Clock Synchronization in Distributed Real-Time Systems

HERMANN KOPETZ, MEMBER, IEEE, AND WILHELM OCHSENREITER

Abstract—**The generation of a fault-tolerant global time base with known accuracy of synchronization is one of the important operating system functions in a distributed real-time system. Depending on the types and number of tolerated faults, this paper presents upper bounds on the achievable synchronization accuracy for external and internal synchronization in a distributed real-time system. The concept of continuous versus instantaneous synchronization is introduced in order to generate a uniform common time base for local, global, and external time measurements. In the last section, the functions of a VLSI clock synchronization unit, which improves the synchronization accuracy and reduces the CPU load, are described. With this unit, the CPU overhead and the network traffic for clock synchronization in state-of-the-art distributed real-time systems can be reduced to less than 1 percent.**

Index Terms—**Distributed real-time systems, external synchronization, fault-tolerant clock synchronization, internal synchronization, synchronization accuracy, VLSI clock.**

I. Introduction

A distributed real-time system consists of a set of nodes which are interconnected by a local area network and communicate via message passing only. The nodes are self-contained computers, which perform a specified function. Each node contains a local real-time clock of quartz accuracy.

In order to measure the time of occurrence of events and the duration of intervals between events, which are observed from different nodes, it is necessary to provide to the application a common time reference of specified accuracy. This common time reference, which is known as an "approximate global time," is generated from the local times of the local real-time clocks by the application of a clock synchronization algorithm.

Clock synchronization algorithms for distributed real-time systems should meet the following requirements.

1) The maximum difference of the time values of any single event observed from two different nodes, measured in the metric of the local time base of these two nodes of the distributed system, must be bounded by a known constant.

2) The approximate global time must be chronoscopic, i.e., it can be used for the accurate measurement of small time intervals at any point in time.

Manuscript received December 19, 1986; revised February 26, 1987. This work was supported in part by the Austrian Microelectronics research program under Contract 8067/1-V/E-ME/85 and by the external research program from Digital Equipment Corporation (DEC) under Contract AU-001/1984.

The authors are with the Institute of Technische Informatik, Technical University of Vienna, A-1040 Vienna, Austria.

IEEE Log Number 8715061.

3) The clock synchronization algorithm must be fault tolerant, i.e., a fault in any local real-time clock or the loss of a message must be tolerated.

4) The execution of the clock synchronization algorithm should not degrade the performance of the total system.

A number of different clock synchronization algorithms have been published [2], [4], [6], [8]–[12], [17].

In the analysis of these algorithms certain assumptions about clock failure modes, granularity of the local time base (this parameter is excluded from most investigations), properties of the communication medium, etc., are made. The main subject of this paper is the analysis of clock synchronization in a loosely coupled distributed real-time system, the derivation of an upper bound on the synchronization accuracy and the presentation of a VLSI clock synchronization unit for the efficient implementation of a fault-tolerant global time base.

This paper is organized into four main sections. In Section II we introduce the basic concepts related to real time and the different time references which must be considered in a distributed system. In the first part of Section III we discuss the internal synchronization with the fault-tolerant average algorithm and calculate upper bounds for the synchronization accuracy under different fault assumptions. The second part of Section III deals with external synchronization. Finally, in Section IV we present the functions of a VLSI clock synchronization unit for distributed systems.

II. Real Time

A progression of time from the past to the future is often called the "arrow of time" or the timeline.

In our model of the world we will introduce two basic notions relating to this timeline, the concepts of event and duration.

Definition: An *Event* is an occurrence at a point in time, i.e., a happening at a cut of the timeline, which itself does not take any time.

Definition: A *Duration* is the time interval between two events, i.e., a section of the timeline.

If we want to measure the position of an event on the timeline or the length of a duration, we need some metric of time and a device which can be used for time measurement.

A metric always requires the existence of a standard in order to calibrate the measuring devices. A standard of time must satisfy two criteria: it must contain a relative reference in order to measure a duration within a system and it must contain an absolute starting point in order to measure the position of an event on the timeline in relation to the

environment. In the course of history, a number of different standards of time have evolved:

Definition: Astronomical Time is a standard of time, which is based on astronomical observations (e.g., UT1, ET [3]).

Definition: External Physical Time is a standard of time, which is based on some periodic physical oscillation (e.g., atomic time TAI [3]). TAI is a chronoscopic time without any irregularities.

