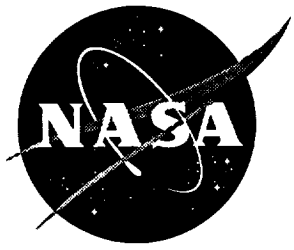
NASA Technical Memorandum 109176



An Optimized Implementation of a Fault-Tolerant Clock Synchronization Circuit

Wilfredo Torres-Pomales
Langley Research Center, Hampton, Virginia

February 1995

National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23681-0001

Abstract

A fault-tolerant clock synchronization circuit was designed and tested. A comparison to a previous design and the procedure followed to achieve the current optimization are included. The report also includes a description of the system and the results of tests performed to study the synchronization and fault-tolerance characteristics of the implementation.

Contents

1. Introduction

Present reliability requirements of critical systems necessitate failure rates less than those of available digital devices. Fault-tolerant architectures are used to achieve the reliability requirements through redundancy. These fault-tolerant systems are designed to maintain correct operation in the presence of a bounded number of faults. In a synchronous redundant computing system, the computers are assumed to be synchronized through some mechanism implemented in hardware or software. Clock synchronization provides coordinated action among redundant processing elements, and that means maintenance of synchronization in the presence of faults is crucial. This report presents an optimized hardware implementation of a fault-tolerant clock synchronization algorithm.

The system presented here is an optimization to the implementation described in [1], which was a realization of the system proposed by Paul S. Miner in [2]. The goal was to design a system that performed the same functions as the original, but it had to use fewer logic gates and be capable of operating at clock frequencies in excess of 33 MHz, given the appropriate hardware. This paper explains how this goal was achieved. In [3], Miner and Johnson present a formal analysis of the optimized implementation of the synchronization algorithm. The following sections describe the new system in detail and the results of some tests are presented. Also, the appendix lists some ideas that could be used to further improve the clock synchronization system.

2. System Description

The clock synchronization system has four nodes and uses a fully connected point-to-point communications network. It is capable of tolerating a maximum of one transient or permanent fault. Each node in the system has its view of what the correct time is. This is called the Local Time of the node. The synchronization algorithm requires the nodes to operate in frames or cycles, and to exchange their Local Times once during each frame. With the information received, the nodes compute and apply adjustments to their Local Times to maintain the system synchronized within a maximum allowed skew (D). In this implementation, the Local Time of each node has two components: the Frame Number (i), and the Local Clock (LC). The Frame Number counts the number of frames since the system first achieved synchronization. The Local Clock is the time elapsed in each frame. If the nominal frame length is R, the Local Time is given by $(i\ R) + LC$.

The nodes in this implementation do not transmit both their Local Clock and Frame Number values. Instead, each node sends its Frame Number to the other nodes some time before its Local Clock reaches time $LC = Q$ (the time to transmit, $Q = R/2$). This transmission provides sufficient information about the Local Time of the node since the Frame Number is included and the time of

arrival of the transmission can be used by a receiving node to determine the skew or deviation of the Local Clock of the transmitting node.

The nodes use voters to maintain agreement on the value of the Frame Number. Each node uses the received Frame Numbers together with its own Frame Number to perform a vote, and the result is the value used in the next frame.

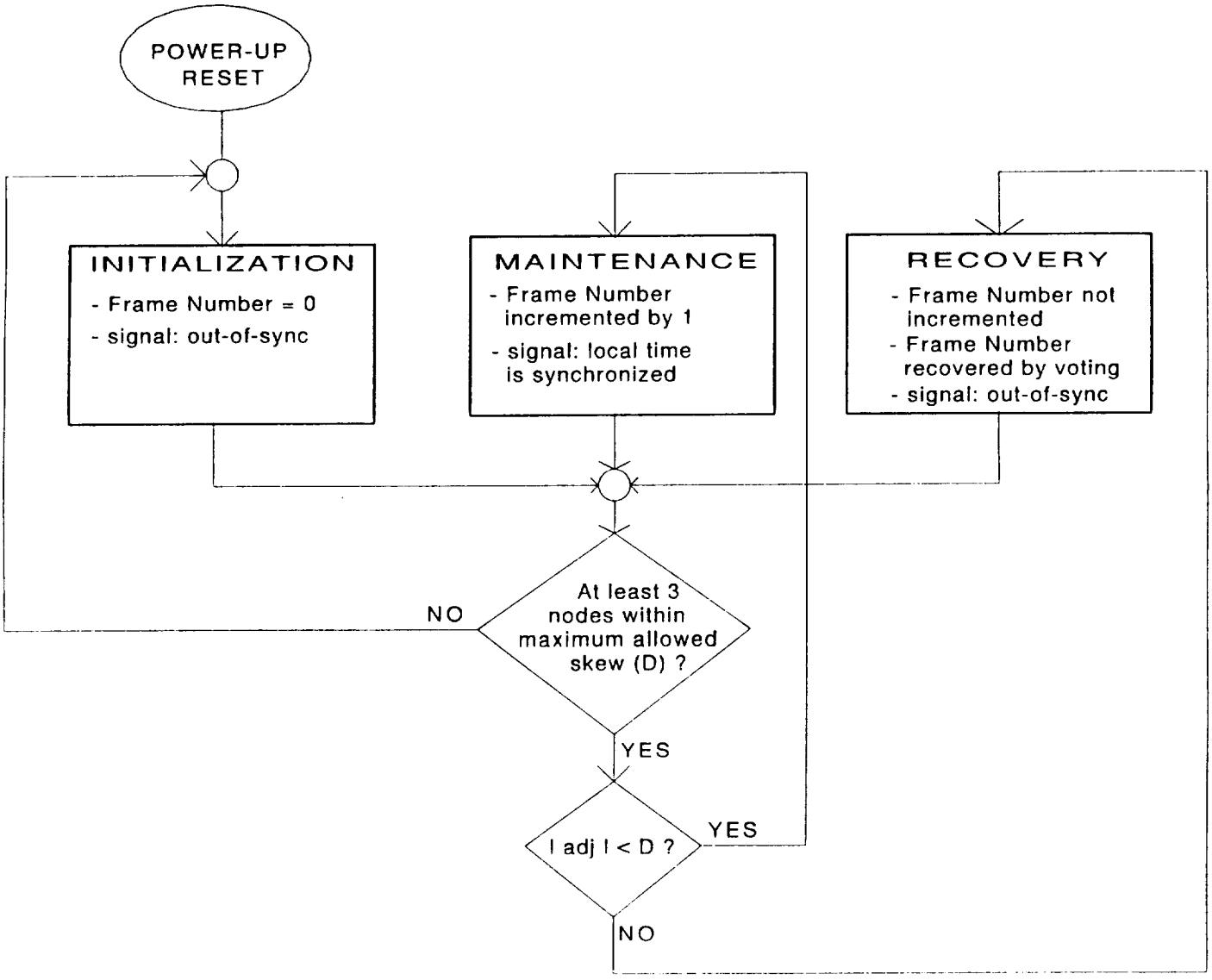
The function used to synchronize the Local Clocks is the fault-tolerant midpoint function [4]. This function is applied independently by each node. A particular node needs the deviations of all the nodes in the system with respect to its Local Clock. The deviations of the other three nodes are computed based on the time of arrival of their transmissions with respect to the expected arrival time. A fourth deviation, the deviation of the node with respect to itself, is also required by the algorithm. To get this deviation a node assumes there is a fourth transmission received in perfect synchronization (i.e., this deviation is always zero). Computation of the Local Time adjustment using the midpoint function consists of taking the average of the deviations of the second and third received transmissions. A node applies its computed adjustment by making the frame shorter or longer than the nominal $\mathbf{R}$ by an amount equal to the adjustment.

