SYNCHRONIZING LARGE SYSTOLIC ARRAYS

by

A.L. Fish=r [9]* H.T. Kung

December, 1932

DRC-15-19-8?

Synchronizing Large Systolic Arrays

Allan L. Fisher and H. T. Kung

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

April 1982

To appear in the *Proceedings of the SPIE, Vol. 341, Real-Time Signal Proceedings V*, Arlington, Virginia, May 1982.

The research 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, and in part by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under Contract F33615-81-K-1539. A. L. Fisher was supported in part by a National Science Foundation Graduate Fellowship.

ABSTRACT

Parallel computing structures consist of many processors operating simultaneously. If a concurrent structure is regular, as in the case of a systolic array, it may be convenient to think of all processors as operating in lock step. This *synchronized view,* for example, often makes the definition of the structure and its correctness relatively easy to follow. However, large, totally synchronized systems controlled by central clocks are difficult to implement because of the inevitable problem of clock skews and delays. 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 systolic arrays are proposed. In general, this paper represents a first step towards a systematic study of synchronization problems for large systolic arrays.

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

Key Words and Phrases

Synchronization, VLSI, large systolic arrays, dock skews, concurrent systems

1. Introduction

Parallel computing structures consist of many processors, or *cells* in the terminology of this paper, operating simultaneously. If a concurrent structure is regular, as in the case of a systolic array [3], it may be convenient to think of all cells as operating in lock Step. This *synchronized view,* for example, often makes the definition of the structure and its correctness relatively easy to follow—indeed, synchronized, moving transparencies arc typically used in talks to illustrate systolic arrays. Perhaps the simplest means of synchronizing an ensemble of cells is the use of broadcast docks. A clocked system in general consists of a collection of functional units whose communication is synchronized by external dock signals. A variety of clocking schemes are possible; the essential point is that by referring to die global time standard represented by the dock, 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 dock signals by different paths, they may not receive docking events at die same time, potentially causing synchronization failure. These synchronization errors due to dock skews can be avoided by lowering dock rates and/or adding delay to circuits, thereby slowing the computation. The usual docking schemes are also limited in performance by the time needed to drive dock 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 docks are difficult to implement because of die inevitable problem of clock skews and delays.

An alternative approach is self-timing [7J, in which cells synchronize their communication locally with some variety of "handshaking" protocols. It is easy to convince oneself that any synchronized parallel system where processors operate in lock step can be converted into a corresponding asynchronous system of this type that computes die same output—the asynchronous system is obtained by simply letting each processor start computing as soon as its inputs become available from other processors. The self-timed, asynchronous sdieme 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 die size of the entire processor array. A serious disadvantage of fully self-timed systems is that they are difficult and expensive to design and tesL

An advantage that self-timed systems often enjoy, in addition to die absence of dock 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 aDows a machine to take advantage of variations in component speed or data-dependent conditions allowing fcter computation. This advantage will seldom exist in systolic systems, however, for two reasons:

- Usually, eadi cell in a systolic array performs the same kind of computation as every other cell; thus there is little opportunity for speed variation.
- 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 systolic arrays. The techniques described below use clock-based approaches, sometimes with a self-timed assist, to allow convenient synchronization of large arrays.

2. Basic Assumptions

The basic model that we will use for considering synchronization of systolic arrays is as follows:

(A1) Inter-cell data communications in an *ideally synchronized* systolic array, in which all processors operate in lock step, are defined by a directed graph COMM, which is laid out in the plane. Each nixie of COMM, also called a *cell,* represents a cell of the systolic 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 die target cell in every cycle of the system. Any two cells connecting 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 die basis for docked implementations of ideally synchronized arrays.

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

(A5) A docked system may be driven with dock period $5 + A + r$, where 8 is the maximum dock skew between any two communicating cells, A is the maximum time for a cell's outputs to be computed and propagated, and T is the time to distribute a docking event on CLK.

This assumption can be justified by appeal to a more detailed model which deals with the periods of time in which cells hold their output edges invariant or are sensitive to the values on their input edges. The constraints between dock events, which are enforced in implementation by the pattern of die dock signals **and** circuit delays, may be adjusted so that any communicating pair is property synchronized with a dock period $4 + A + T$. Induction on the size of an array then shows that the docked system correctly implements the ideally synchronized array.

