99238

P.24

NASA Technical Paper 3209

July 1992

Experimental Validation of Clock Synchronization Algorithms

Daniel L. Palumbo
and R. Lynn Graham

NASA

ERRATA

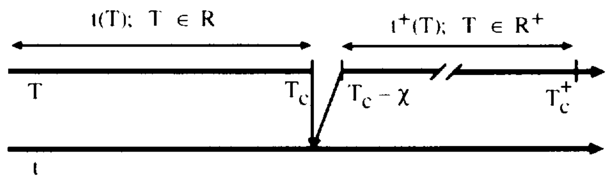
NASA Technical Paper 3209

Experimental Validation of Clock
Synchronization Algorithms

Daniel L. Palumbo and R. Lynn Graham

July 1992

Page 5, figure 5: The figure should appear as follows:



Issued July 1992

NASA
Technical
Paper
3209

1992

Experimental Validation of Clock Synchronization Algorithms

Daniel L. Palumbo
Langley Research Center
Hampton, Virginia

R. Lynn Graham
PRC Kentron, Inc.
Hampton, Virginia

NASA

National Aeronautics and
Space Administration

Office of Management

Scientific and Technical
Information Program

Abstract

The objective of this work is to validate mathematically derived clock synchronization theories and their associated algorithms through experiment. Two theories are considered, the Interactive Convergence Clock Synchronization Algorithm and the Midpoint Algorithm. Special clock circuitry was designed and built so that several operating conditions and failure modes (including malicious failures) could be tested. Both theories are shown to predict conservative upper bounds (i.e., measured values of clock skew were always less than the theory prediction). Insight gained during experimentation led to alternative derivations of the theories. These new theories accurately predict the behavior of the clock system. It is found that a 100-percent penalty is paid to tolerate worst-case failures. It is also shown that under optimal conditions (with minimum error and no failures) the clock skew can be as much as three clock ticks. Clock skew grows to six clock ticks when failures are present. Finally, it is concluded that one cannot rely solely on test procedures or theoretical analysis to predict worst-case conditions.

Introduction

Many theories of clock synchronization have been proposed and subjected to the rigors of mathematical proof of correctness (see refs. 1 and 2). Few of these theories are validated by experiment. One of the difficulties in validating clock synchronization theory is that the theory often predicts the behavior of the synchronization algorithm under failure conditions that are hard to replicate in the lab (e.g., the presence of a "malicious liar," ref. 3). The objective of this work is to select a theory for validation, build a synchronization subsystem that is based on this theory, and subject this subsystem to a series of tests designed to validate the theory.

The Interactive Convergence Clock Synchronization Algorithm (ICCSA) of Lamport and Melliar-Smith (ref. 4) was chosen as a test subject because of its use on the SIFT (Software Implemented Fault-Tolerance) computer (ref. 5) and the fact that the algorithm and the accompanying bounding theory had been recently subjected to the rigors of a mechanical proof (ref. 6). During the process of testing, it was found that the theoretical bound on the clock skew was larger than the observed maximum clock skew. Although the theory only guarantees an upper bound, this discrepancy led to inquiries into why the theory was not more accurate. In the course of this investigation, an alternative method for the derivation of the expression for the clock skew bound was developed. This new expression accurately predicts the observed clock skew for the Interactive Convergence Clock Synchronization Algorithm.

Lundelius has derived a clock skew bound (ref. 7) for the Midpoint Algorithm proposed by Dolev (ref. 8). The Dolev algorithm was programmed into the clock synchronization subsystem and tested. As with the ICCSA theory, the predicted bound was found to be greater than the observed clock skew (although only in extreme cases). With the insight gained from the previous derivation and applying a fresh approach to the worst-case analysis of the Midpoint Algorithm, a new expression is derived that accurately predicts the observed clock skew.

In the following sections, expressions for the clock skew bound for both the ICCSA and the Midpoint Algorithm will be derived. A test plan will be introduced, and the design of the clock subsystem described. Results of the testing are presented and case studies are done. Finally, conclusions concerning this work are drawn.

Symbols

EHDM	extended hierarchical design methodology
f_c	clock counter frequency
f_r	clock reference frequency
HDM	hierarchical design methodology
ICCSA	Interactive Convergence Clock Synchronization Algorithm
m	number of faulty clocks in a synchronizing set
n	number of clocks in a synchronizing set
p, q, r, s	processor designations
R	minimum length of synchronization period
S	minimum length of synchronization process
T	clock time
T_c	time of clock correction
T_{qp}	clock reading of processor p upon receipt of synchronization signal from processor q
T_s	time of synchronization signal
t	real time, $(1-\rho)T+\varepsilon+t_0$
t^i	uncorrected clock function (see fig. 4)

t_0	real-time offset at $T = 0$
v	drift rate setting in clock subsystem peripheral
Δ	limit of perceived skew allowed in ICCSA
Δ_{qp}	perceived skew of processor p with respect to processor q
δ	maximum skew between good clocks in a synchronizing set
δ_{qp}	real-time skew between clocks p and q
$\delta_{qp}(T)$	real-time skew between processors p and q when clock for processor p equals T
δ_0	maximum initial skew
ε	maximum clock read error
ε_0	minimum read error, $1/f_c$
ρ	clock drift rate with respect to real time
ρ_M	maximum drift rate expected between any two clocks
ρ_p	drift rate of clock p
ρ_{qp}	drift rate between clocks p and q
Σ	maximum clock correction
χ_p	clock correction calculated by processor p
∇	perceived skew value derived from faulty clock reading

Clock Fundamentals

The purpose of synchronizing clocks is to provide a global time base throughout a distributed system. Once this time base exists, transactions between members of the distributed system can be controlled based on time. For example, the management of redundant data in a real-time fault-tolerant computer is simplified if the processors are synchronized (ref. 9). In the following discussions, the term *clock* refers to a device that provides a time base for a processor. A processor thus inherits time-related characteristics from its clock. For this reason, we sometimes refer to a processor as *drifting* with respect to other processors when, in fact, the drift is actually a property of the clock.

A common convention has been that real time is denoted by a lowercase letter, as in t or δ, and that clock times are capitalized, as in T and Δ. A clock approximates real time with the relationship between clock time and real time given by

$$t = (1 - \rho)T \tag{1}$$

where t is real time, T is clock time, and ρ is the rate of drift of clock time from real time. A clock may have some nonzero offset at clock time $T = 0$, as represented by the constant t_0 in equation (2).

$$t = (1 - \rho)T + t_0 \tag{2}$$

If ρ is zero, the clock is a perfect clock. If ρ is positive, the clock is a fast clock and accumulates time faster than real time. Clocks are considered to be digital devices consisting of a crystal oscillator and a counter. Ideally, the crystal oscillates at frequency f_c. Deviations from this specification are what cause drift among a set of clocks. The digital nature of the counter causes the relation between t and T to be discontinuous, as shown in figure 1. The error in reading a clock is denoted as ε, and for digital clocks ε has a minimum, ε_0, of $1/f_c$. Thus, for a digital clock the inverse of equation (2) becomes

$$T = \lfloor (t - t_0)/(1 - \rho) \rfloor \tag{3}$$

where $\lfloor \ \rfloor$ represents the *floor* function.

For a set of clocks, a maximum drift rate ρ_M is chosen so that for any nonfaulty clock p in the set

$$|\rho_p| \leq \rho_M/2 \tag{4}$$

The drift between any two clocks p and q in the set of nonfaulty clocks is given by

$$\rho_{qp} = \rho_q - \rho_p \tag{5}$$

with

$$|\rho_{qp}| \leq \rho_M \tag{6}$$

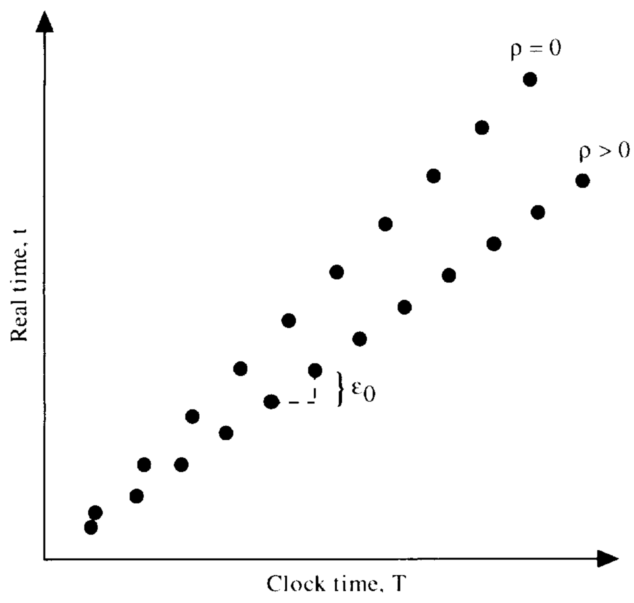


Figure 1. Real time versus clock time.

The real-time skew δ_{qp} that exists between two clocks at some clock time T is given by

$$\delta_{qp}(T) = t_p(T) - t_q(T) \tag{7}$$

Alternatively, the skew can be expressed in terms of the difference between two clock values at some real time t. The form of equation (7) was chosen, as this is the perspective taken in the Lamport and Melliar-Smith proof.

Synchronizing Clocks

In the two algorithms considered here, synchronization is accomplished by periodically executing an algorithm that first computes a clock correction value and then applies the correction to the local processor clock. In order to compute either of the two algorithms, each processor in the synchronizing set must obtain a *perceived* skew Δ_{qp} between its clock and each of the other clocks in the set. To obtain Δ_{qp}, processor p must compute the difference between its local clock and the remote clock. Processor p must, in effect, read the clock of processor q. Figure 2 graphically depicts this process. By design, the algorithm executes every R time units and takes S time units to complete. In the clock subsystem constructed for these tests, actual clock values are not transmitted. Instead, at predetermined time T_s during S, clock q sends a synchronization signal to p. Upon receipt of this signal, p reads its local clock and stores this value, T_{qp}; T_{qp} is then the local clock value for processor p taken at a real time corresponding to T_s, the clock reading for processor q. The perceived skew Δ_{qp} can then be computed as $T_{qp} - T_s$.

