Distributed Synchronous Clocking

Gill A. Pratt and John Nguyen

Abstract— **It has historically been difficult to distribute a well-aligned hardware clock throughout the physical extent of a synchronous processor. Traditionally, this task has been accomplished by distributing the output of a central oscillator over a tree-like network, with repeaters at necessary intervals. While straightforward in concept, this method suffers from poor reliability, poor scalability and high skew.**

In this paper, we present an alternative approach—*Distributed Synchronous Clocking*—that maintains the simplicity of synchronous operation without suffering the drawbacks of centralized clocking. A network of independent oscillators takes the place of the centralized clock source, providing separate clock signals to the physically distant parts of a computing system. A distributed error correction algorithm effects global phase alignment by utilizing local comparisons of neighboring oscillator phase.

In contrast to centralized clock distribution, distributed clocking has the inherent potential for complete scalability and graceful degradation. However, because oscillator phase is a modular quantity, a naive implementation of distributed synchronous clocking can suffer from *mode-lock*—the trapping of local oscillator phase in undesirable stable equilibria where global phase is not aligned [26]. We present a simple method for eliminating this problem in k-ary Cartesian meshes and give a proof of its correctness for two-dimensional networks. An electronic implementation is also presented and several engineering issues relating to error tolerance are discussed.

Index Terms— **Computer hardware clocking, clock synchronization, distributed clocking, oscillator synchronization, phase-locked loops, mode-locking.**

I. Introduction

GLOBALLY synchronous clocking is the most widely used method of timing digital systems. Yet, while globally synchronous clocking is simple, inexpensive, and yields high performance, it also places great demands on the clock distribution network. In large, high-performance digital systems (such as multiprocessors) these requirements often become burdensome to the overall design task, and in some cases may even surpass the performance achievable by economically feasible technology. Why this is so can be understood by considering the following attributes of clock signals in globally synchronous systems:

A. Clock Noise

In most digital systems, clock signals are more sensitive to noise than data signals. This is because data signals only convey information at specific sampling times whereas clock signals transmit information at all times. Thus, while consumers of data can ignore input noise during times of likely uncertainty (i.e. immediately after state changes), receivers of clock information, with no easy way to predict the time of next transition, are always sensitive to input noise. Among the common causes of erroneous clocking are ringing from mis-terminated transmission lines, crosstalk induced by nearby wires, and cross-coupling caused by the di/dt of power leads interacting with the nonzero inductance of the power distribution network. In large parallel computers with high clock fan-out, the expense of properly driving, terminating, and shielding all clock lines to prevent most of these problems can become prohibitive. If distributed synchronous clocking is used instead, clocks are generated locally, obviating the need for the fan-out tree.

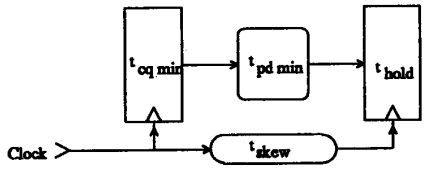


Fig. 1. Skew constraints of single-phase edge-triggered clocking.

B. Clock Skew

Skew must be more tightly controlled for clock signals than for data. Data skew simply adds to propagation time, and can be allowed for by lowering system cycle frequencies. Clock skew, however, affects the simultaneity of system-wide data sampling. Because synchronous systems make quick changes from "old" state to "new" state, excessive clock skew can cause erroneous operation, regardless of cycle rate. This problem is particularly severe in systems that utilize single-phase, edge-triggered clocking, as shown in Fig. 1.

For these systems to operate correctly, the timing criteria:

$$t_{\text{skew}} + t_{\text{hold}} < t_{cq\,\min} + t_{pd\,\min}$$

must be obeyed. As faster devices are used, $t_{cq\,\min} + t_{pd\,\min}$ becomes smaller, lowering (for a given t_{hold}) the allowable t_{skew}.

Despite the need to do so, it is hard to distribute a low-skew hardware clock throughout the physical extent of a large synchronous processor. Computers have historically become smaller as they have become faster, and low clock skew has become somewhat easier to achieve in modern, physically smaller systems than in older, larger designs. But unfortunately, massively parallel architectures disobey this traditionally inverse size versus speed relationship. In parallel

Manuscript received December 26, 1991, revised May 17, 1994. The work was supported by the U.S. Navy under Grant N00014-89-J-1988.

The authors are with the Massachusetts Institute of Technology, Laboratory for Computer Science, Cambridge, MA 02139 USA.

IEEE Log Number 9408138.

machines, technological improvements in hardware density often result in more complex machines of roughly the same physical size. This trend has generated a growing disparity between required and readily achievable clock quality and skew.

In answer to this problem, skew-tolerant design methods can be used to reduce sensitivity to clock skew. For example, dual-port asynchronous FIFO's can be placed between elements with uncertain clock timing, providing very good tolerance to clock skew. Unfortunately, such designs must pay a data latency penalty in return for their clock skew tolerance, and often an added hardware complexity penalty that scales with the number of data pins. Distributed synchronous clocking, in contrast, provides locally low-skew clocks that allow the lowest possible data transfer latency to be achieved with simple synchronous register interfaces.

C. Scalability

The scale of a traditional synchronous clock distribution system is fixed at the time of its construction. This is unimportant for serial machines because once designed, serial computers do not greatly change size. Parallel machines, on the other hand, offer the tantalizing potential of post-manufacture scalability. If a locally connective multi-processor is found inadequate for a given task, it should be possible to increase the available computing power simply by attaching additional processors [25].

Providing for such post-manufacture scalability demands careful design at all levels, including power supply, cooling system, communications network, and system software. A computer is only as scalable as its least scalable component, and because traditional clock distribution systems are not scalable, an alternative clocking method must be used to provide for post-manufacture scalability. Distributed synchronous clocking scales as easily as the communications network—one simply adds additional nodes to the periphery of an existing structure.

D. Reliability

Another motivation for decentralized clocking is improved reliability. The possibility for this improvement exists because locally connective parallel machines have an inherent potential for graceful degradation in the face of localized hardware faults.

As with scalability, insuring graceful degradation also requires careful design at all levels, from low level hardware to high level software. Global dependencies at any level subvert the reliability enhancements provided by redundancy in others. As an example, a single node shorting any point of a traditional clock distribution tree ruins the signal received by many other nodes. The insertion of "firewall" repeaters can somewhat alleviate this problem, but only at the expense of added skew. Redundant clock networks address this problem, but at significantly added expense and complexity, as a change of local clock source must be accomplished without significantly altering skew. Thus, traditional clock distribution systems are a significant weakness in multi-processors that must survive local short circuits.

E. Repair

Clock faults are not a significant problem in present computer designs, but they are likely to cause more trouble in the future. Today's processors are typically constructed from removable cards, two-dimensional backplanes, and connecting ribbon cable. This has made the replacement of defective components a simple matter. In the near future, however, parallel machines will likely be built as dense three-dimensional structures, so that internal defects will be difficult, if not impossible, to repair. Future systems will also have larger numbers of simultaneously active components, making the probability of significant hardware faults higher than in current designs. Thus, as is true in biological systems, continued operation despite the permanent existence of local defects will be required. Long term short circuits and disconnections in the clock network must be expected. Distributed synchronous clocking is capable of providing continued operation despite the existence of permanent faults in the network. Even the simple phase error control system to be described in this paper (which was not designed with fault-tolerance in mind) is tolerant to the majority of such faults.

F. The Alternative of Asynchronous Control

An elegant response to all of the above difficulties is asynchronous control [21]. Networks of asynchronous circuits, no matter how large, require neither a central clock source nor its cumbersome system of distribution. Best of all, without prior planning by the designer, asynchronous circuits can operate automatically at maximum speed.

This guarantee of global efficiency with only local control has an almost irresistible aesthetic appeal. Unfortunately, asynchronous systems have several serious drawbacks. Asynchronous circuits have higher physical and behavioral complexity than synchronous counterparts. They are harder to design, costlier to implement, and more difficult to verify and debug. Synchronous debugging techniques such as state snapshots cannot be used for asynchronous designs, and no equally powerful alternative is available.

Noise control is another serious problem. In synchronous systems, all signal transients, whether locally generated or induced by cross-talk, settle down after a fixed time following the global clock event. No such "quiet time" exists in asynchronous systems, and unpredictable power supply noise and cross-talk may become an extremely elusive source of error [4].

G. Motivations—Summary

As has been discussed above, the traditional distribution tree is an unscalable and fragile mechanism for clocking globally synchronous computer systems. Some esoteric mechanisms (such as optical distribution [14]) have been proposed for distributing synchronous clocks with low skew, but these solutions do not address the issues of scalability and robustness. Wave-propagation methods (such as [7]) have also been proposed as a means of bounding local skew, but these systems suffer from dependency on a single clock source (and are thus inherently unreliable) and can only lower skew to the

propagation time of several gates. Finally, skew tolerant design methodologies can lower clock distribution requirements to constant frequency, rather than zero phase, operation, but these methods are neither scalable nor inherently robust. Furthermore, they exact a latency performance penalty and often require complex hardware on every data input pin.

