UTCLIENT – an ASIC Supporting Clock Synchronization in Distributed Real-Time Systems*

MARTIN HORAUER, DIETMAR LOY
Technical University of Vienna
Department of Computer Technology
Gußhausstraße 27, A-1040 Vienna
Email: {horauer, loy}@ict.tuwien.ac.at

January 14, 1997

Abstract

This paper provides an overview of specification and implementation of an ASIC running under the acronym UTCLIENT that supports external clock synchronization by hardware. It incorporates an adder-based clock to maintain local time, facilities to timestamp packets containing synchronization data, interfaces to GPS receivers and provides moderate application support. This novel clock is distinguished by its fine grained rate adjustability and hardware support for linear continuous amortization.

Keywords: Universal Time Coordinated (UTC), real-time, distributed system, external clock synchronization, Global Positioning System (GPS).

1 Introduction

A distributed real-time system consists of a set of nodes which are interconnected by a local area network. The nodes are self-contained computers, that run applications which are in need to interact both correctly and timely with the environment. Actions like timestamping of events that are observed from different nodes require a common time reference of a specified accuracy provided to the application. The distributed nature aggravates the installation of a time service considerably, because autonomous running clocks have a tendency to drift apart or might fail grossly. Hence clocks need to be synchronized by virtue of a suitable algorithm executed on each node. The problem that is addressed therefore is known as *internal clock synchronization*. When the requirement of a consistent global time is extended in a way that it should also relate to some external time standard, e.g. *Universal Time Coordinated* (UTC), we have to deal with the problem of *external clock synchronization*.

*This work is part of project SynUTC, supported by the Austrian Science Foundation (FWF) under contract no. P10244-ÖMA. For further project information take a look at http://www.ict.tuwien.ac.at/synutc/index.html

Promising sources of UTC are readily worldwide available, most notably the american *Global Positioning System* (GPS) or the russian *Glonass* system, that spray time information with a precision of about 100 ns or even better.

Our SynUTC project is related to external clock synchronization focusing on hard real-time systems. Thus we are targeting $1\mu s$ as *precision*, see [Sch95] for an introduction. —Worst case precision π denotes a maximal bounded deviation between two fault-free clocks $C_i(t)$ and $C_j(t)$, thus it has to satisfy $|C_i(t) - C_j(t)| \leq \pi \quad \forall t \geq t_0$.— To achieve this ambitious goals, proper hardware support is required to combine the time information from external (high accurate but low reliable) time sources with a global time provided by clock synchronization (moderate accuracy but high reliability). This ongoing research led to the development of an ASIC termed *Universal Time Coordinated Synchronization Unit* (UTCSU), see [SS95], [SSHL97] and [Loy96]. Based on this specification a reduced subset, to which we refer as UTCLIENT, was implemented. In Section 2 we present a functional specification, proposing requirements to maintain a local clock, several features for timestamping of *Clock Synchronization Packets* and all necessary interfaces for GPS-coupling. The chip is decomposed into physical units in Section 3 along with an explanation of implementation details. After presentation of some general chip data, some conclusions and directions of further research are appended in Section 4.

2 Functional Specification

The UTCLIENT is designed to operate close to a *Communication Coprocessor* (COMCO), that are available for a wide variety of networks, ranging from fieldbusses like Profibus over Ethernet based networks up to advanced high-speed FDDI or ATM networks. Beside it is accessible by processors, employing data bus widths of 8, 16 or 32 bits and big/little endian byte ordering without additional external logic. The CPU runs the clock synchronization software, that requires a local time and access to external time sources, which is provided via the UTCLIENT. Thus the UTCLIENT must provide a versatile interface to commercially available systems and GPS timing-receivers.