Proper behavior must be defined for when a node does not have enough information to compute an adjustment. The fault-tolerant midpoint function requires the deviations of at least three nodes. Considering the results in [1], it was decided to use the Assumed-End-of-Frame strategy to handle a condition of insufficient information to compute the adjustment. This strategy consists of assuming that the missing information arrived at the end of the frame, and then computing an adjustment.

The nodes can operate in one of three possible states: Initialization, Recovery, or Maintenance. Each node evaluates its operational state at the end of each frame. Figure 1 shows the rules that a node uses to determine its state. A node goes to Initialization after power-up or after detecting there are fewer than three nodes synchronized within the maximum allowed skew $\mathbf{D}$. In this state the Frame Number is set to zero and the node raises a flag signaling its Local Time is not synchronized. If it is detected that there are at least three nodes synchronized, but the computed adjustment is greater than $\mathbf{D}$, the node will switch to Recovery because its Local Time is not synchronized with the others. In this state the Frame Number is not incremented and its value is recovered by a vote on the received Frame Numbers. If there are at least three nodes synchronized and the computed adjustment is smaller than $\mathbf{D}$, a node goes to the Maintenance state. This is the normal operational state of the system. In this state a node increases its Frame Number by 1 at the end of each frame and clears its out-of-sync flag.

The following sections describe the design of the new system in more detail.

FIGURE 1: OPERATIONAL STATES OF THE NODES



2.1. Comparison to previous design

Figure 2 is a block diagram of the first implementation. The goal there was to design a circuit that implemented the algorithm using a 10 MHz clock, used Programmable Logic Devices (PLDs), had variable parameter settings, and included external controls for experiments. Using theoretical analysis, it was decided to use a nominal frame length R of 8192 (= 2000 hex) clock ticks, a transmission time Q of R/2 = 4196 (= 1000 hex) ticks, and a maximum allowed skew D of 11 (= B hex) ticks. These could be varied during experiments to compare the performance for various parameter combinations. To accommodate these requirements it was decided to use a 16-bit data path. This complicated the circuit because it used many comparators, multiplexers, adders, and subtracters. As built, the circuit used a lot of board space and consumed a large amount of power.

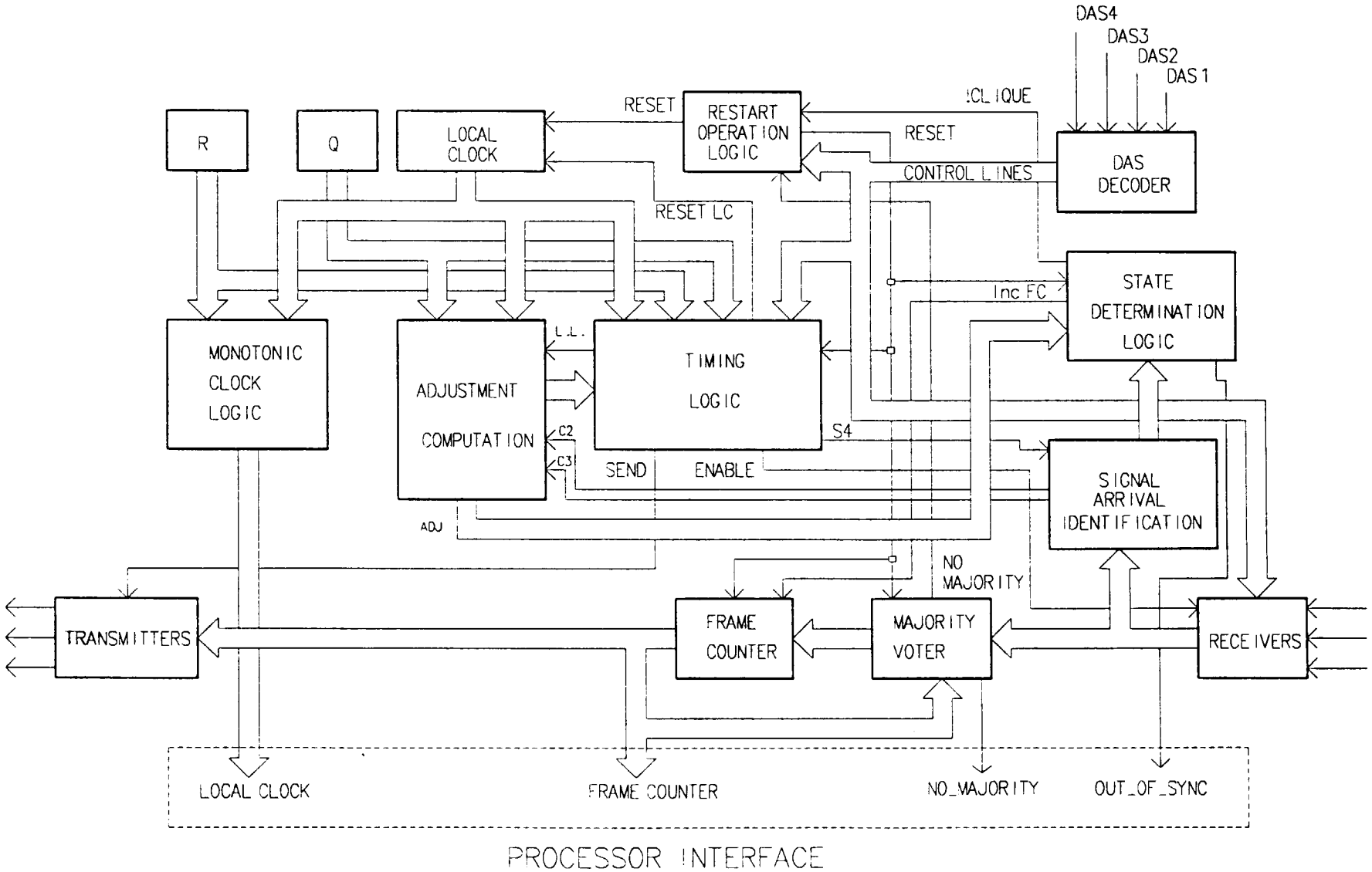
In figure 2 again, the Timing Logic and Adjustment Computation blocks used most of the hardware in the first implementation. These blocks controlled the sequence of events in the circuits and the computed adjustments, respectively. The Timing Logic block used many 16-bit comparators. The Adjustment Computation block used the informal block model present in [2]. This needed a 16-bit adder, a subtracter, a multiplexer, and two 16-bit registers. It was important for the new implementation to reduce the logic used by the Timing Logic and Adjustment Computation blocks.

The goal of the new implementation was to design a circuit that performed the same functions as the first one, but it had to use less logic, have fixed parameters setting, and operate with a 33 MHz clock, given the appropriate hardware. In figure 2, all the blocks were modified except for the Receivers, Majority Voter, Frame Counter, and Transmitters. The circuit was simplified by taking advantage of the fixed parameters setting. The parameters were fixed at the same values as in the original circuit. A 14-bit data path was used. The comparators needed for the Timing Logic were replaced by 14-bit AND gates. However, using a similar approach with the Adjustment Computation block would have resulted in a circuit more complex than desired. Simplification of this block was achieved by reanalyzing the convergence function. The next section explains the procedure followed.