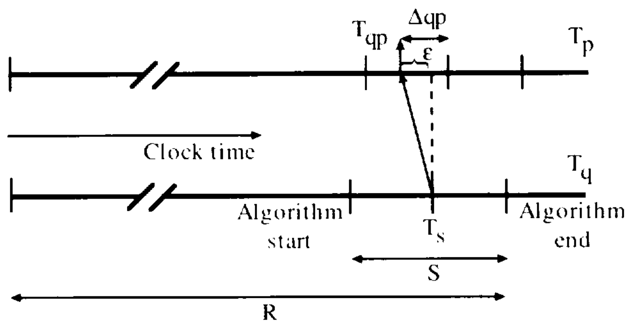


Figure 2. Reading the clock of another processor.

More precisely stated, the perceived skew values are arrived at by the following process:

1. Each processor broadcasts a synchronizing signal at a predetermined time T_s.

2. Upon receipt of the synchronizing signals from other processors, the receiving processor p stores its clock value, T_{qp}.

3. The perceived skew is then the stored value, T_{qp}, minus T_s, or

$$\Delta_{qp} = T_{qp} - T_s \tag{8}$$

Figure 3 represents this process taking place between two processors p and q, with processor q having a clock that is faster than processor p. From the graph it can be seen that T_{qp} can be thought of as the value of clock p at real time $t_q(T_s)$, or

$$T_{qp} = T_p(t_q(T_s)) \pm \varepsilon_{qp} \tag{9}$$

where T_p is the inverse clock function of clock p, and ε_{qp} is the error inherent in taking T_p.

By using equation (8) with equations (9), (2), and (3), the following expression for the perceived skew can be derived (see appendix A):

$$\Delta_{qp} = -\delta_{qp}(T_s) \pm \varepsilon + \rho_p \Delta_{qp} \tag{10}$$

An examination of figure 3 will reveal that if q is faster than p, then $\rho_{qp} > 0$, $\delta_{qp} > 0$, and $\Delta_{qp} < 0$. To correct its clock, the slower processor p must *add* a positive value to the clock. Since the values of Δ_{qp} will be negative, the resulting correcting value χ must be subtracted from clock p (assuming that a sign change does not occur in the algorithm).

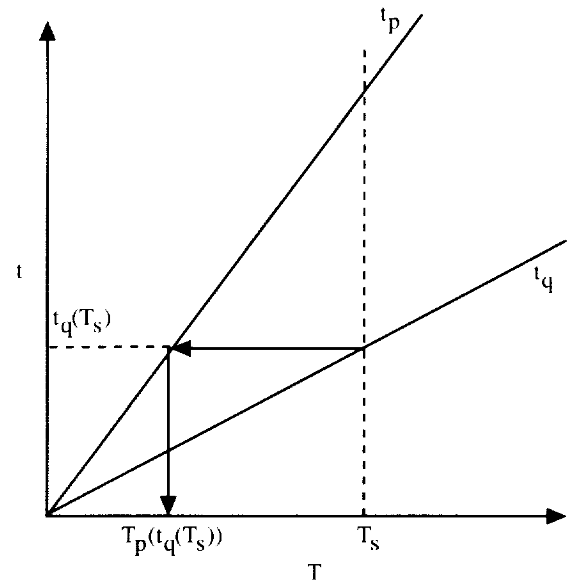


Figure 3. Formulation of T_{qp}.

Figure 4 graphically depicts the effect of applying a correction to a fast clock. The superscripts i and $i + 1$ refer to synchronization periods, as will be discussed in the section "Periods i and $i + 1$." In the figure, t^i refers to the uncorrected clock function and t^{i+1} to the corrected clock function. The correction is applied at clock time T_c. The following relationship exists between the corrected and the uncorrected clock functions:

$$t^{i+1}(T_c) = t^i(T_c) + (1 - \rho)\chi^i \tag{11}$$

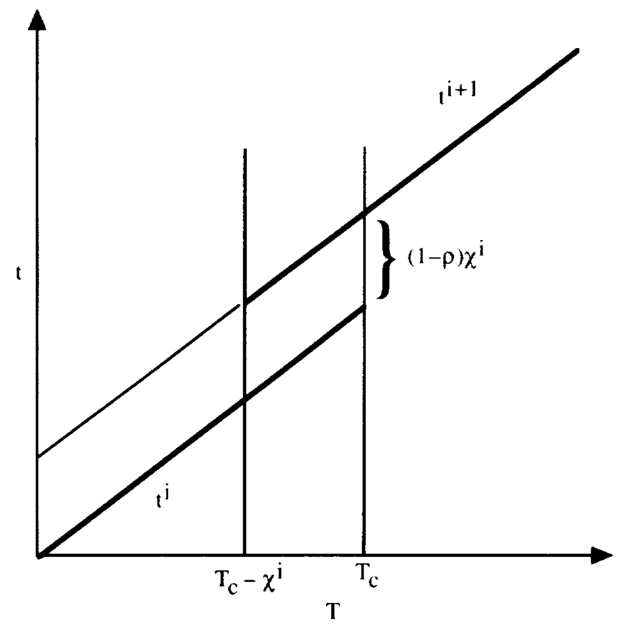


Figure 4. Effect of applying correction.

Some Useful Relations

The following relations will be used to derive the bound formulas. Detailed derivations of these relations are given in appendix A. These relations hold true provided that a clock correction is not applied during the interval from T to $(T + C)$.

$$\delta_{qp}(T + C) = \delta_{qp}(T) + \rho_{qp}C \tag{12}$$

Equation (12) states that the skew between p and q at some time T plus a constant C is equivalent to the skew that exists at time T plus an amount equal to the relative drift rate times the constant.

$$\delta_{qp}(T) = \rho_{qp}(T - T_c) + \delta_{qp}(T_c) \tag{13}$$

Equation (13) states that the skew between p and q during a synchronization period is equivalent to the skew at the beginning of the period (T_c) plus the skew accumulated over the period due to the relative drift ρ_{qp}.

$$\delta_{rq}(T) - \delta_{rp}(T) = \delta_{pq}(T) \tag{14}$$

Equation (14) states a relationship that exists between the skews of three good clocks, p, q, and r.

The Proofs

The statement of the bounding theorem is taken largely from references 4 and reference 6.

Clock Skew Bounding Theorem

For a set of n processors cooperating in the synchronization algorithm for all time T through period i, a bound δ exists on the skew between any two of the processors given that at most m of the n processors are faulty. Stated mathematically,

$$|t_p^i(T) - t_q^i(T)| < \delta \tag{15}$$

Because this theorem is written in terms of consecutive periods of time, it is convenient to use proof by induction. To do this, we will derive an expression for δ for the first interval, $i = 0$, and then show that another expression exists that is true for the following intervals. This latter expression depends on characteristics of the synchronization algorithm, and thus separate derivations are necessary for the ICCSA and the Midpoint Algorithm.

The First Period, $i = 0$

At system start-up, assume a maximum skew δ_0 exists between all good processors in the set. Then, at the end of period 0 with $T = R$,

$$\begin{aligned} t_p^0(R) - t_q^0(R) &= (1-\rho_p)R - (1-\rho_q)R + t_{0p} - t_{0q} \\ &= (\rho_q - \rho_p)R + t_{0p} - t_{0q} \\ &= \rho_{qp}R + t_{0p} - t_{0q} \\ |t_p^0(R) - t_q^0(R)| &\le \rho_M R + \delta_0 \le \delta \end{aligned} \tag{16}$$

where in expression (16) $|t_{0p} - t_{0q}| \le \delta_0$. Expression (16) is thus one constraint on the value of δ, i.e., $\delta \ge \rho_M R + \delta_0$.

Periods i and $i+1$

To continue the proof, we will assume that an expression for the bound is true for period i and show that the same expression is true for $i+1$. As stated above, this expression will depend on the synchronization algorithm. However, we can derive a general expression from which the subsequent proofs can continue. Refer to figure 5 for a graphical representation of the situation that exists between periods i and $i+1$. To reduce clutter in the terms, the lack of a superscript will refer to period i and a + superscript will refer to period $i+1$.

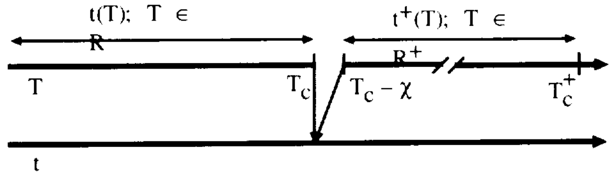


Figure 5. Transition from period i to period $i+1$.

Using equation (7) for period $i+1$, we have

$$\delta_{qp}^+(T_c) = t_p^+(T_c) - t_q^+(T_c) \tag{17}$$

Then using equation (11) to replace the t^+ functions with t, and then equation (7) again to recombine the t functions, we get

$$\delta_{qp}^+(T_c) = \delta_{qp}(T_c) + (\chi_p - \chi_q) + \rho_q\chi_q - \rho_p\chi_p \tag{18}$$

It is assumed that the difference between the $\rho\chi$ terms can be ignored. For an error-free system this is justified because, when considering the worst-case skew condition with ρ_q equal to negative ρ_p, $\rho_q\chi_q$ will be of the same sign and approximately equal to $\rho_p\chi_p$. When clock read errors are present, the worst-case read error effect occurs when the error for clock q is equal to but opposite to the error for clock p. As in the error-free case, the effect is canceled out in the $\rho\chi$ difference terms. In short, when $\chi_p - \chi_q$ is maximized, $\rho_q\chi_q - \rho_p\chi_p$ is minimized.