Note that if we adopt the usual convention that die dock tree is brought to an equipotential state before a **new** dock event is transmitted, eliminating dock skew can lead only to a constant factor increase in performance, since it must always be true that $5 \leq T$. In particular, speed of light considerations impose the following condition:

(A6) The time r required to distribute a docking event on a dock tree CLK in a particular layout is bounded bdow by $a\text{-}P_f$ where $a>0$ is a constant and P is the (physical) length of a longest root-to-leaf path in CLK.

Thus, since die 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[9] 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 die case where an array grows too big for its dock 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[1]. The electrical problems of passing a clean signal 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 r 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, r is a constant independent of the size of the array.

To ensure that successive clock events remain correctly spaced along die dock path, we make die 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 models based on die above assumptions, and Sections 4 and 5 explore die problem of docking under these models. Section 6 considers die case where assumption (A8) does not hold, and hence condition (A6) holds rather than condition (AT).

3. Two Models of Clock Skew

Given a basic model consisting of conditions (A1) through (A5), plus (A7) and (A8), die following sections consider die implications of two models of dock skew. First, in Section 4 we consider die case where dock skew between two cells depends on die difference in their physical distance from the root of die dock tree. This *difference model* corresponds reasonably well with the practical situation in high speed systems made of discrete components, where dock trees are often wired so that delay from the root is the same for all cells. More formally, we assume the following:

(A9) The dock 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 die positive difference between tbe (physical) lengths of the paths on CLK that connect die two nodes to die root

---

[1] TJie attbon *wot* told tint this ~pipdmed" fonn of dodrii^ was actually implemented in some high-speed CDC machines.

This assumption is illustrated in Figure 3-1. The two circles connected by the dashed line have clock skew between them which is no more than a constant times the length of the crosshatched segment This segment represents the difference between the cells' distances to their nearest common ancestor in the clock tree.

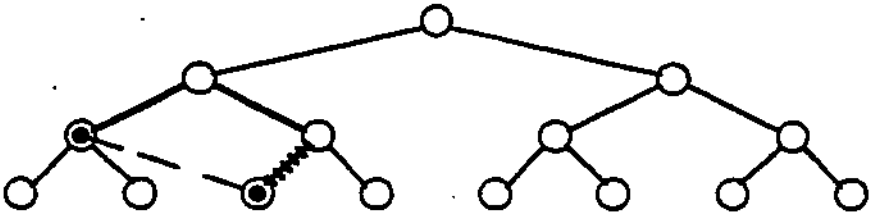


Figure 3-1: Skew in the difference modeL

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 die sum of their lengths. Especially since it is not possible to tune the dock network of a system on a single chip, Section 5 considers a model in which the skew between two nodes depends on the distance between them along die dock tree. Formally, die *summation model* (so called because die distance between two nodes is the sum of their distances irom their nearest common ancestor, while die difference measure used above is the difference between those distances) uses die following upper assumptions:

(A10) The dock 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 dock skew between two nodes of CLK, with respect to a given layout, is bounded below by fis where fi >0 is some constant and sis die (physical) length of the path on CLK that connects die two nodes.

Figure 3-2 illustrates these assumptions; here both die upper and lower bounds on the skew between die two cryfimniiniraring cells depend on the entire length of the path between them, which is die sum of their distances to their nearest common ancestor in die tree.

The two models of dock skew introduced above can be formally derived as follows, for the case when both functions/and g are linear. Let h_x and A* with $h\&h^*$ be the distances of any two cells to their nearest Common ancestor in the Clock tree. Lft $m+?$ and m— f "» :fi» immimiiin and minimum rimg, respectively, tn transmit a dock signal across a wire of unit length, where e corresponds to the variations in electrical characteristics along dock lines. Then the clock skew between the two ceils can be as large as