There is a disagreement between the astronomical and physical standard of time. The standard for measuring the duration is the "second," based on the oscillation of a physical (atomic) clock, i.e., the caesium beam standard. The reference for measuring the position of an event on the timeline is based on the astronomical time. Since there is some deviation between the "physical year" and the "astronomical year" it is necessary to insert occasionally an additional second into the latter. The insertion of this "switching" second is done by the Bureau International de l'Heure, Paris, which is responsible for the standard of time, based on international agreements. The resulting standard of time is called Universal Time Coordinated, UTC for short [3]. UTC forms the basis for local time zones (MET, EDT, etc.) and can be derived from TAI. UTC contains discontinuities at the switching second and, hence, is not chronoscopic. We, therefore, propose to take the chronoscopic external physical time TAI as an external reference for distributed real-time systems.

Definition: Physical Clock is a device which is based on some periodic physical oscillation (e.g., the oscillation of a pendulum, a quartz crystal, or an atom) in order to measure the progression of physical time.

Definition: Granularity of a Physical Clock is the period of oscillation of a physical clock.

The granularity of a physical clock is determined by the duration between two consecutive ticks. It can be measured only if there is another clock available with a finer granularity. At the end, we can only count ticks, but cannot make any statement about granularity.

From the point of view of analysis we assume the existence of a reference clock. Starting from a defined initialization point this reference clock ticks forever. In our model, a metric for the duration between the ticks, i.e., the granularity of our reference clock, is not defined. We consider a tick i of the reference clock as a significant event denoted by ts_i. The domain of i is the set of positive integers. The state of the clock at the time of occurrence of an arbitrary event e is equal to the index i of the last tick ts_i.

At the moment of observation of an event e, we instantaneously record the state of our reference clock and assign this state to the event, abbreviated by $ts(e)$. If two or more events occur between two ticks of our reference clock, we say that the events "occurred at the same time."

It is evident that the granularity of a physical clock limits the resolution of direct time measurements. We will assume that the granularity of the reference clock is chosen such that the effect of the discreteness of the time base is reduced to a sufficiently small value ns. If not stated otherwise, we will neglect errors of the magnitude ns. The duration between two events $e1$ and $e2$ expressed in the number of ticks of the reference clock is then given by

$$|ts(e_2) - ts(e_1)|. \tag{2.1}$$

In order to measure events in the metric of the external physical time (TAI), we need an internal image of the external physical time in each node of our distributed computer system.

Definition: Internal Physical Time is a time reference within each node of a distributed computer system which approximates the metric of the external physical time tp.

The occurrence of an event e in the internal physical time of node k is denoted by $tp^k(e)$, the granularity and ticks accordingly.

A distributed computer system consists of a set of nodes and a communication facility to exchange information between these nodes. Every node has its own local physical clock.

Definition: Local Time is the time of the local physical clock in a node. The abbreviation $tl^k(e)$ will be used to record the occurrence of an event e in the units of the local time of node k. The granularity of the local time of node k, expressed in the metric of the standard physical time, is

$$nl^k = ts(tl^k_{i+1}) - ts(tl^k_i). \tag{2.2}$$

The specified value of the granularity nl of the local time will be denoted by nl_{spec}.

Let us assume that during an arbitrary interval the rate of a fault-free quartz clock can deviate from the rate of the standard physical time by at most ρ^l, i.e.,

$$\forall i,\ k\colon\ 1-\rho^l \le \frac{ts(tl^k_{i+1}) - ts(tl^k_i)}{nl_{\text{spec}}} \le 1+\rho^l. \tag{2.3}$$

A clock which meets this requirement is called a "good" quartz clock.

In a distributed real-time system it is required to measure the duration between two events which have been observed by two different nodes in the metric of physical time or to identify and record the time of occurrence of a distributed event, i.e., an event which can be observed from different nodes in a distributed system. This requires a common time reference, which can be generated by the mutual synchronization of the local clocks.

Two levels of synchronization are necessary in distributed systems, internal synchronization and external synchronization. Internal synchronization refers to the construction of an approximate global time base among the ensemble of nodes of the distributed system. External synchronization refers to the synchronization of this approximate global time base with the external physical time, in order to generate an internal physical time.

In a loosely coupled distributed system the internal synchronization should be realized by the exchange of messages, otherwise separate channels for clock synchronization are needed.

Definition: Approximate Global Time is an approximation of a common time base which is established by internal synchronization.

TABLE I
TIME REFERENCES

time reference	source	notation	granularity	tick
reference clock	atomic clock	$ts(e)$	undef.	ts_i
physical time	TAI	$tp(e)$	np	tp_i
internal physical time at node k	appr. global time via extern. sync.	$tp^k(e)$	np^k	tp_i^k
global time		$tg(e)$	ng	tg_i
local time at k	local quartz	$tl^k(e)$	nl^k	tl_i^k
approximation of global time at node k	local time via internal synchronization	$tg^k(e)$	ng^k	tg_i^k

The abbreviation $tg^k(e)$ will be used if the occurrence of an event e is to be recorded in the metric of the approximate global time of node k. Its ticks are denoted by tg_i^k, its granularity is denoted by

$$ng^k = ts(tg_{i+1}^k) - ts(tg_i^k). \tag{2.4}$$

We will assume that the granularities of the approximate global times are about the same in all the nodes.

