A network interface for highly accurate clock synchronization

MARTIN HORAUER
Institute of Computer Technology

Technische Universität Wien
Gußhausstraße 25–27
A-1040 Vienna, Austria
tel: +43-1-58801-38416
fax: +43-1-58801-38499
e-mail: horauer@ict.tuwien.ac.at

NIKOLAUS KERÖ
Department of Industrial Electronics and Material Sience
Technische Universität Wien
Gußhausstraße 25–27
A-1040 Vienna, Austria
tel: +43-1-58801-36666
fax: +43-1-58801-36099
e-mail: keroe@tuwien.ac.at

ULRICH SCHMID
Department of Automation
Technische Universität Wien
Treitlstraße 1, A-1040 Vienna, Austria
tel: ++43-1-58801-18325, fax: ++43-1-58801-18391
e-mail: s@auto.tuwien.ac.at

Abstract

This paper describes a novel network interface hardware, which enables clock synchronization with sub-μs-range accuracy in network-coupled distributed systems. The basic idea, which works for any packet-oriented data network, is to timestamp data packets at the interface between physical layer transceiver and network controller upon packet arrival and departure. Our particular implementation targets the standardized Media Independent Interface (MII), which is used by almost any modern 10/100/1000 Mb/s Ethernet chipset. A custom FPGA intercepting the MII datastream is used for triggering timestamps and inserting them into the data packets. Local time is supplied by the high-resolution rate-adjustable adder-based clock of our UTCSU-Asic [SSHL97], which also contains all other hardware support required for interval-based external clock synchronization, like local accuracy intervals and interfaces to GPS receivers. Owing to this design, GPS time can be distributed over Ethernet without sacrificing the excellent time accuracy of modern GPS receivers.

Keywords: Networked distributed systems, Global Positioning System (GPS), Fast Ethernet, packet timestamping, Media Independent Interface (MII), Network Interface Card (NIC), Peripheral Component Interconnect (PCI).

1 Introduction

The interaction of components in a distributed system is usually much easier to coordinate if some sort of global time is available. In addition, if specific real-time constraints are to be met by the distributed system, global time is indispensable for timestamping external events, scheduling resources and initiating actions. A time service providing global time can either be implemented as a central time server accessible from all nodes or, preferably, as a distributed time service. In the latter case, each node i hosts a local clock $C_i(t)$, which is periodically adjusted in a way that guarantees $|C_i(t) - C_j(t)| \leq \pi \quad \forall t \geq t_0$, for any two fault-free nodes i and j in the system; π is usually called the worst case precision.

Since distributed applications usually have very different precision requirements, many different approaches for clock synchronization exist. They range from dedicated clocking lines up to purely software-based solutions, thereby spanning a range from ns up to ms for the achievable precision, see [SKMNCK99] for a survey. Still, a separate clocking network is usually only affordable if the interconnected nodes are only a few meters apart. For larger systems, clock synchronization must be accomplished by exchanging time information over data network connections. Most modern distributed systems, however, are based on shared broadcast LANs. Since this type of networks suffer from a considerable medium access uncertainty in the $1\ldots10$-ms-range, high-accuracy clock synchronization is difficult here due to the resulting high variability (uncertainty) of the transmission delays ε.

Fortunately, with moderate hardware support, ε and hence π can be brought down to the μs range. In the project SynUTC[1], we developed a dedicated hardware M-Module termed *Network Time Interface* (NTI), which ensures ε in the range of 1μs over Ethernet. It couples the local clock provided in a custom Asic termed *Universal Time Coordinated Clock Synchronization Unit* (UTCSU), cf. [Lo96], [SSHL97] and an Ethernet controller by means of shared memory: Timestamps are triggered and inserted in packet memory when the network controller grabs/deposits a packet there. Experiments revealed, however, that the internal packet FIFO's of the network controller prohibit a further reduction of ε for this memory-based timestamping method, see [SKMNCK99] for details.

In this paper, we present the implementation of our *Media-Independent Interface*-based timestamping method first proposed in [SHK99], which allows to further reduce the transmission delay uncertainty ε. In fact, by equipping every node of the distributed system with our *MII-Network Time Interface* (MII-NTI), we should be able to push ε (and hence π) down to the *ns*-range in Fast Ethernet networks. Prior to the illustration of the involved principles, we give a short overview of the architecture of a node and its MII-NTI. In the subsequent sections, we elaborate on the mechanisms required for MII-based timestamping and their implementation in a field programmable gate array. Some conclusions and further directions of improvement eventually round off the paper.

