A Provably Correct Design of a Fault-Tolerant Clock Synchronization Circuit

Paul S. Miner Peter A. Padilla Wilfredo Torres

NASA Langley Research Center
Hampton, VA 23665-5225

Abstract

A prototype fault-tolerant clock synchronization system is designed to a proven correct formal specification. The specification is derived from Schneider's general paradigm for Byzantine resilient clock synchronization. One addition to the formal theory is a mechanism for proven recovery from a bounded number of transient faults. A description of a four-clock implementation which satisfies the requirements of the formal theory is presented. In addition, the design provides options for initialization which enable recovery from some correlated transient failures. Extra logic is included to provide experimental control of these options. Simulation results are presented.

1 Introduction

An important function in the design of a fault-tolerant computer system is the synchronization of the clocks of the redundant processing elements. Due to the subtleties involved in reasoning about the behavior of failed components, it is necessary to prove that systems purporting to be fault-tolerant will survive an arbitrary failure. This paper presents a fault-tolerant clock synchronization system designed to a proven correct formal specification.

The circuit is intended to serve as an experimental test-bed for part of the ongoing FLY-BY-LIGHT/POWER-BY-WIRE project. The formal specification for the design is drawn from an extended version [1] of Shankar's mechanical verification [2] of Schneider's generalized Byzantine fault-tolerant clock synchronization protocol [3].

A motivating factor in the design is the ability to recover from transient faults. A proven mechanism for transient recovery is presented. The circuit incorporates an automatic initialization scheme that can be used to trigger recovery from a massive transient upset, provided that no malicious faults remain in the system.

Details of a four clock implementation are discussed. The communications between the clock circuits will be implemented through the use of fiber optic cables. The synchronization period to be used during the implementation will be of 0.8192 ms, with a clock frequency of 10 MHz. Each of the four clocks provides its corresponding processor with the information required to determine the local clock value, and whether or not the clock is synchronized

with the rest of the system. The hardware also provides extra control to perform various experiments and to alter parameters of the design.

In addition, simulation results are presented which confirm predictions of the formal theory. Scenarios studied include: achieving initial synchronization between clocks, resynchronization of a lost clock, and general behavior during upset or in the presence of a faulty clock. The simulation estimates of the times required from power-on to achievement of initial synchronization and to resynchronize a lost clock corroborate the initial analysis of the algorithms and meet the performance criteria and constraints of the theory. The simulation results will provide a series of tests to confirm the hardware implementation's compliance with the specification.

2 Formal Requirements

This section provides an informal description of the theory relevant to this design. A formal description of the theory can be found in [1] and [2]. Although the theory is for a general N clock system, the discussion here focuses on a four clock design. The result guaranteed by the theory is that

Theorem 1 (bounded skew) *For any two logical clocks VC_p and VC_q that are nonfaulty at real time t ,*

$$|VC_p(t) - VC_q(t)| \leq \delta$$

The clocks, VC_p , are mappings from real time to clock time, where real time ranges over the real numbers and clock time ranges over the integers. The intended semantics are for $VC_p(t)$ to be the reading of p 's logical clock at real time t . A mechanically checked proof of this result is presented in [2]. A revised set of conditions sufficient to meet this result are presented in [1].

Synchronization algorithms are generally of the following form:

```
do forever {  
    exchange clock values  
    determine adjustment for this interval  
    determine local time to apply correction  
    when time, apply correction}
```

The logical clocks, VC_p , represent an infinite sequence of interval clocks, IC_p^i , where interval clock i corresponds to the i th pass through the above loop; for the remainder of this paper, i will be called the frame counter. Like VC_p , IC_p^i is a function from real time to clock time. If ADJ_p^i represents the adjustment applied to clock IC_p^i at the end of the i th interval, then we get the following equation:

$$IC_p^{i+1}(t) = IC_p^i(t) + ADJ_p^i$$

In order to compute the adjustment, it is necessary to read the other clocks. The correction is determined from these readings by evaluating a convergence function which returns the desired clock time to begin the next interval. It is necessary to show that the convergence function satisfies certain abstract properties.

