A Synchronization Strategy for a Time-Triggered Multicluster Real-Time System

Hermann Kopetz
Andreas Krüger
Dietmar Millinger
Anton Schedl

Institut für Technische Informatik
Technische Universität Wien, Austria
email: hk@vmars.tuwien.ac.at

Abstract

The provision of a system-wide global time base with a good precision and sufficient accuracy is a fundamental prerequisite for the design of a multicluster distributed real-time system. In this paper we investigate the issues of clock synchronization in a multicluster system, where every node can have a different oscillator. Based on the parameter of a typical automotive distributed system we show that a precision and accuracy in the μsecond range is achievable without undue effort.

Key Words: Clock Synchonization, Distributed Systems, Real Time, Global Time, Fault Tolerance

Introduction

At present, the design of large real-time distributed systems is more an art than an engineering endeavor. Recent spectacular failures of some of these systems underscore our point [Wayt Gibbs 1994]. One key reason for the failures of these large systems is the complexity in the synchronization and coordination of the concurrently executing dynamically scheduled real-time tasks. It is very difficult to sufficiently test or to reason formally about [Rushby 1993] these data dependent dynamic control structures. The "random" occurrence of non reproducible unexpected and untested synchronization problems during the operation of these systems is a main cause for the encountered difficulties.

The situation is more encouraging if we look at the field of experience with safety critical real time systems. In most of these systems the time-triggered control structure is data independent and therefore static. It is thus possible to cleanly separate the validation of the data transformations, i.e., the execution of sequential tasks, from the validation of the static control structure. At present, most of these time-triggered safety critical systems are relatively small and confined to a single cluster. It is thus a research challenge to extend the time-triggered technology to the design of large multicluster real-time systems.

A necessary service of a time-triggered architecture is the provision of a system-wide fault- tolerant global time base of sufficient precision. In multicluster systems--and sometimes even in a single cluster system--it cannot be assumed that all nodes will contain oscillators with the same nominal frequency. The design of a synchronization system within a set of clusters that will generate a uniform time base with a precision in the μsec range despite the fact that each node may have an oscillator with a different nominal frequency, is an interesting research challenge.

The objectives of this paper are the presentation of a fault-tolerant synchronization strategy for a multicluster real-time system, where no assumptions are made about the base oscillator frequency in each node, and the integration of the internal and external clock synchronization into a single coherent time base.

The paper is organized as follows. In the next section we explain our architectural assumptions and introduce a uniform format for the representation of time in a multicluster real-time system. Section three focuses on the problem of internal synchronization and describes a macrotick generation logic that allows the generation of a global time base within the physical second standard from an arbitrary oscillator frequency. Section four is devoted to the topic of external synchronization and discusses the functions of a time-gateway. In section five we analyze the achievable precision and accuracy in a typical scenario from an automotive onboard system. This paper finishes with a conclusion in section six.

Architecture

The assumptions about the base architecture have been influenced significantly by our experiences in the design of the MARS system [Kopetz 1989].

System Structure

Let us consider a time-triggered distributed system that consists of a set of clusters interconnected by gateways (Fig. 1).

Each cluster is composed of a number of nodes that are exchanging messages via a common communication channel. A node consists of a client CPU, a memory, I/O devices and a communication controller for a time-triggered communication protocol (Fig. 2). The controller communicates with the client CPU by a dual ported random access memory (DPRAM) that contains periodically updated images of the state variables in the environment. We call this DPRAM interface between the communication controller and the client CPU the Message Base Interface (MBI).