In a distributed system a global time is an abstract notion. However, if the approximate global times are synchronized to some accuracy and the granularity of the approximate global times is chosen accordingly, then it is useful to introduce the concept of a global time as a notion which refers to those properties which are common to all approximate global times, e.g., the average duration between two ticks, the tick number, etc.

Table I gives an overview of the different times introduced so far [7].

III. Clock Synchronization

A. Internal Synchronization

In the worst case two good clocks of an ensemble of clocks can deviate by at most $2\rho^l$ [s/s]. In order to reduce the deviation of good clocks of an ensemble from each other, a periodic internal synchronization of the clocks is necessary. We call the time interval between two internal resynchronization points the resynchronization period R_{int}. At each internal resynchronization point every node needs some information on the state of the clocks of other nodes and applies a synchronization function [12] on this information in order to calculate a correction term for its local clock. The state of each clock has to be corrected by its correction term in order to sustain the required synchronism. No matter how short the resynchronization period or how good the synchronization function is, there will always be a deviation among the local clocks of the system.

We distinguish between instantaneous and continuous resynchronization. In instantaneous resynchronization the correction term is applied to the local time of node j at an instant of time. In continuous resynchronization the rate of the local clock is readjusted for the duration of the resynchronization interval such that the correction is spread out over the resynchronization interval. Continuous synchronization is an established technique of clock synchronization in closely coupled systems [14], [16]. Let us call the time interval which is needed to read all clock values and to calculate and apply the correction term C_j the synchronization interval R_{read}. In the case that

$$R_{\text{read}} \ll R_{\text{int}}$$

we can neglect the effects of the differing clock rates of the good clocks during R_{read} as second-order quantities. Consider the example of $nl = 1\ \mu$s, $R_{\text{read}} = 10$ ms, and $R_{\text{int}}^r = 1$ s, then assuming $\rho^l = \pm 10^{-5}$ the maximum deviation of two good clocks during R_{read} is 200 ns. This is about one order of magnitude smaller than nl in our proposal (Section IV) and can, therefore, be neglected as a second-order quantity.

Definition: Accuracy of Internal Synchronization: Given any external event e, the occurrence of which can be observed by all of the N nodes of a distributed system, then the accuracy of internal synchronization Δ^{int} is defined by

$$\forall e, i, k\text{: Max } |tl^i(e) - tl^k(e)| = \Delta^{\text{int}}. \tag{3.1}$$

This accuracy is defined in the metric of tl.

Let us assume that there exists an ensemble of N^{total} nodes connected by a broadcast communication medium. Depending on the size of N^{total} either all or only a subset of the nodes N are used for the internal clock synchronization (in order to improve the scaling properties of clock synchronization) [15]. If

$$N < N^{\text{total}} \tag{3.2}$$

then the rest of the clocks can listen to the synchronization traffic generated by the N clocks and set their local clock accordingly. Thus, the proposed clock synchronization method can function with a constant number of N messages per resynchronization period, independent of the total number of nodes N^{total} in the ensemble.

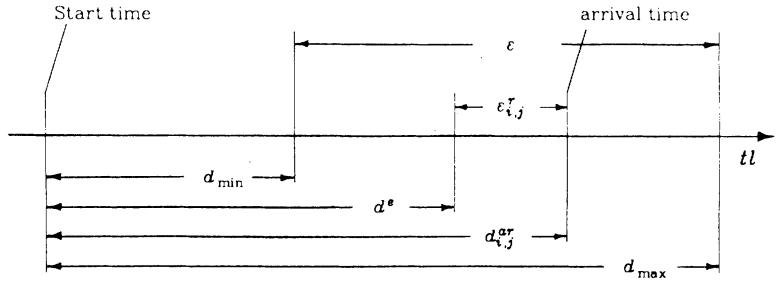
In the following we will use r to identify a particular resynchronization period R_{int}. Let us assume that every node of the N selected nodes in the system sends at about the same time, determined by the accuracy of synchronization Δ^{int} and the bandwidth of the medium, a broadcast resynchronization message to all other nodes. By

$$x_{i,j}^r$$

we refer to the time value of arrival at node j of the resynchronization message of clock i measured in the local time of clock j. The point in time of sending the message in the local time of clock i is contained in the message (it is also the point in time of receiving the message by node i, i.e., $x_{i,i}^r$). We call the variability of the message delay in this period $\epsilon_{i,j}^r$ (the reading delay), and the estimated message delay d^e, (d^e is independent of r).

Fig. 1 shows the message delays on an example.