$$\text{dock skew} = \wedge(m+e\,hA/m-e) = (A_1-A_2)m+(A_1+hje.$$

Noticing that </= ^ - h^* $s= h_x+h^*$ and s£ d>_0, we have

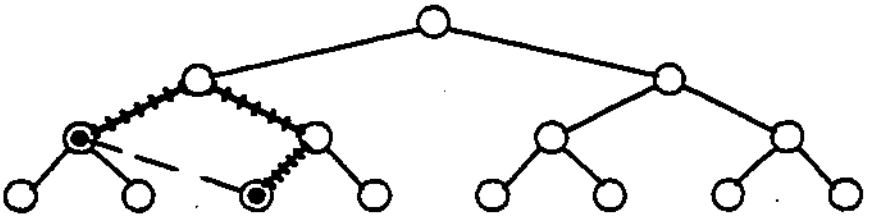


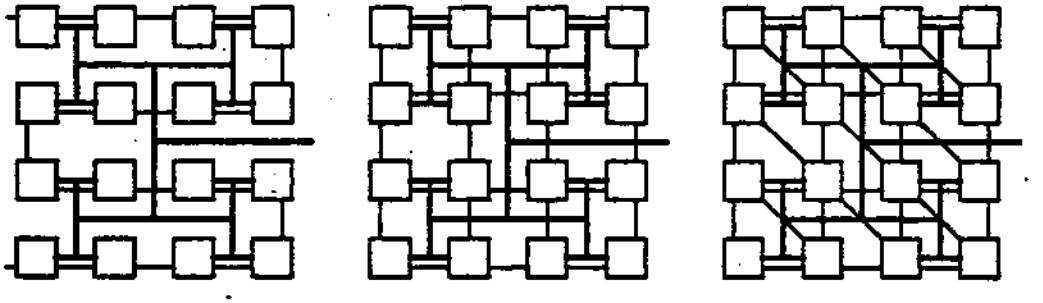
Figure >2: Skew in the summation modeL

(m+ e>5 ≥ dock skew = *ntd*+ e-5 £ «•*

We see that die upper and lower bounds correspond directly to assumptions (A10) and (A11) used in the summation model, whereas the difference model corresponds to the case when terms involving e can be **ignored.**

4. Clocking under the Difference Model

Assuming die basic modd defined above along with condition (A9), which states that the skew between two cells is bounded by a function of the difference between their distances from the root, it is apparent that no dock skew will accur if we assure that all nodes in COMM are equidistant (with respect to die dock layout) from die root of CLJC This can be achieved for any layout for COMM of bounded aspect ratio, without increasing die area of die layout by more than a small constant factor, by distributing the dock through an H-tree [5J. This scheme is illustrated for linear, square, and hexagonal arrays in Figure 4-1, in which heavy lines represent dock edges and thin lines represent communication edges.



(a) **(b)** **(c)**

Figure 4-1: H-tree layouts for docking (a) linear arrays, (b) square arrays, and (c) hexagonal arrays.

More precisely, we have die 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 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 systolic array with computation and communication delay A bounded by a constant can be simulated by a corresponding docked system operating with a clock period independent of the size of the array, with no more than a constant factor increase in layout area.

5. 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 dock 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 modd is weaker than the difference model, any clocking scheme working under the summation modd must also work under the difference modeL The reverse of the statement is not true, however. For example, the docking scheme illustrated in Figure 4-l(a) for linear arrays may not work under die summation model, since two communicating cells (such as the two middle cells on the left 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 docking scheme for linear arrays that works even under the summation model for dock skew; in addition, we show that it is impossible, under this model, to dock a two-dimensional array in time independent of its size. In this sense, linear arrays are especially, suitable for docked implementation.

5.1. Clocking one-dimensional systolic arrays

Given any ideally synchronized one-dimensional systolic array (Figure 5-1 (a)), we propose a corresponding docked array (Figure 5-1 (b)) obtained by running a dock wire along die length of die one-dimensional array. By (A10) die maximum dock 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 dock skew, any ideally synchronized one-dimensional systolic array with computation and communication delay A bounded by a constant can be s''TM[1af]>H by a corresponding docked system, as illustrated in Figure 5-1, operating at a dock period independent of the size of the array.

Skew between die host and the ends of the array can be handled similarly by folding the array in the middle (Figure 5-2), and the array can be laid out with any desired aspect ratio by using a comb-shaped layout (Figure 5-3).