Locally connective parallel processors, unlike serial machines, have an inherent potential for general scalability and outstanding robustness, even in the face of permanent defects. To make use of these potentials, we must either abandon system-wide synchronicity (which has many drawbacks), or generate a system-wide synchronous clock using a mechanism that is as distributed, scalable, and fault-tolerant as the computing network itself. This latter approach is the main philosophy behind distributed synchronous clocking.

II. The Basic Idea

Our goal is to precisely synchronize a group otherwise independent hardware oscillators in the fashion of summer fire-flies synchronizing their flashing while gathered on a tree. Unlike fire-flies (and many previously studied synchronization systems [1], [3], [6], [13], [15]–[17], [19], [20], [22]–[24]), the oscillators of our system only utilize phase comparisons against physically adjacent neighbors.

III. Topology

In this paper, we consider synchronization in K-ary N-dimensional Cartesian mesh networks. This simple topology is both wire density efficient and also succumbs to a simple mechanism of deadlock avoidance, [5] making it a likely candidate for future parallel machines. Distributed synchronous clocking is applicable to other topologies as well, but demonstrating such requires modified analysis. If a particular topology is found to be unsuitable, a K-ary N-dimensional Cartesian clock mesh can be physically superimposed onto any other locally connective N-dimensional communications topology, producing physically local clocks of highly precise alignment.

IV. Preliminary Definitions

Formally, we treat the network as a graph G with vertices V representing computation nodes, and edges E representing communications links between neighboring nodes. For any two vertices $v_a, v_b \in V$, there is an edge between them $((v_a, b_b) \in E)$ if and only if v_a and v_b represent nodes that are neighbors. The network graph is symmetric $((v_a, v_b) \in E \leftrightarrow (v_b, v_a) \in E)$ since every node is a neighbor of its neighbor.

Definition 1: The **phase** $\phi(v)$ of a node v is a value in the range $[-\pi, +\pi)$ describing the instantaneous phase of its clock signal with regard to a global reference.

Definition 2: The **phase error** $\delta(v_a, v_b)$ of two nodes v_a and v_b is defined as

$$\delta(v_a, v_b) = ((\pi + \phi(v_a) - \phi(v_b)) \bmod 2\pi) - \pi$$

and also has a range of $[-\pi, +\pi)$.

Definition 3: A nodes v_a is *synchronized* with all of its neighbors when the phase error on each of its links is zero:

$$\forall v_b (v_a, v_b) \in E \delta(v_a, v_b) = 0.$$

V. A Simple (but Incorrect) Scheme: Phase Averaging

The first scheme to be described is a simple, but unfortunately incorrect, algorithm for synchronizing a network. We present it because its incorrectness will motivate a small but nonobvious change that results in a correct methodology.

In this simple scheme, the phase of each local clock is driven towards the average phase of its topological neighbors.[1] When all nodes have identical phase, all driving forces are zero and a stationary point is reached. Smaller errors bring about proportionately smaller driving forces, so that the zero phase error condition is approached asymptotically.

To simplify matters,[2] we temporarily assume that all nodes have available a clock reference signal of identical frequency. This allows each node to utilize a Voltage Controlled Delay Line (VCDL) to directly control the phase of its output signal. While this makes analysis straightforward, it must be remembered that the VCDL's need for a common frequency reference flies in the face of our main goal—the elimination of the clock distribution tree. To remove this global dependency, we will later substitute a Voltage Controlled Oscillator (VCO) for the VCDL. This is not done now because the control input of a VCO controls output frequency, the time-derivative of phase, making the operation of a VCO phase correction system more difficult to understand.

Each node of the simple phase control system has the form shown in Fig. 2.

A. How the Averaging Scheme Should Work

As seen above, the phase of each node is compared against the phase of its neighbors, and the total local error slews an integrator controlling each local node's phase. The slew rate is proportional to this error, so that each node's phase is driven asymptotically towards the average phase of its neighbors. Because this average is also changing, the dynamics of the entire system seem complex. Fortunately, this is deceiving, and significant insight can be gained by considering the electrical analog (*analog1*), shown in Fig. 3.

Each node in this analog receives current from the resistors attaching it to neighboring nodes. These currents are proportional to the voltage differences between the local and neighboring nodes, scaled by the conductance of the attaching resistors. The sum of these currents is integrated by each node capacitor so that the capacitor's voltage moves asymptotically towards the the weighted average of the neighboring node voltages. If the connecting resistors are of equal value, this weighted average is the true average, and the above circuit implements the localized phase averaging scheme just described, with local clock output phase being modeled in the analog as capacitor voltage.

[1] Because phase is cyclic, this average is not uniquely defined, but this issue is not important to the present discussion and will be resolved later.

[2] For example, to reduce the control system to first order.

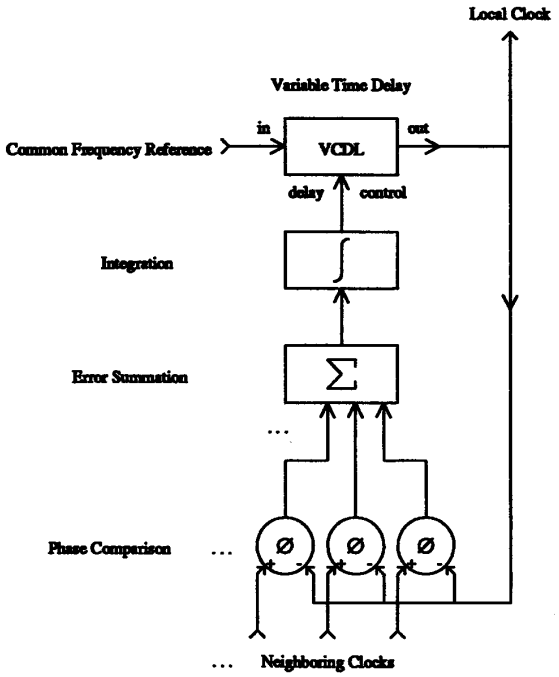


Fig. 2. First-order phase control.

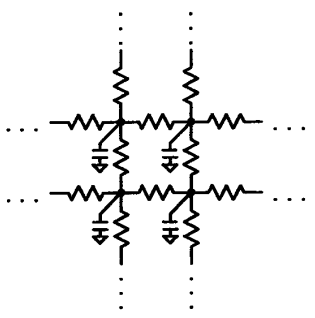


Fig. 3. First-order electrical analog (**analog1**).

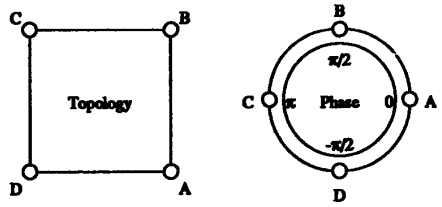


Fig. 4. Mode-locking in a 2 × 2 mesh.

Because this analog contains positive resistors and no power sources, it is unconditionally stable. All nodes will eventually converge to a common voltage (even if the resistors have different values). For our purposes, this final voltage (which can be calculated from conservation of charge) is unimportant; it is only the final uniformity that matters.

B. What's Wrong with Averaging

The simple phase-correction system seems wonderful, and indeed, with minor variation, this method is used by many computer networks to maintain coherent notions of absolute time.

But unlike absolute time, phase is a cyclic, or modular, variable, and the modularity of phase causes trouble. This trouble, known as *mode-locking*, can be seen in the example of a 2 × 2 mesh, shown in Fig. 4, whose 4 oscillators have become "stuck" in a cycle of relative phase.

The left-hand diagram depicts the physical topology and connectivity of the network. In the right-hand diagram, the phase of each node is given by its angular position on the perimeter of the circle and arcs are drawn between nodes to re-iterate the network's connectivity.

In this particular example, each node's oscillator experiences a total influence equal to the average error of its own phase compared to its neighbors. As can be seen, because phase is cyclic, in the configuration above the effect of one neighbor exactly cancels the effect of the other neighbor for every node. The total influence on every node is zero, and despite the fact that their global phases are unequal, there is no reason for the node phases to change. The system is at a stationary point.

This configuration is not only stationary, it is also stable. This means that if we perturb any one (or any set) of the node phases by a minute amount, the dynamics act to restore the system to its previous configuration, not to drive it away. This is shown formally in Section IX-A.[3]

C. Hitting the Stops

Another problem exists due to the cyclic nature of phase, and has to do with the finite voltage range which may be applied to a real VCDL (and equivalently, the finite value which will be output by a real world integrator). If, per chance, our phase-correction system has reached a configuration where a node is required to increase its VCDL time delay when the delay control signal is already at its maximum value, the VCDL delay will not, in fact, be able to increase. As a result, the overall system will no longer operate as expected. If the range of the VCDL is sufficient, the VCDL input should ideally *wrap-around* to a low value that gives the desired output phase. Analog circuitry to detect this condition and carry out the wrap-around is quite complex. The best solution is probably to use a phase lock loop to adjust a multi-tap VCDL to an integral number of clock cycles, and then use a digital multiplexor (controlled by a modulo 2^n counter) to select the desired delay [9]. The modulo 2^n counter implements a digital approximation of the integrator, with the desired wrap-around properties, as shown in Fig. 5.