Let us construct the following matrix of time values, where node j holds its column and its diagonal of the matrix



$d^{ar}_{i,j}$ .. actual delay of message from node i to node j in period r
$\varepsilon^r_{i,j}$ .. reading delay, i.e. variable delay of message from node i to node j in period r
d^e .. estimated delay
d_{min} .. minimum delay
d_{max} .. maximum delay
ε .. $d_{max} - d_{min}$ is called the reading error

Fig. 1. The message delays shown on an example.

(constructed from the time values contained in the received messages).

$$X^r_j = \begin{pmatrix} x^r_{1,1} & \cdot & x^r_{1,j} & \cdot & \cdot \\ \cdot & x^r_{2,2} & x^r_{2,j} & \cdot & \cdot \\ \cdot & \cdot & x^r_{j,j} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & x^r_{n,j} & \cdot & x^r_{n,n} \end{pmatrix}. \quad (3.3)$$

For all good clocks

$$x^r_{i,j} = x^r_{i,i} + (\Delta^r_{i,j} + \epsilon^r_{i,j} + d^e) \cdot (1 - \delta_{i,j}) \quad (3.4)$$

where

$\Delta^r_{i,j}$ refers to the difference between clock i and clock j
$\epsilon^r_{i,j}$ refers to the reading delay of clock i by node j, relative to a estimated message delay
d^e is the estimated message delay

$$\delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j. \end{cases}$$

We can now derive a correction matrix Y

$$Y^t = \begin{pmatrix} 0 & y^r_{1,2} & \cdot & \cdot & y^r_{1,n} \\ y^r_{2,1} & 0 & \cdot & \cdot & y^r_{2,n} \\ \cdot & \cdot & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ y^r_{n,1} & y^r_{n,2} & \cdot & \cdot & 0 \end{pmatrix} \quad (3.5)$$

where each element is given by

$$\begin{aligned} y^r_{i,j} &= x^r_{i,j} - x^r_{i,i} - d^e \cdot (1 - \delta_{i,j}) \\ &= (\Delta^r_{i,j} + \epsilon^r_{i,j}) \cdot (1 - \delta_{i,j}). \end{aligned} \quad (3.6)$$

Each node sees only its view, i.e., its column of the correction matrix. Every node must apply a synchronization function [12] on its column of the correction matrix. The result of the synchronization function is a correction term C_j for each node j.

After we have calculated the correction term C_j for the local clock we can either correct the state of the clock instantaneously (instantaneous resynchronization) or we can spread the correction term over the next synchronization period and apply it continuously (continuous resynchronization).

In instantaneous resynchronization a negative value of C_j can result in a backward correction of the local time, thus invalidating the local time measurements and possibly causing negative interval measurements. In order to avoid this phenomenon, Halpern *et al.* [13] have proposed the creation of a new clock version at every resynchronization point. These different clock versions have to be managed by the clock software for a period of overlapping existence. Continuous resynchronization provides a chronoscopic time base and avoids the problems of negative intervals or of different time versions. It is thus possible to perform precise interval measurements within the local node without any consideration of the clock synchronization activity.

The Fault Tolerant Average Algorithm: The synchronization function analyzed in this paper is the fault tolerant average algorithm, which is abbreviated by FTA [6], [10], [12]. For this algorithm it is necessary to sort each column of matrix (3.5). The correction term for clock j in an ensemble of N clocks with k faulty clocks and $N - k$ good clocks is calculated by

$$C_j = -\frac{1}{N-2k} \sum_{i=k+1}^{N-k} y^r_{i,j}. \quad (3.7)$$

Let us now derive an upper bound on the synchronization accuracy using FTA. As a start condition, let us assume that all clocks have been synchronized within Δ^{int} at the beginning of the resynchronization interval R^r_{int}. Let us introduce a convergence function $\Pi(\Delta^{int}, N, k, \epsilon)$ which gives the maximum (worst case) difference of the clock values of all good clocks immediately after an instantaneous resynchronization (see Fig. 2).

Since the reading error ϵ is an additive term, the convergence function can be decomposed.

$$\Pi(\Delta^{int}, N, k, \epsilon) = \Pi(\Delta^{int}, N, k) + \Pi(\epsilon). \quad (3.8)$$

The accuracy of synchronization is determined by

1) drift rate ρ^l of the local clocks and duration of the resynchronization interval

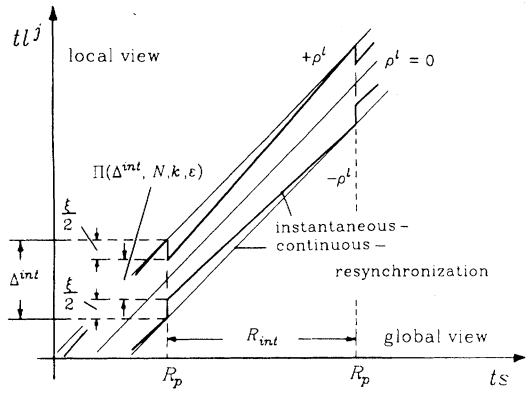


Fig. 2. Synchronization condition.

