



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

Time and Frequency Transfer in Local Networks

by

Jiří Dostál

A dissertation thesis submitted to
the Faculty of Information Technology, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

Dissertation degree study programme: Informatics
Department of Computer Systems

Prague, August 2018

Supervisor:

RNDr. Ing. Vladimír Smotlacha, Ph.D.
Department of Computer Systems
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Copyright © 2018 Jiří Dostál

Abstract

This dissertation thesis deals with these topics: network protocols for time distribution, time transfer over optical fibers, comparison of atomic clocks timescales and network time services. The need for precise time and frequency synchronization between devices with microsecond or better accuracy is nowadays challenging task from both scientific and engineering point of view. Precise time is also the base of global navigation system (e.g. GPS) and modern telecommunications. There are also new fields of precise time application e.g. finance and high frequency trading. The objective is achieved by two different approaches: precise time and frequency transfer in optical fibers and network time protocols. Theoretical background and state-of-the-art is described: time and clocks, overview of network time protocols, time and frequency transfer in optical fibers and measurements of time intervals. The main results of our research is presented: IEEE 1588 timestamper, atomic clock comparison, architectures for precise time measurements, running processor on external frequency, long distance evaluation of IEEE 1588 performance and time services in CESNET network. At the end is mentioned the ongoing and proposed work.

Keywords:

FPGA, IEEE 1588, atomic clock, time interval counter, transparent clock, optical network, timescale.

Abstrakt

Tato disertační práce se zabývá těmito tématy: síťovými protokoly pro přenos času, přenosem času pomocí optických vláken, vzdáleným porovnáním časových stupnic atomových hodin a síťovými časovými službami. Potřeba přesné synchronizace s mikrosekundou nebo lepší přesností je dnes náročným úkolem z hlediska vědeckého i inženýrského. Přesný čas je také základem navigačních systémů (např. GPS) a telekomunikačních technologií. Existují také nové oblasti aplikace přesného času, např. finanční systémy a vysokofrekvenční obchodování na burze. Cíle práce jsou dosaženy dvěma různými přístupy: přesným přenosem času a frekvence pomocí optických vláken a síťových časových protokolů. Je popsán teoretický základ a současná řešení: teorie času, přehled síťových časových protokolů, přenos času a frekvence v optických vláknech a měření časových intervalů. Hlavními výsledky našeho výzkumu jsou: IEEE 1588 timestamper, porovnání časových stupnic vzdálených atomových hodin, architektury pro přesné měření času, napájení hodin procesoru externí frekvencí, vyhodnocení provozu protokolu IEEE 1588 v síti CESNET. V závěru jsou popsány výsledky a navržena možná navazující práce.

Klíčová slova:

FPGA, IEEE 1588, atomové hodiny, čítač časových intervalů, transparentní hodiny, optická síť, časová stupnice.

Acknowledgements

I would like to express my special thanks of gratitude to my supervisor Vladimír Smotlacha who gave me the golden opportunity to do this interesting research in the field of time-keeping. I would also especially like to thank my wife for her endless support and empathy during dealing with this demanding work. Finally, I would like to thank my colleagues and all friends who helped me a lot to finish this thesis.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Problem Statement	1
1.3	Related Work/Previous Results	2
1.4	Goals of the Dissertation Thesis	2
1.5	Structure of the dissertation thesis	3
2	Background and State-of-the-Art	5
2.1	Time and Clocks	5
2.2	Network Time Protocols Overview	7
2.2.1	Daytime Protocol	7
2.2.2	Time Protocol	7
2.2.3	Network Time Protocol	7
2.2.4	IEEE 1588	9
2.2.5	White Rabbit	14
2.3	Time Transfer in Optical Fiber	15
2.4	Time Interval Measurement	17
3	Overview of Our Approach	21
3.1	Time Transfer over Optical Network	21
3.2	Timestamping	23
4	IEEE 1588 Timestamper	25
5	Atomic Clock Comparison	29
5.1	Adapter	29
5.2	Time Interval Counter	30
5.2.1	Coarse counter	31
5.2.2	Interpolator	32