The centerpiece of the UTCLIENT is a local clock, that follows the NTP-time format from [Mil91]. This fixed point representation of the current time utilizes 32 bits for the integer part that are interpreted as standard seconds relative to UTC, and another 32 bits give the associated fractional part.
The local clock can be paced with any oscillator frequency f_o in the range of 1...20 MHz, it is fine-grained rate adjustable in steps of about 30 ps/s, and supports state adjustment and leap second corrections in hardware. Those features are easily implemented internally due to an *adder-based* clock design. The basic idea is to use a large adder (91 bit) for summing up the elapsed time between succeeding oscillator ticks. Of course a proper value for the augend must be provided by the atop running clock synchronization algorithm to gain the desired rate adjustment.
It might appear that such a design is too much of a good thing for 1 μs worst-case precision. But in fact, a detailed analysis of clock synchronization in [SS97], revealed that clock granularity and rate correction uncertainities have tremendous

effects upon an achievable worst-case precision.
When UTC is considered as reference timescale, because of its official character and the worldwide availability through systems as for e.g. GPS, attention must be paid in conjunction with real-time systems due to its non-chronoscopic nature. Therefore strict chronoscopic time standards, such as *Temps Atomique International* (TAI) or *GPS-Time*, should be given preference. Nevertheless the UTCLIENT supports insertion or deletion of pending leap-seconds to follow UTC.

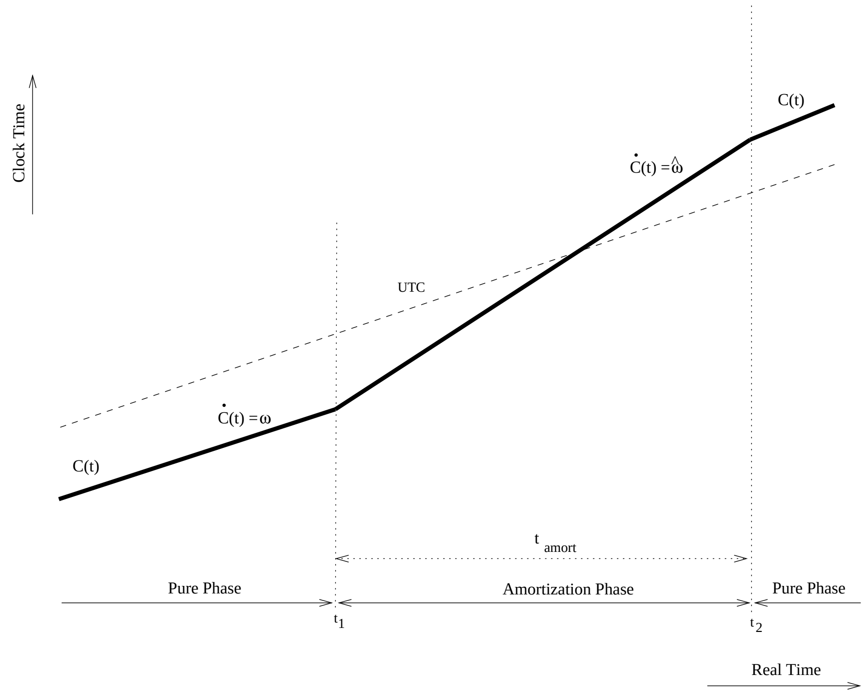


Figure 1: *Continuous Amortization*

Real-time applications dictate specific properties on correction of time, since timestamps from local clocks need to be monotonic and continuous. The technique of *linear continuous amortization* introduced in [SC90] for a software-based clock synchronization is adapted for the UTCLIENT, that provides hardware support for this clock setting.
Instead of adjusting the clock state instantaneously, the clock rate for a specific amount of time is modified. Thus during the *amortization phase* the slope of clock $C_i(t)$ differs from that during *pure phase*, cf. Figure 1. These two phases alternate perpetually, controlled by the clock synchronization algorithm. The latter also determines the proper rate $\hat{\omega}$ for the amortization phase that commences at t_1, and the pure rate $\dot{C}(t) = \omega$ that advances clock C(t) in pure phase before t_1 and after t_2.