2) reading error of one clock by another (ϵ) see Fig. 1
3) system failures caused by message loss or malicious faults.

Let us now estimate the contribution of these items to the synchronization accuracy.

1) Drift rate ρ^l: according to Fig. 2 the drift rate ρ will result in a deviation ξ of

$$\xi = 2 \cdot \rho^l \cdot R_{\text{int}} \tag{3.9}$$

during a resynchronization interval R_{int}.

2) Reading error ϵ: the maximum difference of two reading delays $\epsilon^r_{i,j}$ and $\epsilon^r_{i,k}$ is less than or equal to the reading error ϵ (Fig. 1). Therefore, the convergence function $\Pi(\epsilon)$ will be

$$\Pi(\epsilon) = \epsilon. \tag{3.10}$$

3) Message loss: in the worst case a "good early message" is received by one node and not received by another node, where it is replaced by a "good late message." Thus, the contribution of a single message loss to the average is bounded by

$$\Pi(\Delta^{\text{int}}, N, k) = \frac{\Delta^{\text{int}}}{N-2k}. \tag{3.11}$$

In the presence of Byzantine faults the worst case deviation of clocks is the same as in (3.11).

An upper bound on the deviation of the approximate global times immediately after resynchronization, considering the reading error and k malicious faults (message loss, Byzantine faults), can be expressed by (3.8), (3.10), (3.11):

$$\Pi(\Delta^{\text{int}}, N, k, \epsilon) = \frac{k\Delta^{\text{int}}}{N-2k} + \epsilon. \tag{3.12}$$

Since any two good clocks can deviate by ξ during the next resynchronization interval the following necessary synchronization condition must hold for any resynchronization interval

$$\Pi(\Delta^{\text{int}}, N, k, \epsilon) \leq \Delta^{\text{int}} - \xi \tag{3.13}$$

otherwise two good clocks can diverge more than Δ^{int} (Fig. 2), i.e., the ensemble of clocks can diverge during the next resynchronization interval. This resynchronization condition (3.13) can be used to calculate an upper bound on the achievable synchronization accuracy Δ^{int} of FTA if the reading error ϵ, the resynchronization deviation ξ, and the number of nodes and malicious faults are known.

TABLE II
COST OF FAULT TOLERANCE IN RELATION TO THE SYNCHRONIZATION ACCURACY

Faults k	Number of nodes N									
	4	5	6	7	8	9	10	15	20	30
1	2	1.5	1.33	1.25	1.2	1.16	1.14	1.08	1.06	1.03
2				3	2	1.66	1.5	1.22	1.14	1.08
3							4	1.5	1.27	1.14
4								2.33	1.5	1.22

$$\frac{\Delta^{\text{int}}}{\epsilon + \xi} = \frac{N-2k}{N-3k} = u(N, k). \tag{3.14}$$

The value of the function $u(N, k)$ determines the worst possible accuracy impairment caused by the k faults in an ensemble of N clocks. The given nonfault-tolerant synchronization accuracy ($\epsilon + \xi$) is impaired by the factor $u(N, k)$. It is, therefore, an indication for the cost of fault tolerance.

From Table II it can be seen, that in an ensemble of ten clocks, two arbitrary faults can be tolerated causing an impairment of the synchronization accuracy by 50 percent. We feel that this degree of fault-tolerant clock synchronization is tolerable for most applications. If R_{int} is 1 s it follows that the synchronization traffic generated by such a fault tolerant clocking system is ten messages/second. Considering the throughput of a state-of-the-art 10 Mbaud local area network (LAN), the clock synchronization traffic amounts to less than 1 percent of the LAN throughput.

The Reading Error ϵ: The reading error ϵ (see Fig. 1) is determined by the following durations:

1) the variable time required to assemble and send the message after the local clock of the sender has been read (send time),
2) the variable access time of node k to the communication medium, depending on the access strategy (access time),
3) the variable propagation delay of the message, depending on distance and channel (propagation delay),
4) the variable time required to check the message and to record the time of arrival at the receiver (receive time),
5) the granularity of the local time (local granularity).

Let us now estimate the reading error in a distributed system connected by a local area network. We will distinguish between three alternatives (Table III).

In alternative *A* we assume a layered communication system and no control over retransmission or physical access to the communication medium, i.e., clock synchronization is performed in the application software.

In alternative *B* we assume that clock synchronization is performed in the kernel of the operating system without special hardware assistance. The send time is written into the message immediately before transmitting the message. No retransmission of synchronization messages is permitted.

In alternative *C* we assume special hardware assistance by a clock synchronization unit as described in Section IV. The

send time of a message is inserted into the message after access to the communication channel has been granted.