2 Node Architecture

The MII-NTI node architecture is outlined in figure 1. It consists of an off-the-shelf Network Interface Card (NIC) built around a 10/100 Mbit/s Fast Ethernet network controller with an integrated PCI interface, and a separate physical layer interface device. These two devices are interconnected

[1]The SynUTC-project received support from the Austrian Sience Foundation (FWF) grant P10244-ÖMA, the OeNB "Jubiläumsfonds-Projekt" 6454, the BMfMV research contract Zl.601.577/2-iV/B/9/96, the Gesellschaft für Mikroelektronik (GMe), and the START programme Y41-MAT. See http://www.auto.tuwien.ac.at/Projects/SynUTC/ for further information.

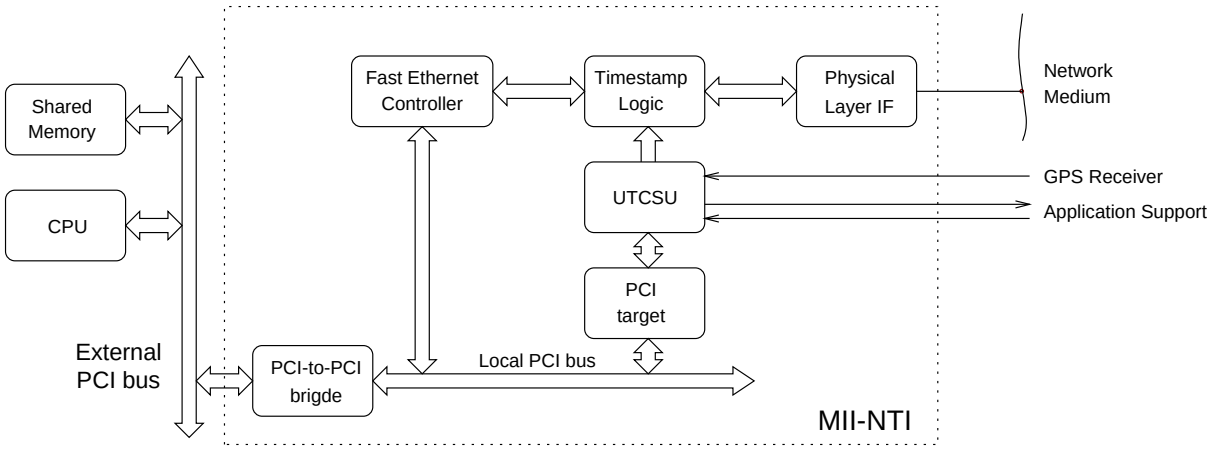

Figure 1: Node Architecture

via a standardized Media Independent Interface (MII) interface. The MII-based timestamping unit is implemented as a field programmable gate array and sits in between the two devices. It works as follows: A *clock synchronization packet* (CSP) is recognized by a special type field value and contains fields for a transmit and a receive timestamp; the latter must be left untouched (reserved) by the device driver. When a CSP is detected on transmission resp. reception, the transmit resp. receive timestamp is automatically inserted on-the-fly into the data stream. Local time is supplied by the UTCSU Asic, see [SSHL97], which maintains the node's local clock and accuracy information on behalf of a clock synchronization algorithm running on the CPU, cf. [SKMNCK99], [SS97].
However, since the UTCSU provides no PCI interface, a PCI target chip is required to access the many UTCSU registers from the CPU. To satisfy the rules of the PCI bus specification, a PCI-to-PCI bridge has to be incorporated between the PCI bus interfacing to the network controller/PCI target and the PCI bus on the CPU side.

3 Timestamping Mechanisms

The MII-based timestamping of Ethernet packets is the key idea of the presented MII-NTI. Figure 2

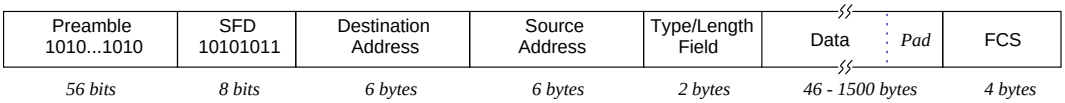

Figure 2: Ethernet/IEEE 802.3 Data Frame

illustrates an Ethernet data frame according to [Ieee85]. The preamble constitutes of 56 subsequent "1010.." bits followed by the Start Frame Delimiter (SFD).[2] The destination ethernet address field

[2]Sometimes the first six "101010" bits of the SFD are considered as being part of the preamble, which results in a SFD of only two bits constituting "11".