Substituting the resulting expression in equation (13) written for period $i+1$, we obtain

$$\delta_{qp}^+(T) \le \delta_{qp}(T_c) + (\chi_p - \chi_q) + \rho_M R \tag{19}$$

with $R \ge (T - T_c)$. Expression (19) will be used in the following sections to derive bound expressions for the associated algorithms.

The Interactive Convergence Clock Synchronization Algorithm

The ICCSA is derived for n clocks synchronizing in the presence of m faulty clocks. In this algorithm, a processor computes the correction by averaging all the perceived skew values Δ_{qp}. To limit the effect of a faulty clock, the Δ_{qp} are subjected to the test that their absolute value be less than some maximum expected value Δ. If Δ_{qp} exceeds Δ, Δ_{qp} is set to 0. More precisely

$$\chi_p = \frac{1}{n}\sum_{q=1}^{n}\overline{\Delta}_{qp} \tag{20}$$

where $\overline{\Delta}_{qp} = 0$ if $|\Delta_{qp}| > \Delta$. A value for Δ is easily derived from equation (10):

$$\Delta \ge \delta + \varepsilon + \frac{\rho_M}{2}\Delta \tag{21}$$

Wishing to replace the correction terms in equation (19) with an expression based on equation (20), we look at the correction terms more closely:

$$\begin{aligned} \chi_p - \chi_q &= \frac{1}{n}\sum_{r=1}^{n}\overline{\Delta}_{rp} - \frac{1}{n}\sum_{r=1}^{n}\overline{\Delta}_{rq} \\ &= \frac{1}{n}\sum_{r=1}^{n}(\overline{\Delta}_{rp} - \overline{\Delta}_{rq}) \\ &= \frac{1}{n}\sum_{r=1}^{n-2-m}(\Delta_{rp} - \Delta_{rq}) \\ &\quad + \frac{1}{n}(\Delta_{pp} - \Delta_{pq}) + \frac{1}{n}(\Delta_{qp} - \Delta_{qq}) + \frac{m}{n}(\nabla_p - \nabla_q) \end{aligned}$$

The final expression contains four terms, the first of which contains values of Δ_{qp} taken from $n-2-m$ good processors. The second and third terms have readings of the local clock, e.g., Δ_{pp}. The last term holds the readings from the m possible faulty clocks (denoted by ∇). In appendix B, each term is taken individually and expanded under assumptions relating to those terms and then recombined to obtain

$$\chi_p - \chi_q \leq \left(\frac{m-n}{n}\right)\delta_{qp}(T_c) + \frac{2(n-1-m)}{n}\varepsilon + \frac{\rho_M(n-m)}{n}\Delta + \frac{2m}{n}\Delta \qquad (22)$$

Substituting equation (22) into equation (19), we get

$$\delta^+_{qp}(T) \leq \left(\frac{m}{n}\right)\delta_{qp}(T_c) + \frac{2(n-1-m)}{n}\varepsilon + \frac{\rho_M(n-m)}{n}\Delta + \frac{2m}{n}\Delta + \rho_M R \qquad (23)$$

Now we create an expression for δ and assume it holds for period i, i.e., that $\delta \geq \delta_{qp}(T)$, with T in i and with δ given by

$$\delta \geq \frac{2(n-1-m)}{n-m}\varepsilon + \rho_M\Delta + \frac{2m}{n-m}\Delta + \left(\frac{n}{n-m}\right)\rho_M R \qquad (24)$$

Under this assumption then, by replacing δ_{qp} with δ in equation (23) we have

$$\delta^+_{qp}(T) \leq \left(\frac{m}{n}\right)\delta + \frac{2(n-1-m)}{n}\varepsilon + \frac{\rho_M(n-m)}{n}\Delta + \frac{2m}{n}\Delta + \rho_m R \qquad (25)$$

Now using equation (24) for δ, it follows that

$$\delta^+_{qp}(T) \leq \frac{2(n-1-m)}{n-m}\varepsilon + \rho_M\Delta + \frac{2m}{n-m}\Delta + \left(\frac{n}{n-m}\right)\rho_M R \leq \delta \qquad (26)$$

which completes the proof.

The Midpoint Algorithm

In the Midpoint Algorithm, as suggested by Dolev (ref. 8), the correction is computed as the midpoint of the span of values of Δ_{qp} after the m largest and smallest values have been discarded. Stated for the case where $m = 1$,

1. Processor obtains all the Δ_{qp} values.
2. The Δ_{qp} are ordered so that $\Delta_{\text{min}} \leq \Delta_{\text{min}'} \cdots \leq \Delta_{\text{max}'} \leq \Delta_{\text{max}}$.
3. Discard Δ_{min} and Δ_{max} and use the new minimum and maximum, $\Delta_{\text{min}'}$ and $\Delta_{\text{max}'}$, to compute the correction as

$$\chi_p = \frac{\Delta_{\text{min}'} + \Delta_{\text{max}'}}{2} \qquad (27)$$

This algorithm has the property that the clock reading of a faulty processor will not be used to compute the correction unless it is bounded by good clock readings. This results in it being possible to derive a tighter bound.

In the following sections, an expression for χ_p is derived by first considering the case with no errors, then with some clock read error ε, and finally with an arbitrary faulty clock reading.

The ideal case. In the absence of a faulty clock and read errors, all good processors in a synchronizing set will place the processor readings in the same order. Take, for example, the four-processor system (p, q, r, s) where

$$t_p(T) \leq t_q(T) \leq t_r(T) \leq t_s(T)$$

Then, for any member i in (p, q, r, s)

$$\Delta_{pi} \leq \Delta_{qi} \leq \Delta_{ri} \leq \Delta_{si}$$

All good processors will then use clock readings from the same two processors to compute their respective corrections. (In the above example, this would be Δ_{qi} and Δ_{ri}.) This is equivalent to the processors using a single clock reading which is at the midpoint of these two clock readings ($\delta_{\text{mid}}(T_s)$). Thus using equation (10) with $\varepsilon = 0$, we have

$$\chi_p = \Delta_{\text{mid},p} = -\delta_{\text{mid}}(T_s) + \rho_p\Delta_{\text{mid},p} \qquad (28)$$

Including read error. Any read error present in the clock readings will affect the clock correction by at most the read error ε:

$$\begin{aligned}\chi_p &= \frac{\Delta_{\text{min}'} \pm \varepsilon + \Delta_{\text{max}'} \pm \varepsilon}{2} \\ &= \frac{\Delta_{\text{min}'} + \Delta_{\text{max}'}}{2} \pm \varepsilon \\ &= \Delta_{\text{mid},p} \pm \varepsilon \\ &= -\delta_{\text{mid},p}(T_s) + \rho_p\Delta_{\text{mid},p} \pm \varepsilon \qquad (29)\end{aligned}$$

Including a faulty clock. In reference to figure 6, consider that the maximum and minimum readings taken from good clocks differ by at most

$\delta + 2\varepsilon$. The algorithm guarantees that if a faulty clock reading is used in computing the correction, it is bounded by good clock readings. Thus, the maximum error that a faulty clock could cause is $\frac{1}{2}(\delta + 2\varepsilon)$. The expression for the correction including both read error and error due to a faulty clock reading becomes

$$\begin{aligned}\chi_p &= \frac{\Delta_{\text{min}'} \pm \left(\frac{\delta + 2\varepsilon}{2} + \varepsilon\right) + \Delta_{\text{max}'} \pm \left(\frac{\delta + 2\varepsilon}{2} + \varepsilon\right)}{2} \\ &= \frac{\Delta_{\text{min}'} + \Delta_{\text{max}'}}{2} \pm \left(\frac{\delta}{4} + \varepsilon\right) \\ &= \Delta_{\text{mid},p} \pm \left(\frac{\delta}{4} + \varepsilon\right)\end{aligned}$$

$$\chi_p = -\delta_{\text{mid},p}(T_s) + \rho_p \Delta_{\text{mid},p} \pm \left(\frac{\delta}{4} + \varepsilon\right) \tag{30}$$

A maximum correction Σ can be obtained by using δ and Δ for the maximum values of $\delta_{\text{mid},p}$ and $\Delta_{\text{mid},p}$, giving

$$\begin{aligned}\Sigma &\geq \delta + \frac{\rho_M}{2}\Delta \pm \left(\frac{\delta}{4} + \varepsilon\right) \\ &\geq \frac{5\delta}{4} + \varepsilon + \frac{\rho_M}{2}\Delta\end{aligned} \tag{31}$$

Now using equation (30) in equation (19), we obtain

$$\begin{aligned}\delta_{qp}^{+}(T) \leq \delta_{qp}(T_c) &+ [\delta_{mq}(T_s) - \delta_{mp}(T_s)] \\ &+ \left(\rho_q \Delta_{\text{mid},q} - \rho_p \Delta_{\text{mid},p}\right) + 2\left(\frac{\delta}{4} + \varepsilon\right) + \rho R\end{aligned} \tag{32}$$

Ignoring the difference between the $\rho\Delta$ terms (as was done in eq. (18) with the $\rho\chi$ terms) and using equation (14) on $[\delta_{mq}(T_s) - \delta_{mp}(T_s)]$, we get

$$\delta_{qp}^{+}(T) \leq \delta_{qp}(T_c) - \delta_{qp}(T_s) + 2\left(\frac{\delta}{4} + \varepsilon\right) + \rho_M R \tag{33}$$

We then use equation (12) with $T_c = T_s + \Delta$ to obtain

$$\delta_{qp}^{+}(T) \leq \delta_{qp}(T_c) - \delta_{qp}(T_c) + \rho_M \Delta + 2\left(\frac{\delta}{4} + \varepsilon\right) + \rho_M R$$

$$\delta_{qp}^{+}(T) \leq \rho_M \Delta + 2\left(\frac{\delta}{4}\right) + \rho_M R \tag{34}$$

Now to continue the induction, we assume the following expression to be true for period i:

$$\delta \geq 4\varepsilon + 2\rho_M \Delta + 2\rho_M R \tag{35}$$

Substituting equation (35) for δ in equation (34), we get

$$\delta_{qp}^{+}(T) \leq 4\varepsilon + 2\rho_M \Delta + 2\rho_M R \leq \delta \tag{36}$$

which completes the proof.