2.2. Optimized Implementation of the Convergence Function

The purpose of the Adjustment Computation block is to compute the correction to the Local Clock using the convergence function. As mentioned above, the function used here is the fault-tolerant midpoint function. The computed adjustment is applied by making the frame shorter or longer than the nominal frame length R by an amount equal to the adjustment. To explain this more specifically, let us define some variables:

FIGURE 2: BLOCK DIAGRAM OF THE FIRST IMPLEMENTATION



LC_2= value of the Local Clock LC when the second transmission is received
LC_3= value of the Local Clock LC when the third transmission is received
Q= value of LC when a received signal is in perfect synchronization with the Local Clock; also equal to the time for transmission
R= nominal frame length (i.e., in perfect synchronization, a frame ends when LC=R)

The computed adjustment adj is equal to the average of the deviations of the second and third received transmissions. So:

$$\text{adj} = \frac{(LC_2 - Q) + (LC_3 - Q)}{2}$$

Then, the condition to end a frame is:

$$LC = R + \text{adj}$$

To get to the formula used in the new implementation, note that for the parameters specified:

$$R = 2Q$$

Substituting in the condition to end the frame:

$$LC = R + \text{adj} = 2Q + \frac{(LC_2 - Q) + (LC_3 - Q)}{2}$$

$$LC = 2Q + \frac{(LC_2 + LC_3) - 2Q}{2} = 2Q + \frac{(LC_2 + LC_3)}{2} - Q$$

$$LC = \frac{(LC_2 + LC_3)}{2} + Q$$

The first term in this summation can be rearranged:

$$LC = [LC_2 + \frac{(LC_3 - LC_2)}{2}] + Q$$

This is the formula implemented in the new optimized version of the clock synchronization circuit. It is implemented in the following manner:

1. When the second transmission is received, the value of $LC=LC_2$ is loaded into a counter

2. After loading, the counter starts counting at one-half the frequency of the system clock until the third transmission is received
3. The output of the counter is then added to Q. This gives the value of LC to end the frame.

Figure 3 shows a diagram of the implementation of this formula. As can be seen, it only requires a 14-bit counter, an adder, and a comparator. Note that for Q=1000 hex, the adder only operates on the two most significant bits of the output of the counter. Specifically, let $A[14..1]$ be the output of the counter. Then the output of the adder $E[14..1]$ is:

$$E[12..1] = A[12..1]$$
$$E[13] = \overline{A}[13]$$
$$E[14] = A[13] \text{ XOR } A[14]$$

This result simplifies the implementation even further. The comparator uses the standard XNOR-AND combination for 14-bits. The next section presents the full implementation at the component level.

2.3. Implementation Description

The optimized implementation had to perform the same functions as the original. Figure 2, without the DAS Decoder block, also applies to the new implementation. Figure 4 shows the logic used in the Local Clock, Monotonic Clock, Timing Logic, and part of the State Determination blocks. The Local Clock (at the top of the figure) is the 14-bit counter driven by the clock oscillator (CLK signal). It has enable (COUNT), reset, and restart (LOAD) inputs.

The logic to compute and signal the end of the frame is in the top right side of figure 4. The A counter is driven by the clock oscillator. The counter is loaded with the value of the Local Clock LC_2 when the second transmission is received. After this, the counter starts incrementing until the third transmission is received. Arrival of the second and third transmissions is signaled by D_C2 and D_C3, respectively. The A counter has a carry-in input (Cin) used to control the count to one-half the frequency of the clock oscillator. The value Q (=1000 hex) is added to the output of the counter, and the result is passed to a two-stage synchronous LC comparator. The two stages of this comparator are the XNOR and AND stages, which are separated by D-type flip-flops. Note that the A counter resets to value 2FFF hex to avoid an early-end-of-frame error (i.e., the output of the adder will be equal to 1000 hex if the counter resets to 0000 hex). The output of the comparator passes through a D flip-flop and becomes the LOAD signal used to end the frame. Note also that because of the delay introduced by the A counter, the comparator and the D flip-flop of LOAD, the nominal frame length becomes R+4 = 2004 hex ticks. The NO_Rx signal is

FIGURE 3: OPTIMIZED IMPLEMENTATION OF THE CONVERGENCE FUNCTION

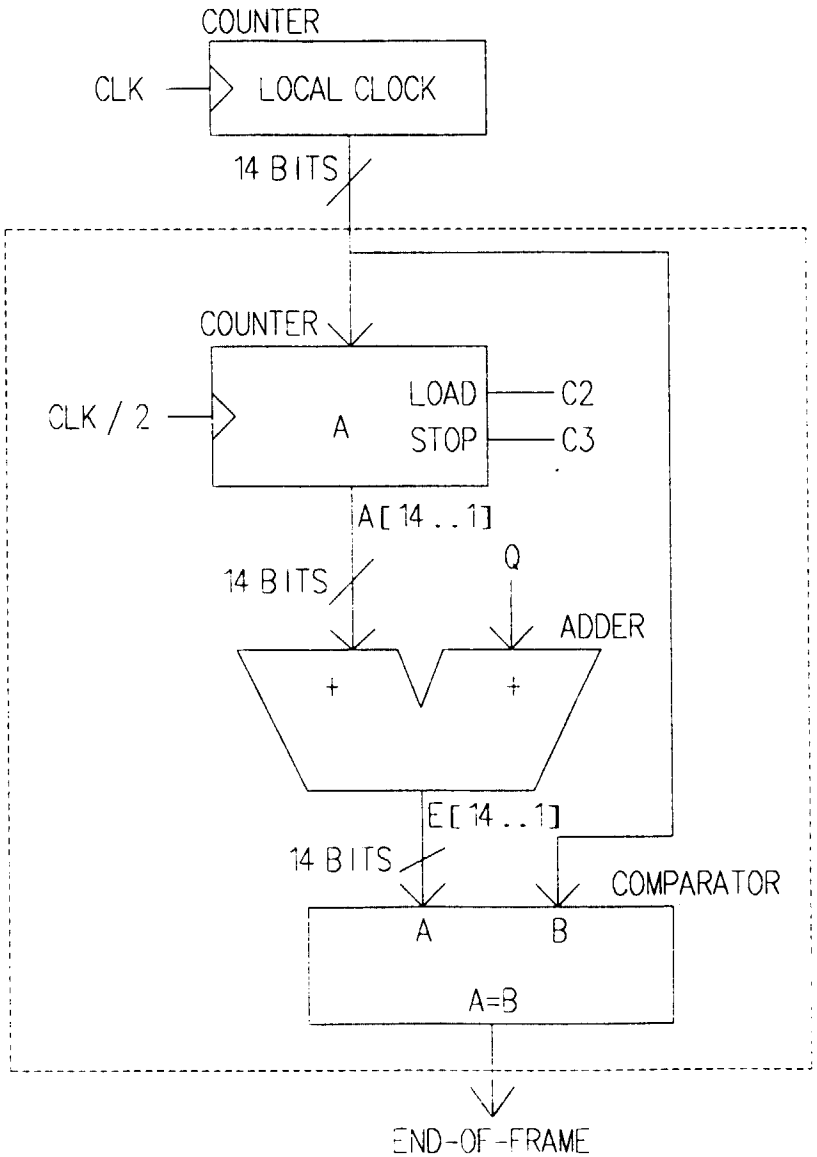
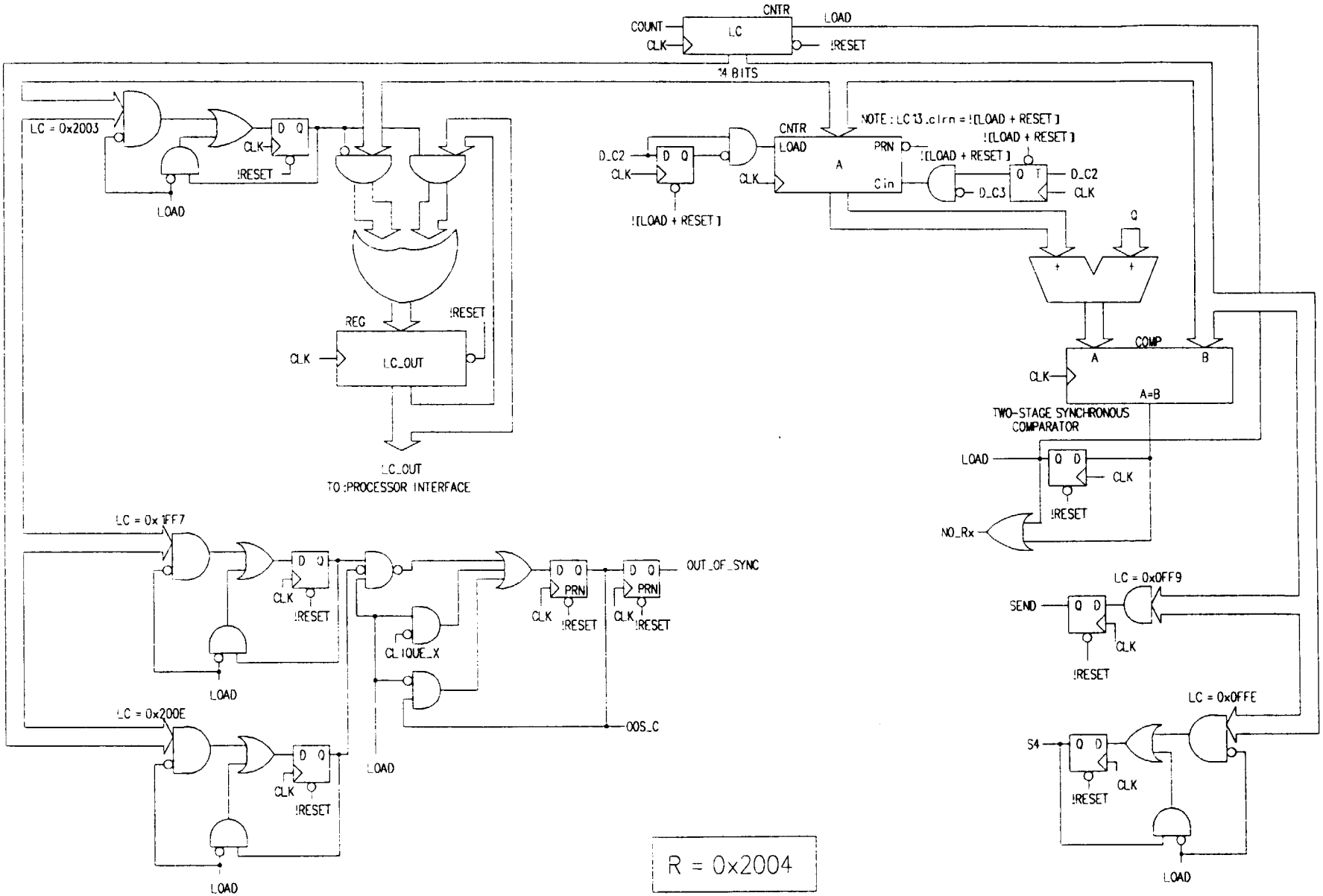