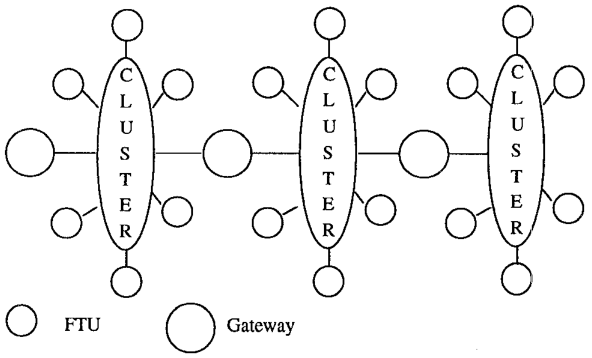

Fig. 1: Set of interconnected clusters

The communication between the nodes is controlled by the Time-Triggered Communication Protocol TTP [Kopetz 1994]. This protocol is an integrated communication protocol that provides all services needed for the implementation of time-triggered fault-tolerant real-time systems, such as predictable message transmission, internal clock synchronization, membership service, prompt error detection and fault management.

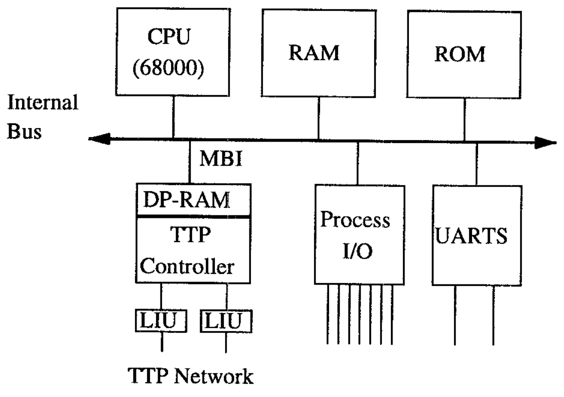

Fig. 2: Structure of a node

In TTP the access to the communication channel is controlled by a time-division multiple access strategy (TDMA) that is based on the fault-tolerant common time base generated by TTP. Since in a time-triggered system the point in time, when a message has to be sent, is known a priori to all nodes, the time difference between the expected and the observed arrival time of a message is a measure of the deviation of the sender's clock from that of the receiver. This information, which is continuously collected during system operation, is sufficient for the implementation of a fault-tolerant global clock synchronization algorithm, such as the Fault Tolerant Averaging Algorithm FTA [Kopetz 1987].

Time Representation

In a multicluster system, where a node may have an oscillator with a nominal frequency that is relative prime to the oscillator frequencies of other nodes, a common representation of time must be chosen that is independent of the characteristics of the individual node. This representation of time has to satisfy a number of criteria:

(i) it should be understandable to humans
(ii) it should be independent of detailed implementation decisions and the speed of the communication channel
(iii) it should be easy to manipulate by the computer

An accepted international standard of time is the chronoscopic International Atomic Time TAI. TAI has as a measure of time the physical second and has as the start of its epoch the midnight of January 1st 1958 [Becker 1975]. TAI serves indirectly as the basis for all international time zones, such as UTC. In contrast to UTC, which requires the insertion of an occasional switching second, TAI is chronoscopic, i.e., uniform.

We propose a data structure for the representation of time as shown in Fig.3.

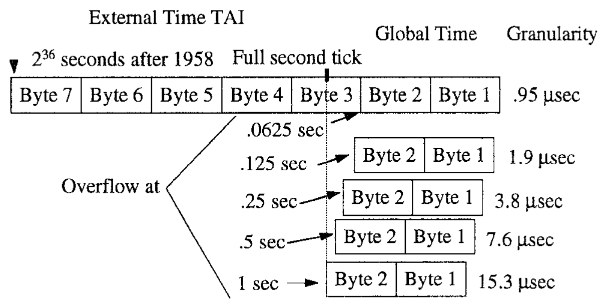

Fig. 3: Representation of the time

This seven byte data structure consists of two fields:

- a five byte (byte 3 to byte 7) external time field that contains in the first four and a half bytes the TAI representation of the current full second and in the last half byte the fractional part of a second (granularity 1/16 seconds, i.e., 62.5 msec). Considering that TAI starts at Jan.1st, 1958 this data format will not overflow until after the year 4000.
- a two byte (byte 1 and byte 2) global time field that contains the fractional part of the second as a fraction of the power of two, down to a smallest granularity of $1/(2^{20})$ second, i.e., exactly a granularity of 0.953 674 316 406 25 µsec.