Instead of pursuing this notion further, we note that the trouble really stems from the use of a monotonic but finite signal to control the value of a modular variable. Fortunately,

[3]Those readers blessed with mechanical intuition can view the right-hand diagram as a pulley around which a necklace of 4 beads connected by 4 rubber bands has been stretched. It is clear from this view that the equilibrium is stable.

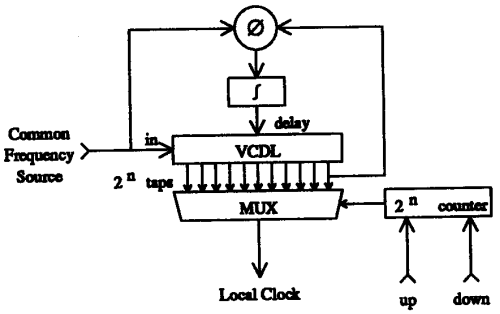


Fig. 5. Digital VCDL with wrap-around.

a VCO that is fed a slightly high control voltage will output a *frequency* slightly higher than other nodes, and will naturally wrap its phase in relation to its neighbors. Thus, we have reason to hope that the adoption of a VCO to replace the VCDL and system-wide clock input distribution tree (something we plan to do anyway) will resolve the wrap-around problem. Indeed, this will prove to be the case.

VI. Eliminating Mode-Locking—First Method

Because the problem of mode-locking is caused by the cyclic nature of phase interacting with a physical cycle in the connection mesh, one solution is to break the physical cycle until all phases are sufficiently close to be assured of convergence to zero error.

In a two-dimensional phase correction mesh, this can be implemented by having all nodes only consider lower and left-hand neighbors for a certain period of time. In this configuration, phase information propagates as a wave from the lower left corner of the mesh to the upper right corner, and cycles are impossible. After a while, all node phases are assumed sufficiently close to each other so that every node can connect to its four neighbors without fear of mode-lock.

Unfortunately, this is a global strategy, requiring a global control signal. In an effort to eliminate this global dependence, we can program each node to sense the "tension" it is under (by measuring the maximum magnitude of adjacent phase error, for example), and switch into the two neighbor mode whenever this tension exceeds some threshold.

We have conducted simulation and hardware experiments on small networks using this algorithm, and it seems to work. However, this method is both complex to implement in hardware and, because the clock network topology is constantly changing, hard to prove stable. To be truly robust, we must show it capable of recovering from all possible mode-locks. This seems difficult. If we make each node permanently look at only two of its four neighbors, no cycles are possible and mode-lock cannot occur, but phase error is higher than necessary and faults have effects that propagate far.

We can solve the problem another way. Instead of trying to change the network topology when mode-locking conditions exist, we alter the action of each link's restoring function under large errors, as explained below.

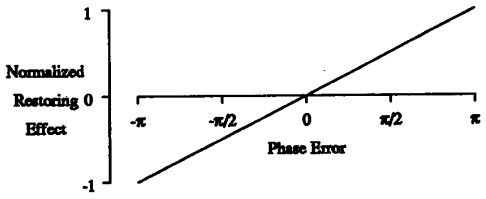


Fig. 6. Linear phase detector function.

VII. Phase Detectors

Without stating so explicitly, we have been assuming that each phase detector has the linear transfer characteristic shown in Fig. 6.

For phase correction to work at all, we need positive restoring effect for positive phase errors and negative restoring effect for negative phase errors. Formally:

Definition 4: The error function err maps a phase difference $\delta(v_a, v_b)$ into $[-1, 1]$ with the following conditions:

$$err([-\pi, 0)) \rightarrow [-1, 0)$$
$$err(0) \rightarrow 0$$
$$err((0, \pi)) \rightarrow (0, 1].$$

In other words, to maintain negative feedback, the error function must preserve the sign of its argument. The linear error function shown above:

$$err(\delta(v_a, v_b)) = \frac{\delta(v_a, v_b)}{\pi}$$

obeys this rule and is typical of "sequential" or "state machine" phase detectors. A clock synchronization system using this type of detector was proposed by A. Nowatzyk [18]. Nowatzyk's error correction network is similar to the one described here, but fails to address mode-locking phenomena. The resultant nonzero delays are constant but unpredictable on power-up, making Nowatzyk's system appropriate for equalizing frequency, but unusable for minimizing skew. Sequential phase detectors may also have an undesirable "dead band" near their zero point which can cause excessive phase jitter [9], [10].

VIII. Energy Functions and Stability

Before we consider other phase detectors, we must decide on a method for establishing stability. A common method for analyzing the stability of stationary points in dynamic systems is to look at measures of stored "energy" for different static configurations.[4] A particular link's stored energy w is equal to the integral of its error function:

$$\forall (v_a, v_b) \in E \quad w(v_a, v_b) = \int_{\delta=0}^{\delta(v_a, v_b)} err(\delta) d\delta.$$

Fig. 4's stationary mode-lock is stable because the sum of the stored energy function for all of its links is at a local minimum.

[4]This energy is exactly equal to the potential energy stored in the mechanical "rubber bands" of the previous footnote's necklace.

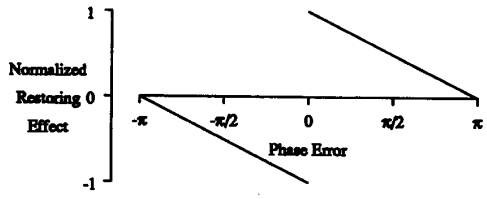


Fig. 7. Harsh negative slope phase detector.

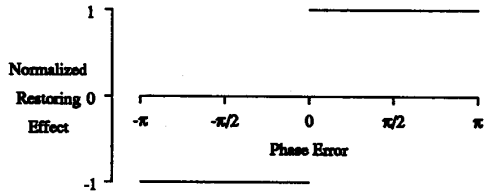


Fig. 8. Flip-flop phase detector.

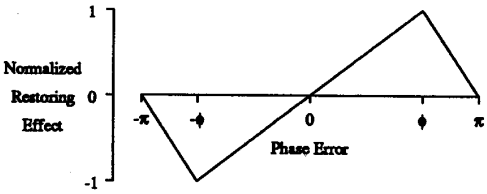


Fig. 9. Nonmonotonic phase detector.

IX. Eliminating Mode-Lock-Second Method

If we change the phase detector's error function slope to be negative instead of positive, the stored energy function would be at a local maximum, indicating an unstable (or metastable) stationary point. In this circumstance, the system would eventually move away from the stationary mode-lock point.

The ability of negative error function slope to destabilize stationary points leads us to consider phase detector transfer functions that satisfy the negative feedback criterion of definition 4 (required for convergence) while still having negative incremental slope. If an error function is found that destabalizes all mode-lock conditions while retaining stability when all phases are equal, the network will eventually resolve all mode-lock conditions on its own.

A first attempt is shown in Fig. 7.

Unfortunately, the large discontinuity of this detector's characteristics cause large influences to "bang around" the phase of the system's clocks near zero error. A similar problem exists in phase locked loops that use flip-flops as phase detectors, [12] which have the transfer function shown in Fig. 8.

Slow filter response is necessary in these systems to remove the strong effects of high gain near zero phase error. A more attractive alternative is the nonmonotonic error function shown in Fig. 9, which has positive slope near zero, but negative slope at phase angles greater than $\pm\phi$:

This type of error function raises the question of what value of ϕ, if any, is required for destabilizing all possible mode-locks.

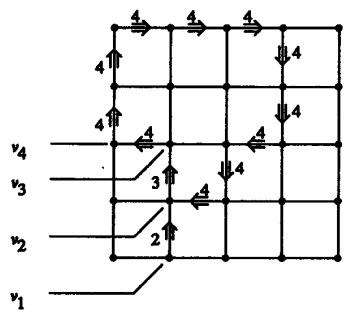


Fig. 10. Illustration of Proof 1 steps 1–4.

It turns out that for 2-dimensional mesh networks, $\phi = \pi/2$ does the job. This is shown in two proofs, given below. In the first, we show that mode-locking can only occur in n-dimensional mesh networks when the magnitude of at least one link's phase error is at least $\pi/2$. In the second proof, we show that Fig. 9's error function (with $\phi = \pi/2$) will destabilize any 2-dimensional network under proof 1's conditions.

Definition 5: Formally, a node v_a is said to be *stationary* when the sum of the error function err applied to all of its links is zero:

$$\sum_{v_b(v_a,v_b)\in E} err(\delta(v_a, v_b)) = 0.$$

An entire network is stationary when all of its nodes are stationary. A network is *mode-locked* when it is stationary and at least one node is unsynchronized.