FIGURE 4: LOCAL CLOCK, MONOTONIC CLOCK, TIMING LOGIC, AND OUT_OF_SYNC LOGIC



used to disable the Receivers during the last two ticks of the frame to allow time to process the received information and to clear the logic for the next frame.

The other two signals in the bottom right part of figure 4 control the timing of the transmissions of the node. The SEND signal enables the Transmitters to broadcast the Frame Number. The S4 signal controls the timing of the assumed transmission in perfect synchronization (see the beginning of section 2).

The Monotonic Clock logic is at the top left side of figure 4. This logic passes the value of the Local Clock until it reaches 2003 hex. If LC is larger than this, the output will be kept constant at 2003 hex.

The bottom left section of figure 4 shows part of the State Determination Logic. The OUT_OF_SYNC signal goes low if the node is in the Maintenance state (see figure 1). A node stays in this state if there are at least three synchronized nodes (i.e., there is a clique) and if the computed adjustment is smaller than the maximum allowed skew D. The CLIQUE_X signal indicates whether or not there is a clique. Also, there is logic to implement two "flags". One flag goes high when LC=1FF7 hex and the other when LC=200E hex. If the LOAD signal comes in the time between the first flag and the second flag, then the computed adjustment is smaller than the maximum skew D.

The logic in figure 5 complements the Out_Of_Sync logic presented in figure 4. It is the Clique Detection logic. Counters C13 and C24 are used to count the number of clock ticks between the first and third and between the second and fourth received transmissions, respectively. Their outputs go to comparators (implemented with AND gates) to determine if they are larger than the maximum allowed skew D. The block labeled LOGIC implements the CLIQUE_X function, which uses the results of the comparisons and the number of received transmissions to determine whether or not there is a synchronized group of nodes. The logic function implemented by CLIQUE_X is shown in figure 5.

Figure 6 shows the Signal Identification and Receivers blocks. The Receivers are a direct link to the other nodes in the system. Their outputs include the Frame Number (which are sent to the Majority Voter) and control signals to indicate the arrival of valid data. These control signals are latched (S1d, S2d, and S3d) and sent to a synchronizer. This section is needed because the receivers are synchronized to the clock of the node from which the information was sent. After the synchronizer, there are logic functions to identify the first (C1), second (C2), third (C3), and fourth (C4) received transmissions. Note that signal S4 is used in this logic. Also, the Gate section uses the bit LC14 to generate D_C2 and D_C3 from the logic of C2 and C3. This is a simple way to implement the Assumed-End-of-Frame strategy mentioned at the beginning of section 2. If the second or third transmissions are not received, LC14 forces D_C2 and D_C3 high near the end of the frame so an adjustment can be computed. D_C2 and D_C3 are used by the A counter in figure 4.

Figure 7 shows the reset circuit. There are three conditions that generate a reset: power-up, manual reset, and "return to Initialization". The 74122 chip is a One-Shot circuit that is used to reset after power-up and after a manual reset.