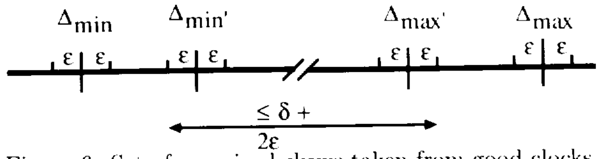


Figure 6. Set of perceived skews taken from good clocks.

If the effect of faulty processors were to be ignored ($m = 0$), then equation (35) becomes

$$\delta_{qp}^{+}(T) \leq 2\varepsilon + \rho_M \Delta + \rho_M R \tag{37}$$

and the clock bound is

$$\delta \geq 2\varepsilon + \rho_M \Delta + \rho_M R \tag{38}$$

Experimental Verification

To experimentally verify the derived skew bounds, several tests were performed in which the effect of varying one parameter of the skew bound expression was measured while the remaining parameters were either held constant or zero. The parameters are $\delta_0, \varepsilon, \rho, m, n$, and R. For the clock subsystem that was actually tested, the number of clocks n was kept constant at four, and thus m was limited to (0,1) for both algorithms. It was decided that if ρ is tested, it is not necessary to test the effect of varying the synchronization period R. The following test cases were then generated:

1. $\delta = 0$ with $\delta_0 = 0, m = 0, \varepsilon = 0$, and $\rho = 0$
2. $\delta = f(\delta_0)$ during the first period with $\rho = 0$
3. $\delta = f(\delta_0)$ during the first period with $\rho = C$
4. $\delta = f(\varepsilon)$ with $m = (0, 1), \rho = 0$, and $\delta_0 = 0$
5. $\delta = f(\varepsilon)$ with $m = (0, 1), \rho = C$, and $\delta_0 = 0$
6. $\delta = f(\rho)$ with $m = (0, 1), \varepsilon = 0$, and $\delta_0 = 0$
7. $\delta = f(\rho)$ with $m = (0, 1), \varepsilon = C$, and $\delta_0 = 0$

In all the tests, the read error is treated as a random variable with a mean of zero. This is not the case in most communication systems. However, the expected value of the communication delay is often known and can be subtracted from the clock readings in the synchronization algorithm, so that the resulting effect is a read error with zero mean.

In addition to functioning as a synchronizing circuit, the clock subsystem must be able to support the test plan. The following capabilities were then designed into the clock subsystem and the experiment support environment:

1. Ability to sustain long-duration data acquisition of internal variables without perturbing the system function
2. Availability of a *global* clock that can be read by each processor under test; the global clock will represent real time
3. Ability to set the starting skew δ_0 of each clock
4. Ability to set the drift rate of each clock with respect to real time, i.e., the global clock
5. Ability to set the read error of each clock
6. Ability to emulate a faulty clock, especially a malicious liar

The following sections describe the clock subsystem and experiment environment.

Design of Clock Subsystem

The clock subsystem is designed as a synchronization peripheral. This primary function is then augmented to provide the data acquisition and control necessary to accomplish the tests proposed in the previous section. The next section will describe the design of the primary synchronization function. This is followed by a section on the actual design, which includes the test augmentations. In these and the subsequent sections, the term clock tick is used to refer to one increment of digital time. Practically all the parameters are stated in terms of clock ticks instead of time. A clock tick is easily converted to time once the base frequency of the clock is known.

A clock synchronization peripheral. As mentioned previously, the ICCSA was first used in the SIFT computer. This implementation was tested (ref. 10), and it was found that the clock skews were due primarily to large clock read errors. It was proposed then that a simple hardware enhancement could greatly reduce the read error, tighten the clock synchronization, and thus increase the efficiency of interprocessor communication. While it is possible to put the entire clock function in hardware, for the purposes of this test it is convenient to have the algorithm in software so that alternate algorithms can be tested. Having the algorithm in software also enhances data acquisition and fault simulation.

Figure 7 is a block diagram of how the clock functions are distributed between the clock peripheral hardware and the synchronization software. The clock hardware monitors a communication channel for the presence of a synchronizing signal. When a sync signal is detected, the hardware latches the local clock value and stores it in a register related to the processor that sent the signal. The clock hardware also generates a sync signal at a specified time T_s and places the signal on the communication channel. These functions are done most efficiently (i.e., the lowest read error is realized) if they are integrated with the communications and networking protocols. The clock peripheral also generates an interrupt to the host processor to indicate the end of the period. The processor then executes the clock algorithm, reading the clock read registers, computing the correction, and correcting the clock.

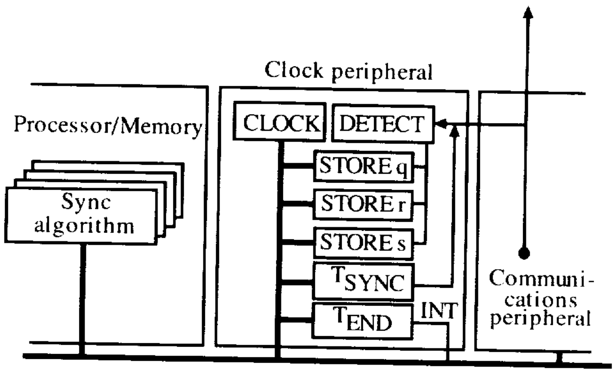


Figure 7. Block diagram of clock functions.

Several considerations must be made to properly design the clock peripheral. The ICCSA requires that all clock readings greater then Δ be ignored. This is equivalent to a buffer of size Δ existing before and after the synchronization time T_s (see fig. 8). The clock hardware can easily be designed to enforce the rejection of signals received outside this window by clearing all clock read registers at the beginning of the window and inhibiting the update of the registers at the end of the window (when the interrupt to the processor is generated).

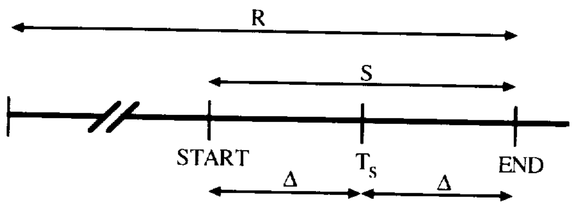


Figure 8. Synchronization window.

Thought must also be given to the clock itself. The clock must be corrected. While at first this may sound trivial, several factors should be considered. A read error equivalent to $1/f_c$ could be induced every time a clock is read or written. Thus, by reading the clock, adding the correction, and writing the new value, two clock ticks of read error can be accumulated. Also, since it takes the processor a finite amount of time to perform the correction, it is possible that additional ticks will be lost during the correction. Correcting the clock by adding the correction is undesirable because clock time will be either "lost" or repeated, and then care must be taken not to "skip over" or "reschedule" an event. Alternative correction methods can be designed that add pulses to or delete pulses from a clock oscillator input, as necessary. As will be seen, this is the method used to adjust the drift rate between the processors. To avoid possible interaction between the application of the correction and the drift rate setting, another correction method was developed.

In the clock circuit tested, the correction is applied by moving the synchronization window (which defines the end of the frame). Normally this would result in larger skews because the clocks will drift for an additional frame before the correction takes effect. This is indeed what would happen. However, during this test no other tasks are scheduled off the clock during the frame. Thus, moving the synchronization window is a way of applying the correction for the purpose of this test. Measurements are not affected because data are only taken during the execution of the synchronization algorithm, and by this time, the correction for the last frame has already been applied. An additional benefit of using this method is that the length of time taken to compute the algorithm (including any interrupt latency) does not affect the experiment. This allowed a great deal of freedom in coding different algorithms, fault models, and data acquisition.

Test augmentation. The clock peripheral design is augmented to allow the adjustment of the oscillator drift rate, the setting of read error, and the simulation of a malicious liar. To adjust the drift rate, the oscillator input of the clock counter is driven by a *pulse deletion* circuit. The pulse deletion circuit has as input a reference oscillator signal (the global clock oscillator) and a 16-bit unsigned integer value. The circuit loads the 16-bit value in a down-counter and deletes a pulse from the reference oscillator signal on overflow. A value of 0 will cause every other pulse to be deleted; a value of 1 will delete every third pulse, and so on, so that the clock frequency is defined as

$$f_c = \frac{v+1}{v+2} f_r \qquad (39)$$

where v is the 16-bit value and f_r is the reference clock frequency. If we let ρ be defined as

$$\rho = \frac{f_r - f_c}{f_r} \qquad (f_c < f_r) \qquad (40)$$

then equation (39) can be written as

$$v = \frac{1-2\rho}{\rho} \qquad (0.0 < \rho \leq 0.5) \qquad (41)$$

For a drift rate of 10^{-5}, $v = 99998$.

Read errors and faulty clock behavior can be programmed by varying the sync strobe time. To present different errors to each of the remote clocks (a form of malicious behavior), a SYNC pulse must be independently generated for each remote clock. Thus, three SYNC register/comparators were used in the final circuit design.

Figure 9 is a block diagram of the clock synchronization peripheral. The circuit is designed for four clocks (one local and three remote) and assumes a dedicated connection to the remote clocks. An oscillator drives a counter of sufficient length to resolve a frame. Five register/comparator blocks define the START window time, SYNC times, and END window time (T_c). The START strobe clears and enables the STORE n registers. The SYNC strobes are broadcast to the remote clocks. The END strobe disables the STORE n registers, interrupts the processor, and clears the clock (counter), beginning a new frame. Three remote clock strobes are gated through the enable circuitry to the STORE n registers. On receipt of a synchronization strobe, the current clock value is latched into the associated STORE n register.