5.3	Time scale comparison	33
5.4	Embedded counter evaluation	35
5.5	Summary	36
6	Dual Interpolating Counter Architecture	37
6.1	Interpolator Feed	37
6.2	Dual Interpolator Design	39
6.3	Calibration Method	39
6.4	Summary	40
7	Architecture for Precise Time Measurements	43
7.1	Timescale comparison method overview	43
7.2	Timestamping	44
7.2.1	Multiple Channels	44
7.2.2	Priority Encoder	44
7.3	Zynq SoC platform	44
7.4	Summary	45
8	Running Processor on External Frequency Source	47
9	Long Distance IEEE 1588 Evaluation	51
10	Time Services in CESNET Network	55
10.1	Accurate time and frequency source	55
10.2	NTP – Network Time Protocol	55
10.2.1	Hardware	56
10.2.2	Software	58
10.2.3	Performance	59
10.3	TSA – Timestamp Authority	59
10.3.1	TSA Hardware	60
10.3.2	Clock Synchronization	61
10.4	IEEE 1588	62
10.5	Time and Frequency Transfer Infrastructure	62
10.5.1	Summary	64
11	Future Work	65
11.1	New Generation of NTP Servers Platform	65
11.2	Transparent Clock with Deterministic Delay	65
11.3	Expanding the System on Chip Design	66
12	Conclusions	69
12.1	Comparison of Goals and Achieved Results	69
12.2	Original Results	70

Bibliography	71
Reviewed Publications of the Author Relevant to the Thesis	75
Remaining Publications of the Author Relevant to the Thesis	77
Supervised Publications Relevant to the Thesis	79
A Laboratory Equipment	81
B Working with Socket Options	83
C Processor External Frequency Source	89
D IEEE 1588 PTPmon Monitoring Tool	91
E Pipelined Priority Encoder Source Code	93

List of Figures

2.1	NTP operation principle	8
2.2	PTP packet format	10
2.3	Master–Slave PTP hierarchy	11
2.4	Latency origin	12
2.5	PTP packet timestamping	13
2.6	PTP messages transfer	13
2.7	End-to-end transparent clock	14
2.8	End-to-end transparent clock network	14
2.9	Peer-to-peer transparent clock	15
2.10	Peer-to-peer transparent clock network	15
2.11	White Rabbit switch in a laboratory	16
2.12	Unidirectional fiber-optic transfer	16
2.13	Bidirectional fiber-optic transfer	17
2.14	Coarse counter principle	17
2.15	Coarse counter errors	18
2.16	Vernier method	19
2.17	Tapped delay line method	19
3.1	Time transfer method	21
3.2	Time transfer between Prague and Vienna	22
4.1	Simplified block diagram of the IEEE 1588 timestamper.	25
4.2	PTP timestamp generation model	26
4.3	Simplified block diagram of the FPGA counter	26
4.4	IEEE 1588 timestamper evaluation	27
5.1	Time and frequency transfer adapter	30
5.2	The first generation adapter scheme.	30
5.3	Virtex-5 XC5VLX50T FPGA floorplan	31
5.4	Simplified block diagram of the FPGA counter	32

5.5	TIC evaluation	33
5.6	Time stability	34
5.7	Absolute error of measurements	34
5.8	Embedded FPGA counter performance	35
6.1	FPGA counter with carry chain interpolation	38
6.2	Interpolator feed input logic	39
6.3	Dual interpolating counter floorplan in Virtex-5 FPGA	40
7.1	Pipelined priority encoder	45
7.2	Delay line signal propagation – unsorted	46
7.3	Delay line signal propagation – sorted	46
8.1	Pierce CMOS oscillator	47
8.2	XTAL pins measurement	48
8.3	External source signals measurement	48
8.4	External clock source	49
9.1	Unicast PTP communication Erlangen–Prague	51
9.2	Unicast PTP communication Prague(data center)–Prague(lab)	52
9.3	Multicast PTP communication Prague(data center)–Prague(lab)	53
9.4	Multicast SolarFlare PTPd	53
10.1	OCXO module	57
10.2	PPS capture card diagram	58
10.3	Crystal frequency offset	59
10.4	OCXO frequency offset	59
10.5	NTP server internals	61
10.6	TSA clock accuracy	62
10.7	Single-fiber lines	63
10.8	Time stability	64
11.1	SoC block diagram	66
11.2	ZedBoard	67
11.3	Mini-ITX platform	68
A.1	Meinberg M600	81
A.2	5071A Atomic Clock	81
A.3	SR620 Counter	82
A.4	Meinberg PTP207PEX	82
A.5	Pendulum CNT–91 Counter	82
C.1	IDT frequency source schematics	89
C.2	SiLabs frequency source schematics	90

