Synchronizing Large VLSI Processor Arrays

ALLAN L. FISHER, MEMBER, IEEE, AND H. T. KUNG

Abstract — **Highly parallel VLSI computing structures consist of many processing elements operating simultaneously. In order for such processing elements to communicate among themselves, some provision must be made for synchronization of data transfer. The simplest means of synchronization is the use of a global clock. Unfortunately, large clocked systems can be difficult to implement because of the inevitable problem of clock skews and delays, which can be especially acute in VLSI systems as feature sizes shrink. For the near term, good engineering and technology improvements can be expected to maintain the feasibility of clocking in such systems; however, clock distribution problems crop up in any technology as systems grow. An alternative means of enforcing necessary synchronization is the use of self-timed asynchronous schemes, at the cost of increased design complexity and hardware cost. Realizing that different circumstances call for different synchronization methods, this paper provides a spectrum of synchronization models; based on the assumptions made for each model, theoretical lower bounds on clock skew are derived, and appropriate or best possible synchronization schemes for large processor arrays are proposed.**

One set of models is based on assumptions that allow the use of a pipelined clocking scheme where more than one clock event is propagated at a time. In this case, it is shown that even assuming that physical variations along clock lines can produce skews between wires of the *same* length, any one-dimensional processor array can be correctly synchronized by a global pipelined clock while enjoying desirable properties such as modularity, expandability, and robustness. This result cannot be extended to two-dimensional arrays, however; the paper shows that under this assumption, it is impossible to run a clock such that the maximum clock skew between two communicating cells will be bounded by a constant as systems grow. For such cases, or where pipelined clocking is unworkable, a synchronization scheme incorporating both clocked and "asynchronous" elements is proposed.

Index Terms — **Clock skew, processor arrays, synchronization, systolic arrays, VLSI complexity.**

I. INTRODUCTION

In describing a processor array algorithm, it is often convenient to picture the processors as running in lock step. This *synchronized* view, for example, often makes the definition of the structure and its correctness relatively easy to follow: computations proceed in discrete steps which may be cleanly characterized. Perhaps the simplest means of synchronizing an ensemble of cells is the use of broadcast clocks. A clocked system in general consists of a collection of functional units whose communication is synchronized by external clock signals. A variety of clocking implementations are possible; the essential point is that by referring to the global time standard represented by the clock, communicating cells can agree on when a cell's outputs should be held constant and when a cell should be sensitive to its input wires. When different cells receive clock signals by different paths, they may not receive clocking events at the same time, potentially causing synchronization failure. These synchronization errors due to clock skews can be avoided by lowering clock rates and/or adding delay to circuits, thereby slowing the computation. The usual clocking schemes are also limited in performance by the time needed to drive clock lines, which will grow as circuit feature size shrinks relative to total circuit size. Therefore, unless operating at possibly unacceptable speeds, very large systems controlled by global clocks can be difficult to implement because of the inevitable problem of clock skews and delays.

As a practical aside, we should note that at current LSI circuit densities, clock distribution is still a solvable problem. Two somewhat pessimistic studies of which we are aware [3], [5] do not take into account either the tricks that a circuit designer can use to reduce the RC constant of his clock tree or the promise of multiple-layer metallization and low-resistance silicides. Given these factors, it seems that the usual clocking schemes should remain feasible for on-chip synchronization in the near term. Moreover, for some specific structures, such as one-dimensional arrays, clocking can be effectively used even in the presence of large signal propagation delays (see Section V-A).

An alternative approach to clocking is self-timing [10], in which cells synchronize their communication locally with some variety of "handshaking" protocol. It is easy to convince oneself that any synchronized parallel system where cells operate in lock step can be converted into a corresponding asynchronous system of this type that computes the same output — the asynchronous system is obtained by simply letting each cell start computing as soon as its inputs become available from other cells. The self-timed asynchronous scheme can be costly in terms of extra hardware and delay in each cell, but it has the advantage that the time required for a communication event between two cells is independent of the size of the entire processor array. A serious disadvantage of fully self-timed systems is that, given current digital de-

Manuscript received September 24, 1982; revised September 19, 1984. This work was supported in part by the Office of Naval Research under Contracts N00014-76-C-0370, NR 044-422 and N00014-80-C-0236, NR 048-659, in part by the Defense Advanced Research Projects Agency (DOD), ARPA Order 3597, monitored by the Air Force Avionics Laboratory under Contract F33615-81-K-1539, in part by a National Science Foundation Graduate Fellowship, and in part by an IBM Graduate Fellowship.