Since this smallest granularity may be below the precision that can be achieved in a low bandwidth distributed system, we provide four further options for the base granularity, leading up to a granularity of 15.3 µsec. One of these options has to be selected during system configuration. In all cases we want to keep the global time field sixteen bit wide. Consequently, the last four bits of the external time field and the first four bits of the global time field may overlap.

The External Time TAI and the global time are part of the message base interface MBI. The communication controller updates the time fields autonomously. In a sixteen bit architecture the host CPU can read the global time without any integrity concerns. When reading the external time, the NBW access protocol [Kopetz 1993] has to be used to guarantee the integrity of the time information.

The described representation of time is closely related to the representation of time proposed in the Network Time Protocol (NTP) [Mills 1991]. The fractional part of the second is identical, however the full second count is different. NTP refers to the non uniform UTC, while we propose the chronoscopic TAI as the reference of the full second.

Timescales

The local oscillator within a node has to serve the following purposes (Fig. 4)

(i) it has to generate the timing signals for all computational units of the node (oscillator tick);
(ii) it has to provide the time reference for node-local time measurements and for the generation of the bit encoding and sampling of the incoming bit stream at the communication link (micro tick);
(iii) it has to generate the timing signals for the global and external time (macrotick).

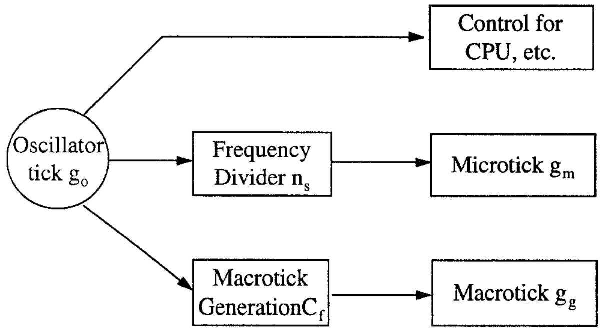

Fig. 4: Functions of the oscillator

In our view it is important to keep these conceptually different functions separate during the system design.

Oscillator tick (g_o) The oscillator tick is determined by the physical characteristics of the oscillator. In the general case it cannot be assumed that the selection of the nominal oscillator frequency is in the sphere of control of the computer system designer. This selection is often determined by electromagnetic interference concerns. Other sensitive electronic equipment within a vehicle dictates the use of an oscillator with a frequency within a limited frequency band. Furthermore it cannot be assumed that all nodes within a cluster will be driven by an oscillator with the same nominal frequency f^{nom}.

Microtick (g_m) --The granularity of the microtick within a node is determined by the required quality of the time measurements within a node. The most important local time measurement in a TTP system is the measurement of the interval between the expected arrival time of a message and the actual arrival time of a message. The quality of this time measurement depends on the reproducibility of the rise time measurement of the incoming bitstream signal on the communication channel. It is thus determined by the bit cell length, its shape, and the sampling granularity of the communication controller. A bit sampling rate of 16 samples/bitcell seems to be accepted practice. TTP systems that are to be deployed in the automotive environment have to support a transmission rate between 10 kbit/seconds and 1 Mbit/second, implying a bitcell length of between 100 µsec and one µsec. We propose a microtick granularity in the order of 1/16 of the chosen bitcell length.The microtick granularity is derived from the oscillator by a frequency divider n_s.