Proof 1:

1) Consider a mode-locked mesh network. By Definition 5, we can pick one unsynchronized node and label it v_1.
2) By definition 3, an unsynchronized node must have at least one link (v_1, v_x) whose phase error is nonzero. By Definition 4, that link's error function output is also nonzero:

$$\exists v_x(v_1, v_x) \in E \, s.t. \, err(\delta(v_1, v_x)) \neq 0.$$

By Definition 5, since the sum of error function outputs on v_1 must be zero, at least two nonzero error function outputs of opposite polarity must actually be present:

$$\exists v_b(v_1, v_b) \in E \, s.t. \, err(\delta(v_1, v_b)) < 0$$

and

$$\exists v_c(v_1, v_c) \in E \, s.t. \, err(\delta(v_1, v_c)) > 0.$$

We pick one v_c node and label it v_2. Thus, by Definition 4:

$$(v_1, v_2) \in E$$

and

$$\delta(v_1, v_2) > 0.$$

We draw a directed arrow, labeled as "2" in Fig. 10, from v_1 to v_2. The direction of this arrow indicates the polarity of the phase error between v_1 and v_2.

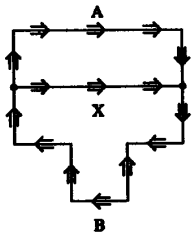


Fig. 11. Illustration of Proof 1 step 6.

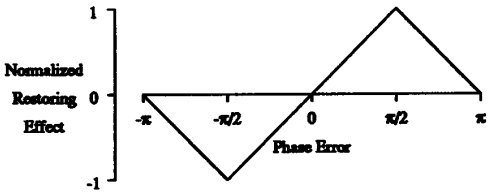


Fig. 12. Triangle wave phase detector.

3) By Definition 2, since $\delta(v_1, v_2) \neq 0$:

$$\delta(v_2, v_1) \neq 0.$$

Thus, by Definition 3, node v_2 is also unsynchronized. Furthermore, by Definition 4, since $err(\delta(v_1, v_2)) > 0$

$$err(\delta(v_2, v_1)) < 0.$$

In other words, if the phase error between v_1 and v_2 causes a "push" on v_1, it also causes a "pull" on v_2. Since v_2 is stationary, we know by Definition 5 that there must exist another link on v_2 to balance the polarity of v_1's error:

$$\exists v_3 (v_2, v_3) \in E \; s.t. \; err(\delta(v_2, v_3)) > 0.$$

Thus, it must be possible to draw another directed arrow, from node v_2 towards another node v_3, indicating continued phase error. This second arrow is labeled "3" in Fig. 10.

4) We can now repeat step 3, drawing another arrow from node v_3 to a new node v_4, from v_4 to yet another new node v_5, and so on. Because the network is finite, the resulting chain of arrows (all labeled as "4" in Fig. 10) cannot find untouched links forever; the chain must eventually "run into" its old path. Note that it may also loop back without exhausting all links:

5) We next consider the sum of phase errors added directionally around any cycle. From the definition of phase error (Definition 2), we see that the sum of all phase errors in a cycle of nodes v_j through v_{k-1} must be a multiple of 2π:

$$\left(\sum_{i=j}^{k-1} \delta(v_i, v_{i+1}) \right) \bmod 2\pi = 0$$

where

$$v_k = v_j.$$

This modular version of Kirchoff's voltage law is easily verified by substituting in Definition 2 for $\delta(v_i, v_{i+1})$ and removing the redundant inner $\bmod 2\pi$.

For the directed loop in step 4, each of the phase errors is positive, so the loop sum must in fact be a **nonzero** multiple of 2π.

6) In the following induction, we discard the attribute of the incremental phase errors around the directed path being positive, but keep invariant that the total of these phase errors is a nonzero multiple of 2π. Because a 2-D Cartesian grid is finite and a nontorus, if any cycle encloses an area larger than one square, it is possible to divide that cycle with a connecting path X it into two cycles, each of smaller area. This is shown schematically in Fig. 11.

7) Let the symbols A, B, and X, indicate the sum of the phase error along their respective directed paths. By step 5 we know that for integers i, j, and k:

$$\begin{aligned} A + B &= i\,2\pi \qquad (i \neq 0) \\ B + X &= j\,2\pi \\ A - X &= k\,2\pi \end{aligned}$$

thus,

$$j + k = i \quad (i \neq 0).$$

Since i is nonzero:

$$j \neq 0 \quad \text{or} \quad k \neq 0.$$

Thus, one of the two subcycles must inherit the parent cycle's property of loop sum being a nonzero multiple of 2π.

8) We recursively apply steps 6 and 7 to the subcycle with nonzero total phase until the area enclosed by the cycle is only a single square, resulting from a path only 4 links long. Since this cycle still has the property that its total phase error is a nonzero multiple of 2π, at least one of its links must have a phase error whose absolute value is at least $\pi/2$. Thus, every mode-locked mesh network has at least one link whose absolute phase error is at least $\pi/2$.

Note that this proof (which is valid in two or more dimensions) assumed nothing about the particular characteristics of the error function except the negative feedback criterion of Definition 4. It was also critical that the grid was a nontorus, as only nontoroidal grids are susceptible to the recursion used in steps 6 and 7.

A. A Practical Phase Detector

Since we now know that any mesh suffering from mode-lock will have at least one link whose phase error magnitude is at least $\pi/2$, there is reason to believe that the error function might only require negative slope for phase errors in excess of $\pi/2$. The corresponding error function, hereafter called the "triangle wave phase detector," is shown in Fig. 12. The proof of its ability to destabilize mode-lock makes use of the Hessian

matrix, a matrix of the energy function's second order partial derivatives. The definiteness of this matrix determines whether a given static configuration of node phases is a local minimum, maximum, or saddle point of the energy function. If the energy function is never at a local minimum except when phase is globally aligned, stable mode-lock is impossible.

Aside from issues of stability, it is reasonable to wonder whether the convergence of **analog1** is compromised by the use of nonlinear incremental resistance. In fact, convergence is still guaranteed because the transfer function stays out of the upper left and lower right quadrants: those regions where an equivalent resistor would deliver, rather than dissipate, absolute power. Thus, if we can show that the triangle wave detector guarantees instability (i.e., meta-stable behavior) at all stationary points except when all phase errors are zero, we know that the overall system will converge to a uniform phase from any initial configuration. A proof of this for 2 dimensional meshes is given below:

Proof 2:

1) The stability of the network's total stored energy function:

$$W = \sum_{(v_a,v_b)\in E} w(v_a, v_b)$$

can be determined by computing the definiteness of the Hessian matrix A composed of the second partial derivatives of W:

$$A = \begin{bmatrix} \frac{\partial^2 W}{\partial\phi_0\partial\phi_0} & \frac{\partial^2 W}{\partial\phi_0\partial\phi_1} & \cdots & \frac{\partial^2 W}{\partial\phi_0\partial\phi_{n-1}} \\ \frac{\partial^2 W}{\partial\phi_1\partial\phi_0} & \frac{\partial^2 W}{\partial\phi_1\partial\phi_1} & \cdots & \frac{\partial^2 W}{\partial\phi_1\partial\phi_{n-1}} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 W}{\partial\phi_{n-1}\partial\phi_0} & \frac{\partial^2 W}{\partial\phi_{n-1}\partial\phi_1} & \cdots & \frac{\partial^2 W}{\partial\phi_{n-1}\partial\phi_{n-1}} \end{bmatrix}.$$

For a given configuration of phases $\Phi = \{\phi_0 \cdots \phi_{n-1}\}$, if the matrix A is negative definite, negative semidefinite, or indefinite, then there is an incremental change we can make to the phase vector Φ which will lower the system's stored energy W. In other words, the system is not at a local energy minimum.

2) Being able to lower W through an incremental change of node phases is equivalent to showing that there exists some vector x such that:

$$x^T Ax < 0.$$

3) As discussed in Section VIII, each link's energy is the integral of its error function. The change in total network energy due to a change in a single node phase is thus:

$$\frac{\partial W}{\partial \phi_i} = F_i$$

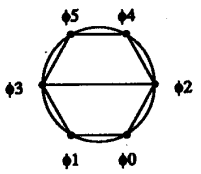


Fig. 13. Example network for Proof 2.

where F_i is the total restoring influence generated on node v_i:

$$F_i = \sum_{v_j(v_i,v_j)\in E} \delta(v_i, v_j).$$

The elements of A are thus:

$$A_{ij} = \frac{\partial^2 W}{\partial\phi_i\partial\phi_j} = \frac{\partial F_i}{\partial\phi_j},$$

i.e., the rate of change in the total restoring influence on node v_i resulting from a phase perturbation of node v_j.