FIGURE 5: CLIQUE DETECTION

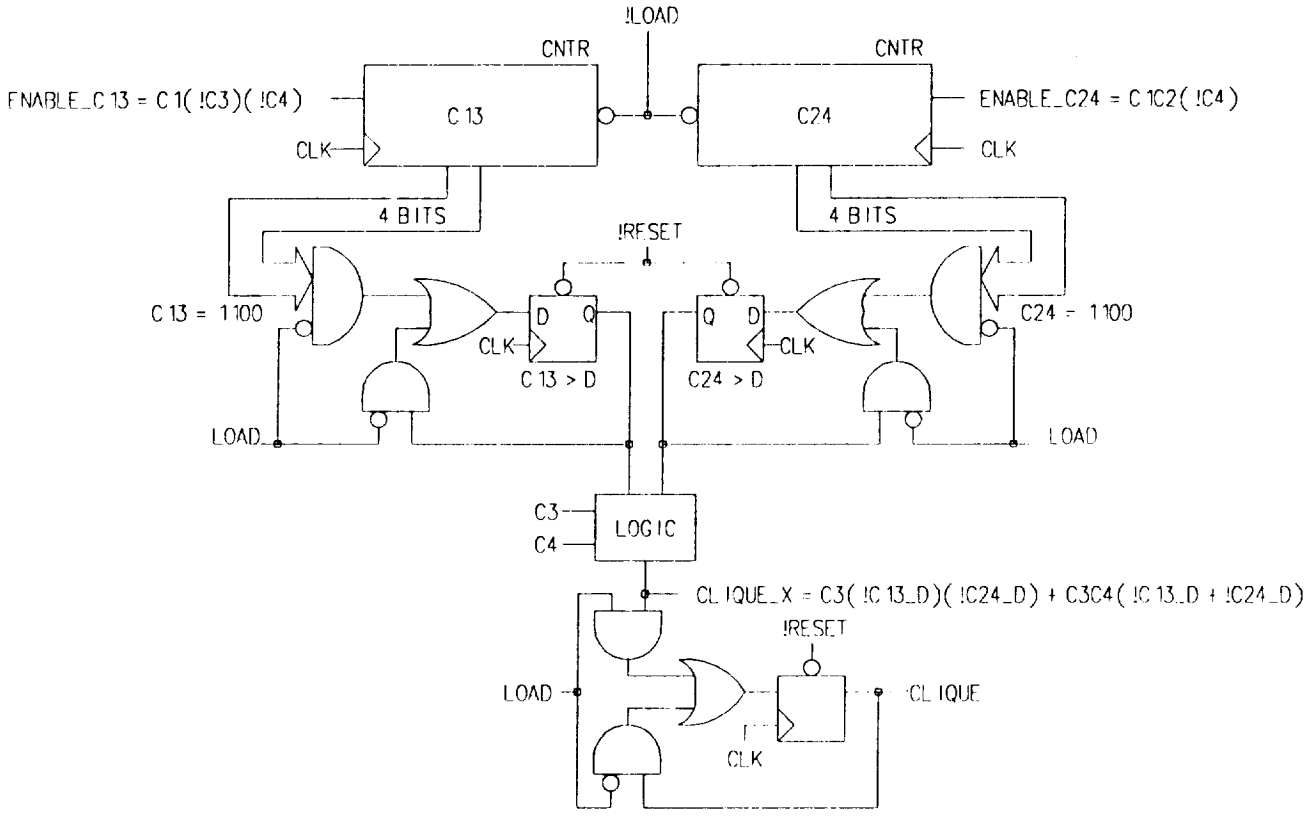


FIGURE 6: SIGNAL ARRIVAL IDENTIFICATION

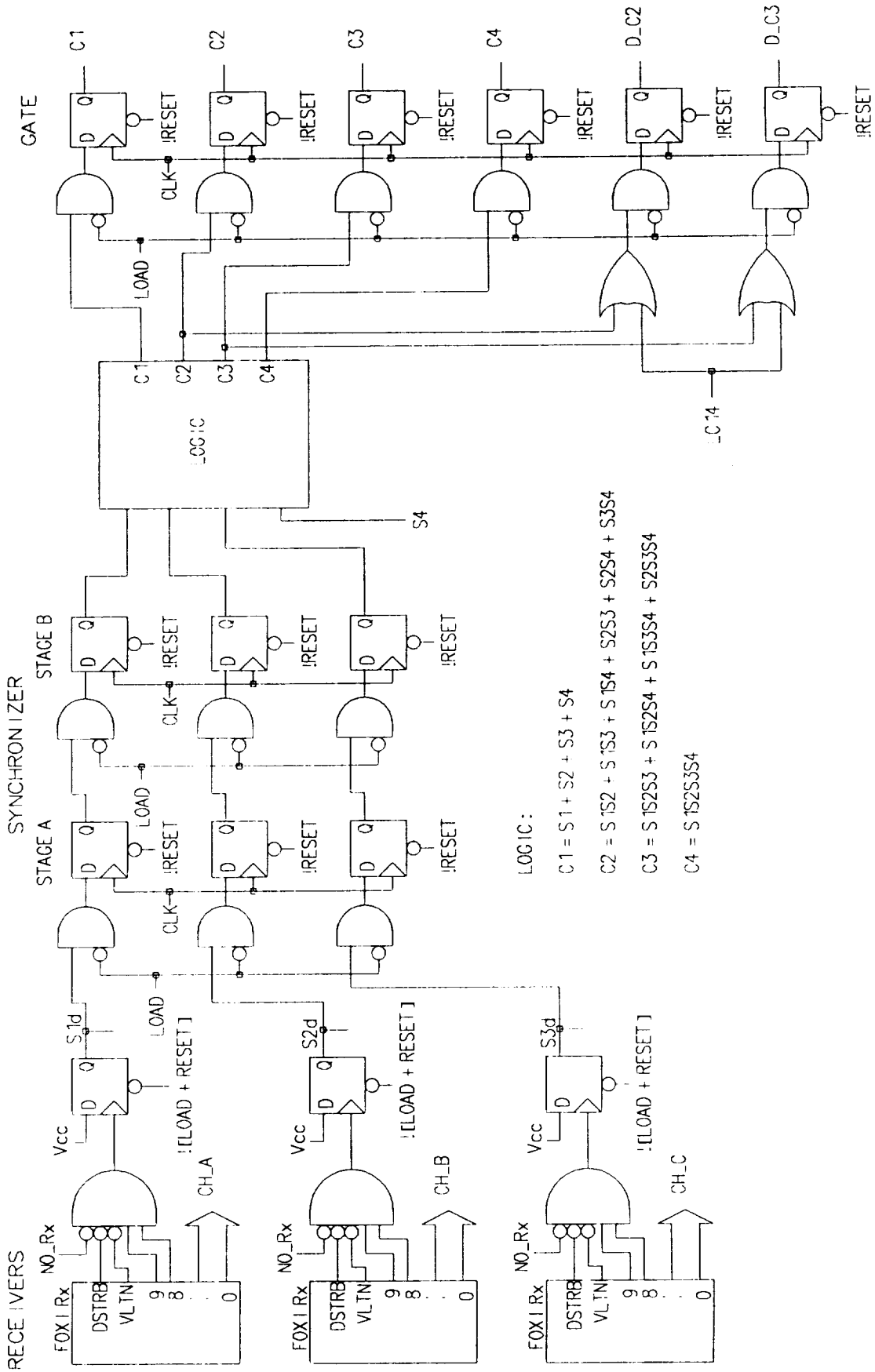
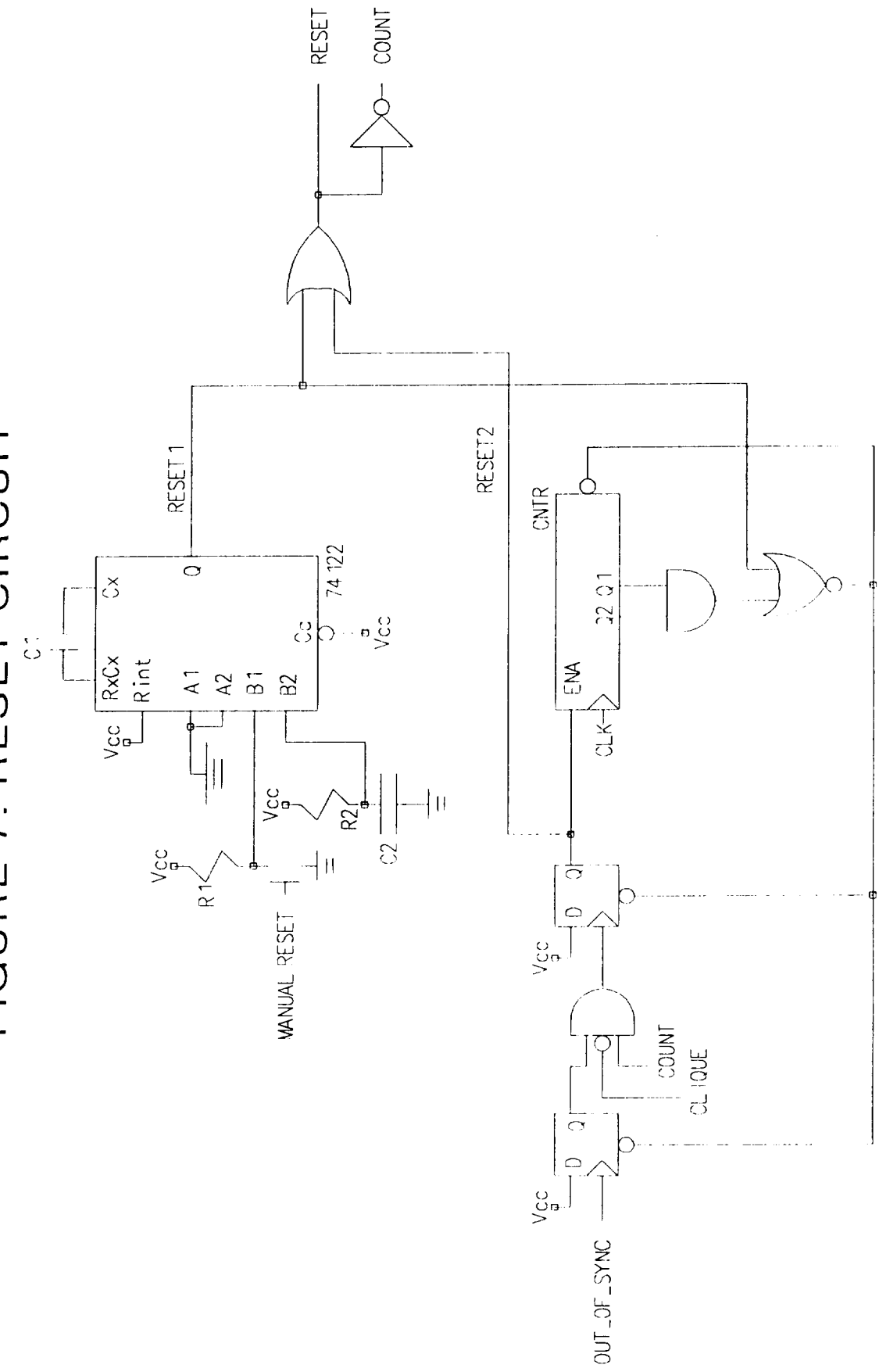