Up to now we considered elements to maintain a local clock, which can be used for event timestamping and generating purposes. Exact timestamping of *Clock Synchronization Packets* (CSP) at both sending and receiving side virtually influences the quality on clock synchronization. This requirement is justified by the well known result of [LL84], saying that even n ideal clocks cannot be synchronized with a precision less than $\epsilon(1-1/n)$ in case of transmission/reception uncertainity ϵ. Hence CSPs need to be stamped with local time just the moment they are actually leaving the node resp. arriving at one. Coordinated hardware support from the UTCLIENT and the surrounding hardware, in particular, the COMCO is required. The UTCLIENT is notified of CSP transmission/reception via dedicated input lines and saves the current time into corresponding registers. A similar method is used to link an external time source, e.g. the GPS system to the UTCLIENT. Thus the *one pulse per second* (1pps) signal, that is provided by most commercially available receivers is used to trigger a timestamp. This signal marks the beginning of a second, while the associated information identifying the pulse along with other data is provided afterwards via a serial interface.

3 Unit Implementation

Figure 2 portrays several internal units of the UTCLIENT along with some of their interfaces. The interrupt lines provided by the UTCLIENT are drawn next to the system bus on the left side, and signal lines for GPS-coupling and CSP stamping are shown on the right.

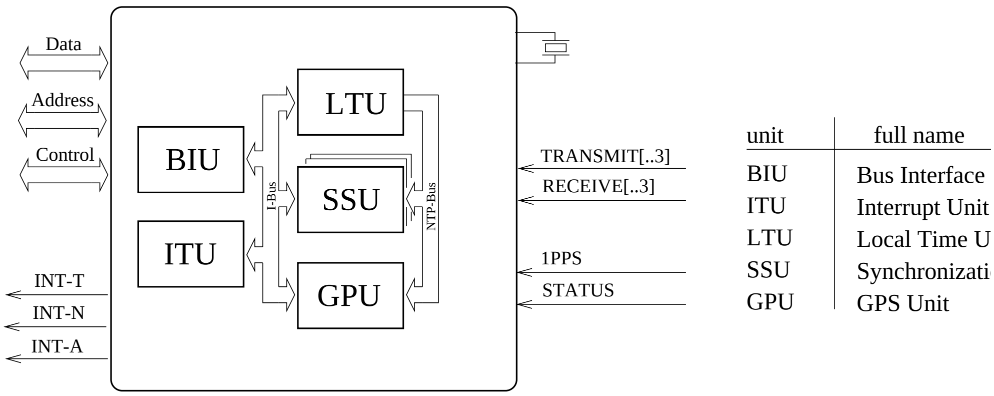


Figure 2: *UTCLIENT block diagram*

The *Bus Interface Unit* (BIU) makes the UTCLIENT applicable to System-Bus widths of 8, 16 or 32 bits, little or big endian byte ordering and different access times. Therefore *dynamic bus sizing* is implemented as outlined in [Mot90].
Several internal interrupt sources are hardwired to three different interrupt outputs INTN (network-related), INTT (timer-related) and INTA (application-related). The interrupt logic along with enable/disable and status registers is incorporated in the *Interrupt Unit* (ITU).

The *Local Time Unit* (LTU) maintains a very fine-grained local clock, supports continuous amortization and insertion/deletion of leap seconds as well. Figure 3 is depicted to a short presentation of the LTUs architecture. —Therefore the following notation is used to denote a specific part of the NTP-time format: (u, v) covers bits ranging from the more significant bit position u to the less significant one v.—

The addend is formed as a positive feedback via a multiplexer `NTPMUX` and register `NTPTIME` holding local time. Initialization during startup takes place by loading a preload value, stored in the register set `MSSET`, `TSSET` and `USSET`, via `NTPMUX` to the local time register. The augend is composed of the $(-20, -51)$-registers `STEPPUREACT` resp. `STEPAMORT` during pure resp. amortization phase along with `STEPLOW`, which holds a value in the $(-52, -59)$-range in order to account for a non-binary oscillator frequency. The duration of the amortization phase is programmed into the 32 bit counter `AMORTTIMER`, that switches via the multiplexer `AMORTMUX` between these two clock-step values. The preload register `STEPPURE` is necessary to hold the clock step value for the following pure phase when the counter ceases to run.