4) Because the triangle wave error function of Fig. 12 is odd (i.e., antisymmetric), the change in node v_i's total restoring influence due to its own perturbation is equal and opposite to the sum of changes due to the perturbation of its neighbors:

$$A_{ii} = -\sum_{j\neq i} A_{ij}.$$

5) For nodes v_i and v_j connected by an error function of slope k,

$$A_{ij} = -k(i \neq j).$$

6) For nodes i and j that are unconnected,

$$A_{ij} = 0.$$

It is useful at this point to consider an example. The phase diagram in Fig. 13 shows one mode-locked stationary phase configuration of a 2×3 grid:

If the positive slope error function of Fig. 6 is used, we have $k = 1$ for all links, yielding a Hessian matrix of:

$$A = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 2 & 0 & -1 & 0 & 0 \\ -1 & 0 & 3 & -1 & -1 & 0 \\ 0 & -1 & -1 & 3 & 0 & -1 \\ 0 & 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix}.$$

If we substitute the "triangle wave" phase detector of Fig. 12, where $k = 1$ for phase errors of magnitude less than $\pi/2$ and $k = -1$ for phase errors greater than $\pi/2$, the Hessian becomes:

$$A = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 2 & 0 & -1 & 0 & 0 \\ -1 & 0 & 1 & 1 & -1 & 0 \\ 0 & -1 & 1 & 1 & 0 & -1 \\ 0 & 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix}.$$

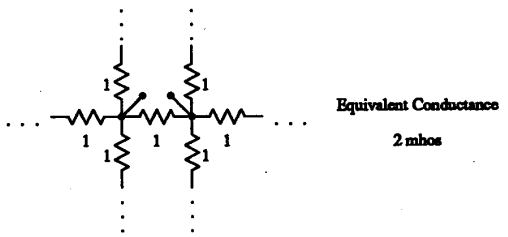


Fig. 14. An infinite 1 Mho resistor grid.

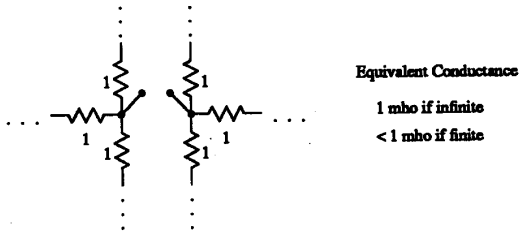


Fig. 15. An infinite grid with one resistor removed.

Determining whether or not some x exists which makes $x^T Ax < 0$ seems difficult. However, another electrical analog can help.

7) Consider a resistive electrical network **analog2** isomorphic in topology to a given mesh network (this is the same as **analog1** but without the capacitors). If the conductance of each resistor in **analog2** is made equal to k (the slope of the phase error function) for the corresponding link in the original phase correction network, then if an external voltage vector x is applied to the nodes of **analog2**, a current vector:

$$i = Ax$$

will flow into the network's nodes. Thus, the equation

$$x^T Ax$$

is exactly equal to the total **power** $p = x^T i$ absorbed and dissipated by **analog2** upon application of a voltage vector x.

8) If we change k from $+1$ to -1 on any link in the phase correction network, we change the corresponding resistor conductance in **analog2** from $+1$ to -1 mhos. The ability to make $x^T Ax$ negative can thus be answered by seeing if it is possible to find a set of node voltages that cause **analog2** to dissipate negative power, i.e. to deliver power to its inputs.

9) For a two-dimensional mesh network under conditions of mode-lock, if we use the triangle-wave error function, then because at least one phase error has magnitude at least $\pi/2$, then at least one resistor in **analog2** will have a conductance of -1 mho.

10) To show that this allows a voltage vector x to be found which dissipates negative power, we consider the infinite two-dimensional grid of 1 mho resistors, shown in Fig. 14.

    It is well-known that from the perspective of the two indicated terminals, this grid has an equivalent conductance of 2 mhos. If we remove the middle resistor, we end up with a conductance of 1 mho, as shown in Fig. 15.

    If the grid is made finite, fewer conducting paths are present and the conductance *decreases* to less than 1 mho. By placing a new -1 mho middle resistor in parallel with this finite grid, we generate an equivalent conductance that is negative, as shown in Fig. 16.

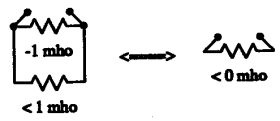


Fig. 16. A finite grid with one -1 Mho resistor.

11) Any nonzero voltage source connected across this negative conductance will receive power. Thus, by step 7, if one phase error in a finite, 2-D, triangle wave error function, cartesian mesh has magnitude greater than $\pi/2$, at least one -1 resistor will be present in **analog 2** and it will be possible to find a voltage vector x such that $x^T Ax < 0$. The original phase correction network will be unstable under these conditions.

12) If more than one phase error has magnitude greater than $\pi/2$, we can draw even more power from the grid. This is tested by setting all but one of the negative conductances to 1 mho, applying a voltage across the remaining -1 mho conductance, and allowing the network (which we know delivers power to the one voltage source) to determine the remaining node voltages. Voltage sources set to these network-determined voltages are then connected to all nodes, and all original -1 mho conductances are restored. With all node voltages the same as before the restoration, all resistors dissipate the same power as before, except for those restored to -1 mho, which now deliver instead of dissipate power. Thus, the total power delivered by the network increases.

B. Three or More Dimensions

The translation of the network convergence problem into a question of power absorption by a isomorphic resistive net under the application of some voltage vector is also valid in three or more dimensions, i.e., a network of nonlinear resistors with capacitive nodes is stable at a particular voltage vector (even if the "voltage" is considered in a modular fashion) if and only if an isomorphic network of linearized resistors absorbs power for all possible voltage vectors. Unfortunately, an infinite three-dimensional network of 1 mho resistors, when viewed from the terminals of one resistor, is equivalent to a conductance of 3 mhos, which, when the middle resistor is replaced by a -1 mho resistor, still leaves a positive conductance of 1 mho. Thus, except for very small networks, one negative resistance will not be sufficient to destabilize a three-dimensional mesh in mode-lock.

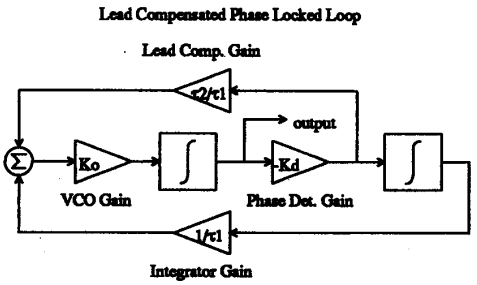


Fig. 17. Lead-compensated control loop.

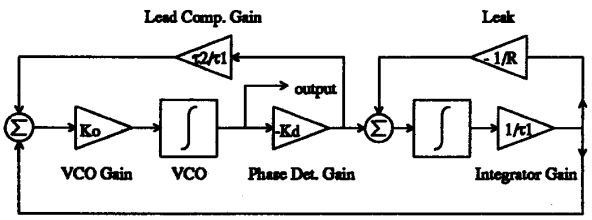


Fig. 18. Weak dc feedback.

Fortunately, 3-D networks in mode-lock have many links with phase error magnitudes greater than $\pi/2$. Furthermore, the negative resistance caused by these large phase errors has, under simulation, been sufficient to destabilize every mode-lock for thousands of cases. A rigorous proof of this result and a thorough study of higher dimensional networks are topics for future research.

X. Substituting a VCO for the VCDL

Until now, we have only considered control systems that directly alter the phase of each local clock. Unfortunately, this requires all nodes to have available a clock source of common frequency. This global dependency is clearly undesirable. What happens if we simply substitute a locally self-sufficient VCO for the globally dependent VCDL at each node?

The result is marginal instability. This is most easily seen by realizing that the control loop now has two integrators in series—that of the error filter, and that of the VCO (integrating frequency into phase). Thus, a feedback loop phase-shift angle of π is created, converting negative feedback to positive feedback, and oscillation ensues.[5]

As with conventional phase-locked loops, these oscillations are easily damped by adding "lead compensation" to the control loop, as shown in Fig. 17.

Because the clock phase alignment system deals with strong, unvarying signals, the actual degree of loop damping is not particularly important. However, to provide for reasonable settling times, a critical damping of

$$\zeta = \frac{\tau_2}{2}\left(\frac{K_o K_d}{\tau_1}\right)^{1/2} = \frac{1}{\sqrt{2}}$$

can be used.

A. VCO Sensitivity and Noise

Low control input sensitivity is one reason VCDLs are preferable to VCOs for low jitter phase locking [12], [27]. Indeed, a standard multi-vibrator VCO typically has such high sensitivity that achieving low phase jitter in the presence of control signal and power supply noise is difficult. However, since our entire system is meant to operate at a known frequency, it is quite reasonable to use a voltage controlled crystal oscillator (VCXO) for the VCO. These devices, with typical frequency pull ranges of 100 parts per million, are now being produced for 15 dollars each, making them practical alternatives to multivibrators in this application [2]. We require that the pull range of each oscillator to be sufficiently wide so as to allow any set of manufactured devices to operate at a common frequency.