List of Tables

2.1	Timescales used by computer and/or network protocols	6
2.2	PTP multicast group IP addresses	12
9.1	IEEE 1588 measurements evaluation	54

Abbreviations

1PPS	1 Pulse Per Second
ACONet	Austrian national research and education network
ASCII	American Standard Code for Information Interchange
ASIC	Application Specific Integrated Circuit
BC	Boundary Clock
BEV	Bundesamt für Eich- und Vermessungswesen
BMC	Best Master Clock algorithm
CESNET	Czech Education and Scientific NETwork
CTU	Czech Technical University in Prague
CV	Common View
DWDM	Dense Wavelength Division Multiplexing
E2E	End-to-End
FAU	Friedrich-Alexander-Universitaet Erlangen-Nuernberg
FEE	Faculty of Electrical Engineering
FIFO	First In First Out
FPGA	Field-Programmable Gate Array
GMT	Greenwich Mean Time
GPS	Global Positioning System
I/O	Input/Output
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IP Core	Intellectual Protperty Core
ISO	International Organization for Standardization
ITU	International Telecommunication Union
LAN	Local Area Network
MAC	Medium Access Control
MAN	Metropolitan Area Network

LIST OF TABLES

NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
OCXO	Oven-controlled Crystal Oscillator
OSI	Open Systems Interconnection
P2P	Peer-to-Peer
PDU	Protocol Data Unit
PHY	Physical layer
PPP	Precise Point Positioning
PTP	Precision Time Protocol
Rb clock	Rubidium atomic clock
RFC	Request For Comments
SFP	Small Form-factor Pluggable
SI	International System of Units
TC	Transparent Clock
TCP	Transmission Control Protocol
TCXO	Temperature-compensated Crystal Oscillator
TIC	Time Interval Counter
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
XO	Crystal Oscillator

Introduction

1.1 Motivation

Time is a SI base physical quantity and has very broad area of influence for all people and application fields. The need for a precise time and frequency synchronization between devices with microsecond or better accuracy is nowadays challenging task for both scientific and engineering point of view. There are also new fields of the precise time applications, e.g., finance and high frequency trading. Timekeeping is a specialized branch that deals with the precise time management. As we have the precise time, another problem is the distribution of this time to other timekeeping devices. Many methods of time transfer are employed (e.g., satellite transmission). In recent times, a new method is proposed – time transfer over universal optical networks.

1.2 Problem Statement

Time Protocols In the scope of computer networks the NTP protocol is the most dominant but does not require such a strict resolution of timestamps. Another case is a modern IEEE 1588 protocol also known as a Precision Time Protocol (PTP) which operates with seconds/nanoseconds resolution of timestamps. The research is focused also on the PTP protocol especially on the time distribution infrastructure. A transparent clock (TC) node is a part of PTP hierarchy but in the present days TC are not fully available with desired quality of operation. Another area for HW-based precise time measure systems are specialized applications for a time distribution, e.g., for a time scale comparison between distant atomic clocks.

Clock Comparison Nowadays, an accurate time signal is typically acquired from the global positioning system (GPS). The Common-View GPS satellite method [1] is used for the atomic clocks time-scale calibration as well. Since the GPS time transfer is prone to the accuracy degradation at distances over 1000 km or there are issues with a GPS antenna

or receiver installation, an alternative method has been developed – the precise time and frequency transfer in optical networks. A fiber optic based network can carry a signal up to 2000 km utilizing only an optical amplification.