FIGURE 7: RESET CIRCUIT



The circuit at the bottom of the figure controls the "return to Initialization" reset. The D flip-flop is used to latch a high value when the OUT_OF_SYNC signal goes low (i.e., when the node is synchronized). If later the Clique Detection logic signals that the system has lost synchronization, the reset goes high and a counter is started. After three ticks the reset is cleared, the COUNT signal is enabled, and the node continues operation in the Initialization state.

The logic of the Frame Counter and the Majority Voter will not be presented because they are identical to the ones used in the first implementation. The functions performed by these blocks is explained at the beginning of section 2.

3. Implementation Evaluation

The new design was tested by implementing it on PLDs and using a 10 MHz clock frequency. This frequency was selected so the new design could be tested using three nodes from the first implementation to complete the system. The setting of the nodes that used the first design was tuned so they matched those in the new implementation. Three cases were investigated: initialization, recovery from a single transient, and recovery from a massive transient. Also, the tightness of synchronization was measured.

Figure 8 shows a sample plot of the response of the system during initialization after power-up. The Local Clocks of the nodes start at random values. After an initial delay, the nodes reset their circuits and start operating. Synchronization was achieved after 58,000 clock ticks (= 5.8 milliseconds) from power-up. The time to achieve initial synchronization was close to this value during all of the observations. Figure 9 shows the Local Time for the same run in figure 8. Note how the Local Time was kept under 9,000 until the nodes synchronized, after which the Frame Number began to increment. This was the expected behavior and, as can be seen, the new optimized implementation behaves identically to the previous one.

Figure 10 shows a sample response of the system during a single transient fault on the optimized clock circuit. Because the new design does not include external controls, the transient fault was simulated by forcing a reset at Reference Time 20,000 clock ticks. The figure shows how the optimized node recovered synchronization by extending its frame. This behavior is typical, and the time to recover synchronization depends on the value of the Local Clock at the time the reset occurs.

Figure 11 shows a sample plot of the response of the system to a massive transient fault. Here again the fault was a reset forced at a random time during the operation of the system. In this sample case, the fault occurred in nodes 1, 3, and the optimized implementation. This massive fault caused the nodes to move to the Initialization state until resynchronization. Note that the system recovered within 20,000 clock ticks (= 2.0 ms) after the transient.