B. Deciding on the Reference Phase (or Frequency)

Although we have shown our clock network stable and convergent to zero relative phase, we have not shown how the common phase, as regarded by an external observer, is determined. Although the zero relative phase error condition is an energy "cup" as a function of relative phase, it is an energy "trough" as a function of absolute phase. In other words, the phase (and frequency) of all nodes can be manipulated as a group without affecting the system's energy function whatsoever.[6]

One means of achieving external reference stability is to insert a fixed clock signal into some part of the network. This will pull the entire network into alignment with the reference. However, the point of injection of any external reference and the reference source itself constitute global dependencies that significantly handicap reliability. Thus, some distributed means for agreeing upon a common reference frequency would be preferable to the use of an external reference.

An additional problem is the real limits of each VCO's control signal. Somehow, the agreed upon reference frequency must fall within the range of each VCO.

The solution is to modify each node's loop filter so that weak feedback pulls the control voltage towards the center of its range, as shown in Fig. 18.

This amounts to placing a "leak" resistor around the integrator, so that all else being equal, the integrator output relaxes towards the VCO control's mid-range. Thus, network-wide weak DC feedback exists which centers all VCO control voltages. Stronger phase-error correction keeps the oscillators phase locked.

As it turns out, the filter DC leak also compensates for certain types of implementation nonidealialities, which are discussed later.

[5]This can also be understood by considering once again a mechanical necklace of rubber bands and beads, but this time with lossless rubber-bands (i.e., perfect springs) and beads with nonzero mass. With no energy-dissipating components, oscillation is inevitable. The solution is obvious—the introduction of viscous drag, or damping, to all motion. With the system passive and damped, it becomes stable.

[6]In the mechanical analog, this would correspond to rotating the entire system of rubber bands and beads as a uniform assembly. With no absolute anchoring mechanism, the entire assembly is free to rotate at any speed. Thus the operating frequency of the system is unspecified.

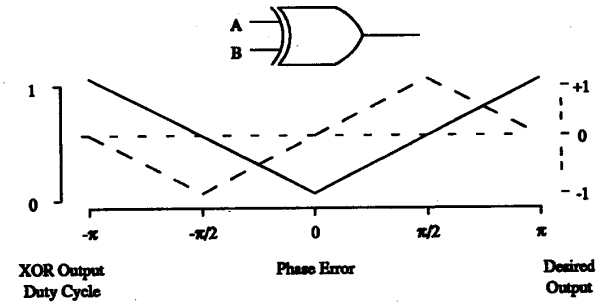


Fig. 19. XOR gate phase detector.

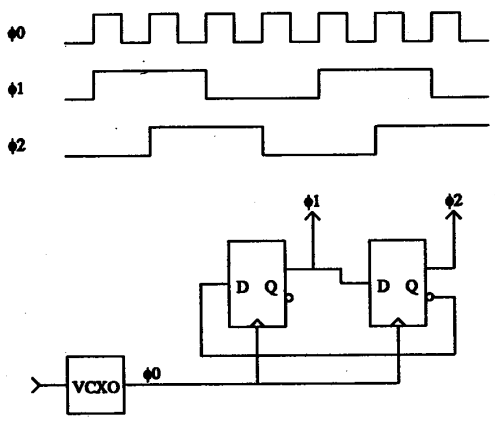


Fig. 20. Ring counter quadrature generator.

XI. Implementation

As shown below, distributed synchronous clocking is surprizingly easy to implement with a small number of simple components.

A. Triangle-Wave Phase Detector

As is well known, [8] triangle wave phase detectors can be implemented digitally by utilizing XOR gates, as shown in Fig. 19.

In most phase locked loops, the negative slope portion of the XOR's transfer function is parasitic. Here it is essential. As shown in Fig. 19, the straight XOR gate characteristic must be modified in two ways. The dc bias is wrong (easily remedied by referencing to a 50% duty cycle voltage instead of 0 volts) and the input axis must be rotated by $\pi/2$ (i.e., one input must be delayed by $\pi/2$).

The $\pi/2$ input delay is most easily accomplished by feeding each VCO's output into a local 2 bit ring counter, producing two 1/4 frequency waveforms in quadrature, as shown in Fig. 20.

This circuit is insensitive to the duty cycle of the VCXO output, but causes the network to lock to the one quarter frequency waveforms, not the VCXO output frequency. This means that t_{c-q} variations in the ring counter's flip-flops will generate phase error between otherwise aligned VCXO outputs. This phase error is not cumulative across the network and can be eliminated by utilizing a VCXO of four times the desired clock rate and using the ϕ_1 and ϕ_2 outputs directly.

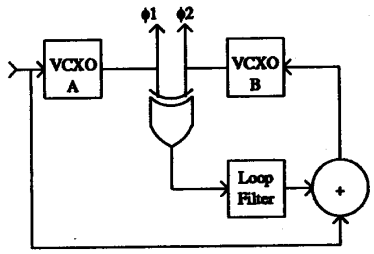


Fig. 21. Phase-locked quadrature generator.

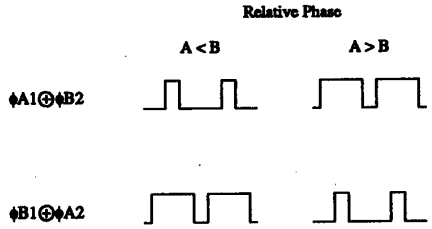


Fig. 22. Differential phase detection.

An alternative solution, shown in Fig. 21, avoids this problem at somewhat greater cost.

This circuit utilizes a simple phase-locked loop within each node to lock a second VCXO to the primary VCXO's phase with a quadrature delay introduced by the XOR phase detector. A voltage controlled time or phase delay can be used instead of the second VCXO.

Regardless of how the quadrature signals ϕ_1 and ϕ_2 are generated, the phase error of neighboring nodes a and b is determined by taking the XOR of the leading output (ϕ_1) of node a and the lagging output (ϕ_2) of node b. To generate the inverse transfer function, we XOR the lagging output (ϕ_2) of node a with the leading output (ϕ_1) of node b, as shown in Fig. 22.

Subtracting these two signals generates the desired triangle-wave error function, as given by the dashed line in Fig. 19.

B. Loop Filter

The availability of differential error signals allows us to detect the zero phase condition by subtraction instead of comparison with an absolute reference. This leads to a minimization of offset errors, and can be accomplished within the loop filter, as shown in Fig. 23.

On the fast time scale of the clock signals, the loop filter's op-amp output is almost constant, and the loop filter has perfect symmetry for its positive and negative inputs. This is purposefully arranged to prevent high frequency differential signals from being fed to the op-amp (which can generate slew asymmetry errors). Once the system is phase locked, the $A > B$ and $A < B$ inputs are identical, and the op-amp will have zero differential drive.

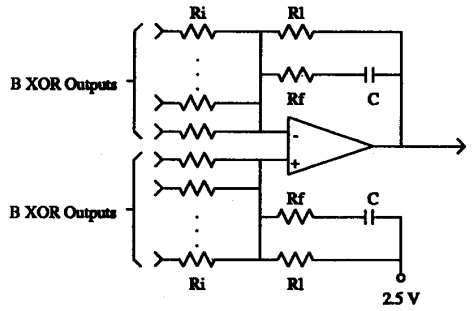


Fig. 23. Loop filter.

C. A 4 Node Example

Shown in Fig. 24 is a block diagram of a 4 node (2 × 2) network. In this diagram, the differential loop filter is represented by an amplifier symbol with an internal integral sign.

Note that each node requires four wires to connect to its neighbors. Alternatively, by sacrificing the advantages of differential operation, two wire links may be used, with subtraction from fixed references. In this case, each node would send opposite quadrature outputs on its opposite sided links, e.g., the north and east links would transmit ϕ_1 to their neighbors while the south and west links would transmit ϕ_2. A single XOR gate would receive the opposite quadrature signals in the middle of each link, and send the result back to both nodes.

The network of Fig. 24 was constructed using 1% resistors and 100 μV maximum offset op-amps. Measured neighboring phase error was under 0.1% of a cycle. Much larger 2 and 3 dimensional networks were simulated to test for convergence and fault tolerance. Some initial results of this fault tolerance study are discussed below.