Experiment Environment

The clock peripherals were installed on an existing fault-tolerant processor (FTP) test-bed (ref. 11). The FTP is hosted from a VAX computer through a dual port memory. In addition, each channel of the quad FTP has an additional dual port memory channel to separate VAX computers. These channels were dedicated to data acquisition. A sixth VAX computer with a windowing interface was used to control the experiment. The FTP is a tightly coupled computer. Initial skew is then easily controlled from the base skew of $\delta_0 = 0$ provided by the FTP. The synchronization algorithm is loaded into FTP RAM and configured for the test trial. The FTP operating system is then started from ROM. After the FTP stabilizes, control is passed to the synchronization algorithm and the FTP clock synchronization is disabled.

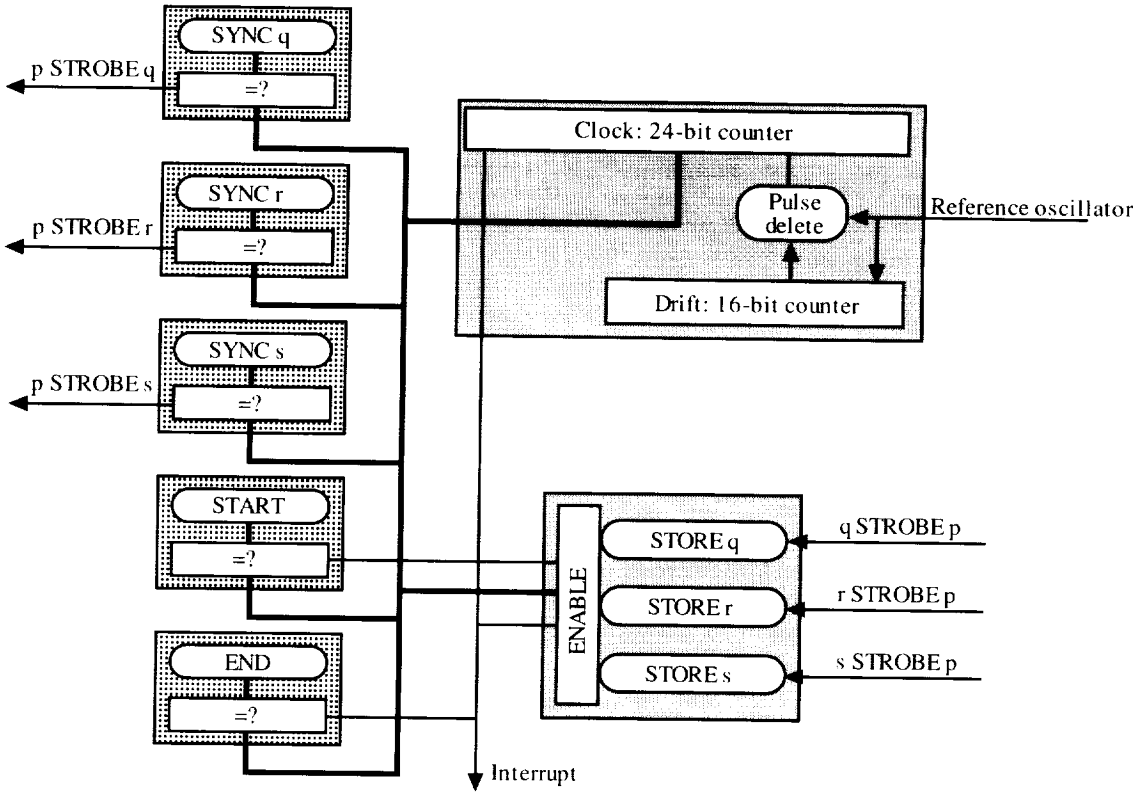


Figure 9. Detail of clock synchronization peripheral.

Another component of the experiment environment is the global clock. The global clock has a base frequency of 2 MHz and a resolution of 32 bits. The output of the global clock can be read by each channel and is assumed to be real time. To establish the global clock as real time, its 2-MHz base frequency is fed to the clock synchronization peripherals as the reference frequency. Thus, in the absence of any programmed drift rate, the clock synchronization peripherals are perfectly synchronized.

Results

Several tests were run to verify the functionality of the system. The following runs were made with the synchronization algorithm disabled:

1. ($m = 0, \rho = 0, \varepsilon = 0$, and $\delta_0 = 0$) to test the global clock
2. ($m = 0, \rho > 0, \varepsilon = 0$, and $\delta_0 = 0$) to test drift rate circuits
3. ($m = 0, \rho = 0, \varepsilon = 0$, and $\delta_0 > 0$) to test setting initial skew
4. ($m = 0, \rho = 0, \varepsilon > 0$, and $\delta_0 = 0$) to test setting the read error

With the synchronization algorithm enabled, several tests were run with $\delta_0 > 0$ and $\rho > 0$, and it was found that equation (16), the $i = 0$ synchronization constraint, held. The next several sections present the results of testing the ICCSA and the Midpoint Algorithm.

The ICCSA. In reference 6, six constraints are listed that must be met if the bounding theorem is to hold for a clock synchronization system executing the ICCSA (see table I). These constraints include the skew bounds (C5 and C6), the maximum perceived clock skew Δ (C4), the maximum clock correction Σ (C3), the minimum time allocated to the synchronization process S (C2), and the minimum length of the synchronization frame R (C1). A synchronization subsystem based on these constraints must have the property that a processor can read a remote clock at a time when the remote processor is not executing the synchronization process. That is, the remote clock must be accessible for external reads outside the scope of its own synchronization process. This is clearly not the case with the design used in this test.

Because a remote clock is read with the cooperation of the remote synchronization process, the synchronization windows must allocate adequate time before and after the synchronization time T_s in order to be sure of capturing all good clocks. This time is at least $\delta + \varepsilon$. In these tests, the window was set at 2 times the maximum perceived clock skew Δ, with the synchronization time T_s in the center of the window (see fig. 8). Thus, the period R is determined by the END window register. The START window register is set to END $- 2\Delta$ and the SYNC registers are set to END $- \Delta$.

Table I. Constraints for Old Theory ICCSA

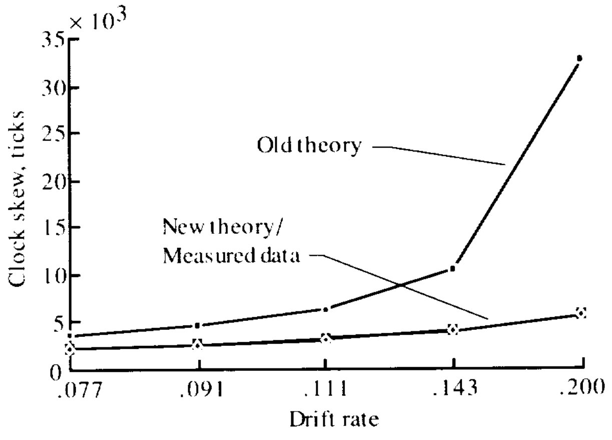
Constraint definition	Constraint relation
C1: minimum period time	$R \geq 3S$
C2: minimum algorithm time	$S \geq \Sigma$
C3: maximum correction	$\Sigma \geq \Delta$
C4: maximum perceived skew	$\Delta \geq \delta + \varepsilon + \frac{\rho_M}{2} S$
C5: maximum skew	$\delta \geq \delta_0 + \rho_M R$
C6: maximum skew	$\delta \geq 2\varepsilon + \rho_M(2S + \Delta) + \frac{2m}{n - m}\Delta$ $+ \left(\frac{n}{n - m}\right) \rho_M (R + \Sigma)$

Table II. Constraints for New Theory ICCSA

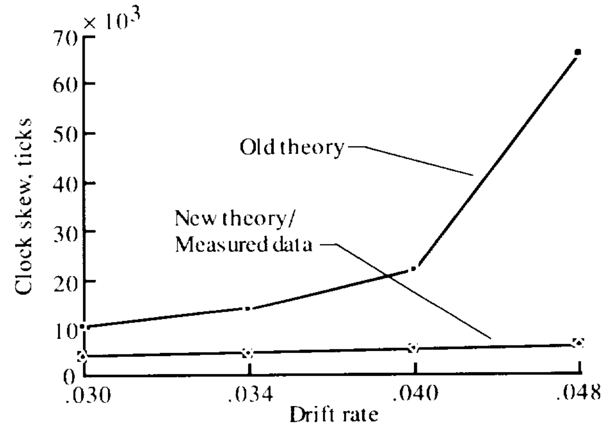
Constraint definition	Constraint relation
C1: minimum period time	$R \geq S + \Sigma$
C2: minimum algorithm time	$S \geq 2\Delta$
C3: maximum correction	$\Sigma \geq \left(\frac{n - 1}{n}\right) \Delta$
C4: maximum perceived skew	$\Delta \geq \delta + \varepsilon + \frac{\rho_M}{2} \Delta$
C5: maximum skew	$\delta \geq \delta_0 + \rho_M R$
C6: maximum skew	$\delta \geq \frac{2(n - 1 - m)}{n - m} \varepsilon + \rho_M \Delta$ $+ \frac{2m}{n - m} \Delta + \left(\frac{n}{n - m}\right) \rho_M R$

The constraints as defined for these tests are listed in table II. The only expression that remains equivalent to table I is C5. The difference in C4 may be due to the difference in S as described in the previous paragraph; C3 defines the maximum correction possible if all $n - 1$ clocks return a difference of Δ; C2 comes directly from the above discussion. Finally, R must be at least as big as S, with room for a correction.

Figure 10 shows, for one series of tests, the bound for the old theory (table I, C6), the bound as derived in this paper (table II, C6), and the actual data. These plots are of maximum clock skew (in ticks) versus drift rate. The data were taken at large drift rates with a constant read error of 200. Figure 10(a) displays zero-fault-tolerant performance ($m = 0$), and figure 10(b), single-fault-tolerant performance ($m = 1$). The bound as derived in this paper exactly predicts the performance of the result.



(a) $m = 0$.



(b) $m = 1$.

Figure 10. ICCSA test results.