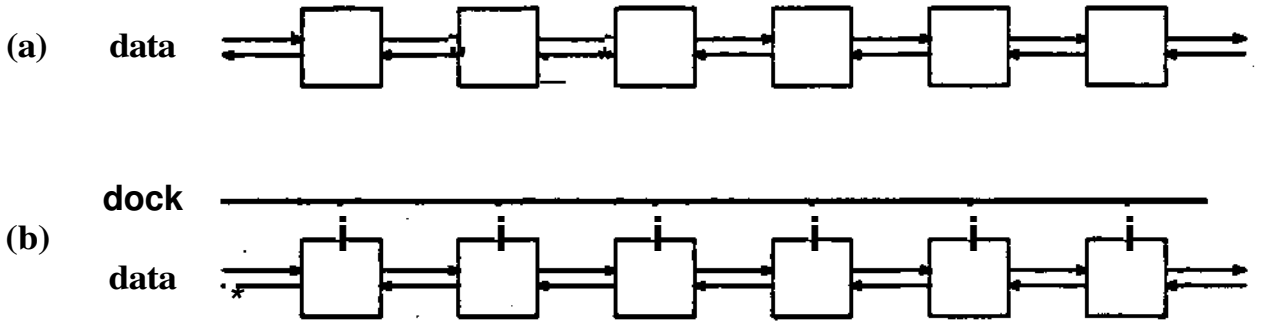


Figure 5-1: (a) Ideally synchronized one-dimensional systolic array and (b) corresponding clocked array.

With the clocking schemes illustrated, we see that the dock period for any one-dimensional systolic array can be made independent of the size of the array. As a result, die clocked array may be extended to contain any number of cells using the same clocked cell design. Therefore, we can say that these clocked schemes are 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 [3].

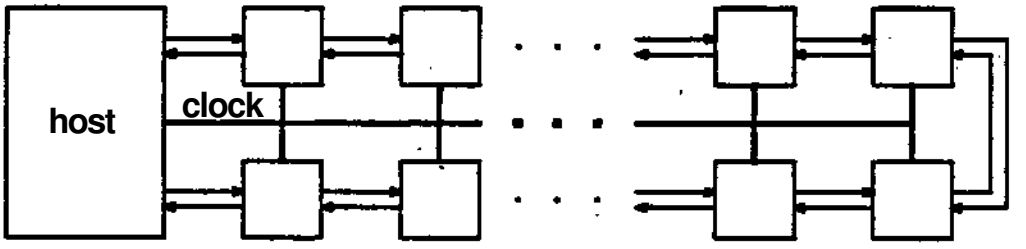


Figure 5-2: Array folded to bound skew with host

5.2. A lower bound result on clock skew

We show here that the result of Theorem 3 for die one-dimensional array cannot be extended to two-dimensional structures. Consider any layout of an *nxn* array and a global dock tree CLK whose nodes include all cells of die array. Let *8* be the *maximum dock skew* between two communicating cells of the array. We want to prove that *8* can not be bounded above by any constant independent of *n.* We use die following well known result {4J:

> **Lemma 4:** To bisect an *nxn* mesh-connected graph at least *on* edges have to be removed, where c > 0 is a constant independent of IL

Bisecting a graph means partitioning die graph into two subgraphs, each containing about half of die nodes of

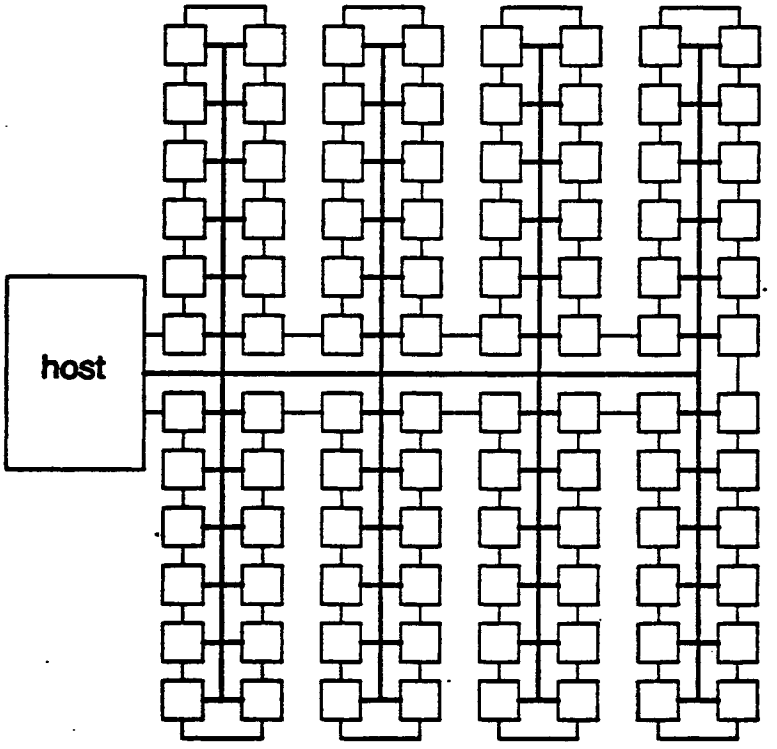


Figure 5-3: Comb layout.