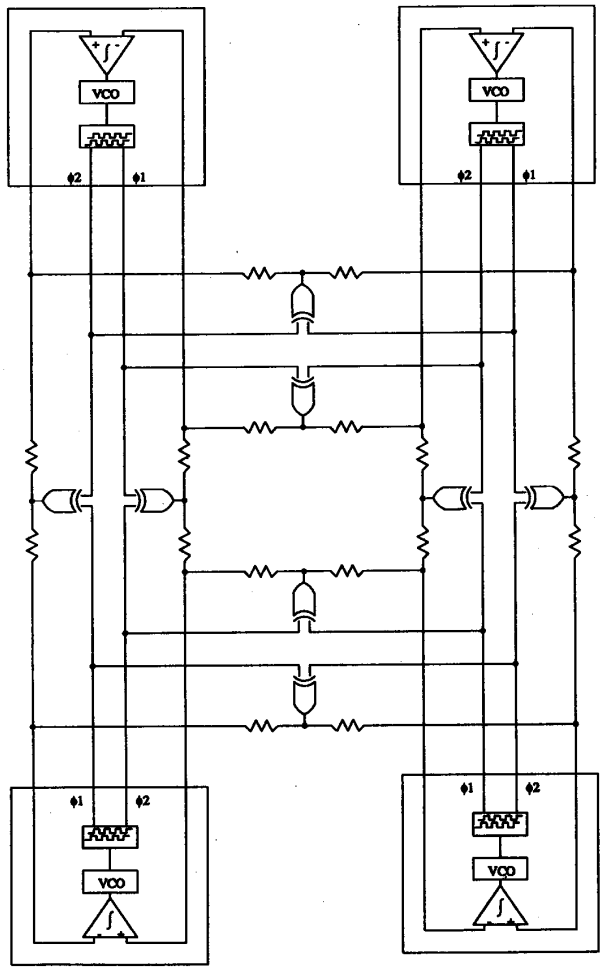


Fig. 24. Four-node prototype.

D. Real Word Tolerances

The hypothetical resistors of **analog1** are ideal: No current flows when there is zero voltage drop and the same current that flows in one lead flows out the other. The first and third quadrant occupancy of these ideal resistor's transfer curves was essential to guarantee system-wide stability.

In the actual clock synchronization system, two uni-directional resistor pairs implement each ideal bi-directional resistance of **analog1.** These uni-directional resistor pairs can be modeled as shown in Fig. 25.

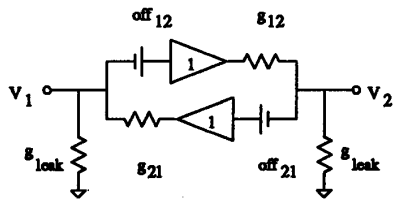


Fig. 25. Interconnect model.

The unity-gain amplifiers serve to make each resistor influence only one terminal, in uni-directional fashion. Due to real-world tolerances, four types of linear error are present:

Common mode offset error	$(\text{off}_{12} + \text{off}_{21})/2$
Differential offset error	$(\text{off}_{12} - \text{off}_{21})/2$
Common mode gain error	$(g_{12} + g_{21})/(2g_{\text{ideal}})$
Differential gain error	$(g_{12} - g_{21})/(2g_{\text{ideal}})$.

1) Common Mode and Differential Offset Errors: Offset errors are created by nonideal op-amps and asymmetric delays in clock lines and phase comparison circuitry. Luckily, both common mode and differential offset errors can be modeled exactly as a fixed, single ended current source attached to each end of **analog1** (i.e., between each node and ground), as shown in Fig. 26.

Each network node can be expected to have a different offset current. If the sum of all the offset currents in the network is zero, stability will not be affected. However, if the total network offset current is nonzero (as it probably will be) the capacitor voltages of **analog1** will all eventually become extremely positive or extremely negative.

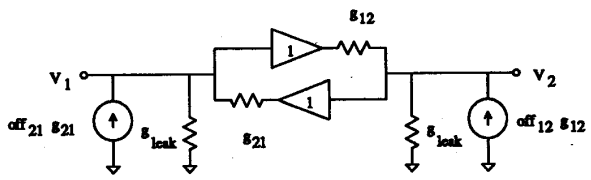


Fig. 26. Interconnect model with offset errors.

Fortunately, the dc "leak" resistors already used in our actual filter implementation to center the VCO controls (and decide on a common reference frequency) act to prevent such divergence. The conductance of these resistors is modelled above as g_{leak}. Offset currents will cause slight increases in local phase error, but the leak resistors provide the offset current required and the network remains stable as a result.

2) Common Mode Gain Error: Common mode gain error corresponds exactly to variability in ideal bi-directional resistor values, from resistor to resistor. As discussed previously, this type of error causes no voltage error (and hence no phase error) in the final state, and does not affect stability.

3) Differential Gain Error: Differential gain error, like offset error, has the potential for destabilizing our control system. From the point of view of the network, the power-absorption characteristics of each uni-directional implementation of a bi-directional resistor determines its effect on system wide stability. Because of the uni-directional amplifier buffers, the power absorbed by the circuit above is not equal to the power dissipated by its internal resistors. However, the power absorbed form the two terminals is still easy to calculate. For zero offset:

$$\begin{aligned} P_{\text{absorbed}} &= v_1 I_1 + v_2 I_2 \\ &= v_1 g_{21}(v_1 - v_2) - v_2 g_{12}(v_1 - v_2) \\ &= (g_{21} v_1 - g_{12} v_2)(v_1 - v_2). \end{aligned}$$

From this equation, we realize that if $g_{12} \neq g_{21}$, it is possible for the uni-directional resistor implementation to deliver instead of absorb power from the network. To prevent network destabilization, we call on the very same "leak" resistors that were used to determine the common operating frequency and to stabilize against offset errors. The effect of these resistors can be seen by adding the power absorbed by the leak resistors to the above equation:

$$\begin{aligned} P_{\text{leak}} &= g_{\text{leak}} v_1^2 + g_{\text{leak}} v_2^2 \\ P_{\text{total}} &= P_{\text{absorbed}} + P_{\text{leak}} \\ &= (g_{21} + g_{\text{leak}}) v_1^2 \\ &\quad - (g_{21} + g_{12}) v_1 v_2 \\ &\quad + (g_{12} + g_{\text{leak}}) v_2^2. \end{aligned}$$

To ensure that a quadratic form like $A x^2 + B x y + C y^2$ is never negative, we must have $B^2 \leq 4AC$. Thus, to insure network stability, we must have

$$(g_{21} + g_{12})^2 \leq 4(g_{21} + g_{\text{leak}})(g_{12} + g_{\text{leak}}).$$

It is helpful to change the individual gains (g_{12} and g_{21}) into common mode and differential components:

$$g_{\text{cm}} = \frac{g_{21} + g_{12}}{2} \qquad g_{\Delta} = \frac{g_{21} - g_{12}}{2}$$

so that:

$$g_{21} = g_{\text{cm}} + g_{\Delta} \qquad g_{12} = g_{\text{cm}} - g_{\Delta}.$$

This makes the above inequality

$$g_{\text{cm}}^2 \leq (g_{\text{leak}} + g_{\text{cm}})^2 - g_{\Delta}^2.$$

Solving for g_{leak} (where $g_{leak} \geq 0$) yields

$$g_{\text{leak}} \geq \sqrt{g_{\text{cm}}^2 + g_{\Delta}^2} - g_{\text{cm}}$$

because of the triangle inequality, we know that

$$\sqrt{g_{\text{cm}}^2 + g_{\Delta}^2} \leq g_{\text{cm}} + g_{\Delta}$$

so that

$$\sqrt{g_{\text{cm}}^2 + g_{\Delta}^2} - g_{\text{cm}} \leq g_{\Delta}.$$

This shows that $g_{\text{leak}} \geq g_{\Delta}$ (the conductance tolerance of any uni-directional resistor) is sufficient to stabilize against any differential gain error.

XII. Catastrophic Faults and Stability

Many types of catastrophic faults, including open circuits, stuck-at-0 faults, and stuck-at-1 faults have very little effect on our clock network's stability. This fortunate result is due to several factors.

1) Internode control currents are zero during lock-up. Thus, if a wire is broken, the control system remains stationary.
2) All node control signals are double-redundant. Differential signals are used to convey phase error. Because of the leak resistors (which pull input terminals to their mid-range voltage), a degraded but still usable control signal is conveyed from one node to another even if only one wire exists.
3) The duty cycle output of an XOR gate phase detector to one stuck-at-0 or stuck-at-1 input is identical to its response during phase lock. In both cases, the output is identical in frequency to the oscillating input and has a 50% duty cycle. Thus, a stuck-at-0 or stuck-at-1 input fault will cause a phase detector to output a zero error signal.
4) The low bandwidth of the loop filter reduces sensitivity to a node operating at a frequency far from normal. This is ordinarily a source of trouble in phase lock loops, which is why combination frequency/phase detectors are used to bring about capture. In our phase control system, this bug becomes a feature (just like the negative slope portion of the XOR detector's transfer curve), and provides for a degree of errant node tolerance.
5) The weak leak resistors allow the system to track a free-running oscillator which for some reason is not listening to its control input. As long as the stubborn oscillator's frequency is within the controlled range of all others, the

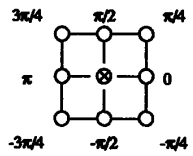


Fig. 27. 3 × 3 network with dead middle node.

rest of the network will track the demanded frequency. Node-to-node phase error increases, of course, in proportion to the leak resistor currents, but these currents are small.

XIII. Error Tolerance, Faults, and Convergence

We have shown above how many sources of error and even catastrophic faults have little effect on system stability near the desired lock point. Once the system has locked, there is little that will upset it. Unfortunately, our proof of instability for other stationary states is not valid for all conceivable errors. Offset and gain errors, if they are sufficiently severe, may affect the system energy function so that other energy minima are created when phase errors are not all zero. The exact sensitivity of the system to such conditions is difficult to calculate, because network size plays a crucial role. This is a topic for future work.