holds the address of the intended receiver; for broadcast addresses this field is all 1's. The source ethernet address field is the unique ethernet address of the sending station. The address fields are followed by the length field, which holds the number of data bytes within the frame (for IEEE 802.3) or the type descriptor of the packet (for Ethernet I & II). Data frames hold between 46 and 1500 bytes of data, while shorter frames must be padded to 46 bytes. Note that type field codes are larger than 1500, which allows both IEEE and Ethernet I/II frames to co-exist. The frame check sequence (FCS) is a 32 bit CRC calculated of destination/source address, type/length field, data and optional pad bytes using the Autodin-II polynomial:

$$g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Preamble, SFD, Source Ethernet Address and FCS are usually inserted by the Media Access Controller (MAC), whereas the destination address, the length/type field and the data is provided via a device driver routine from the local host processor.

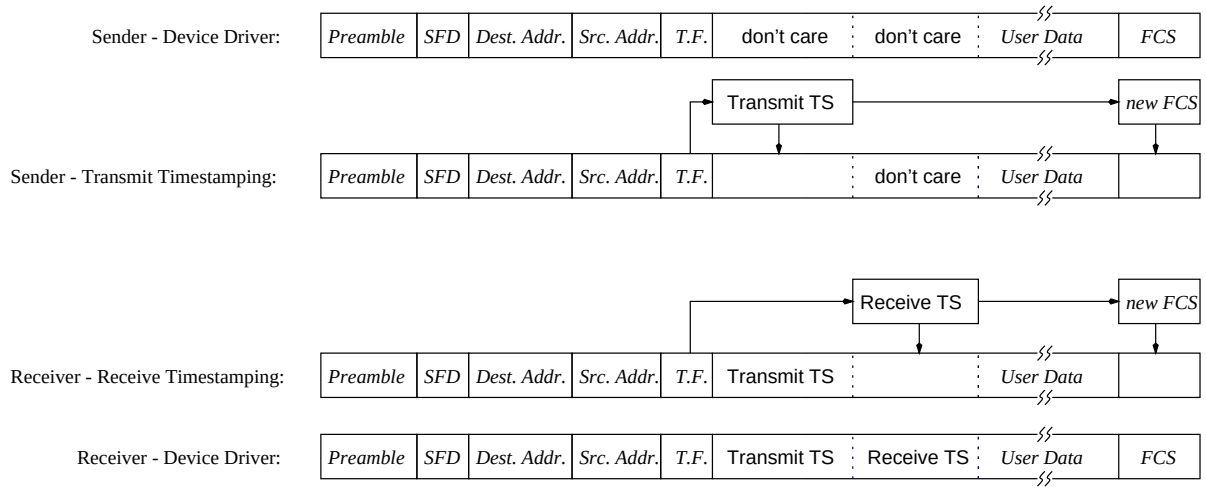

Figure 3: CSP timestamping

Timestamps are automatically inserted into the incoming resp. outgoing datastream when a Clock Synchronization Packet (CSP) is received resp. transmitted. Note that the CRC values are to be modified here as well to ensure correct packet transmission. Figure 3 shows the modification of a CSP when being sent resp. received. The MII-NTI hardware recognizes CSPs by their unique type field value; all other frames are passed through without modification.

4 Timestamp Logic

Figure 4 shows a block diagram of the timestamp logic, which is split into a transmit timestamp unit (TTU) and a receive timestamp unit (RTU). The TTU monitors the data-stream from the MAC unit of the Fast Ethernet controller to the physical layer device. When a CSP is sent, a timestamp is sampled from the UTCSU and transparently inserted into the outgoing packet, and the frame check CRC is adjusted accordingly. In the reverse direction, the RTU monitors the incoming data-stream. Upon CSP reception, a receive timestamp is sampled and inserted into the packet. When the original

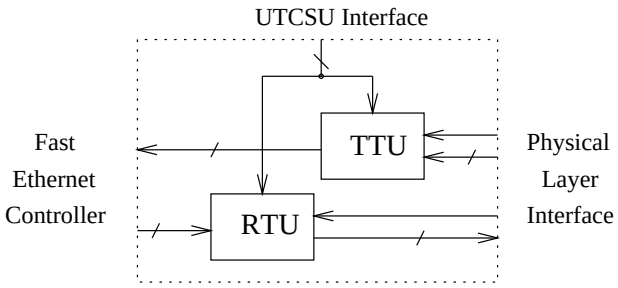

Figure 4: Timestamp Logic Block-diagram

CRC value was correct, a new CRC value is computed and inserted into the packet. Otherwise, the inverted new CRC value is passed on to the media acces unit (MAC) to allow for error detection within the MAC. Both units need access to the local time information provided by the UTCSU for timestamping purposes. Although the TTU and RTU have a similar layout, resource sharing between them is not possible since the Fast Ethernet Controller may occupy both channels simultaneously in full-duplex mode.