Macrotick (g_g) -- The granularity of the macrotick g_g (the global time) has to be selected from one of the values of Fig. 3, in order that the physical second is an integer power of two of the macrotick. According to [Kopetz 1992] the granularity of the global time g_g must also satisfy the following relation

$$g_g > \Delta_{int}$$

where Δ_{int} is the synchronization precision of the ensemble of clocks. This precision Δ_{int} will be estimated in a later section. If no assumptions can be made about the frequency of the oscillator, then the conversion factor C_f from the oscillator tick to the macrotick cannot be assumed to be an integer. In general the conversion factor C_f will thus have an integer part I_f and a fraction F_f. To achieve the synchronization objectives, a macrotick generation logic has to generate macroticks of slightly different durations in order to synchronize the macroticks within a cluster. The maximum difference in the duration of the macroticks is one oscillator tick g_o.

Internal Synchronization

The internal synchronization establishes a global time base within a cluster by the mutual fault-tolerant synchronization of the local clocks. The global time within a given cluster is an abstract notion that is approximated by the local view of the global time in each node.

Principle of Operation

The heart of the global time is the local oscillator of each node that oscillates with its given frequency f_{osc} determined by the physical shape of the crystal. The actual oscillator frequency f_{osc} can deviate from the nominal oscillator frequency f^{nom} because of a mechanical imprecision of the crystal or because of environmental effects (e.g. varying temperature). Since in our system the granularity of the global time, the macrotick, is a predetermined fraction of the full second, an adjustable oscillator-to-macrotick frequency conversion has to be put in place:

$$f_{macro} = f_{osc}/C_f$$

where C_f is the adjustable oscillator-to-macrotick conversion factor and f_{macro} is $1/g_g$, the frequency of the global time. This conversion factor C_f can be decomposed into two parts

$$C_f = C_n + C_c.$$

Hereby C_n is the conversion factor between the nominal oscillator frequency f^{nom} and the nominal macrotick frequency f_{macro}, whereas C_c is that part of the conversion factor that has to be periodically adapted in order to correct the systematic and stochastic variations of the oscillator and to bring the local view of the global time into synchronism with the ensemble of clocks.

The calculation of C_c is carried out periodically in each node by the Byzantine resilient Fault-Tolerant Average Algorithm [Kopetz 1987]. This algorithm requires as its input the number of clocks, the maximum number of faulty clocks, and the measured deviations between the clock of the node from all other clocks. These deviations are measured continuously by the Time Difference Capture Logic described below.

Macrotick Generation Logic

Figure 5 shows the schematic structure of the Macrotick Generation Logic.

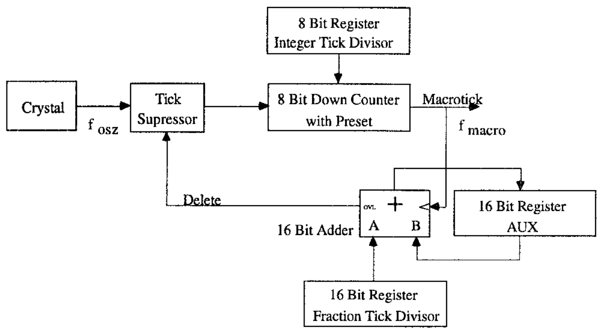

Fig. 5: Macrotick Generation Logic

This hardware performs a software-controlled frequency division by a fixed point value C_f. The correction factor C_f combines both, the nominal conversion factor from the nominal oscillator frequency f^{nom} to the macrotick frequency and the necessary (dynamically changing) correction factor to synchronize the local clock with the ensemble. C_f can be decomposed into an integer part I_f and a fraction F_f. These two values are assigned to Macrotick Generation Logic registers as shown in figure 6.