The authors are with the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213.

0018-9340/85/0800-0734$01.00

sign methodology, they can be difficult and expensive to design and validate.

An advantage that self-timed systems often enjoy, in addition to the absence of clock skew problems, is a performance advantage that results from each cell being able to start computing as soon as its inputs are ready and to make its outputs available as soon as it is finished computing. This allows a machine to take advantage of variations in component speed or data-dependent conditions allowing faster computation. This advantage will seldom exist in regular arrays such as systolic systems, however, for two reasons.

1) Usually, each cell in a regular array performs the same kind of computation as every other cell; thus, there is little opportunity for speed variation.

2) In cases where variations do exist, the throughput of computation along a path in an array is limited by the slowest computation on that path. The probability that a worst case computation will appear on a path with k cells is $1 - p^k$ where p is the probability that any given cell will *not* be performing a worst case computation. This quantity approaches unity as k grows, so large arrays will usually be forced to operate at worst case speeds.

The result of these considerations is that clocking is generally preferable to self-timing in the synchronization of highly regular arrays. This paper derives techniques for synchronizing large arrays, using clocking where possible and preserving some of the advantages of clocked schemes where clocking breaks down.

II. Basic Assumptions

The basic model that we will use for considering synchronization of VLSI processor arrays is as follows.

A1) Intercell data communications in an *ideally synchronized* processor array, in which all processors operate in lock step, are defined by a directed graph COMM, which is laid out in the plane. Each node of COMM, also called a *cell,* represents a cell of the array, and each directed edge of COMM, called a *communication edge,* represents a wire capable of sending a data item from the source cell to the target cell in every cycle of the system. Any two cells connected by a communication edge are called *communicating cells.*

A2) A cell occupies unit area.

A3) A communication edge has unit width.

We now add assumptions which provide the basis for clocked implementations of ideally synchronized arrays.

A4) A clock for a clocked processor array is distributed by a rooted binary tree CLK, which is also laid out in the plane. A cell of COMM can be clocked if the cell is also a node of CLK.

A5) A clocked system may be driven with clock period $\sigma + \delta + \tau$ where σ is the maximum clock skew between any two communicating cells, δ is the maximum time for a cell's outputs to be computed and propagated to a communicating cell, and τ is the time to distribute a clocking event on CLK.

This assumption is an abstraction of properties common to all clocking schemes. The detailed relationships between these parameters and other more specific parameters such as flip-flop setup and hold times depend on the exact clocking method used. An exact representation of minimum clock period might be something like $\max(\tau, 2\sigma + \delta)$ in a particular case, but such formulas will exhibit the same type of growth with respect to system size as the simple sum used here.

Note that if we adopt the usual convention that the clock tree is brought to an equipotential state before a new clock event is transmitted, eliminating clock skew can lead only to a constant factor increase in performance since it must always be true that $\sigma \leq \tau$. In particular, speed of light considerations impose the following condition.

A6) The time τ required to distribute a clocking event on a clock tree CLK in a particular layout is bounded below by $\alpha \cdot P$ where $\alpha > 0$ is a constant and P is the (physical) length of a longest root-to-leaf path in CLK.

Thus, since the clock tree must reach each cell in the array, large arrays which are synchronized by equipotential clocking must have clock periods at least proportional to their layouts' diameters. Note that in the remainder of this paper we will relate transmission delays to wire length; delays are caused by other factors, of course, but we choose to treat them together as a "distance" metric.

In the case where an array grows too big for its clock tree to be driven at the desired speeds due to the time needed to bring long wires to an equipotential state, it is possible to take advantage of the propagation delay down a long wire by having several clock cycles in progress along its length. This mode of clocking is often used in large mainframe computers, but not, to our knowledge, on chips. The electrical problems of passing a clean signal on a chip in this fashion are severe, due to analog phenomena such as damping and reflections. We can instead simulate this behavior by replacing long wires with strings of buffers, which will restore signal levels and prevent backward noise propagation. These buffers are spaced a constant distance apart; a good candidate is that distance which will cause wire delays between buffers to be of the same size as a buffer's propagation delay. This allows us to replace assumption A6) with the following.

A7) If CLK is a buffered clock tree, the time τ required to distribute a clocking event on a particular unbuffered segment of CLK is the maximum delay through a buffer and its output wire. Thus, τ is a constant independent of the size of the array.

To ensure that successive clock events remain correctly spaced along the clock path, we make the following assumption.

A8) The time for a signal to travel on a particular path through a buffered clock tree is invariant over time.