Transmit Timestamp Unit (TTU)

The task of the transmit timestamp unit is to filter outgoing packets for transmit clock synchronization packets (CSP). When a CSP is recognized by the TTU logic, a timestamp is pulled off the UTCSU NTP-interface, cf. [Lo96] and inserted into the CSP data field. A CRC module within the TTU re-calculates the checksum, based on the modified data, and replaces the CRC value in the packet. All other packets are simply passed through by the TTU. Figure 5 shows a block diagram of

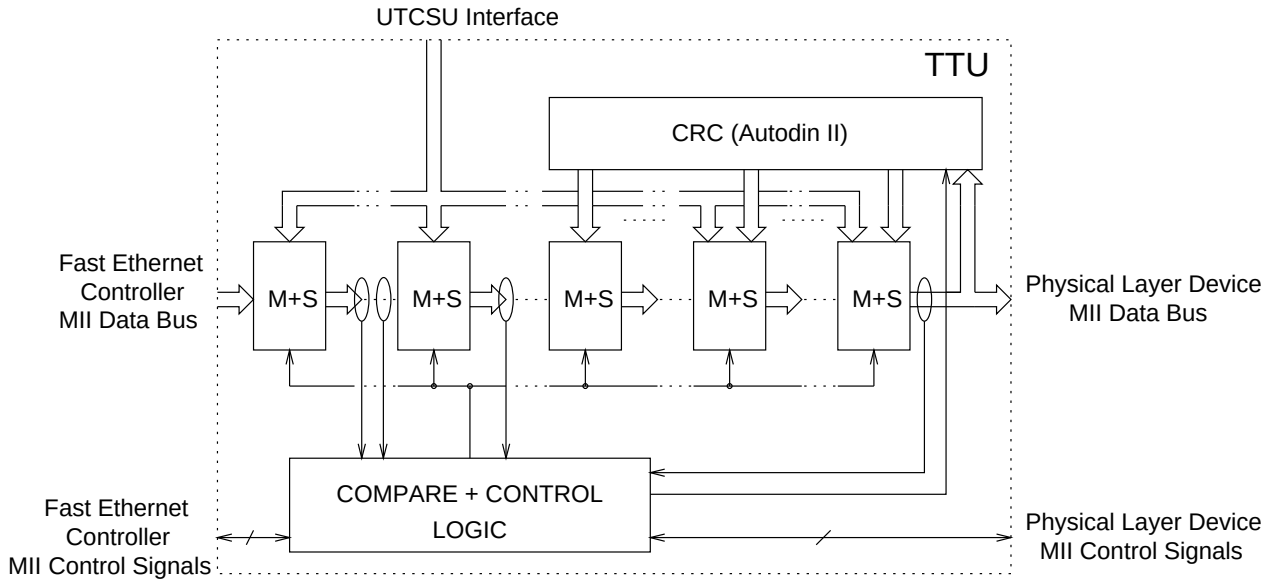

Figure 5: Transmit Timestamp Unit Block Diagram

the TTU. It consists primarily of several nibble wide multiplex and shift building blocks, a compare and control unit, and a CRC module. The compare and control unit detects the transmission of a CSP by analyzing the type field of every packet. The compare logic triggers when the unique type value of a CSP is recognized at a fixed offset from the start frame delimiter, which in turn is identified by the first “1011” sequence following the beginning of a frame. This trigger event is eventually used to feed the time information read from UTCSU NTP-bus into the multiplex and shift units, which finally insert the timestamp into the data stream. Last but not least, a new CRC value, calculated from the modified data, needs to be inserted into frame overwriting the previous checksum. Since the frame check sequence is calculated from the address, type, and data field, it must be initiated after the start frame delimiter has been detected and stopped after the last data nibble within the current frame. Insertion of the resulting CRC value into the outgoing datastream is done in the same way as for the timestamp.