The Resynchronization Deviation ξ: The local real-time clock of a node is a quartz clock. Quartz clocks have a drift rate ρ^l on the order of 10^{-6}, i.e., on the average a quartz clock will deviate about 1 μs every second in relation to the standard of physical time. We will assume that the drift rate is less than $5 \cdot 10^{-6}$ (including the rate adjustment for resynchronization). The following table contains the resynchronzation deviation ξ:

resynchronization interval s	1	10	100	1000
resynchronization deviation μs	10	100	1000	10 000.

In the case of hardware support for clock synchronization, a resynchronization interval of 1 s and the assumption of a single malicious fault in an ensemble of more than four clocks, we get

$$(\epsilon + \xi) \cdot u(N, k) = \Delta^{\text{int}} < 29\ \mu\text{s}. \tag{3.15}$$

If we make other fault assumptions we can refer to Table II for the appropriate value of the function $u(N, k)$, the cost of fault tolerance.

B. External Synchronization

The synchronization between an external time and the approximate global time is called external synchronization. The internal physical time is generated as a result of this external synchronization. The external synchronization requires access to an external time reference.

External synchronization is necessary for the following three reasons:

1) to record events in the metric of a chosen external time reference

2) to take action at a point in time determined by the metric of an external time reference

3) to improve the metric of internal interval measurements.

If the external time reference is used for interval measurement, then the chosen standard must be chronoscopic. The external physical time TAI, which has been introduced in Section II, conforms to this property. It is, therefore, suggested to use TAI as an external time reference and to provide a known mapping function from TAI to the time in the given time zone (e.g., MET). It must be understood that, by definition, this mapping function contains points of unsteadiness (e.g., switching second, change from summer time to winter time). It is up to a specific application to decide how to handle these points.

Definition: Accuracy of External Synchronization, Δ^{ext}, expressed in the number of ticks of the physical time, is given by

$$\forall e, j\text{: Max } |tp(e) - tp^j(e)| = \Delta^{\text{ext}}. \tag{3.16}$$

In many distributed systems the internal physical time at node j will be derived from the approximate global time tg^j by a simple function

$$tp^j(e) = a^r \cdot tg^j(e) + \text{offset}. \tag{3.17}$$

If the offset is zero, the parameter a^r is determined by the ratio of the rates of the respective times including the continuous correction term. The offset depends on the initialization.

Since the internal physical time is derived from the approximate global time, the external synchronization accuracy Δ^{ext}

$$\Delta^{\text{ext}} = \Delta^{\text{int}} + K \tag{3.18}$$

will always be greater than the internal synchronization accuracy Δ^{int}. The constant K depends on the external resynchronization interval, the deviation of the rate of the approximate global time from the rate of the external time reference and an external reading error.

If the external time reference has the fail stop property, then an error of the external time reference will result in a loss of this reference. Since the internal synchronization provides a continuous time reference with drifts from the external reference by a given ρ^{ext}, it is possible to bridge given periods of unavailability of the external reference without overrunning a known external synchronization error. Thus, the reliability of the external time reference is not as critical to the operation of the distributed system as the reliability of the internal time reference. Therefore, in many applications a nonfault-tolerant external synchronization (i.e., a single receiver to an external reference clock, e.g., DCF77 [3]) can be justified on economic grounds.

IV. Clock Synchronization Unit

It is the objective of our hardware clock synchronization unit (CSU) to improve the synchronization accuracy, to reduce the load on the CPU for clock synchronization, and to provide one single common time reference in a distributed real-time computer system, which integrates the following properties of the local time, the approximate global time, and the internal physical time:

- chronoscopic time reference without any unsteadiness
- synchronized with all clocks of the ensemble (internal synchronization accuracy Δ^{int})
- synchronized to the external physical time TAI (external synchronization accuracy Δ^{int}).

Although the finest granularity of the common time base of the CSU corresponds to the granularity of the local time, it is to be understood that the approximate global time and the physical time, which have a coarser granularity, can be obtained by eliminating the appropriate least significant digits of this single common time base. The number of significant digits, which have to be eliminated, depends on the accuracy of the internal and external synchronization.

The clock synchronization unit CSU supports the following functions.

1) It provides a register for the local time tl with a granularity of 1 μs.

2) It provides a register for the global time tg with a granularity of 100 μs.