2.1 Convergence Function

The convergence function selected for this design is the fault-tolerant midpoint function used in Welch and Lynch's algorithm [4]. For a four clock design, this consists of averaging the second and third readings. To satisfy the formal theory, it is necessary to show that the function satisfies the following properties: translation invariance, precision enhancement, and accuracy preservation.

Translation invariance allows the computation of the convergence function based on values offset from the actual readings. Precision enhancement states that if the readings employed by two nonfaulty clocks are sufficiently similar, then, after applying the correction, the two clocks are sufficiently close to maintain synchronization. Accuracy preservation ensures that the result of evaluating the convergence function is sufficiently close to any reading from a non-faulty clock. In addition to helping bound the separation of clocks, this also bounds the magnitude of the adjustment in any interval. Formal definitions of these properties and mechanically checked proofs that the fault-tolerant midpoint function satisfies them can be found in [5]. Other properties required by the theory are assumptions about physical properties of the system, and a collection of inequalities relating various quantities determined by the implementation.

2.2 Physical Properties and Inequalities

Some of the assumptions required by the theory are physical properties that cannot be established formally. A four clock system can tolerate at most one Byzantine failure. All of the proofs assume that there is never more than one failed clock in the system. Another requirement for a good clock is that its rate of drift from real time be bounded by a small positive constant ρ . For $s \geq t$ and all i , the following is assumed:

$$[(s - t)/(1 + \rho)] \leq IC_p^i(s) - IC_p^i(t) \leq [(1 + \rho)(s - t)].$$

For the purposes of the design presented here, it is assumed that $\rho < 10^{-5}$. Finally, a bound must be established on the

inherent error in reading a remote clock. This design has an estimated reading error of 0.5 ticks. Once again, formal definitions of these requirements can be found in [1].

A number of inequalities relating various aspects of the design must be satisfied. These include requirements that the interval between synchronization events is sufficiently large, and terms determining the tightness of synchrony possible for a particular implementation. For the design presented here, the nominal duration of a synchronization interval, R , is 8192 ticks; this easily satisfies the requirements of the theory. Since $\rho R \leq 0.1$, drift has a minimal effect on reading error. Given a reading error of 0.5 ticks, the theory establishes that this design maintains synchrony within $\delta \leq 11$ ticks. Of course, the above constraints assume that the circuit is already synchronized. It is still necessary to provide a means to attain initial synchronization.

2.3 Initialization

This design obtains its reading of other clocks by computing the difference between the time a signal arrives, and the expected arrival time if the clocks were perfectly synchronized. This allows the exchange of clock values to occur anywhere in the synchronization interval. By setting the expected time for observing synchronization signals at $R/2$ into the current interval, we establish the largest possible window for reading remote clocks. If any three clocks send their signal within slightly less than $R/2$ of each other, then the hypotheses of precision enhancement are satisfied and the clocks will converge to within δ in approximately $\log_2(R/2)$ intervals. During the initialization process, the frame counter is kept at 0. A Byzantine exchange is required to switch to the maintenance protocol. We wish to identify situations in which the clocks do not converge. First, we need to define proper behavior when the circuit observes fewer than three pulses in a given interval. The design provides two options:

Assumed Perfection — pretend all unobserved clocks are observed to be in perfect synchrony, or

End of Interval — pretend that unobserved clocks are observed at the end of the synchronization interval.

Neither of these techniques guarantee initialization; all explored variations possess a pathological 2-2 split, even if no faults are present. Preliminary explorations suggest that the first is most effective for fault-tolerant initialization, but the second provides for provable transient recovery. If we assume only benign failures, a simple time out mechanism added to either of the above approaches can guarantee initial synchronization. It is likely that some combination of the two approaches will be adopted. Simulation and experimentation will be necessary to determine an appropriate approach.

