing $n_\epsilon\sim N^2$. In other words, for almost all practical values of ϵ, taking $n\sim N^2$ is as good as $n=\infty$.

References

[1] E.A. Volkov, Numerical methods (Mir Publishers, Moscow, 1986).

[2] R.N. Bracewell, The Fourier transform and its application, 2nd Ed. (McGraw-Hill, New York, 1986).

[3] W.D. Hillis, Intern. J. Theor. Phys. 21 (1982) 255.

[4] W.D. Hillis, The connection machine (MIT Press, Cambridge, MA, 1985).

[5] A. Orlitsky and A. El Gamal, Communication complexity, in: Complexity in information theory, ed. Y.S. Abu-Mostafa (Springer, Berlin, 1988).

[6] H.M. Ozaktas, Opt. Eng. 31 (1992) 1563.

[7] A.C. Hartmann and J.D. Ullman, Model categories for theories of parallel systems, in: Parallel computing: theory and experience, eds. G.J. Lipovski and M. Malek (Wiley, New York, 1986).

[8] W.J. Dally, A VLSI architecture for concurrent data structures (Kluwer, Dordrecht, 1987).

[9] G. Frazier. An ideology for nanoelectronics, in: Concurrent computations, ed. S.K. Tewksbury (Plenum, New York, 1988) ch. 1.

[10] H.M. Ozaktas and J.W. Goodman, in: Frontiers of computing systems research, ed. S.K. Tewksbury, Vol. 2 (Plenum Press, New York, 1991) p. 61.

[11] H.M. Ozaktas and J.W. Goodman, Optimal partitioning of very large scale optoelectronic computing systems, in: Optical Society of America 1990 Annual Meeting Technical Digest, 1990.

[12] H.M. Ozaktas, A physical approach to communication limits in computation, PhD thesis, Stanford University, Stanford, California, 1991.

[13] H.M. Ozaktas, H. Oksuzoglu, R.F.W. Pease and J.W. Goodman, Intern. J. Electron. 73 (1992) 1227.

[14] J.D. Ullman, Computational aspects of VLSI (Computer Science Press, Rockville, Maryland, 1984).

[15] C.E. Leiserson, IEEE Trans. Comp. 34 (1985) 892.

[16] Athanasios Papoulis, Probabi ity, random variables and stochastic processes, 2nd Ed. (McGraw-Hill, New York, 1965).

[17] S.Y. Kung, VLSI array ›rocessors (Prentice-Hall, Englewood Cliffs, NJ, 1988).

[18] H.T. Kung, Computer January (1982) 37.

[19] H.T. Kung, in: IEEE 13th An ual Intern. Symposium on Computer Architecture (1986 p. 49.

[20] D. Casasent, in: Optical data ›rocessing, ed. D. Casasent (Springer, Berlin, 1978) ch. 8.

[21] B. Gold and C.M. Rader, D gital processing of signals (McGraw-Hill, New York, 19(9).

[22] A.V. Oppenheim and R.W. Sha 'er, Digital signal processing (Prentice-Hall International, L ondon, 1975).

[23] J.W. Goodman, Introduction to Fourier optics (McGraw-Hill, New York, 1968).

[24] C.D. Thompson, in: Proc. 11th Annual ACM Symposium on the Theory of computing, Association for Computing Machinery, 1979, p. 81.

[25] H.B. Bakoglu, Circuits, interconnections and packaging for VLSI (Addison-Wesley, Reading, MA, 1990).

[26] J.P. Boyd, Chebyshev and Fourier spectral methods (Springer, Berlin, 1989).

[27] B.S. Wherrett, J.F. Snowdon, S. Bowman and A. Kashko, in: Optical computing, SPIE Proceedings 1806, SPIE, Bellingham, Washington (1992) p. 164.