3) It contains the correction logic for the continuous correction of the state and the rate of the local and global time base. The correction term can be set in the range from 100 ns

TABLE III
ANALYSIS OF THE READING ERROR

	A μs	B μs	C μs
send time	10	10	1
access time	10000	50	1
propagation delay	5	5	5
receive time	1000	10	1
local granularity	1000	50	1
reading error ε	102015	125	9

Block Diagram

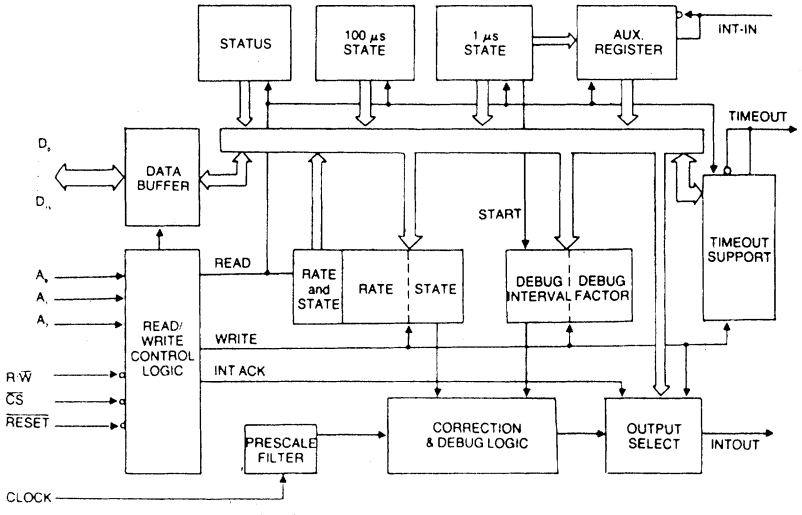


Fig. 3. Block diagram of the CSU.

to 10 ms. The large correction term is needed for initialization of the CSU.

4) It contains a sample and hold register for the accurate determination of an interrupt signal (arrival time of an incoming message).

5) It contains a DMA compatible interface such that the CSU registers can be accessed with a memory fetch operation. This facility is used to timestamp an outgoing message after the media access to a local area network (LAN) has been granted.

6) It generates a periodic interrupt after a programmable period R_{int}.

During internal synchronization, a message containing the local time of the sender is broadcasted on a LAN with a controlled access strategy (e.g., Token bus) within R_{read}. A receiving node calculates the correction factor according to (3.7) and writes the correction term into the state correction register of the CSU. The CSU then corrects the clock continuously during the next resynchronization interval.

During external synchronization, the rate of the whole ensemble of clocks has to be adjusted, i.e., every clock has to modify its local correction factor by the same additive term for external synchronization. This term is calculated by the node which has access to the external time reference. If a fault-tolerant external synchronization is desired, then the corresponding nodes have to agree on a "global" correction term by an appropriate protocol.

The rate register of the CSU is used to correct systematic rate errors of the individual clock, e.g., rate errors caused by mechanical imprecision of the quartz crystal.

The chip, designed in this project, contains eight specific read/write registers which can be selected by three address lines. The rate and state correction term can be stored into the appropriate CSU register by a single write operation. The dynamic (range) of the correction mechanism is $2^{15} - 1$. The chip, which is manufactured by a double metal 2 μm CMOS process, can be used as a CPU clocking coprocessor in a DMA driven system.

If the CSU is used as a DMA device, such that the timestamp is inserted into the outgoing message after the transmission has started, then the reading error can be reduced to about 9 μs (Table III).

Our experiments have shown that the use of our clock synchronization unit reduces the software overhead for clock synchronization in a state-of-the-art system to considerably less than 1 percent of the CPU load.

A functional overview of the CSU is given in the block diagram of Fig. 3. A detailed description is given in [5].

V. Conclusion

A number of different time references have to be synchronized in a distributed real-time system. We have shown that many important properties of these time references can be integrated into a single common time base with the aid of a

VLSI synchronization unit. The granularities of the different times are determined by the accuracy of external and internal synchronization. Upper bounds on these accuracies have been derived, depending on the types and number of tolerated faults, the duration of the resynchronization interval, and the reading error of reading one remote clock by another node and of reading an external time source. The concept of continuous versus instantaneous synchronization has been introduced to support interval measurements at all levels.

In order to reduce the overhead of clock synchronization, it is suggested to form a subset of nodes which are active in clock synchronization. The rest of the nodes can synchronize themselves to the global time base of the synchronized nodes by listening to the synchronization traffic of this subset. The analysis, which has been presented in this paper, shows that a subset of ten nodes is sufficient to tolerate two arbitrary faults without impairing the accuracy of internal synchronization by more than 50 percent. The overhead for clock synchronization can thus be reduced to less than 1 percent of the CPU load and network traffic.

Acknowledgment

We would like to thank F. B. Schneider from Cornell University, S. Shrivastava from the University of Newcastle, G. Le Lann from INRIA, and the referees for extensive comments on a previous version of this paper. Many stimulating discussions within the research group on real-time computing at the Technical University of Vienna, particularly with W. Schwabl, are acknowledged.

References

[1] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. Ass. Comput. Mach.*, vol. 26, no. 11, 1983.