The following section describes two clock skew models based on the above assumptions, and Sections IV and V explore the problem of clocking under these models. Section VI considers the case where assumption A8) does not hold, and hence condition A6) holds rather than condition A7). Section VII briefly discusses the practicality of the models and the results obtained.

III. Two Models of Clock Skew

Given a basic model consisting of conditions A1) through A5), plus A7) and A8), the following sections consider the implications of two models of clock skew. First, in Section IV we consider the case where clock skew between two cells depends on the difference in their physical distances from the root of the clock tree. This *difference model* corresponds reasonably well to the practical situation in high-speed systems made of discrete components, where clock trees are often wired so that delay from the root is the same for all cells. Formally, we assume the following.

A9) The clock skew between two nodes of CLK, with respect to a given layout, is bounded above by $f(d)$ where f is some monotonically increasing function and d is the positive difference between the (physical) lengths of the paths on CLK that connect the two nodes to the root.

This assumption is illustrated in Fig. 1. The two circles connected by the dashed line have clock skew between them which is no more than $f(d)$ where d is the length of the crosshatched segment. This segment represents the difference between the cells' distances to their nearest common ancestor in the clock tree.

As systems grow, small variations in electrical characteristics along clock lines can build up unpredictably to produce skews even between wires of the same length. In the worst case, two wires can have propagation delays which differ in proportion to the sum of their lengths. Especially since it is not possible to tune the clock network of a system on a single chip, Section V considers a model in which the skew between two nodes depends on the distance between them along the clock tree. Formally, the *summation model* (so called because the distance between two nodes is the sum of their distances from their nearest common ancestor, while the difference measure used above is the difference between those distances) uses the following upper and lower bound assumptions.

A10) The clock skew between two nodes of CLK, with respect to a given layout, is bounded above by $g(s)$ where g is some monotonically increasing function and s is the (physical) length of the path on CLK that connects the two nodes.

A11) The clock skew between two nodes of CLK, with respect to a given layout, is bounded below by $\beta \cdot s$ where $\beta > 0$ is some constant and s is the (physical) length of the path on CLK that connects the two nodes.

Fig. 2 illustrates these assumptions; here both the upper and lower bounds on the skew between the two communicating cells depend on the entire length of the path between them, which is the sum of their distances to their nearest common ancestor in the tree.

The two models of clock skew introduced above can be formally derived as follows, for the case when both functions f and g are linear. Let h_1 and h_2, with $h_1 \geq h_2$, be the distances of any two cells to their nearest common ancestor in the clock tree. Let $m + \varepsilon$ and $m - \varepsilon$ be the maximum and minimum time, respectively, to transmit a clock signal across a wire of unit length where ε corresponds to the variations in electrical characteristics along clock lines. Then the clock skew σ between the two cells can be as large as

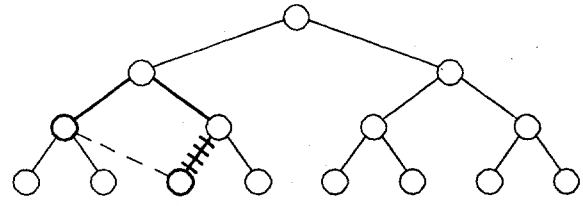


Fig. 1. Skew in the difference model.

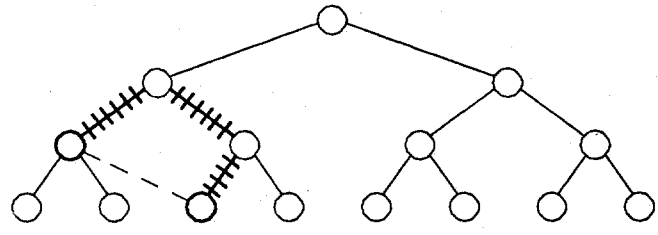


Fig. 2. Skew in the summation model.

$$\sigma = h_1(m + \varepsilon) - h_2(m - \varepsilon) = (h_1 - h_2)m + (h_1 + h_2)\varepsilon.$$

Noticing that $d = h_1 - h_2$, $s = h_1 + h_2$, and $s \geq d \geq 0$, we have

$$(m + \varepsilon) \cdot s \geq \sigma = m \cdot d + \varepsilon \cdot s \geq \varepsilon \cdot s.$$

We see that the upper and lower bounds correspond directly to assumptions A10) and A11) used in the summation model, while the difference model covers the case when terms involving ε can be ignored.