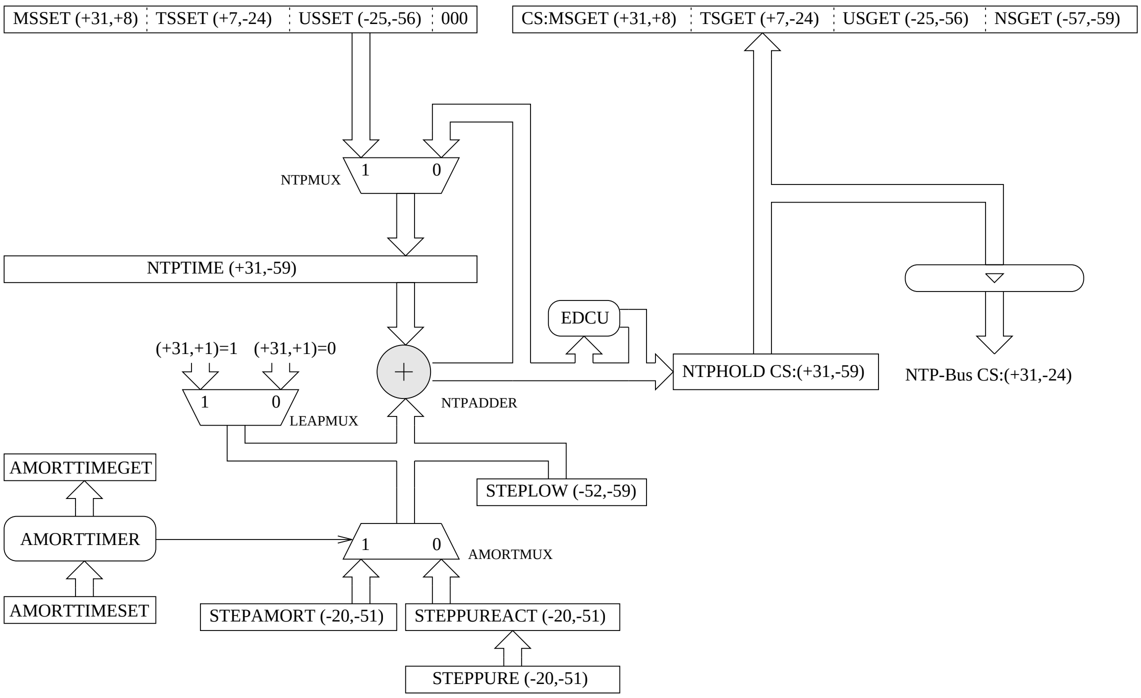


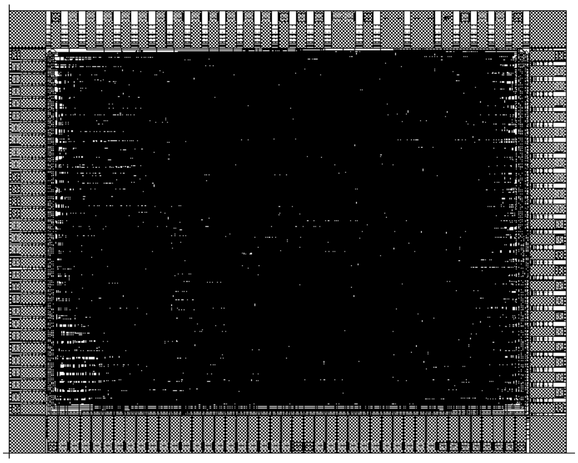
Figure 3: *Local Time Unit*

Leap seconds are inserted/deleted by switching multiplexer `LEAPMUX`, that affects the upper 32 bits of the augend.
An 8 bit Checksum, computed by the *Error Detection and Correction Unit* (EDCU), is stored along with bits $(+31, -24)$ of the result derived from the `NTPADDER` in register `NTPHOLD`. The value of this register is than fed, via the so-called NTP-Bus, to other units and presented to the exterior through register set `CS:MSGET`, `TSGET`, `USGET` and `NSGET`.