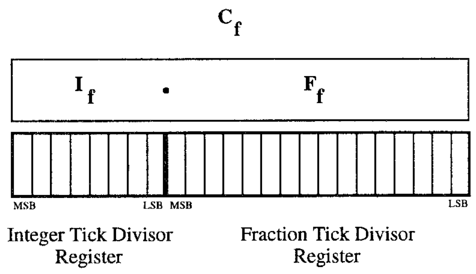

Fig. 6: Macrotick Generator Register Assignment

In the following the function of the Macrotick Generation Logic shown in figure 5 is explained:

The *8 Bit Down Counter* performs the division of the oscillator frequency f_{osc} by the value I_f. It is initialized with the value I_f stored in the Integer Tick Divisor register (ITD). It then decrements its contents whenever it receives a signal from the oscillator. If the counter reaches zero, it produces a macrotick and presets its counting register with the value stored in ITD.

The integer division of the oscillator frequency performed by the 8 Bit Down Counter neglects the fractional part of the divisor, F_f. Thus after $1/F_f$ macroticks (i.e. $1/F_f$ integer divisions) the division error due to the neglected fraction exceeds 1 oscillator tick per macrotick. This error is compensated by suppressing the next oscillator tick, i.e, by delaying the 8 Bit Down Counter by one tick.

The actual quantity of the division error is computed by the *16 Bit Adder* unit. At every macrotick, the 16 Bit Adder adds the value F_f stored in the FTD (Fraction Tick Divisor) register to the binary value stored in the *AUX register* and writes the newly calculated value back to the AUX register. This means that at every macrotick the emerging division error is accumulated in the AUX register. If the accumulated error value exceeds 1.0 (oscillator ticks per macrotick), the signal *Delete* is set to high, telling the *Tick Suppresser* to suppress one tick of the oscillator signal. At the same time, the cumulated error value is modified to reflect the suppression of one oscillator tick. This is done by truncating the error value at 1.0 and storing the remaining fractional part in the AUX register.

Time Difference Capture Logic

It is the objective of the *Time Difference Capture* (TDC) Logic to measure the time difference between the expected arrival time of a message and the actual arrival time of this message. The TDC measures these time differences with the granularity of the receiver microtick. The expected message arrival time is an event determined by the receiver's clock, whereas the actual message arrival time is an event determined by the sender's clock. The observed time difference is thus a measure for the deviation of the sender's clock from the receiver's clock.

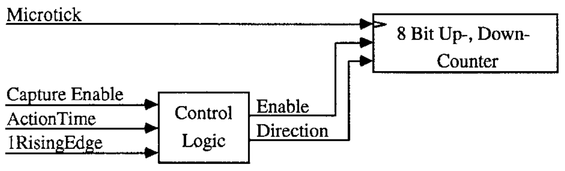

Fig. 7: Time Difference Capture Logic

The actual measurement is done by the TDC logic shown in Figure 7. The TDC unit consists of an *8 Bit Up- and Down Counter* which is clocked by the microtick signal. Three additional control signals control the counting direction and duration. The capturing operation is enabled by the *Capture Enable* signal, which is controlled by the protocol software. The signal *Rising_Edge* is generated by the receiver unit of the communication controller. Rising_Edge is set to high after the first rising edge of a new message frame is detected on the physical communication channel (i.e., this is the point in time of the actual message arrival). The signal *Action_Time* is derived from the local view of the global time. It is set to high as soon as the local view of the global time reaches the expected message arrival time. When either the Action_Time signal or the Rising_Edge signal is set to high, the counter starts to count the number of microticks until the other signal arrives. The accumulated number of ticks is the deviation between the expected and the actual message arrival time given in microticks. The sign of this number (i.e. the counting direction of the counter) is determined by the order of the Action_Time signal and the Rising_Edge signal. This order denotes the direction of the deviation of the local clock.

External Synchronization

External synchronization is concerned with linking the global time of a cluster to an external standard of time. For this purpose we need a *time server*: an external time source that periodically broadcasts the reference time in the form of a *time message*. This time message has to raise a synchronization event (such as the beep of a wrist watch) in a designated node of the cluster and has to identify this synchronization event on the agreed time scale. Such a time scale needs a constant measure of time, e.g., the physical second, and has to relate the synchronization event to a defined origin of time, the so called *epoch*. We call the node that interfaces to a time server a *time gateway* .