[2] K. Marzullo and S. Owicki, "Maintaining time in a distributed system," in *Proc. Second ACM Symp. Principles Distributed Comput.*, Canada, Aug. 1983, pp. 295–305.

[3] *The Astronomical Almanac for 1984*, Washington, 1984.

[4] F. Cristian, H. Aghili, and R. Strong, "Clock synchronization in the presence of omission and performance faults, and processor joins," in *Proc. FTCS 16*, Vienna, Austria, July 1986, pp. 218–223.

[5] Austria Mikrosysteme International, *S65C60 Datasheet*, June 1986.

[6] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl, "Reaching approximate agreement in the presence of faults," in *Proc. Third Symp. Reliability Distributed Software Database Syst.*, Oct. 1983, pp. 145–154.

[7] H. Kopetz, "Accuracy of time measurement in distributed real time systems," in *Proc. Fifth Symp. Reliability Distributed Software Database Syst.*, Los Angeles, CA, 1986, pp. 35–41.

[8] L. Lamport, "Time, clocks and the ordering of events in a distributed system," *Commun. Ass. Comput. Mach.*, vol. 21, pp. 558–565, July 1978.

[9] L. Lamport and P. M. Melliar Smith, "Synchronizing clocks in the presence of faults," *J. Ass. Comput. Mach.*, vol. 32, pp. 52–78, Jan. 1985.

[10] J. Lundelius and N. Lynch, "A new fault-tolerant algorithm for clock synchronization," in *Proc. Third ACM SIGACT-SIGOPS Symp. Principles Distributed Comput.*, Vancouver, Canada, Aug. 1984, pp. 75–88.

[11] S. R. Mahaney and F. B. Schneider, "Inexact agreement: Accuracy, precision and graceful degradation," in *Proc. Fourth ACM SIGACT-SIGOPS Symp. Principles Distributed Comput.*, Minaki, Ont., Canada, pp. 237–249, Aug. 1985.

[12] F. B. Schneider, "A paradigm for reliable clock synchronization," in *Proc. Advanced Seminar Real Time Local Area Network*, Apr. 1986, pp. 85–104.

[13] J. Halpern, B. Simons, and H. R. Strong, "Decentralized synchronization of clocks," U.S. Patent 4 584 643, Apr. 22, 1986.

[14] T. B. Smith, "Fault tolerant clocking systems," in *Proc. FTCS 11*, Portland, June 1981, pp. 262–264.

[15] K. G. Shin and P. Ramanathan, "Synchronization of a large clock network in the presence of malicious faults," in *Proc. IEEE Real-Time Syst. Symp.*, Dec. 1985, pp. 13–24.

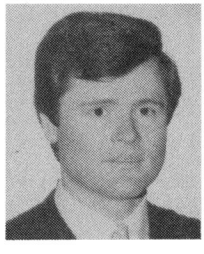
[16] W. M. Daly, A. L. Hopkins, J. F. McKenna, Jr., and J. F. McKenna, "A fault tolerant clocking system," in *Proc. FTCS 3*, Palo Alto, CA, June 1973, Cat. 73CH0772-4C.

[17] J. H. Wensley, L. Lamport, J. Goldberg, M. W. Green, K. N. Levitt, P. M. Melliar-Smith, R. E. Shostak, and C. B. Weinstock, "SIFT: Design and analysis of a fault-tolerant computer for aircraft control," *Proc. IEEE*, vol. 66, Oct. 1978.

Hermann Kopetz (M'74) received the Dr.Phil. degree (summa cum lauda) in physics from the University of Vienna, Vienna, Austria, in 1968.

After several years in industry as a manager of a department for industrial process control, he joined the Technical University of Berlin as Professor for Process Control Computers in 1978. Since 1982 he has been Professor of Computer Science at the Technical University of Vienna, Austria, where he teaches courses in real-time systems, operating systems, and fault tolerance and directs research in the area of distributed fault tolerant real-time systems. He authored over 50 papers on real-time systems, fault-tolerance, and software engineering. His book on software reliability has been translated from German into English and Polish. He is the principle investigator on the research project MARS on a maintainable real-time system.

Dr. Kopetz is a charter member of the Austrian Computer Society and the IFIP WG 10.4 on Reliable Computing. He is a member of the Association for Computing Machinery. He has served on the Program Committee of many international conferences and was the General Chairman of FTCS 16 in Vienna, Austria, in 1986.



Wilhelm Ochsenreiter was born in Bozen, Italy, in 1954. He received the Dipl.-Ing. degree in electrical engineering from the Technical University of Vienna, Vienna, Austria, in 1983.

Since 1983 he has been a Ph.D. student in the Department of Computer Science at the Technical University of Vienna. His interest is the area of real-time systems VLSI design and testing.