The Midpoint Algorithm. A theory based on the Midpoint Algorithm was derived in reference 7 and interpreted in reference 2. Table III lists the constraints for the old theory in terms of the symbols used in this paper (see appendix C). Table IV contains the constraints for the theory for the Midpoint Algorithm as derived in this paper. The synchronization process was identical to the ICCSA with the exception that the Midpoint Algorithm was executed at T_c. Figure 11 plots the clock skew bound predicted by the old theory, the theory derived in this paper, and the actual measured results versus drift rate. As can be seen, the measured clock skew is well below that predicted by the new theory. This is not due to an inaccuracy in the theory, but to an inability to replicate worst-case conditions with the clock subsystem. This phenomenon will be explained in more detail in the section Simulating a Malicious Liar.

Case Studies

The parameters used in the verification tests are obviously far worse than can be expected in an actual system. However, now that the theory has been verified under these extreme conditions, it is reasonable to ask what level of performance can be expected under nominal conditions. The case studies listed in table V were generated to probe this area. The case studies deal primarily with read error and synchronization period, as these are the most significant contributors to the clock skew.

A read error occurs every time a digital clock is read. It is believed that the minimum read error that will be obtainable in most synchronization systems is 1 tick. This tick of read error is added when, as is the case with the subject clock subsystem, the local clock is read in response to the strobe generated by the remote clock. In this case the remote clock is not actually *read*, but generates an event signal that, by definition, occurs at clock time T_s and, therefore, does not include an error component. A similar situation would exist if the remote clock were to be read in response to a request from the local clock (given that there were no other overhead). Case 1 covers this best-case situation.

Table III. Constraints for Old Theory—Midpoint Algorithm

Constraint definition	Constraint relation
C1: minimum period time	$R > 3\Delta + \frac{\rho_M}{2}\delta_0$
C1a: required lower bound on δ	$\delta \geq \frac{4\varepsilon\left(1 - \frac{\rho_M^2}{4}\right) + 2\rho_M(5\Delta + \delta_0) + 2\rho_M^2\delta_0}{\rho_M^2 + 1}$
C2: minimum algorithm time	$S \geq \Delta$
C3: maximum correction	$\Sigma \geq \frac{\delta}{4} + \Delta$
C4: maximum perceived skew	$\Delta \geq \delta + \varepsilon + \frac{\rho_M}{2}\Delta$
C5: maximum skew	Assume C6 dominates
C6: maximum skew	$\delta \geq \frac{4\varepsilon\left(1 - \frac{\rho_M^2}{4}\right) + 2\rho_M(2\Delta + \delta_0 + R) + \rho_M^2\delta_0}{\rho_M^2 + 1}$

Table IV. Constraints for New Theory—Midpoint Algorithm

Constraint definition	Constraint relation
C1: minimum period time	$R \geq S + \Sigma$
C2: minimum algorithm time	$S \geq \Delta$
C3: maximum correction	$\Sigma \geq \frac{\delta}{4} + \Delta$
C4: maximum perceived skew	$\Delta \geq \delta + \varepsilon + \frac{\rho_M}{2}S$
C5: maximum skew	$\delta \geq \delta_0 + \rho_M R$
C6: maximum skew	$\delta \geq 4\varepsilon + 2\rho_M\Delta + 2\rho_M R$

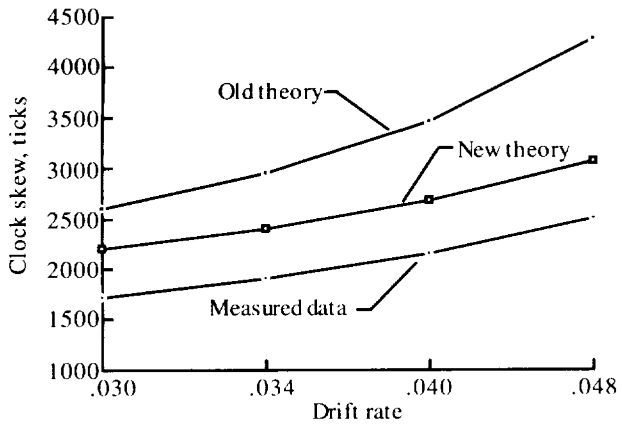


Figure 11. Midpoint Algorithm test results with $m = 1$.

Table V. Case Study Parameters

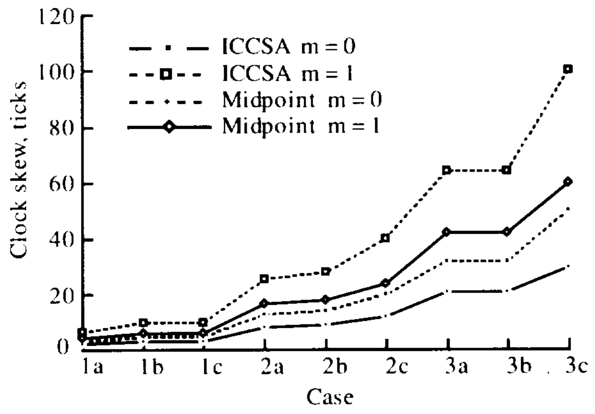
Case	Drift rate, ρ	Period, R, ticks	ρR, ticks/period	Read error, ε, ticks
1a	1.00×10^{-5}	1.00×10^{4}	0.1	1
1b	|	1.00×10^{5}	1.0	1
1c	|	1.00×10^{5}	1.0	1
2a	|	4.00×10^{4}	.4	4
2b	|	1.00×10^{5}	1.0	4
2c	|	4.00×10^{5}	4.0	4
3a	|	1.00×10^{5}	1.0	10
3b	|	1.00×10^{5}	1.0	10
3c	↓	1.00×10^{6}	10.0	10

If both the local clock and the remote clock are read in response to asynchronous events generated by the processor, then 2 ticks of error would be added to a clock read. Similarly, 2 ticks of read error can also be added when a clock is corrected. This is again due primarily to the asynchronous nature of clock reads and writes. If the clock correction circuitry is designed properly, this error will not be incurred. Case 2 covers the situation when the read error ε is 4, with 2 ticks added during clock read and clock correction.

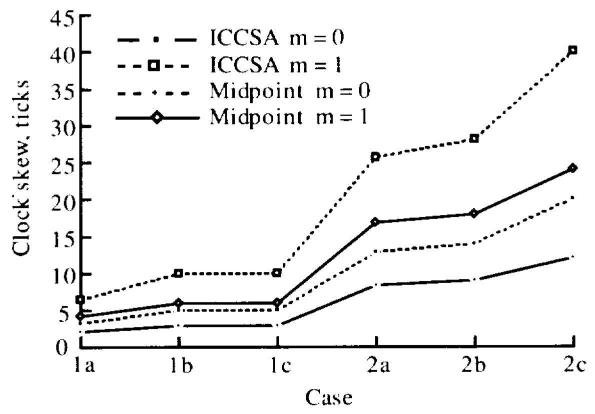
To include a somewhat less than optimal situation, the read error is set to 10 in case 3.

Each case consists of three subcases where the drift rate is set so that the accumulated drift over one period is equal to 1/10 of the read error in subcase "a," 1.0 tick in subcase "b," and the entire read error in subcase "c." This leads to two redundant cases (1c and 3b).

Figure 12(a) is a plot of all three cases. Figure 12(b) plots cases 1 and 2, which represent best-case conditions. Data are plotted for both the ICCSA (dashed lines) and the Midpoint Algorithm (solid lines) and for both zero-fault-tolerant (filled symbols) and single-fault-tolerant cases (empty symbols).



(a) Cases 1, 2, and 3.



(b) Cases 1 and 2.

Figure 12. Case study results.

Discussion of Results

The results of this study span a broad spectrum of subject matter including clock algorithm performance, design methodology, and techniques of worst-case testing. The following sections address these issues.

Clock Algorithm Performance

As can be seen from comparing the fault-free and single-fault cases in figures 12(a) and 12(b), a performance penalty of 100 percent is paid to protect the system from faults. It is interesting to note that this penalty is the same for both algorithms. If a clock skew dead band is made part of every communications exchange, then designers must consider whether they are willing to pay this penalty to protect the system from a rare form of malicious behavior.

The equations for the clock skew upper bound suggest that the component of clock skew due to actual drift (ρR) can be reduced to an insignificant level if R is made small enough. This is not thought to be possible, since, in the absence of read error, no correction will be made for a series of intervals until a significant skew has accumulated. A correction will then be made. This was in fact observed indirectly. Direct observation was not possible because our system had 1 tick of read error, minimum.

The indirect observation was made by first taking one data set with zero additional read error and zero drift rate. What is observed is the minimum read error of the system. This was done for several thousand clock readings, with none exceeding ± 1 tick. To observe the effect of $\rho R < 1$, the same system was then run with $\rho R = 0.1$. Within this series, occasional readings of ± 2 were observed, thus supporting the conjecture that the ρR term actually contributes an amount equivalent to the function *ceiling*(ρR).

The Midpoint Algorithm outperforms the ICCSA and is the clear choice. Remembering that the "a" series subcases are hypothetical with $\rho R < 1$, the next best design is case 1b ($\varepsilon = 1, \rho R = 1$), which yields a single-fault-tolerant skew bound of 6 ticks. While this kind of performance is possible over dedicated links, it may not be possible to design a general-purpose communication protocol that can support both efficient transfer of normal traffic and very low read error.

If it is necessary to allow for greater read error, as represented by case 3, the designer has a wider choice in selecting the synchronization period. In this case, the use of a minimum synchronization period (i.e., with $\rho R = 1$) may yield only marginally tighter clock skews because the read error dominates. The frequent synchronizations may produce more overhead on the communications channel than is saved by virtue of the resultant tighter clock skews.