Principle of Operation

Let us assume that the time-gateway is connected to a GPS receiver. This GPS time server periodically broadcasts time messages containing a synchronization event and the information such that this synchronization event can be placed on the TAI scale. The time gateway has to synchronize the global time of its cluster with the time received from the time server. This synchronization is unidirectional and thus asymmetric, as shown in Fig.8.

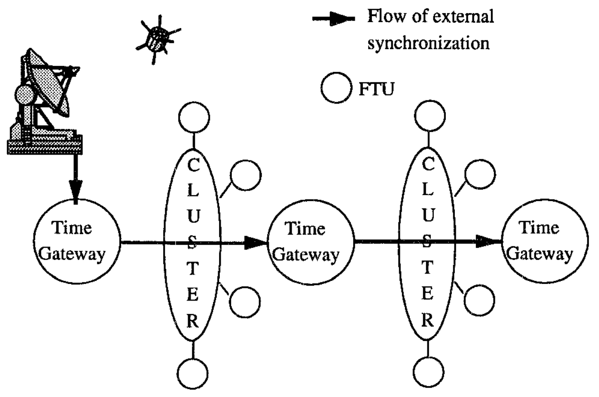

Fig. 8: Flow of External Synchronization

If another cluster is connected to this "primary cluster" by a "secondary time gateway" then the unidirectional synchronization functions in the same manner. The secondary time gateway considers the synchronized time of the primary cluster as the time reference and synchronizes the global time of the secondary cluster.

Whereas internal synchronization is a cooperative activity among all members of a cluster, external synchronization is an authoritarian process: the timeserver forces its view of external time on all its subordinates. From the point of view of fault-tolerance, such an authoritarian regime has a problem: If the authority sends an incorrect message all its obedient subordinates will behave incorrectly. However, in the case of external clock synchronization the situation is under control because of the "inertia" of time. Once a cluster has been externally synchronized, the fault-tolerant global timebase within a cluster acts as a *monitor* of the time server. A time gateway will only accept an external synchronization message if its content is sufficiently close to its view of the external time. The time server has only a limited authority to correct the clock rate of a cluster. The enforcement of a maximum common mode correction rate--we propose less than 10^{-4} sec/sec--is required to keep the relative time-measurement error small. The maximum correction rate is checked by the software in each node of the cluster.

The implementation must guarantee that in no case it is possible for faulty external synchronization to interfere with the proper operation, i.e., with the global time, within a cluster. The worst possible failure scenario in case the external time server is failing maliciously is a common mode deviation of the global time from the external time with the maximum correction rate. The internal synchronization within a cluster will not be affected by this controlled drift from the external time.

Time Message

The time message is a special message for external clock synchronization that is formed in the time gateway and processed in the communication controller of the receiving node. The format of the time message is shown in Fig. 9. The time message has a length of six bytes--the same length as the initialization (I-frame) message of the TTP protocol [Kopetz 1994]. It is thus possible to use the same basic TDMA slot for the time message and the I-message. The time message has to be sent periodically (presumably with a long period) on the bus.

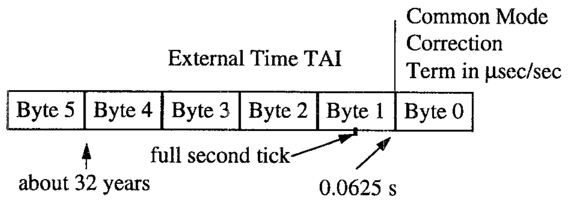

Fig. 9: Format of the time message

The time message contains two fields. Bytes one to five of the time message contain the external TAI time value of the current 1/16th second (see also Fig.3). Byte zero of the time message is the rate correction byte. It contains the requested common mode rate change in µsec/sec for all nodes in the cluster to bring the global time of the cluster in alignment with the current external time. The rate change byte is processed by the protocol software in each node and causes a modification of the C_C part of the conversion factor from the oscillator frequency to the macrotick granularity.