Catastrophic failures can create new, undesired, energy minima. A simple example is a 3 × 3 mesh where the middle node has gone dead, as shown in Fig. 27.

Because of the dead central node, this mesh has become a ring. As such, it is possible for each of the ring's nodes to find the stationary point where all neighboring phase errors are $\pi/4$. Because the slope of the XOR phase detector is positive at $\pi/4$, this stationary configuration will be stable.

Despite these troubles, the situation is better than one might expect. Even in two dimensions, a well-balanced fault condition such as that shown above is not likely. More commonly, faults are not so well placed and mode-locking does not occur. In three dimensions, the connectivity of the network is higher and the ability of single faults to created undesired energy minima is even less. Preliminary simulations for thousands of trials have failed to create a mode-lock state for three-dimensional networks of medium size (a few hundred nodes) with one dead node. More theoretical work to discover the extent of this type of fault tolerance remains to be done.

The nonzero probability of mode-lock due to hardware faults in this very simple implementation of distributed synchronous clocking should not be taken as a weakness of the general approach. A higher level software solution to both mode-lock and fault tolerance can be implemented at little expense using a network of inexpensive single-chip microcontrollers. These microcontrollers could control the activation of links in a lower level phase lock network such as has been described here, guaranteeing globally synchronous phase operation in the working, nonisolated, parts of a network despite severe fault conditions elsewhere. Well-established techniques could be used to insure proper operation in such a system even in the event of Byzantine faults.

XIV. Conclusions

Aside from their obvious performance advantages, multiprocessors have an inherent capability for post-manufacture scalability and fault tolerance. Traditional clock distribution systems, in contrast, are neither scalable nor fault tolerant. Thus, in the coming era of massively parallel architectures, some alternative to centralized clock distribution must be found.

One such alternative, asynchronous control, abandons synchronicity altogether. While intellectually attractive, the abandonment of synchronicity has associated costs that are burdensome for many designs.

In this paper, we have described another alternative to traditional clock distribution—Distributed Synchronous Clocking—where independent clocks are generated locally and a distributed, fault-tolerant, algorithm effects system-wide synchronicity.

Synchronizing a network of oscillators using only phase information is tricky because the 2π modularity of phase can cause lock-up in undesired stable equilibria. To prevent such *mode-locking*, we described a nonlinear error transfer function that was shown (in two dimensions) to destabilize every undesired state while retaining the desired stable state of zero phase error. A method of proof for three dimensions was also suggested. An electronic circuit was then described to implement a lockup-free distributed clocking system. In this implementation, a voltage controlled crystal oscillator, two flip-flops, and one op-amp were used per node, and two XOR gates and four resistors made up each internode link.

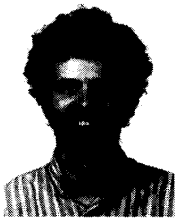
We showed how the common operating frequency can be determined without resorting to external sources, and considered the effects of nonideal circuit components on stability. Finally, a brief discussion of fault behavior was presented, and it was shown that few faults could destabilize the clock network, although some faults could generate new mode-lock states such as were ordinarily prevented by the nonlinear error transfer function. Nevertheless, the circuit is quite fault tolerant considering its simplicity, and higher level software-based supervision of the phase-control hardware can improve the fault tolerance to any level desired.

We believe Distributed Synchronous Clocking to be a simple, effective way to achieve low cost, high quality, low skew clock generation in a synchronous parallel processor. This method is inherently scalable and to a large extent fault tolerant. Measurements of a prototype 4-node system using inexpensive components yielded phase errors of less than 0.1% of a cycle and demonstrated the fault-tolerant properties of the system.

References

[1] K. Alnes, M. Schanke, and E. Kristiansen, "SCI clock synchronization," *IEEE P1596 Std. Committee Doc. 123*, Sept. 1989.
[2] AT&T, *STUGLA VCXO Specification*, 1991.
[3] R. P. Christfield, D. T. Doan, K. R. Williams, and R. H. Wrange, "Phase-locked clocking," *IBM Tech. Discl. Bull.*, vol. 23, no. 7A, pp. 2924–2926, Dec. 1980.
[4] B. Dally, personal communication.
[5] B. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. C-36, pp. 547–553, May 1987.

[6] W. M. Daly, A. L. Hopkins, Jr., and J. F. McKenna, "A fault-tolerant digital clocking system," in *Proc. 3rd Int. Symp. Fault-Tolerant Comput.*, 1973, pp. 17–21.
[7] A. El-Amawy, "Branch-and-combine clocking of arbitrarily large computing networks," in *1991 Int. Conf. Parallel Processing*, 1991, pp. I409–I417.
[8] F. Gardner, *Phaselock Techniques*. New York: Wiley, 1979.
[9] F. M. Gardner, "Charge-pump phase-lock loops," *IEEE Trans. Commun.*, vol. COM-28, pp. 1849–1858, Nov. 1980.
[10] ——, "Phase accuracy of charge-pump PPL's," *IEEE Trans. Commun.*, vol. COM-30, pp. 2363–2364, Oct. 1982.
[11] D.-K. Jeong, G. Borriello, D. A. Hodges, and R. H. Katz, "Design of PPL-based clock generation circuits," *IEEE J. Solid State Circuits*, vol. SC-22, pp. 255–261, Apr. 1987.
[12] M. G. Johnson and E. L. Hudson, "A variable delay line PLL for CPU-coprocessor synchronization," *IEEE J. Solid State Circuits*, vol. 23, pp. 1218–1223, Oct. 1988.
[13] J. L. W. Kessels, "Two designs of a fault-tolerant clocking system," *IEEE Trans. Comput.*, vol. C-33, pp. 912–919, Oct. 1984.
[14] R. K. Kostuk, L. W., and Y-T. Huang, "Optical clock distribution with holographic optical elements," in *Proc. SPIE*, vol. 977, Aug. 1988, pp. 24–36.
[15] C. M. Krishna, K. G. Shin, and R. W. Butler, "Ensuring fault tolerance of phase-locked clocks," *IEEE Trans. Comput.*, vol. C-34, pp. 752–756, Aug. 1985.
[16] W. C. Lindsey, A. V. Kantak, and A. Dobrogowski, "Network synchronization by means of a returnable timing system," *IEEE Trans. Commun.*, vol. COM-26, pp. 892–896, June 1978.
[17] M. R. Miller, "Feasibility studies of synchronised-oscillator systems for P.C.M. telephone networks," *Proc. IEE*, vol. 116, pp. 1135–1143, July 1969.
[18] A. Nowatzyk, "A communication architecture for multiprocessor networks," CMU Ph.D. dissertartion, Carnegie-Mellon CS-89-181, Dec. 1989.
[19] T. Saito, "Application of phase-locked oscillator for PCM network synchronization," *IEEE Trans. Commun.*, vol. COM-30, pp. 2344–2354, Oct. 1982.
[20] M. Schanke, "Proposal for clock distribution in SCI," *IEEE P1596 Std. Committee Doc. 77*, May 1989.
[21] C. L. Seitz, "System timing," in *Introduction to VLSI Systems*, C. A. Mead and L. A. Conway, Eds. Reading, MA: Addison-Wesley, 1980, ch. 7.
[22] K. G. Shin and P. Ramanathan, "Transmission delays in hardware clock synchronization," *IEEE Trans. Comput.*, vol. 37, pp. 1465–1467, Nov. 1988.
[23] T. B. Smith, "Fault-tolerant clocking system," in *Proc. 11th Int. Symp. Fault-Tolerant Comput.*, 1981, pp. 262–264.
[24] N. Vasanthavada, P. N. Marinos, and G. S. Mersten, "Design and performance evaluation of mutually synchronized fault-tolerant clock systems," in *Proc. 16th Int. Symp. Fault-Tolerant Comput.*, 1986, pp. 206–210.
[25] S. Ward, "The NuMesh: A scalable, modular, 3D interconnect," in *NuMesh Industrial Partners Meeting*, Cambridge, MA, Feb. 1991.
[26] M. W. Willard, "Analysis of a system of mutually synchronized oscillators," *IEEE Trans. Commun. Technol.*, vol. COM-18, pp. 467–483, Oct. 1970.
[27] R. Woudsma and J. M. Noteboom, "The modular design of clock-generator circuits in a CMOS building-block system," *IEEE J. Solid State Circuits*, vol. SC-20, pp. 770–774, June 1985.



Gill A. Pratt received the S.B., S.M., and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology in 1983, 1987, and 1989, respectively.

He is currently an Assistant Professor of Electrical Engineering and Computer Science at M.I.T. His research interests include computer architecture and robotics.

John Nguyen received the S.B., S.M., and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology in 1987, 1989, and 1993, respectively.

He is currently a Post-Doctoral Researcher at the University of Michigan, Ann Arbor. His research interests include computer architecture and compilers for multiprocessors.