IV. Clocking Under the Difference Model

Assuming the basic model defined in Section II, along with condition A9), which states that the skew between two cells is bounded above by a function of the difference between their distances from the root, we note that only a bounded amount of clock skew will occur if we ensure that all nodes in COMM are equidistant (with respect to the clock layout) from the root of CLK. This can be achieved for any layout for COMM of bounded aspect ratio, without increasing the areas of the layout by more than a small constant factor, by distributing the clock through an H-tree [8]. This scheme is illustrated for linear, square, and hexagonal arrays in Fig. 3, in which heavy lines represent clock edges and thin lines represent communication edges.

More precisely, we have the following result.

Lemma 1: For any given layout of bounded aspect ratio, it is possible to run a clock tree such that all nodes in the original layout are equidistant (with respect to the clock tree) from the root of the tree, and the clock tree takes an area no more than a constant times the area of the original layout.

By a theoretical result [1] that any rectangular grid (for example, an $n^{2/3} \times n^{1/3}$ grid) can be embedded in a square grid by stretching the edges and the area of the source grid by at most a constant factor, we have the following theorem.

Theorem 2: Under the difference model of clock skew, any ideally synchronized processor array with computation and communication delay δ bounded by a constant can be simulated by a corresponding clocked system operating with a clock period independent of the size of the array, with no more than a constant factor increase in layout area.

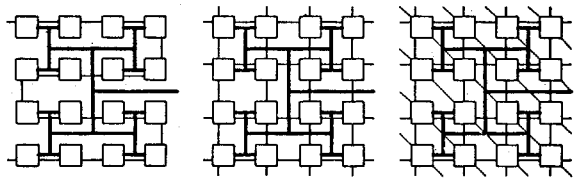


Fig. 3. H-tree layouts for clocking. (a) Linear arrays. (b) Square arrays. (c) Hexagonal arrays.

V. Clocking Under the Summation Model

This section relaxes the assumption of the previous section by using the summation model rather than the difference model for clock skews. The clock skew between two nodes of CLK, with respect to a given layout, is related to the (physical) length of the path on CLK that connects the two nodes. Note that because the summation model is weaker than the difference model, any clocking scheme working under the summation model must also work under the difference model. The reverse of the statement is not true, however. For example, the clocking scheme illustrated in Fig. 3(a) for linear arrays may not work under the summation model since two communicating cells (such as the two middle cells on the left-hand side of the layout) could be connected by a path on CLK whose length can be arbitrarily large as the size of the array grows. In the following we give another clocking scheme for linear arrays that works even under the summation model for clock skew; in addition, we show that it is impossible, under this model, to clock a two-dimensional array in time independent of its size. In this sense, linear arrays are especially suitable for clocked implementation.

A. Clocking One-Dimensional Processor Arrays

Given any ideally synchronized one-dimensional array [Fig. 4(a)], we propose a corresponding clocked array [Fig. 4(b)] obtained by running a clock wire along the length of the one-dimensional array. By A10) the maximum clock skew between any two neighbors is bounded above by a constant $g(s)$ where s is the center-to-center distance between neighboring cells. Thus, we have the following result.

Theorem 3: Under the summation model of clock skew, any ideally synchronized one-dimensional processor array with computation and communication delay δ bounded by a constant can be simulated by a corresponding clocked system, as illustrated in Fig. 4, operating at a clock period independent of the size of the array.

Skew between the host and the ends of the array can be handled similarly by folding the array in the middle (Fig. 5), and the array can be laid out with any desired aspect ratio by using a comb-shaped layout (Fig. 6).

With the clocking schemes illustrated, we see that the clock period for any one-dimensional processor array can be made independent of the size of the array. As a result, the clocked array may be extended to contain any number of cells using the same clocked cell design. These clocked schemes are probably the most suitable for synchronizing one-dimensional arrays due to their simplicity, modularity, and expandability. Note that one-dimensional arrays are especially important in practice because of their wide applicabilities and their bounded I/O requirements [4].

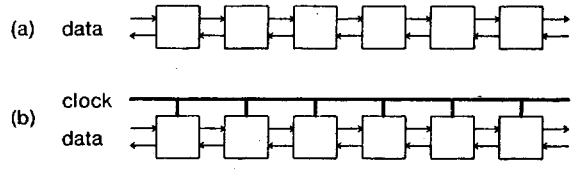


Fig. 4. (a) Ideally synchronized one-dimensional array. (b) Corresponding clocked array.

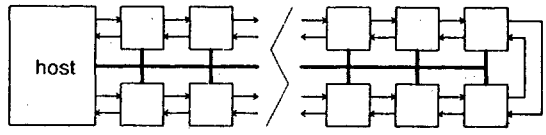


Fig. 5. Array folded to bound skew with host.