FIGURE 8: LOCAL CLOCKS DURING INITIALIZATION

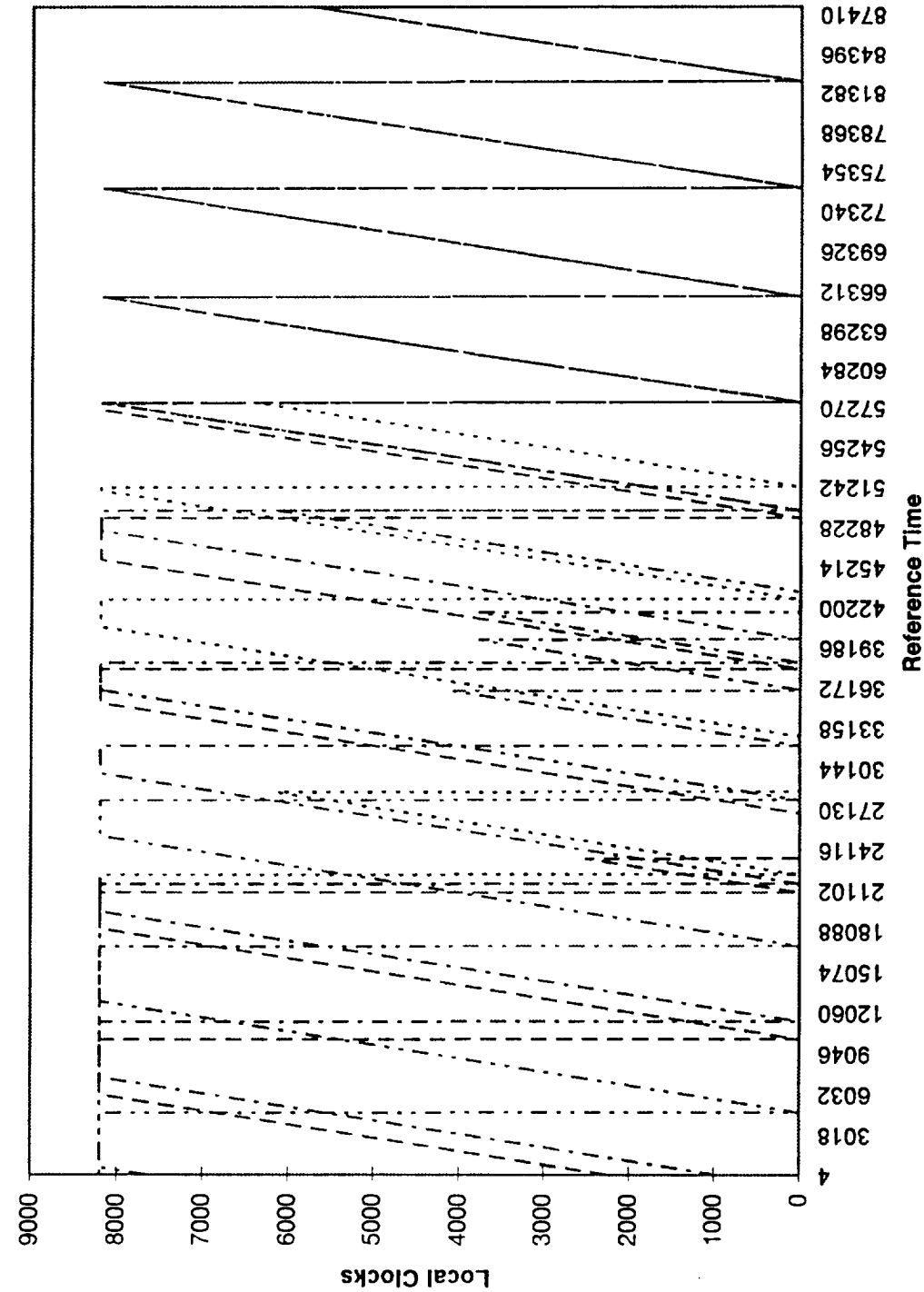
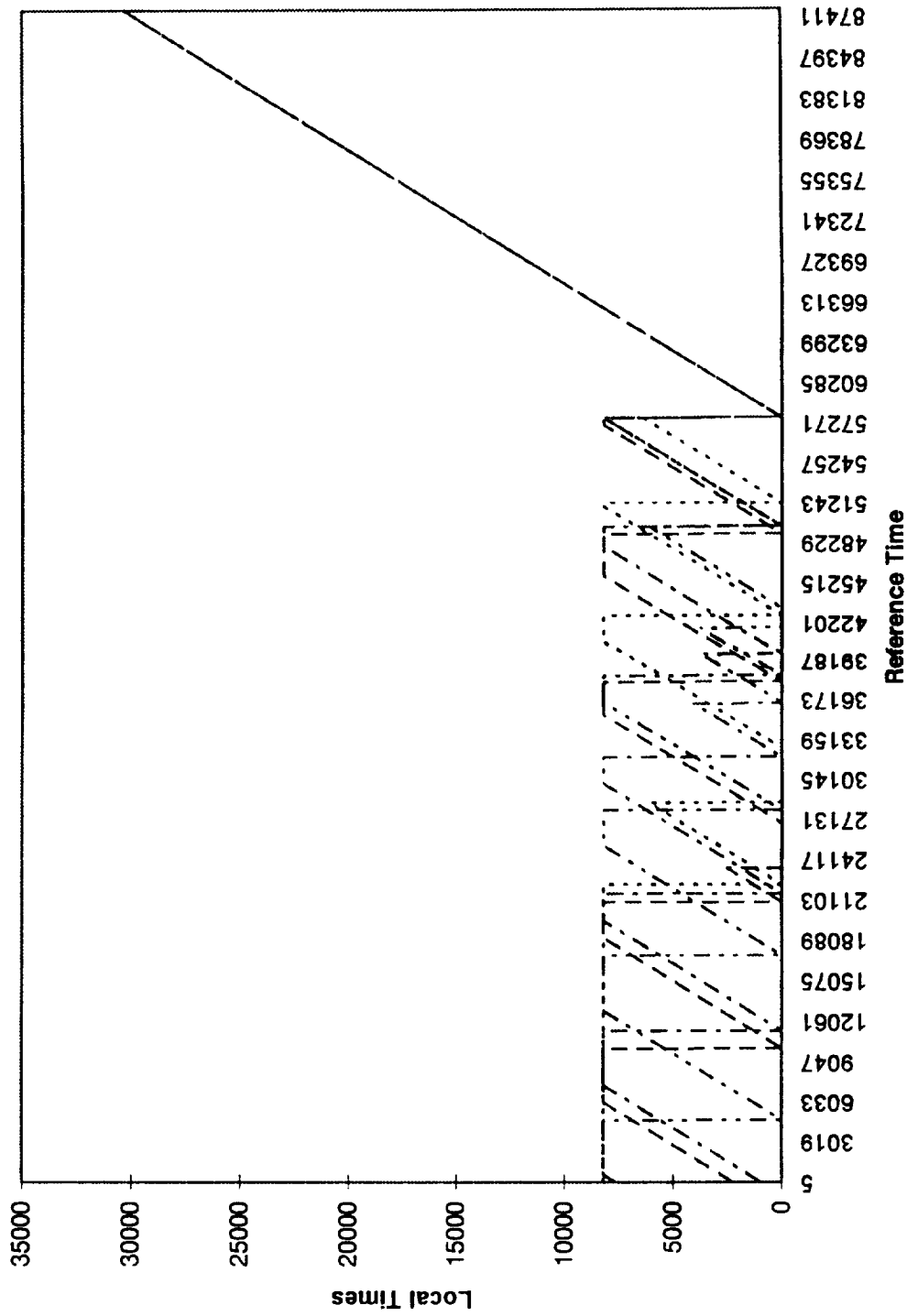