Receive Timestamp Unit (RTU)

The receive timestamp unit performs similar operations as the transmit timestamp unit, although in the reverse direction. An additional CRC check is required in order to check for error free packet reception. When a frame is received, a CSP is recognized in the same way as within the TTU, by analyzing the type-field of the incoming packet. When a CSP is recognized, a receive timestamp is inserted at its appropriate location after the transmit timestamp. The unmodified data is additionally fed to a CRC unit, which calculates the checksum and compares it with the checksum included in the frame check sequence field. When a mis-match is detected, the data is passed on to the MAC with a wrong CRC. Otherwise the checksum, calculated by a second CRC unit, replaces the one in the modified receipt frame.

Figure 6 shows a block-diagram of the RTU consisting of two CRC units, several multiplex and

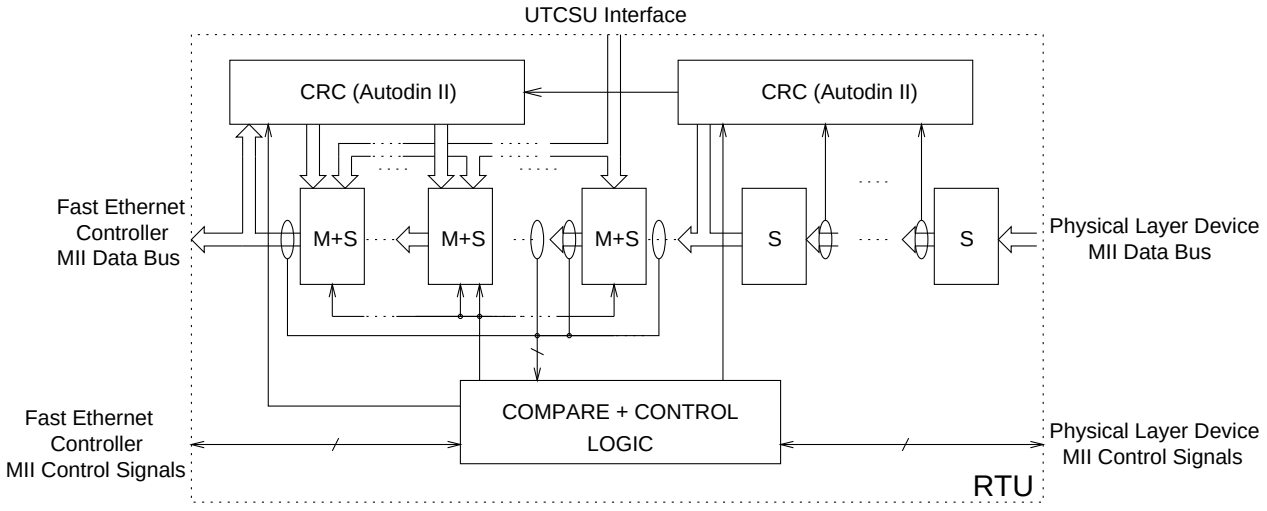

Figure 6: Receive Timestamp Unit Block Diagram

shift building blocks and a compare and control unit. The CRC module at the input stage of the RTU has some additional compare logic built-in. This logic is used to compare the CRC calculated from the destination and source address, type, and data fields with the received frame check sequence.

The result of this comparison is fed to the compare and control logic in order to decide whether the CRC unit at the output stage of the RTU should insert a correct CRC or an inverted (false) one into the modified CSP frame.

Cyclic Redundancy Check Module

As mentioned earlier, the Autodin-II polynomial is used for frame check sequence generation, see [Ro96] for an informal introductory tutorial. Destination, source address, type, and data field are used as input for this CRC polynomial. The computed checksum is appended at the end of the Ethernet frame. At the receiver, the same CRC is calculated and matched against the transmitted CRC value. The steps involved in encoding with k data bits, $n-k$ control bits and a generator polynomial $g(x) \in GF^2(x)$ of degree $r = n-k$ are:

1. Multiply the message $m(x)$ with x^{n-k}

$$x^{n-k}m(x) = \sum_{i=n-k}^{n-1} c_i x^i.$$

2. The above result is divided in $GF^2(x)$ by the generator polynomial g(x). The resulting remainder is

$$r(x) = \sum_{i=0}^{r-1} c_i x^i.$$

3. Finally the resulting code polynomial is constructed by

$$c(x) := x^{n-k}m(x) + r(x) = \sum_{i=0}^{n-1} c_i x^i$$

and transmitted to its destination address.