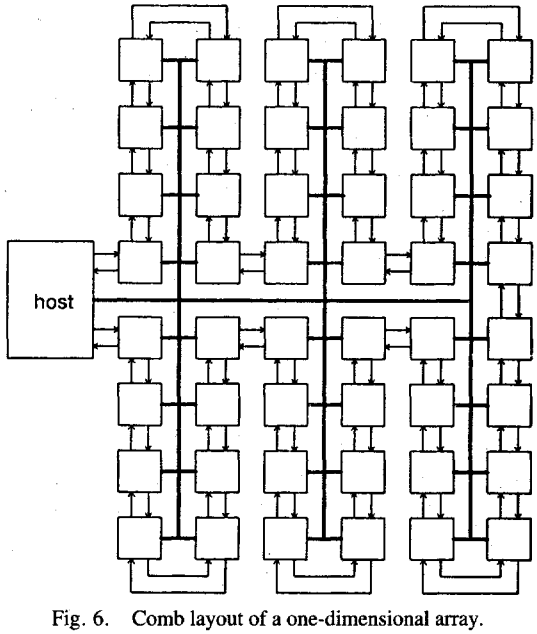


Fig. 6. Comb layout of a one-dimensional array.

B. A Lower Bound Result on Clock Skew

We show here that the result of Theorem 3 for the one-dimensional array cannot be extended to two-dimensional structures. Consider any layout of an $n \times n$ array clocked by a global clock tree CLK; the nodes of CLK include all cells of the array. Let σ be the *maximum clock skew* between two communicating cells of the array. We want to prove that σ cannot be bounded above by any constant independent of n. We use the following well-known result [6].

Lemma 4: To bisect an $n \times n$ mesh-connected graph at least $c \cdot n$ edges have to be removed, where $c > 0$ is a constant independent of n.

Bisecting a graph means partitioning the graph into two subgraphs, each containing about half of the nodes of the original graph. Here, for the $n \times n$ mesh-connected graph we assume that none of the subgraphs contain more than $(23/30) \cdot n^2$ nodes. We also use the following simple lemma without giving a proof.

Lemma 5: For any subset M of at least two nodes of a binary tree, there exists an edge of the tree such that its

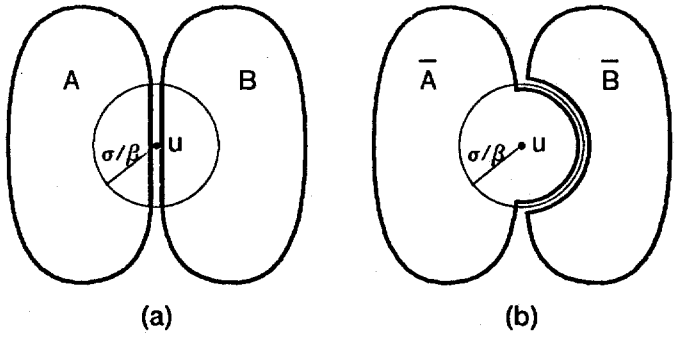


Fig. 7. (a) Original partition of the communication graph. (b) New partition of the communication graph.

removal from the tree will result in two disjoint subtrees, each having no more than two-thirds of the nodes in M.

The n^2 cells of the $n \times n$ array form a subset of nodes of CLK. By Lemma 5 we know that by removing a single edge, CLK can be partitioned into two disjoint subtrees such that each subtree has no more than $(2/3) \cdot n^2$ cells. Denote by A and B the sets of cells in the two subtrees. Let u be the root of the subtree that contains cells in A. Consider the circle centered at u and with radius of σ/β where β is defined in A11) [Fig. 7(a)]. If there are $\geq (1/10) \cdot n^2$ cells inside the circle, then by A2)