2.4 Transient Recovery

A slight modification of the theory presented in [1] gives sufficient conditions for proven recovery from a bounded number of transient faults. Since this design consists of only four clocks, we only need to recover from a single transiently affected channel. Furthermore, we can assume that the remaining clocks are nonfaulty (else the assumed bound on failed clocks would be exceeded). All that is necessary is to ensure that the recovering clock will eventually observe the remaining three clocks. This is guaranteed to occur within two intervals if the *end of interval* approach is used for unobserved signals. If the synchronization signal includes the frame number for the current interval, a simple majority vote of the received indices will restore the frame counter, the convergence function ensures the rest. A more detailed sketch of this argument can be found in [5].

3 Design Description

The dominant parameter in the formal theory is the inherent error in reading a remote clock. The system requires an efficient communications network to achieve a tight synchronization. Our system will use point-to-point optical communications. The optical transmitter-receiver interface is composed of off-the-shelf components and provides a transmission rate of 125 Mbits/sec. Engineering analysis of the communication network predicts a maximum reading error of 0.5 ticks.

Each clock circuit has to perform several functions, including: keeping a local time, performing the transmissions and receptions of the frame counter values, voting the frame values, computing and applying an adjustment to local time, determining if the system is synchronized, determining if its local time is synchronized with the system, and provide some control lines for the experiments. Also, based on the current frame information and its past states, a clock should be capable of determining what its next state should be: initialization, maintenance or recovery. Figure 1 shows a block diagram of a clock circuit, and an overview of the operation follows.

The local time has two components: the Local Clock value, and the Frame Counter. If i is the current frame and LC is the current value of the Local Clock, then the elapsed time since the beginning of the protocol is $iR + LC$. The Local Clock is built using a 16-bit counter driven by a 10 MHz oscillator. It goes through the Monotonic Clock Logic before being sent to the outputs. The purpose of this logic is to ensure that the local time is a monotonically non-decreasing function of real time. The logic limits the output of the Local Clock to R , so that backward jumps on local time are inhibited.

The value of the frame counter is determined by the synchronization state of the clock. If a clock circuit is synchronized, its frame counter will be incremented by one at the beginning of each new frame. During initialization, the value will be set to zero until synchronization is achieved. In the recovery state, the frame counter will not be incremented and the majority voter will recover the correct value.

Some time before the clock reaches time Q (the transmission time, $Q = R/2$), the Timing logic will send a signal to the transmitters to transmit the frame counter. The delays of transmission and signal propagation on the optic fiber, together with the processing time on the receiving clocks, must be taken into account so that synchronized clocks can be identified. The algorithm also requires it to send a signal to itself (s4 on the block diagram). When the second or third signal is received, the difference between Q and the signal arrival time is captured. These values are used for the computation of the convergence function.

The Timing logic will also control other functions. It will use the Local Clock value and the computed adjustment to identify when the clock is close to the end of the frame. At that time it will disable the receivers to allow the electronics to process the received information before starting the next frame. At the end of the window the Timing Logic will reset the computed adjustment and Signal Arrival Identification logic to prepare the circuit for the next frame. The logic will also tell the Local Clock when it is time start a new frame and reset its value to zero.

As mentioned in Section 2.3, a clock circuit may take one of two actions when it receives less than three signals during a synchronization interval. It will assume either that all signals arrived in perfect synchrony or that missing signals arrived at the end of the frame. The desired action can be selected by setting a switch in the Timing logic.

The Adjustment Computation logic is based on the loop presented in Section 2. The Signal Arrival Identification logic will identify when the second and third signals are received and will pass this information to compute the adjustment. The resulting adjustment will also be used by the Timing logic to control start of the next interval.