Time Gateway

The time gateway has to control the timing system of its cluster in the following ways:

(i) it has to initialize the cluster with the current external time

(ii) it has to periodically adjust the rate of the global time in the cluster to bring it into agreement with the external time and the standard of time measurement, the second.

(iii) it has to periodically send the current external time in a time message to the nodes in the cluster in order that a reintegrating node can reinitialize its external time value.

The time gateway achieves this task by periodically sending a time message with a rate correction byte. This rate change byte is calculated in the time gateway software. First the difference between the occurrence of a significant event, e.g., the exact start of the full second in the time server, and the occurrence of the related significant event in the global time of the cluster is measured by using the local timebase (microticks) of the gateway node. This measurement is supported by the communication controller hardware. Then the required rate adjustment in ppm (parts per million) is calculated, considering that the rate adjustment is bounded by the agreed maximum rate correction in order to keep the maximum deviation of relative time measurements in the cluster below an agreed threshold and to protect the cluster from malicious faults of the server.

Precision and Accuracy

The key quality parameter of the internal synchronization is the precision, i.e., the maximum time interval between respective ticks of any member of the cluster. In [Kopetz 1987] a detailed analysis of the precision of the fault-tolerant average algorithm (FTA) for internal clock synchronization has been carried out. According to this analysis, the precision Δ_{int} is given by

$$\Delta_{int} = (\varepsilon + \xi)\ ((N-2k)/(N-3k))$$

where ε is the reading error, i.e. the maximum random error in reading the state of the clock of one node by another node, ξ is the drift of the clocks, determined by the product $2\rho.R_{int}$. The drift rate ρ is the maximum drift rate of the oscillator. R_{int} denotes the length of the resynchronization interval. The number N refers to the number of clocks in the ensemble and k refers to the maximum number of Byzantine faulty clocks.

Let us now investigate the parameters that can be achieved in a typical automotive system in order to get a good estimate for the achievable precision.

Reading error ε: The reading error ε is determined by the variability in the edge detection of the start of a new message (Fig.11) plus the variability in the message transmission times.

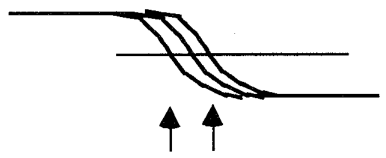

Fig. 11: Variability in the detection of the start of a new message

This variability is determined by the physical shape of the signal on the transmission channel and the granularity of the local time measurement (the microticks) in the node that records the edge detection. Considering that a bitcell consists of sixteen microticks, we can assume that the rising edge of the bitcell can be determined within three microticks. If we assume a bitrate of 100 kbit/sec and thus a bitcell of 10 µsec, then the reading error will be 3.(10/16) µsec, i.e., 1.875 µsec.

Drift ξ: The drift ξ is the product $2R_{int}.\rho$ of the length of the resynchronization interval R_{int} and the drift rate ρ of the oscillator. In a typical automotive application, such as the automotive benchmark of [SAE94], a cluster consists of 7 nodes with a TDMA cycle of 2.5 msec and a cluster cycle of about 10 msec[Kopetz 1994]. If we resynchronize once within a cluster cycle, then R_{int} is 10 msec. According to our experiments [Schedl 1995], the short term drift of a crystal oscillator of a typical computer within an interval of 10 msec is less than 10^{-6} sec/sec. The drift ξ within R_{int} is thus in the order of 10^{-8} second.

Byzantine error factor: If we assume an ensemble of seven nodes and at most one Byzantine error in a cluster cycle, then the Byzantine error factor is 5/4, i.e., 1.25.