$$\pi(\sigma/\beta)^2 \geq n^2/10, \qquad \text{or } \sigma = \Omega(n),$$

and thus σ cannot be bounded above by any constant independent of n. Suppose now that there are fewer than $(1/10) \cdot n^2$ cells inside the circle. Note that any of those cells in A which are outside the circle cannot reach any cell in B by a path on CLK with (physical) length $\leq \sigma/\beta$. Thus, these cells cannot have any communicating cells in B (with respect to the $n \times n$ array) since by A11) the clock skew between these cells and any cell in B would be greater than $\beta \cdot \sigma/\beta = \sigma$, and the clock skew between any two neighboring cells is assumed to be no more than σ. Now let $\overline{A}$ be the union of A and the set of cells in the circle, and $\overline{B}$ be B minus the set of cells in the circle, as in Fig. 7(b). Then $\overline{A}$ and $\overline{B}$ form a partition of the $n \times n$ array, and each of them has no more than $(1/10) \cdot n^2 + (2/3) \cdot n^2 = (23/30) \cdot n^2$ cells. From Fig. 7(b), we see that any edge in the $n \times n$ array connecting a cell in $\overline{A}$ and a cell in $\overline{B}$ must cross the boundary of the circle. Since the length of the boundary is $2\pi\sigma/\beta$, by A3) $\overline{A}$ and $\overline{B}$ are connected by no more than $2\pi\sigma/\beta$ edges. By Lemma 4 we have $2\pi\sigma/\beta \geq c \cdot n$, or

$$\sigma = \Omega(n).$$

Therefore, as n increases, σ grows at least at the rate of n; we see that it is impossible to run a global clock for the $n \times n$ array such that the maximum clock skew σ between communicating cells will be bounded above by a constant, independent of n.

The above proof for two-dimensional mesh graphs can be generalized to deal with other classes of graphs. For the generalization, we need to define the *minimum bisection width* of a graph [11], which is the number of edge cuts needed to bisect the graph. For example, by Lemma 4 the minimum bisection width of an $n \times n$ mesh-connected graph is $\mu(n)$. We have the following general result.

Theorem 6: Suppose that the minimum bisection width of an N-node graph is $\Omega(W(N))$ and $W(N) = O(\sqrt{N})$. Then

$$\sigma = \Omega(W(N)).$$

Since under the summation model of clock skew, two-dimensional $n \times n$ processor arrays cannot be efficiently implemented by clocked controls, their implementation should be assisted by some self-timed scheme, as discussed in the next section.

VI. Hybrid Synchronization

In the absence of the invariance condition A8), in which case pipelined clocking fails, or for communication graphs with asymptotically growing clock skews under the summation model, global clocking is unable to provide constant clock rates as a system grows. In this case, a scheme similar to one described by Seitz [10], where local clocks are controlled by a self-timed handshaking synchronization network, can be used.

In this approach, we break up the layout into bounded-size segments called *elements* and provide each element with a local clock distribution node. The clock distribution nodes employ a handshaking protocol to synchronize among themselves and then distribute clock signals to the cells in their elements. Given assumptions about the maximum delay of a computation node and its communication wires, we can clock the cells in each element in constant time. This structure is illustrated in Fig. 8, in which the heavy lines and black boxes represent the self-timed synchronization network and the narrow lines represent local clock distribution to the cells in each element. Note that the subordination of the local clocks to the self-timed network avoids the possibility of synchronization failure due to a flip-flop entering a metastable state since an element stops its clock synchronously and has its clock started asynchronously.

This provides the asymptotic performance of a self-timed system by making all synchronization paths local while isolating the self-timed logic to a small subsystem and allowing the cells to be designed as if the entire system were globally clocked. The hybrid approach has the additional advantage that a single synchronization design can be used for many different structures. This simplification of the usual fully self-timed scheme is made possible by the fact that we are willing to assume a maximum delay for the cells; this is the same assumption made in ordinary clocked schemes. Note that we are willing to let the entire array operate at worst case cell speed since even a fully self-timed array would usually wind up operating at that speed regardless.

VII. Practical Implications

This section addresses some practical aspects of the material of this paper. Perhaps the most important practical issue raised in this study is the question of the use of pipelined clocking on chips; this section discusses some of the potential limits to this technique and presents some simple

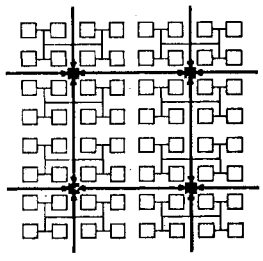


Fig. 8. Hybrid synchronization scheme.

experimental evidence suggesting its practicability. It also discusses the practical relevance of the difference and summation models of clock skew.

The practicality of pipelined clocking hinges on two issues: the limitations of the uniformity assumption A7), and the delay associated with distributing clocks in a conventional fashion. Pipelined clocking only makes sense if clock event transmission is uniform enough to gain an advantage over equipotential clocking.

This relationship, in turn, depends on the relative speeds of logic and interconnect. Pipelined clock trees, with short interconnection paths, will run at logic switching speeds (to the extent that uniformity obtains). The speed at which equipotential clock trees can run is determined by the impedance of the interconnect and by its physical dimensions. We would thus expect pipelined clocking to be most applicable where switching speeds are high and interconnect is long and has high impedance; for example, wafer-scale gallium arsenide may be a likely candidate.