The State Determination logic will determine the state of the clock circuit based on the information present at the end of the frame. It uses two counters to determine the time from the moment of arrival of the first signal to that of the third, and from the second to the fourth. At the end of the frame the value on these counters and the magnitude of the computed adjustment will be compared to the maximum allowed skew D ($= 10 \text{ ticks} = 1 \mu\text{s}$ at a 10 MHz frequency).¹ If both counters are greater than D , the system is not synchronized and the clock will go to initialization. When one or both counters and the computed adjustment are less than or equal to D , the clock is synchronized and it will maintain the synchronization. If one counter is less than or equal to D but the computed adjustment is greater, the clock is not synchronized with the rest of the system and it will go to the recovery state. The State Determination logic will decide when the Frame Counter should be incremented and will also provide a signal (OUT_OF_SYNC) to indicate when the local time is not synchronized with the remainder of the system. This enables the processing element to initiate its own recovery process.

¹At this point in the synchronization interval, the skew is less than 10 ticks.

Figure 1: Block Diagram

The Restart Operation logic will use the information coming from the State Determination logic and the Majority Voter to reset the clock circuit when necessary. This logic will also reset the clock circuit after a power-up.

Because of the experimental nature of this implementation, control lines (DAS1 to DAS4 on the block diagram) can be used to perform various experiments. The control lines allow to override the Restart Operation logic so that the circuits can be reset externally, either by using a push-button reset or by computer control. The transmissions and receptions can be enabled or disabled as desired. This allows laboratory simulation of Byzantine failures.

The desired action when a clock circuit does not receive sufficient information to compute an adjustment can be selected by using a switch. Other parameters that can be set by DIP switches include: the frame length (R), the value loaded into the local clock counter at the end of every frame (normally set to zero), the time to transmit information to the other channels (Q), and the maximum allowed skew (D).

Future implementations will include several enhancement to the system. It will incorporate the necessary hardware to perform Byzantine exchanges between the clock circuits since there is the possibility that the system will not tolerate Byzantine behavior during initialization.

4 Simulation and Experiments

During the formal analysis of the algorithm, several conditions appeared to create problems for the initialization algorithm. The conditions always involve a pair of clocks separated by $\approx R/2$. These were explored in the simulation.

The simulation has not been formally verified, but all analytically predicted behavior has been observed in the simulated algorithm's response. Also, no unexpected behavior was observed in the simulation. Full data is available during a simulation run, therefore, any observation can be traced in the data to its cause and history. In addition, more complex conditions can be analyzed using the simulator than is feasible analytically. For example, jitter was added to the clocks behavior and the performance of the synchronization algorithm studied. The simulation results also provide a model of expected behavior for the planned hardware experiments.

The parameters for the simulation results are as follows. The synchronization interval is .8192 ms, a clock tick is 100 ns. For the simulations that included clock drift, the drift rate used was 10^{-3} . The drift was set much higher than what might be expected in reality (10^{-6}) in order to observe the effect in the simulation within a reasonable time. For the simulations that included clock jitter, for each clock tick a random value was selected to simulate a maximum clock jitter of 10ns. Higher values of clock jitter, beside not being realistic for the clock rate simulated,

also created havoc with the algorithm (clocks with random behavior are hard to synchronize).

The plots in Figures 2 through 4 show $IC^i(t) - iR$ (i.e. the value of the Local Clock) for a four clock system. The Local Clock increments by one every clock tick of 100 ns. Since the nominal duration of a synchronization interval is .8192 ms, the number of clock ticks per interval for clock p is $8192 + ADJ_p^i$.

When two or more ideal clocks (i.e., no jitter or drift) are synchronized, the traces of the synchronized clocks merge into one trace. See Figure 2 for an example. There is only one trace per dual since the two clocks in the dual track each other.

4.1 Initialization

The initialization algorithm (referred to as the algorithm below) is intended to proceed from power-on to synchronize the clock system. The initialization scenarios are varied since each channel will start at different times. To simulate different scenarios, the start times for each clock can be varied. In addition, the simulator allows the state of a running clock to be perturbed. This allows simulation of transient faults and massive upset.