Design Methodology

One of the areas in which clock synchronization is used is highly reliable fault-tolerant architectures such as those in military and commercial aircraft. The high reliability requirements put on these designs (probability of failure $= 10^{-9}$ per mission) preclude testing as a means of validating that this requirement has been met. One of the methods that has been suggested for this purpose is *formal verification.* A formal verification methodology would entail the use of a specification language and the construction of a hierarchical theory written in that language that could be proven to show that the final design meets the highest level specification. Automated *theorem provers* are often used to facilitate this task. A good example of this method is HDM (ref. 12) as used on SIFT. Most recently this has matured to EHDM, which was used by Rushby (ref. 6) to rederive the clock theory originally invented by Lamport and Melliar-Smith. In reference 6, Rushby reports that the rigor enforced by the use of the theorem provers led to the uncovering of several inconsistencies in the original, hand-derived theory.

The purpose of *experimental verification* as reported in this paper was to demonstrate that the formal theory was indeed correct. What was found was that although the theory was correct in that it predicted a bound that was never violated, the bound was only a bound and not a model for the actual circuit performance. With the insight gained by experimentally observing the behavior of the circuit, it was possible to derive a more accurate theory. Thus, although testing cannot be relied upon to verify highly reliable components, it becomes an integral part of deriving the theory, which can then be used to predict the performance of the circuit into the unobservable regions. While this may sound obvious to those who have practiced such techniques, it has been observed that individuals tend to be heavily biased toward either the "design and debug" or "theorize and prove" camps.

Figure 13 is an attempt to illustrate an optimal design methodology. The two axes delineate time spent testing and theorizing. A vector **DMV** is drawn whose length represents design optimality. It is proposed that the optimality is directly proportional to the correctness of a design and inversely proportional to its cost. The locus of points traced by this vector suggests that if too much emphasis is placed on either testing or theory, design optimality suffers and that the optimum design is reached by applying those techniques best suited for the particular problem. As demonstrated in this work, verification of predicted values of physical quantities is well suited to testing. Testing will also provide behavioral insight, which aids in the construction of provable and realizable theory. As will be seen in the next section, testing cannot be relied upon to quantify worst-case behavior.

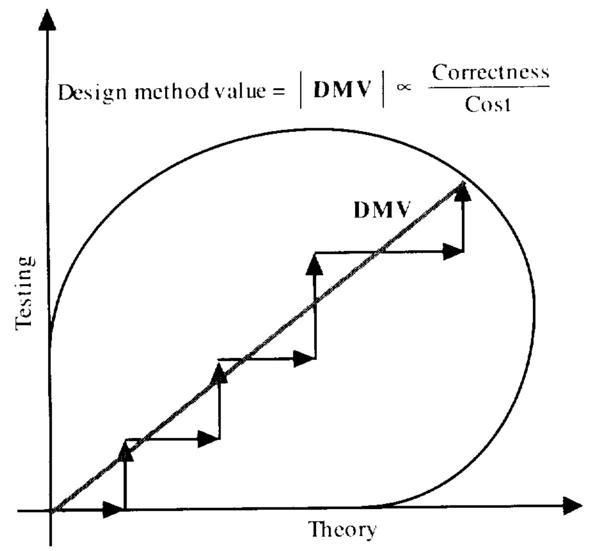


Figure 13. Design method value.

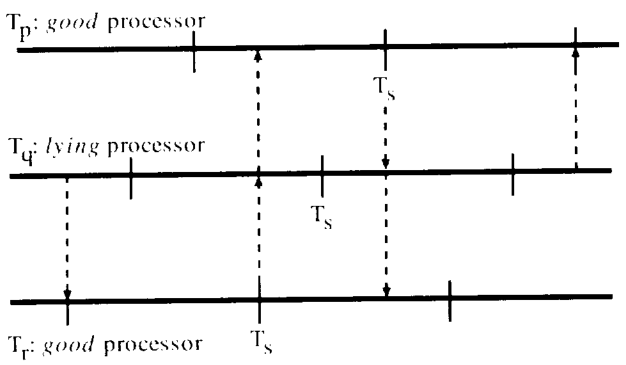


Figure 14. Anticipated malicious liar behavior.

Simulating a Malicious Liar

To experimentally verify the clock theory, special circuits were added to the clock peripheral circuitry to enable the simulation of malicious faults (see the section "Test augmentation"). During testing of the ICCSA, the worst-case behavior of a lying clock was more difficult to simulate than originally anticipated, and the special circuitry could not be used to simulate worst-case conditions without great difficulty. Moreover, for the Midpoint Algorithm, worst-case conditions could not be simulated at all.

Figure 14 shows the faulty behavior that was assumed during the design of the test equipment. The figure illustrates the time line of three processors p, q, and r, with p and r being good processors and q being a lying processor. If p is a slow processor with respect to r, then q would send a synchronization signal to p just prior to the end of the synchronization window to give p the perception that it was a good deal faster than q and thus cause p to apply a correction that would slow its clock even further. Conversely, q would signal r at the beginning of the window and cause r to apply a correction that would speed up its clock.

In practice, the difficulty with doing this is that although it is possible to anticipate the beginning and ending window times for r and p with respect to q for the first frame, it was observed that worst-case skew is not obtained until several frames later. This behavior is illustrated in figure 15. Consider the case in which processor p uses the ICCSA. Processor p will read a clock difference of Δ from q in frame 1. Processor p uses this value as part of the averaging process to compute the correction. The correction computed by processor p will thus have an error of $\Delta/4$ (for four processors). Processor r, on the other hand, will apply a correction with an equal but opposite error with the result that the synchronization windows of p and r have been driven $\Delta/2$ farther apart. Thus, for q to again send worst-case synchronization signals, it must now take this additional skew into account, as illustrated by the second frame in the figure. The correction error would then become $(\Delta + \Delta/4)/4$. The correction is then increasing by amount $\Delta/4^k$, where k is the frame number. The skew between p and r would increase until the additional error becomes insignificant, i.e., $\Delta < 4^k$. This typically took five frames when large drift rates made large synchronization windows necessary.

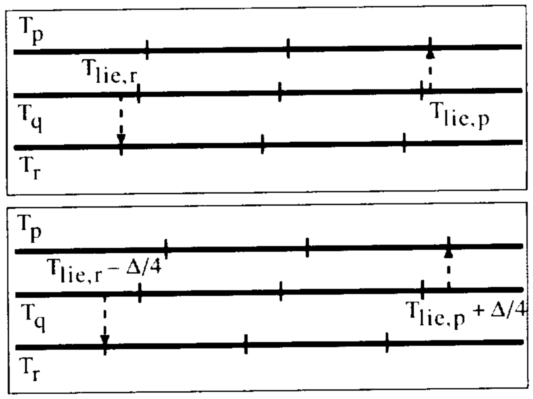


Figure 15. Observed malicious liar behavior.

It was decided, after having observed this behavior, to model the malicious behavior from the perspective of the good processors instead of creating the erroneous signal on the faulty processor. This was done by providing the synchronization algorithm with a parameter that indicated which remote clock was to be considered a liar and in which direction it was lying. The good processor then substituted its START or END window value for the actual reading of the faulty clock, thus simulating the effect described above.

Worst-case conditions could not be simulated with the Midpoint Algorithm because of the lack of sufficient processors to create the necessary conditions. Worst-case conditions are a combination of maximum drift, maximum read error, and the presence of a malicious liar. In the Midpoint Algorithm, the two outlying clock differences are discarded and the remaining two averaged (for four processors). When a malicious liar is present and behaves as described above, it will cause the fastest and slowest clocks to include their clock difference readings (0) in the correction computation. Normally, the fastest and slowest clocks would be at the extremes and not be used. The "self" clock readings do not contain any read error, so that the worst-case skew is not achieved. In a system of five or more clocks, it would have been possible to arrange the parameters to create worst-case conditions.

In conclusion, testing cannot be relied upon to create worst-case behavior. The complex interactions often confound cursory analysis; the result is that something other than worst case may be observed, with the danger then that the system will be designed around these misleading specifications. Developing a theory that predicts worst case provides a checking mechanism that when the theory prediction does not match the observation, immediately raises the question of which is at fault. For a highly reliable design, these kinds of discrepancies must be known and resolved.

Concluding Remarks

New theory has been developed and experimentally verified for the Interactive Convergence Clock Synchronization Algorithm and the Midpoint Algorithm. The Midpoint Algorithm is capable of achieving tighter synchronization than the Interactive Convergence Clock Synchronization Algorithm. Both algorithms suffer a 100-percent penalty to protect against one fault. The new theory outperforms existing theory that was developed without the benefit of the insight gained during experimental verification. However, it is not adequate to rely on testing procedures to uncover worst-case behavior. Testing and theory go hand in hand to produce optimal designs. This is especially true for highly reliable systems.

NASA Langley Research Center
Hampton, VA 23665-5225
May 5, 1992

Appendix A

Proving Equations (10), (12), (13), and (14)

Proving Equation (10)

To prove equation (10), that is,

$$\Delta_{qp} = -\delta_{qp}(T_s) \pm \varepsilon + \rho_p \Delta_{qp} \tag{10}$$

we start with the definition of Δ_{qp}, equation (8), which is

$$\Delta_{qp} = T_{qp} - T_s \tag{8}$$

and using equation (9) to expand T_{qp} and expressing T_s as the value of clock p at a real time when clock p reads T_s, we get

$$\Delta_{qp} = T_p(t_q(T_s)) - T_p(t_p(T_s))$$

Using equation (3) to expand the clock functions T_p and realizing that the second term incurs no read error, we have

$$\Delta_{qp} = \left[\frac{t_q(T_s) \pm \varepsilon - t_{0p}}{1-\rho_p}\right] - \left[\frac{t_p(T_s) - t_{0p}}{1-\rho_p}\right]$$

Finally, combining terms and using equation (7), we get

$$\begin{aligned}\Delta_{qp} &= \frac{-\delta_{qp}(T_s) \pm \varepsilon}{1-\rho_p} \\ &= -\delta_{qp}(T_s) \pm \varepsilon + \rho_p \Delta_{qp}\end{aligned}$$

Proving Equation (12)

To prove equation (12), that is,

$$\delta_{qp}(T+C) = \delta_{qp}(T) + \rho_{qp}C \tag{12}$$

we start with equation (7) and substitute equation (1) as follows:

$$\begin{aligned}\delta_{qp}(T+C) &= t_p(T+C) - t_q(T+C) \\ &= \left[(1-\rho_p)(T+C) + t_{0p}\right] \\ &\quad - \left[(1-\rho_q)(T+C) + t_{0q}\right] \\ &= \delta_{qp}(T) + (\rho_q - \rho_p)C \\ &= \delta_{qp}(T) + \rho_{qp}C\end{aligned}$$

Proving Equation (13)

To prove equation (13), that is,

$$\delta_{qp}(T) = \rho_{qp}(T - T_c) + \delta_{qp}(T_c) \tag{13}$$

we rearrange equation (12) and substitute $C = -(T - T_c)$.