The uniformity of transmission of clock events is subject to a number of factors. One obvious limitation is the uniformity of a buffer in passing rising and falling edges. For an nMOS superbuffer, for example, making transit times for rising and falling edges the same requires careful circuit tuning, and the resulting circuit will be very sensitive to manufacturing process parameters. One solution to this problem is to make each buffer respond only to rising edges on its input and to generate its own falling edges with a one-shot pulse generator. This solution has the disadvantage that the pulse width must be either wired into the circuit or programmable by some means. This may actually be convenient in some cases, if the pulse generating circuitry can be designed to model the delay of the logic circuitry. Beyond static considerations, however, the transmission of clock events can also be affected by noise, for example, internet capacitive coupling in MOS circuits. This problem can only be avoided by careful design, and further research is needed in estimating its magnitude.

Another, simpler approach to the rising/falling edge problem is to build a distribution line as a string of inverters. If the impedance of the outputs of the odd inverters is the same as that of the even inverters, rising and falling edges should traverse the string at essentially the same speed. Although this approach eliminates any inherent bias in favor of one type of edge, it does not result in speed independent of the length of the inverter string. Assume that the discrepancy between rising and falling transit times for a pair of inverters is normally distributed with a mean of zero and variance V. The sum of the discrepancies of n inverter pairs will be similarly distributed, with variance nV. If a fixed yield, independent of n, is desired, chips with a discrepancy sum proportional to the standard deviation, hence proportional to $\sqrt{n}$, must be accepted. Since a minimum condition for a given chip to run with cycle time T is that the sum of discrepancies be no greater than T, some chips will run with cycle times at least proportional to $\sqrt{n}$. The constant factors involved may still be small enough, however, to make this scheme feasible in practice.

As a simple trial of practical issues, an nMOS chip consisting of a string of 2048 minimum inverters was designed, without any special attention paid to making interconnect impedance uniform. An equipotential single phase clock signal could be run through the entire string with a cycle time of approximately 34 μs; even with the disadvantage of a slight bias in the circuit design toward falling edges, a pipelined clock could be run with a cycle time of 500 ns, 68 times faster. The same speedup was observed on five separate chips, indicating that the effect of the bias in the circuit design dominated the type of probabilistic effects described above. Assuming that transit times and any discrepancy between rising and falling edges scale linearly, a similar inverter string of any length could be clocked 68 times faster in pipeline mode than in equipotential mode. This figure does not indicate that pipelined clocking is actually applicable in this case; a chip of this size in this technology could easily be clocked with a 50 ns cycle time with a well-designed low-resistance equipotential clock. However, it does suggest that pipelined clocking may well be feasible where switches are fast and wires are slow.

The second practical issue related to this work is the question of the applicability of the difference and summation models of clock skew. First, both models apply only where pipelined clock distribution is possible; otherwise, clock period inevitably grows with the system, and the only means of improving performance are technology improvement, clever design, and self-timing. For the difference model to apply and for H-tree or other equidistant clocking schemes to be useful, it must be possible to closely control the "length" (that is, the delay characteristics) of the clock tree. This is possible in systems where wires are discrete entities that can be tuned, and indeed this is common practice in such systems. Whether this is true for integrated circuits is another question, hinging on the variability of the fabrication process and on noise characteristics.

The summation model is much more robust. Given the possibility of pipelined clocking, almost any imaginable means of transmitting clock events will have the property that cells close together on the clock tree will have bounded skew between them. We can thus be confident that linear arrays and similar structures will work as well as pipelined clocking can work.

VIII. Concluding Remarks

In this paper, we have analyzed the effect of clocked synchronization on the performance of large processor arrays. We have identified the key issues on which this depends

(clock delay and clock skew) and have proposed means of implementing pipelined clocking for integrated circuits. We have considered two models of the dependence of clock skew on layout properties and have derived upper and lower bounds for the performance of processor arrays of varying topologies. The key results here are that one-dimensional arrays can be clocked at a rate independent of their size under fairly robust assumptions, while two-dimensional arrays and other graphs with similar properties cannot. We have also discussed some of the practical implications of these theoretical results.