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 trivial but useful lemma without giving a proof.

> **Lemma 5:** For any subset M of nodes of a binary tree, there exists an edge of the tree such that its 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 δ/β, where β is defined in (A11). If there are $\geq (1/10) \cdot n^2$ cells inside the circle, then by (A2)

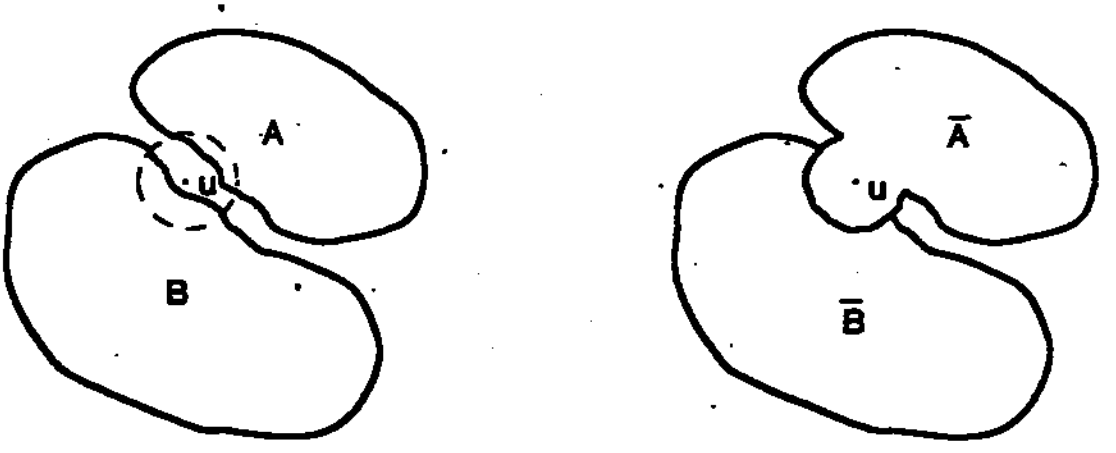
$$\pi(\delta/\beta)^2 \geq n^2/10, \quad \text{or } \delta = \Omega(n),$$

and thus δ cannot be bounded above by any constant independent of n. Suppose now that there are $< (1/10) \cdot n^2$ 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 $\leq \delta/\beta$. 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

is *>fi*8/fi = 8* and the clock skew between any two neighboring cells is assumed to be ≤ 5. These sets are illustrated in Figure 5-4(a). Let $\bar{A}$ be the union of *A* and the set of cells in the circle, and *S* be *B* minus the set of cells in the circle. See Figure 5-4(b). Then $\bar{A}$ and *B* form a partition of the *nxn* array, and each of them has no more than $(1/10>/I^2+(2/3>/I^2 = (Unoyn^1$ cells. From Figure 5-4(b), we see that any edge in the *nxn* array connecting a cell in $\bar{A}$ and a cell in *S* must cross the boundary of the circle. Since the length of the boundary is $2v5//?_t$ by (A3) $\bar{A}$ and $\bar{B}$ are connected by no more than *2*8/ft* edges. By Lemma 4 we have $2\pi\delta/\beta \geq c \cdot n$, or

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

Therefore as *n* increases, *8* grows at least at the rate of *n;* we see that it is impossible to run a global dock for the *nxn* array such that the maximum clock skew *8* between communicating cells will be bounded above by a constant* independent of /L



(a) (b)

Figure 5-4: (a) original partition and (b) new partition of die communication graph.