Several scenarios are considered as the most difficult to handle by the algorithm. The results for the worst such scenario are shown in Figure 2. The scenario involves splitting the clocks from a quad into duals. This scenario, in combination with properly selected read errors, prevents the algorithm from achieving synchronization since, due to read errors and the particulars of the algorithm, the algorithm fails to compute a correction for any clock. As shown in the figure, the dual split continues with no change or correction by the algorithm. The results of Figure 2 were obtained for perfect clocks with no jitter or drift.

For this scenario to continue indefinitely, the starting offsets between clocks must be a set of very particular values. Also, as mentioned above, the required read errors that allow this situation to continue are very specific and unlikely to be encountered in a real application. In addition, as will be made clear below, perfect clocks are required to allow the dual split to continue indefinitely. In summary, this scenario is not one likely to be encountered in nature. Still the results are of obvious theoretical interest.

In Figure 3, the clocks' drift and jitter are restored. As shown, synchronization was achieved within 4 ms. For this scenario, the shorter synchronization times occur when the drift rates between the clocks are large, such that none of the clocks stay in the same relative position with respect to the other clocks. For example, increasing the drift rate to 10^{-2} results in even better performance. Expected clock drifts range below 10^{-6} , therefore, if the clocks ever encountered this scenario, the extrapolated synchronization time is larger than 40 sec.

4.2 Upset Response

It is of interest to study the response of the synchronization algorithm to a random upset. An upset is simulated

by adding random values to the Local Clocks. The time at which the clocks are randomly perturbed is selected by the experimenter. The clocks can be started either synchronized or unsynchronized.

Typical results when the clocks are started synchronized and then subjected to an upset are shown in Figure 4. In Figure 4, all the clocks converged at 4 ms, but minimum skews were not achieved until 10 ms.

5 Concluding Remarks

The development of this synchronization system was an exercise in the application of formal methods in the development of a real design. This design attempts to use the same basic algorithm for proven initialization, transient recovery, and maintenance of synchronization. With the exception of some rare pathological initialization scenarios, these goals were met. The prototype design includes logic to provide experimental control. This will allow for demonstration of Byzantine fault-tolerance and transient fault recovery. In addition, the design allows experiments to explore different initialization approaches.

The pathological initialization scenarios were explored via simulation. These results show the advantage of using simulations to understand complex behavior. While formal methods had not addressed clock jitter, and drift rates were considered undesirable, simulation shows that a moderate amount of drift is beneficial. These results will be used to focus experimentation with the hardware.

Although formal methods provide added assurance that a design is correct, they will not completely replace conventional approaches to hardware design and development. They can serve to focus on appropriate aspects of the system to explore with simulation and experimentation.

References

- [1] P. S. Miner, "An extension to Schneider's general paradigm for fault-tolerant clock synchronization," Technical Memorandum 107634, NASA, Langley Research Center, Hampton, VA, June 1992.
- [2] N. Shankar, "Mechanical verification of a schematic Byzantine clock synchronization algorithm," Contractor Report 4386, NASA, July 1991. Author's affiliation: SRI International, Computer Science Laboratory, Menlo Park, CA.
- [3] F. B. Schneider, "Understanding protocols for Byzantine clock synchronization," Tech. Rep. 87-859, Department of Computer Science, Cornell University, Ithaca, NY, Aug. 1987.
- [4] J. L. Welch and N. Lynch, "A new fault-tolerant algorithm for clock synchronization," *Information and Computation*, vol. 77, pp. 1-36, Apr. 1988.
- [5] P. S. Miner, "A verified design of a fault-tolerant clock synchronization circuit: Preliminary investigations," Technical Memorandum 107568, NASA, Langley Research Center, Hampton, VA, Mar. 1992.

Figure 2: Initialization pathological case with no clock jitter or drift.

Figure 3: Initialization pathological case with clock jitter and drift.

Figure 4: Massive upset simulation

PAPER WITHDRAWN