This study has concentrated on the interaction of clock skew models with the communication structure of arrays with bounded communication delay; future work should also examine cases where asymptotically growing delays occur. One interesting such case is that where the communication graph COMM, neglecting edge directions, is a binary tree. It has been shown that a planar layout of a tree with N nodes of unit area must have an edge of length $\Omega(\sqrt{N}/\log N)$ [9]. Under the summation model of Section V then, if we make the additional assumption that communication delays grow with path length in the same way as clocking delays, a tree may be clocked at no loss in asymptotic performance simply by distributing clock events along the data paths.

Furthermore, if COMM is acyclic, as in the tree machine algorithms described in a paper by Bentley and Kung [2], and the ratio between lengths (in the layout) of any two edges at the same level in the graph is bounded, pipeline registers can be added on the long edges, with the same number of registers on all of the edges in a given level. This makes all wires have bounded length, thus causing the time needed for a cell to operate and pass on its results to be independent of the size of the tree. Adding the registers increases the layout area by at most a constant factor since they, in effect, just make wires thicker. For example, an H-tree layout has this property and allows a tree machine of N nodes to be laid out in area $O(N)$ with delay through the tree of $O(\sqrt{N})$ and constant pipeline interval.

REFERENCES

[1] R. Aleliunas and A. L. Rosenberg, "On embedding rectangular grids in square grids," *IEEE Trans. Comput.*, vol. C-31, pp. 907–913, Sept. 1982.

[2] J. L. Bentley and H. T. Kung, "A tree machine for searching problems," in *Proc. 1979 IEEE Int. Conf. Parallel Processing*, Aug. 1979, pp. 257–266.

[3] M. A. Franklin and D. F. Wann, "Asynchronous and clocked control structures for VLSI based interconnection networks," in *Proc. Ninth Annu. Symp. Comput. Architecture*, Apr. 1982, pp. 50–59.

[4] H. T. Kung, "Why systolic architectures?" *IEEE Comput. Mag.*, vol. 15, pp. 37–46, Jan. 1982.

[5] S. Y. Kung and R. J. Gal-Ezer, "Synchronous vs. asynchronous computation in VLSI array processors," in *Proc. SPIE Symp., Vol. 341, Real-Time Signal Processing V*, Soc. Photo-Opt. Instrument. Eng., May 1982.

[6] R. J. Lipton, S. C. Eisenstat, and R. A. DeMillo, "Space and time hierarchies for classes of control structures and data structures," *J. ACM*, vol. 23, no. 4, pp. 720–732, Oct. 1976.

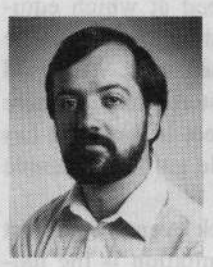
[7] C. A. Mead and L. A. Conway, *Introduction to VLSI Systems*. Reading, MA: Addison-Wesley, 1980.

[8] C. A. Mead and M. Rem, "Cost and performance of VLSI computing structures," *IEEE J. Solid-State Circuits*, vol. SC-14, pp. 455–462, Apr. 1979.

[9] M. S. Paterson, W. L. Ruzzo, and L. Snyder, "Bounds on minimax edge length for complete binary trees," in *Proc. Thirteenth Annu. ACM Symp. Theory Comput.*, May 1981, ACM SIGACT, pp. 293–299.

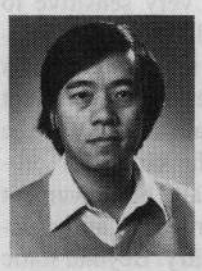
[10] C. L. Seitz, "System timing," in *Introduction to VLSI Systems*, C. A. Mead and L. A. Conway. Reading, MA: Addison Wesley, 1980, ch. 7.

[11] C. D. Thompson, "A complexity theory for VLSI," Ph.D. dissertation, Dep. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, Aug. 1980.



Allan L. Fisher (S'82–M'83) received the A.B. degree in chemistry from Princeton University, Princeton, NJ, and the Ph.D. degree in computer science from Carnegie-Mellon University (CMU), Pittsburgh, PA.

Since the fall of 1984 he has been an Assistant Professor of Computer Science at CMU. His research interests are in computer architecture and VLSI systems.



H. T. Kung received the Ph.D. degree from Carnegie-Mellon University (CMU), Pittsburgh, PA, in 1974.

He joined the faculty of CMU in 1974 and is now a Professor of Computer Science. He leads a research team at CMU in the design and implementation of high-performance computer systems. In 1981 he was a full-time architecture consultant to ESL, Inc., a subsidiary of TRW. His current research interests are in systolic array architectures and their applications.

Dr. Kung is a Guggenheim Fellow (1983–1984) and has served on editorial boards of several journals and program committees of numerous conferences in VLSI and computer science.