For application and CSP-timestamping purposes three *Synchronization Subnet Units* (SSU) are incorporated in the UTCLIENT. Each of these units comprises a register set for receive and transmit timestamping, that is performed by sampling the current value on the NTP-Bus into dedicated registers when a change of the signal value on the according input lines is dedected. Moreover two *duty timers* are introduced in each SSU, that perform a comparison of the NTP-Bus against a 48 bit preload value, implemented as a register-comparator couple. They raise an interrupt whenever the NTP-Bus becomes equal or greater to the preloaded value, given that the interrupt is enabled.

To facilitate external clock synchronization, an external time reference must be linked up to the UTCLIENT. The *GPS Unit* (GPU) couples GPS-receivers, by timestamping the 1pps signal. Therefore with the rising edge of this signal the value of the NTP-Bus is sampled into registers, thus providing the exact begin of a UTC second to the UTCLIENT. This value is than used by the synchronization algorithm to adjust the local clock.

To illustrate the implemenation of the UTCLIENT, the following graphic along with Table 1 presents some related chip data.



Technology:	CMOS 0.7μm
Manufacturer:	ES2: ECPD07
Tools:	Synopsys/Cadence
Transistors:	121.820
Area:	32.41 mm^2
Package:	PGA 180
Fabrication:	Sept. 9^{th},1996
Cost:	7291 ECU (10 pieces)

Table 1: *UTCLIENT chip data*

4 Conclusions

We presented an overview of an ASIC termed UTCLIENT depicted to support external clock synchronization with a precision in the 1 μs-range. We motivated the features packed into our chip and revealed details about their implementation up to some reasonable level.
In a forthcoming step we will pass on the design of the Universal Time Coordinated Synchronization Unit (UTCSU) to a manufacturer. This chip extends the functionality of the UTCLIENT in a way to support also an accuracy of 1 μs, thus hosting an interval around local time that includes UTC.

References

[LL84] J. Lundelius, N. Lynch. *An Upper an Lower Bound for Clock Synchronization*, Information and Control, vol. 62, p. 190–204, 1984.

[Loy96] D. Loy. *GPS-Linked High Accuracy NTP Time Processor for Distributed Fault-Tolerant Real-Time Systems,* PhD thesis, Vienna University of Technology, Department of Computer Technology, April 1996.

[Mil91] D.L. Mills. *Internet Time Synchronization: The Network Time Protocol*, IEEE Transactions on Communications, 39(10), p. 1482–1493, October 1991.

[Mot90] Motorola. *MC68030 Enhanced 32-bit Microprocessor Users's Manual*, Prentice Hall, 1990.

[Sch95] U. Schmid. *Synchronized Universal Time Coordinated for Distributed Real-Time Systems*, Control Engineering Practice 3(6), p. 877–884, 1995.

[SC90] F. Schmuck, F. Christian. *Conitnuous Clock Amortization need not affect the Precision of a CSA*, 9^{th} ACM Symposium on Principles of Distributed Computing, p. 133–143, 1990.

[SS95] K. Schossmaier, U. Schmid. *UTCSU Functional Specification*, Technical Report 183/1-56, Vienna University of Technology, Department of Automation, July 1995.

[SS97] U. Schmid, K. Schossmaier. *Interval-based Clock Synchronization*, Journal of Real-Time Systems 12:2, March 1996.

[SSHL97] K. Schossmaier, U. Schmid, M. Horauer, D. Loy. *Specification and Implementation of the Universal Time Coordinated Synchronization Unit (UTCSU)*, Journal of Real-Time Systems 12:3, May 1997.