At transmission the receiver gets the polynomial $d(x)$ and performs the following decoding steps:

1. Divide $d(x)$ in $GF^2(x)$ by $g(x)$ in order to get the remainder r(x).
2. If $r(x) \neq 0$ report an error and request a re-transmission.
3. In case $r(x) = 0$ and $d(x) = \sum_{i=0}^{n-1} c_i x^i$ set

$$m(x) := \sum_{i=0}^{k-1} c_{n-k+i} x^i.$$

CRC codes are used for error detection and can be efficiently implemented in hardware. Since CRC is based on polynomial division, it is possible to compute n steps of the serial CRC scheme at once. A formal proof of this method is given in [BFGL96]. All CRC modules within the TTU and RTU operate with data nibbles since MII is a nibble wide interface.

5 Conclusion

We presented an overview of the prototype implementation of our MII-based Network Time Interface for PCI-based nodes, which facilitates high-accuracy time distribution in Ethernet-based distributed systems. By timestamping data packets at the standardized MII interface between Ethernet network controller and physical transceiver, a time distribution accuracy down to the 10 ns-range can be achieved in Fast Ethernet networks.

The research prototype of the MII-NTI was primarily designed for experimental evaluation purposes and can hence be improved in many ways. Apart from the fact that the pivotal UTCSU-Asic was designed solely for research purposes and hence contains many redundant blocks, its traditional bus interface requires a quite complex PCI board as well: A PCI target chip and a seperate PCI-to-PCI bridge is required on-board the MII-NTI to make the UTCSU registers accessible to the CPU.

In order to circumvent those shortcomings, we plan to re-design the UTCSU. More specifically, we will reduce its die size (currently $110mm^2$, implemented in $0.7\mu m$ ATMEL-ES2 process) by migrating to a smaller technology and by eliminating unnecessary redundancy. In addition, we will incorporate the MII-based timestamping logic provided by the FPGA directly in the UTCSU, and will add a programming interface that allows accessing the UTCSU registers via *data packets* as well. The latter will eliminate the need for a PCI target chip and the PCI-to-PCI bridge on the MII-NTI, will allow re-use of existing network controller device drivers, and will finally open up many interesting possibilities for remote clock synchronization.

References

[BFGL96] M. Braun and J. Friedrich and Th. Grün and J. Lembert. *Parallel CRC Computation in FPGAs*, Field Programmable Logic - Smart Applications, New Paradigms and Compilers - 6^{th} International Workshop on Field-Programmable Logic and Applications pp.156–165, Darmstadt Germany, September 1996.

[HSS98] M. Horauer and U. Schmid and K. Schossmaier, *NTI: A Network Time Interface M-Module for High-Accuracy Clock Synchronization*, Proceedings of the 6th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS), Orlando Florida, March 30 – April 3 1998.

[Lo96] D. Loy, *GPS-Linked High Accuracy NTP Time Processor for Distributed Fault-Tolerant Real-Time Systems*, Faculty of Electrical Engineering, Vienna University of Technology, April 1996.

[Ieee85] ANSI/IEEE Std.802.3-1985, *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specification*, IEEE Computer Society, 1985.

[Ro96] N.W. Ross, *A Painless Guide to CRC Error Detection Algorithms*, Rocksoft PTY Ltd. Hazelwood Park, Australia, 1996.

[SN99] U. Schmid and H. Nachtnebel, *Experimental evaluation of high-accuracy time distribution in a COTS-based Thernet LAN*, Proc. 24th IFAC/IFIP Workshop on Real-Time Programming (WRTP'99), pages 59–68, Schloß Dagstuhl, May/June 1999.

[SKMNCK99] Ulrich Schmid and Johann Klasek and Thomas Mandl and Herbert Nachtnebel and Gerhard R. Cadek and Nikolaus Kerö, *A Network Time Interface M-Module for Distributing GPS-time over LANs*, J. Real-Time Systems vol. 18 no. 1, 2000.

[SS97] U. Schmid and K. Schossmaier, *Interval-based Clock Synchronization*, Journal of Real-Time Systems vol. 12 no. 2 pp.173–228, March 1997.

[SSHL97] K. Schossmaier and U. Schmid and M. Horauer and D. Loy, Specification and Implementation of the Universal Time Coordinated Synchronization Unit (UTCSU), Journal of Real-Time Systems vol. 12 no. 3 pp. 295–327, May 1997.

[SHK99] Ulrich Schmid, Martin Horauer, and Nikolaus Kerö. How to distribute GPS-time over COTS-based LANs. In *Proceedings of the 31th IEEE Precise Time and Time Interval Systems and Application Meeting (PTTI'99)*, Dana Point, California, December 1999. (to appear).