Achievable Precision: In our example the achievable precision is given by

$$(1.875 + 0.01) * 1.25\ \mu\text{sec} = 2.35625\ \mu\text{sec}$$

In such a system the proper base granularity of the global time would be

$$3.8\ \mu\text{sec}$$

as seen from figure 3, since the precision has to be better than the granularity.

The *accuracy* is the maximum deviation between the external time standard and the internal representation of the external time. It depends on the quality of the external time signal. If we assume that a good GPS receiver will have a time resolution of better than one µsec, then the time measurement within a time gateway will introduce an additional digitalization error of one microtick. It is smaller than the granularity error of the global time.

If we increase the speed of the communication system to 500 kbits/second, the dominating error term, the reading error, will be reduced to 0.35 µsec. In such a system, the smallest granularity of figure 3, 0.95 µsec, can be supported.

Conclusion

In this paper we described a synchronization strategy for a fault-tolerant distributed time-triggered real-time system that contains a number of autonomous clusters. We presented two hardware units for the support of clock synchronization, the macrotick generation logic and the time difference capture logic. The topic of internal and external synchronization has been discussed.

The limiting term for the precision and accuracy of such a system is the performance of the time difference capture logic that measures the time difference between the clocks of a cluster by capturing the exact time of arrival of the leading edge of a bitstream arriving at a node. This term depends on the size of the bitcell, i.e. the bandwidth of the communication system.

We have shown that in a typical automotive system with a bandwidth of 100 kbits/second a timebase with a global precision of better than 4 μsec can be supported.

ACKNOWLEDGMENT

This work has been supported in part by a research grant from HP Labs in Palo Alto, California and by the ESPRIT BRA PDCS 2.

References

[Becker 1975] G. Becker, "Die Sekunde", *PTB Mitteilungen* vol. 85, Jan 1975, pp.14-28,

[Kopetz 1987] H. Kopetz, W. Ochsenreiter, "Clock Synchronization in Distributed Real-Time Systems," *IEEE Transactions on Computers*, August 1987, pp.933-940

[Kopetz 1989] Kopetz, H., Damm, A., Koza, C., Mulazzani, M., Schwabl, W., Senft, C., Zainlinger, R., Distributed Fault-Tolerant Realtime Systems: The MARS Approach, *IEEE Micro,* Vol. 9, No. 1, pp., 25-40, Febr. 1989

[Kopetz 1992] H. Kopetz, "Sparse Time versus Dense Time in Distributed Real-Time Systems", *Proc. of the 14th Distributed Computing System Conference,* Yokohama, Japan, IEEE Press, June 1992,

[Kopetz 1993] H. Kopetz, J. Reisinger, NBW: A Non-Blocking Write Protocol for Task Communication in Real-Time Systems, *Proc. of the IEEE Real-Time System Symposium,* IEEE Press, Dec. 1993

[Kopetz 1994] H. Kopetz, G.Grünsteidl, TTP- A Protocol for Fault-Tolerant Real-Time Systems, *IEEE Computer,* January 1994, pp. 14-23

[Mills 1991] D.L. Mills, Internet Time Synchronization: The Network Time Protocol, *IEEE Transactions on Communications,* Vol. 39, No. 10, Oct. 1991, pp. 1482-1493

[Rushby 1993] J. Rushby, Formal Methods and the Certification of Critical Systems, *SRI International SCL Technical Report SRI-CSL-93-07,* Menlo Park, Cal., USA, November 1993

[SAE 1994] SAE paper J2056/1 June 93, Class C Application Requirements, published in *1994 SAE Handbook,* Vol, 2, pp.23.366 - 23.272, Society of Automotive Engineers, Warrendale, PA, 1994

[Schedl 1995] The short-term Stability of Crystal Oscillators: Experimental Results, Research Report No. 1/1995 , Institut für Technische Informatik, Technical University of Vienna, Austria, 1995

[Wayt Gibbs 1994] W. Wayt Gibbs, Software's Chronic Crisis, *Scientific American,* September 1994, pp.72-81