FIGURE 9: LOCAL TIMES DURING INITIALIZATION



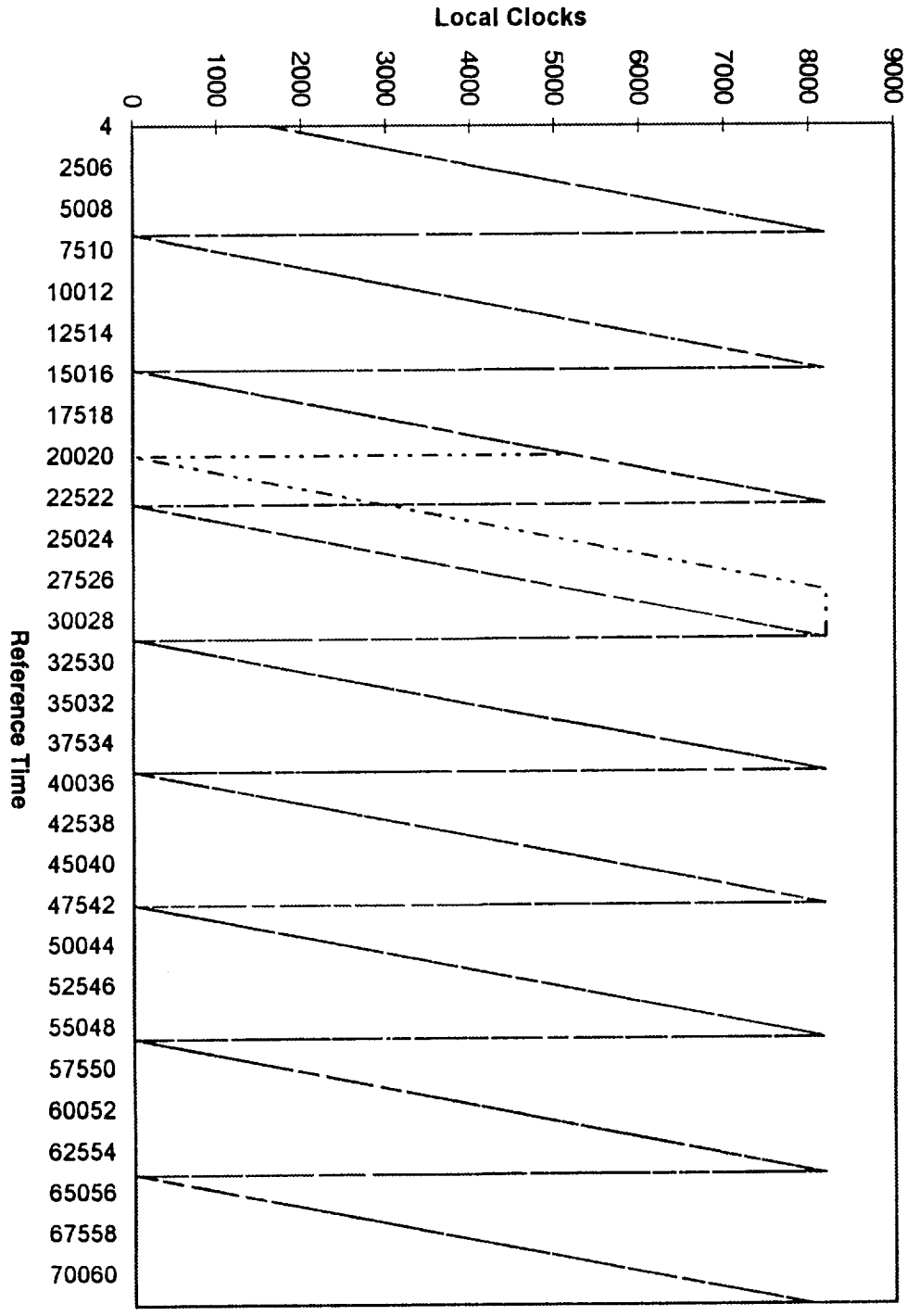
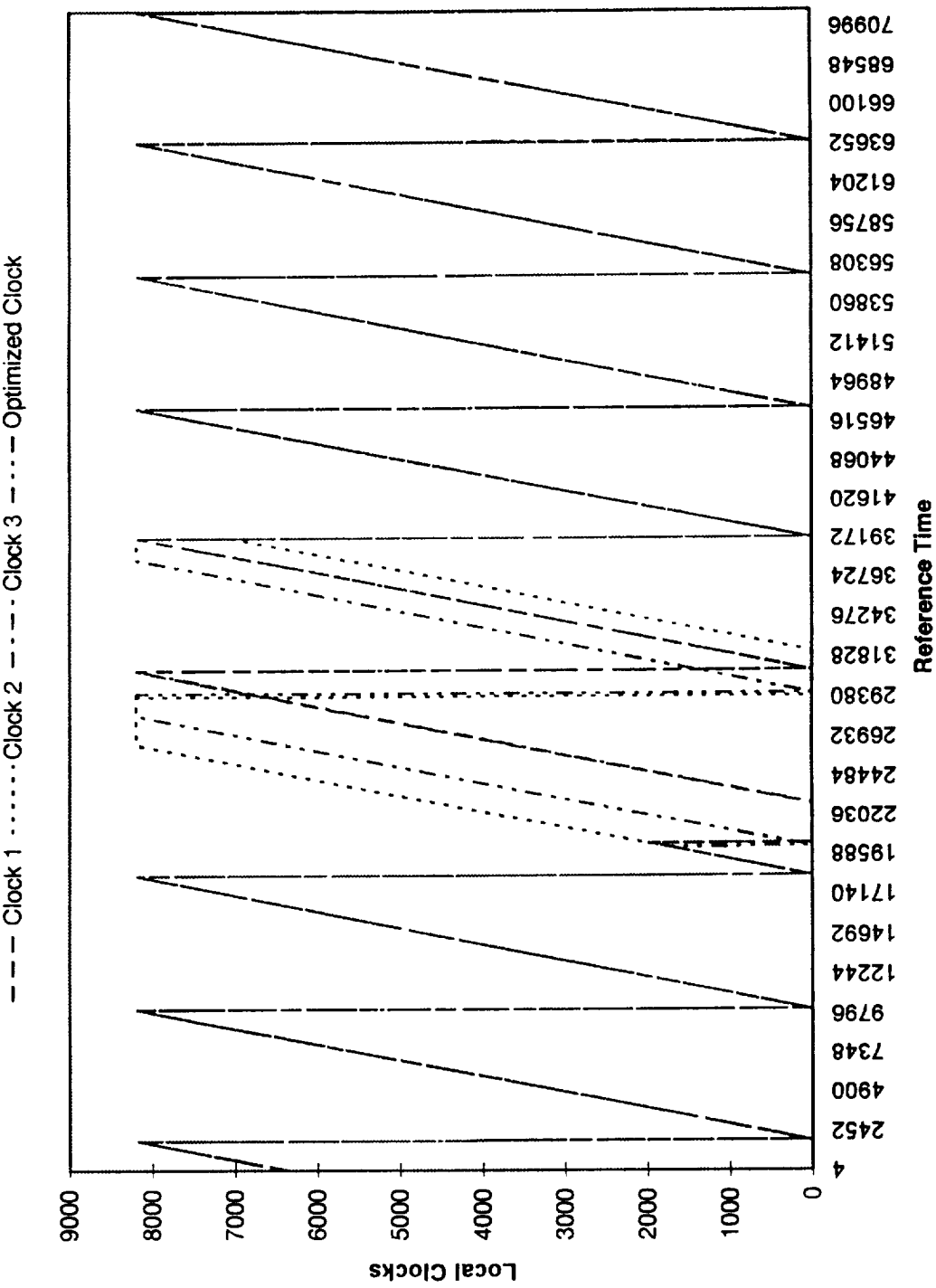


FIGURE 10: RECOVERY FROM A SINGLE TRANSIENT FAULT

Fault: Reset on Optimized Clock at Reference Time 20 000

FIGURE 11: RECOVERY FROM A MASSIVE TRANSIENT FAULT

Fault: Reset on clocks 1, 3, and Optimized at Reference Time 20 000



The tightness of synchronization was measured using an oscilloscope. The system was able to maintain synchronization within 300 nanoseconds or 3 clock ticks. This is an excellent performance considering that the implementation of the convergence function is completely different in the new optimized design. This result also gives added confidence on the equivalency of the two implementations.

4. Concluding Remarks

A new design of a clock synchronization system has been presented. The goal of designing a new circuit that uses fewer logic gates and performs the same functions as the initial design has been accomplished. It has been shown that the circuit behaves correctly during all the possible operating states when tested using three nodes from the initial design. It remains to be demonstrated whether the design can be directly implemented on a 33 MHz circuit. However, it seems reasonable to expect that this will not pose a problem given the simplicity of the design.

5. References

[1] Wilfredo Torres-Pomales: "A Hardware Implementation of a Provably Correct Design of a Fault-Tolerant Clock Synchronization Circuit", NASA TM 109001; Langley Research Center, Hampton, VA; July 1993.

[2] Paul S. Miner: "Verification of Fault-Tolerant Clock Synchronization Systems", NASA TP 3349; Langley Research Center, Hampton, VA; November 1993.

[3] Paul S. Miner and Steven D. Johnson: "Formal Analysis of an Optimized Fault-Tolerant Clock Synchronization Circuit", submitted to the 1995 International Conference of Computer Design, Austin, TX; October 1995.

[4] J. Lundelius Welch and N. Lynch: "A New Fault-Tolerant Algorithm for Clock Synchronization", Information and Computation, 77(1):1-36, April 1988.

Appendix: Ideas that could help improve the clock circuit design

1. The Local Clock and the Frame Counter use independent sections of the design. This could be exploited in a high speed design using fast logic for the circuit directly connected to the Local Clock, and slower logic for the circuit connected to the Frame Counter.
2. The Majority Voter could be simplified by using a sequential machine instead of combinational logic. This could result in a reduction in the amount of hardware needed.
3. The SEND signal could be derived from one of the bits of the Local Clock similarly to the way the LC14 bit is used for the D_C2 and D_C3 signals. Also: Is S4 really needed?
4. Eliminate the NO_Rx signal. In this design, NO_Rx is only needed by the Majority Voter because it uses combinational logic.
5. Use a shorter frame length during Initialization. A shorter frame length will force the nodes to compute adjustments more frequently. This could result in a shorter time to achieve initial synchronization and to recover from a massive transient.
6. Consider reducing the number of transmitters in each node. The current implementations use three transmitters on each node to accomplish the point-to-point communications. However, the same result can be accomplished by using one transmitter with a one-to-three splitter to distribute the signal to the other nodes.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 1995	Technical Memorandum

4. TITLE AND SUBTITLE
An Optimized Implementation of a Fault-Tolerant Clock Synchronization Circuit

5. FUNDING NUMBERS
WU 505-64-10-13

6. AUTHOR(S)
Wilfredo Torres-Pomales

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER

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 TM-109176

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unclassified-Unlimited

Subject Category 62

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

A fault-tolerant clock synchronization circuit was designed and tested. A comparison to a previous design and the procedure followed to achieve the current optimization are included. The report also includes a description of the system and the results of tests performed to study the synchronization and fault-tolerance characteristics of the implementation.

14. SUBJECT TERMS
Fault Tolerance, Clock Synchronization, Convergence Function, Implementation

15. NUMBER OF PAGES
24

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		