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 [8J, which is the number of edge cuts needed to bisect the graph. For example, by Lemma 4 the minimum bisection width of an *nxn* mesh-connected graph is *Qjn).* We have the following general result:

Theorem 6: Suppose that die minimum bisection width of an iV-node graph is $Q(W\{N))$ and $W(N) = O(V7T$ XThen

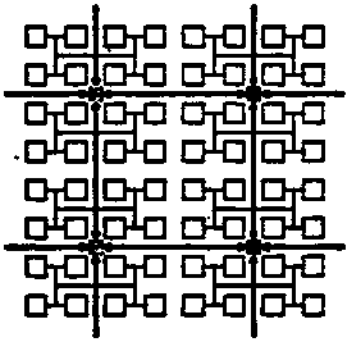
$$\delta = \Omega(W(N)).$$

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

6. Hybrid Synchronization

In the absence of the invariance condition (A8), provisions must be made to ensure that a clock event does not "catch up with" a previous event This requires diat each clock buffer refrain from passing on an event until die processing of die previous event has been acknowledged. In order to implement this constraint, we can essentially replace the buffers of die previous sections with a handshaking network which operates on dock events.

In this approach, we break up the layout into bounded-size segments, and provide each segment with a local clock distribution node. The clock distribution nodes employ a handshaking protocol to pass clock events among themselves. Given assumptions about the maximum delay of a computation node and its wires and the maximum delay for a handshake transaction in the dock distribution network, we can dock the cells in each neighborhood in constant time. As before, we balance the delay within each element with die wire delays between elements. This structure is illustrated in Figure 6-1, in which the heavy lines and black boxes represent die self-timed synchronization network, and die narrow lines represent local dock distribution to the cells near each synchronizing element.



Figyre 6*li Hybrid synchronization scheme.

This provides the 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 computational elements to be designed as if the entire system were globally docked. The hybrid approach has the additional advantage that a single synchronization design can be used for many different structures. This simplifkation of die usual self-timed

scheme is made possible by the fact that we are willing to assume a maximum delay for the computational elements; 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.

7. Concluding Remarks

We have described a series of models in which synchronization schemes can be studied, and have indicated some of the implications of these models. Future work should include refinement of the models and some quantification of when they apply to real systems, as well as further weak on their implications. This paper 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 $Q(V7T / \log N)$ [6\. Under die summation model of Section 5, then, if we make the additional assumption that communication delays are proportional to path length, a tree may be docked at no loss in asymptotic performance simply by distributing dock events along the datapaths.

Furthermore, if COMM is acyclic as in the tree machine algorithms described in a paper by Bentley and Kung [2J, and die 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 die edges in a given level. This makes all wires have bounded length, thus causing die 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 die layout area by at most a constant factor, since they in effect just make wires thicker. For example, an H-tree layout has diis property, and allows a tree machine of//nodes to be laid oat in area $O(N)$ with delay through the tree of 0(VW") and constant pipeline interval.

Acknowledgments

We thank Doug Jensen and Hank Walker of CMU for helpful discussions.

References

[1] Aleliunas, R. and Rosenberg, AX.
On Embedding Rectangular Grids in Square Grids.
Technical Report RC 8404 (#36095), IBM Thomas J. Watson Research Center, Yorktown Heights, New York, June, 1980.

[2] Bcntley, JX. and Kung, H.T.
A Tree Machine for Searching Problems.
In *Proceedings of 1979 International Conference on Parallel Processing,* pages 257-266. IEEE, August, 1979.
Also available as a CMU Computer Science Department topical report, August 1979.

[3] Kung, H.T.
Why Systolic Architectures?
Computer Magazine 15(l):37-46, January, 1982.

[4] Lipton, RJ., Eisenstat, S.C and DeMillo, R.A.
Space and Time Hierarchies for Classes of Control Structures and Data Structures.
Journal of the ACM 23(4):720-732, October, 1976.

[5] Mead, GA. and Rem, M.
Cost and Performance of VLSI Computing Structures.
IEEE Journal of Solid State Circuits SC-14(2):455-462, April, 1979.

[6] Paterson, M£., Ruzzo, WJ Land Snyder. L.
Bounds on Minimax Edge Length for Complete Binary Trees.
In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing,* pages 293*299. ACM SIGACT, May, 198L

[7] Seitz, CL.
Self-Timed VLSI Systems.
In *Proceedings of Conference on Very Large Scale Integration: Architecture, Design, Fabrication** pages 345-355. California Institute of Technology, January, 1979.

[8] Thompson, CD.
A Complexity Theory for VLSI.
PhD diesis, Carnegie-Mellon University, Computer Science Department, 1980.