Proving Equation (14)

To prove equation (14), that is,

$$\delta_{rq}(T) - \delta_{rp}(T) = \delta_{pq}(T) \tag{14}$$

we use equation (7) and write

$$\begin{aligned}\delta_{rq}(T) - \delta_{rp}(T) &= \left[t_q(T) - t_r(T)\right] - \left[t_p(T) - t_r(T)\right] \\ &= \left[t_q(T) - t_p(T)\right] \\ &= \delta_{pq}(T) = -\delta_{qp}(T)\end{aligned}$$

Appendix B

The Expansion of $\chi_p - \chi_q$

We expand the term $\chi_p - \chi_q$:

$$\chi_p - \chi_q = \frac{1}{n}\sum_{r=1}^{n-2-m}(\Delta_{rp} - \Delta_{rq}) + \frac{1}{n}(\Delta_{pp} - \Delta_{pq})$$

$$+ \frac{1}{n}(\Delta_{qp} - \Delta_{qq}) + \frac{m}{n}(\nabla_p - \nabla_q) \qquad \text{(B1)}$$

This expression contains four terms that can be considered in three groups. The first term represents the $(n - 2 - m)$ *good* processors. The second and third terms represent *good* processors, one of which is a *local* clock. The third term represents readings taken from *bad* processors.

The *Good* Processors

We will first reduce the term $\Delta_{rp} - \Delta_{rq}$ using equation (10).

$$\Delta_{rp} - \Delta_{rq} = \left[-\delta_{rp}(T_s) \pm \varepsilon + \rho_p\Delta_{rp}\right]$$

$$- \left[-\delta_{rq}(T_s) \pm \varepsilon + \rho_q\Delta_{rq}\right]$$

$$\Delta_{rp} - \Delta_{rq} \leq \left[\delta_{rq}(T_s) - \delta_{rp}(T_s)\right] + 2\varepsilon$$

$$+ (\rho_p\Delta_{rp} - \rho_q\Delta_{rq})$$

$$\leq \left[\delta_{rq}(T_s) - \delta_{rp}(T_s)\right] + 2\varepsilon \qquad \text{(B2)}$$

Here, as before, the difference between the $\rho\Delta$ values is ignored. These results are replaced in the sum (B1) and simplified as follows:

$$\frac{1}{n}\sum_{r=1}^{n-2-m}(\Delta_{rp} - \Delta_{rq}) \leq \frac{1}{n}\sum_{r=1}^{n-2-m}$$

$$\times \ \{[\delta_{rq}(T_s) - \delta_{rp}(T_s)] + 2\varepsilon\}$$

$$\frac{1}{n}\sum_{r=1}^{n-2-m}(\Delta_{rp} - \Delta_{rq}) \leq \frac{1}{n}\sum_{r=1}^{n-2-m}$$

$$\times \ [\delta_{rq}(T_s) - \delta_{rp}(T_s)] + \frac{2(n-2-m)}{n}\varepsilon \qquad \text{(B3)}$$

The *Local* Processors

Taking the two terms that include *local* processor readings, we write

$$\frac{1}{n}(\Delta_{pp} - \Delta_{pq}) + \frac{1}{n}(\Delta_{qp} - \Delta_{qq})$$

$$= \frac{1}{n}\{[-\delta_{pp}(T_s)] - [-\delta_{pq}(T_s) \pm \varepsilon + \rho_q\Delta_{pq}]\}$$

$$+ \frac{1}{n}\{[-\delta_{qp}(T_s) \pm \varepsilon + \rho_p\Delta_{qp}] - [-\delta_{qq}(T_s)]\}$$

$$= \frac{1}{n}\{[-\delta_{pp}(T_s)] - [-\delta_{pq}(T_s) \pm \varepsilon]\}$$

$$+ \frac{1}{n}\{[-\delta_{qp}(T_s) \pm \varepsilon] - [-\delta_{qq}(T_s)]\}$$

$$+ \frac{\rho_p}{n}\Delta_{qp} - \frac{\rho_q}{n}\Delta_{pq}$$

Finally, ignoring the difference between the $\rho\Delta$ terms we obtain

$$\frac{1}{n}(\Delta_{pp} - \Delta_{pq}) + \frac{1}{n}(\Delta_{qp} - \Delta_{qq})$$

$$= \frac{1}{n}\{[-\delta_{pp}(T_s)] - [-\delta_{pq}(T_s)]\}$$

$$+ \frac{1}{n}\{[-\delta_{qp}(T_s)] - [-\delta_{qq}(T_s)]\} + \frac{2}{n}\varepsilon \qquad \text{(B4)}$$

Good Plus *Local* Processors

Combining equations (B3) and (B4) by replacing the first two terms on the right-hand side of equation (B4) in the summation of equation (B3), we get

$$Good + Local = \frac{1}{n}\sum_{r=1}^{n-m}\left[\delta_{rq}(T_s) - \delta_{rp}(T_s)\right]$$

$$+ \frac{2(n-1-m}{n}\varepsilon \qquad \text{(B5)}$$

Taking a closer look at the expression within the summation we have, with $T_c = T_s + \Delta$,

$$\delta_{rq}(T_s) - \delta_{rp}(T_s) = \delta_{rq}(T_c - \Delta) - \delta_{rp}(T_c - \Delta)$$

Now using equations (12) and (5), we get

$$\delta_{rq}(T_s) - \delta_{rp}(T_s) = \delta_{rq}(T_c) - \delta_{rp}(T_c) + \rho_{qp}\Delta$$

Finally, application of equation (14) yields

$$\delta_{rq}(T_s) - \delta_{rp}(T_s) = -\delta_{qp}(T_c) + \rho_{qp}\Delta \qquad \text{(B6)}$$

Now, substituting equation (B6) into equation (B5),

$$\mathit{Good} + \mathit{Local} = \left(\frac{m-n}{n}\right)\delta_{qp}(T_c) + \frac{2(n-1-m)}{n}\varepsilon + \frac{\rho_M(n-m)}{n}\Delta \qquad \text{(B7)}$$

The *Bad* Processors

Recalling from the ICCSA that all perceived skews are limited to a maximum of Δ, we have

$$\frac{m}{n}\left(\nabla_p - \nabla_q\right) \le \frac{2m}{n}\Delta \qquad \text{(B8)}$$

Good Plus *Local* Plus *Bad* Processors

Using equations (B7) and (B8) in the original expression gives

$$\chi_p - \chi_q \le \left(\frac{m-n}{n}\right)\delta_{qp}(T_c) + \frac{2(n-1-m)}{n}\varepsilon + \frac{\rho_M(n-m)}{n}\Delta + \frac{2m}{n}\Delta$$

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	
2. REPORT DATE	July 1992
3. REPORT TYPE AND DATES COVERED	Technical Paper
4. TITLE AND SUBTITLE	Experimental Validation of Clock Synchronization Algorithms
5. FUNDING NUMBERS	WU 505-64-10-07
6. AUTHOR(S)	Daniel L. Palumbo and R. Lynn Graham
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	NASA Langley Research Center Hampton, VA 23665-5225
8. PERFORMING ORGANIZATION REPORT NUMBER	L-17015
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	National Aeronautics and Space Administration Washington, DC 20546-0001
10. SPONSORING/MONITORING AGENCY REPORT NUMBER	NASA TP-3209
11. SUPPLEMENTARY NOTES	Palumbo: Langley Research Center, Hampton, VA; Graham: PRC Kentron, Inc., Hampton, VA.
12a. DISTRIBUTION/AVAILABILITY STATEMENT	Unclassified Unlimited Subject Category 62
12b. DISTRIBUTION CODE	

13. ABSTRACT *(Maximum 200 words)*

The objective of this work is to validate mathematically derived clock synchronization theories and their associated algorithms through experiment. Two theories are considered, the Interactive Convergence Clock Synchronization Algorithm and the Midpoint Algorithm. Special clock circuitry was designed and built so that several operating conditions and failure modes (including malicious failures) could be tested. Both theories are shown to predict conservative upper bounds (i.e., measured values of clock skew were always less than the theory prediction). Insight gained during experimentation led to alternative derivations of the theories. These new theories accurately predict the behavior of the clock system. It is found that a 100-percent penalty is paid to tolerate worst-case failures. It is also shown that under optimal conditions (with minimum error and no failures) the clock skew can be as much as three clock ticks. Clock skew grows to six clock ticks when failures are present. Finally, it is concluded that one cannot rely solely on test procedures or theoretical analysis to predict worst-case conditions.

14. SUBJECT TERMS	Clock synchronization; Formal methods; Verification; Validation
15. NUMBER OF PAGES	22
16. PRICE CODE	A03
17. SECURITY CLASSIFICATION OF REPORT	Unclassified
18. SECURITY CLASSIFICATION OF THIS PAGE	Unclassified
19. SECURITY CLASSIFICATION OF ABSTRACT	
20. LIMITATION OF ABSTRACT	

NSN 7540-01-280-5500

Standard Form 298(Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

NASA-Langley, 1992