Timestamping Modern applications for the time distribution demand a precise time-stamping of external asynchronous events. There is a need for a sub-nanosecond resolution in such cases – this means that the required timestamp resolution is below the clock period of most digital systems. Generally, it is necessary to generate and evaluate timestamps with a time interval which is shorter than a system clock period. Examples of the precise time-stamping are embedded interval counter in a control systems or a network-based application for the time distribution (details about HW support in [A.12]).

1.3 Related Work/Previous Results

Problems mentioned in this Dissertation thesis are related to the time and frequency transfer over optical networks (chapter 3.1) and TCP/IP network based time synchronization: protocol IEEE 1588 (in chapter 2.2.4) and NTP (in chapter 2.2.3).

1.4 Goals of the Dissertation Thesis

1. Distant atomic clock time scale comparison over optical link

Research and development of proposed time and frequency transfer over optical link method. This means dealing with asynchronous FPGA design to develop an embedded time converter suitable to measure time events on optical fiber and subsequent time data processing.

2. IEEE 1588 evaluation

There are plenty of existing IEEE 1588 enabled products (computer network cards, dedicated servers, software solutions...). Despite the IEEE 1588 specification is widely implemented, the real behavior and performance of IEEE 1588 systems is not well examined. This goal includes performance testing and evaluating of existing products in various setups and potential development of supportive tools.

3. Contribution to CESNET network time services

Beside the newly deployed time and frequency transfer over optical links, there is a long tradition of CESNET network time services originating in 1999 with launching first NTP servers. E.g., after years, there is a need of a new NTP platform development, as the original systems tend to be obsolete. The goal is to contribute with new time network services.

1.5 Structure of the dissertation thesis

The dissertation thesis is organized into chapters as follows:

1. *Introduction*: Describes the field of timekeeping and main problems that are necessary to deal with.
2. *Background and State-of-the-Art*: Introduces the reader to the necessary theoretical background and surveys the current state-of-the-art.
3. *Overview of Our Approach*: Describes an overview for solving the goals of the
4. *Main Results*: In chapters from 4 to 10 are described results and contributions of our research.
5. *Future Work*: Describes ongoing research, subsequent tasks and near future research topics.
6. *Conclusions*: Summarizes the results of our research, suggests possible topics for the further research and concludes the dissertation thesis.

Background and State-of-the-Art

2.1 Time and Clocks

Time is a fundamental phenomenon in the Universe and is dependent on frequency that describes periodically repeated events. Measurement of time is in reality counting of these periods. Device that measures time is called clock.

Development of ancient civilization, especially with respect to agriculture, was not possible without knowing seasons and shorter periods of year, so calendar systems played crucial role in that epoch. Later, it was also important to divide days into shorter units and clocks improved during human history as more advanced principles and more stable sources of frequency had been discovered. Since 17th century, when pendulum clocks reached the accuracy of about 10 second per day, we can speak about accurate clock defining own time scale.

In order to measure time we need a time standard that specifies rate at which time passes or points in time. Time standards were usually based on earth rotation. Typical example is Greenwich Mean Time (GMT) that is derived from solar time at the meridian passing Royal Greenwich Observatory. However, irregularities of Earth rotation were later discovered when clock stability were further improved and so call "constructed time standards" based on sets of atomic clocks were defined [2]:

Atomic Time with the unit of duration the Systeme International (SI) second TAI is the International Atomic Time scale, a statistical timescale based on a large number of atomic clocks.

Universal Time (UT) is counted from 0 hours at midnight, with unit of duration the mean solar day, defined to be as uniform as possible despite variations in the rotation of the Earth.

- UT0 is the rotational time of a particular place of observation. It is observed as the diurnal motion of stars or extraterrestrial radio sources.

2. BACKGROUND AND STATE-OF-THE-ART

- UT1 is computed by correcting UT0 for the effect of polar motion on the longitude of the observing site. It varies from uniformity because of the irregularities in the Earth's rotation.

Coordinated Universal Time (UTC) differs from TAI by an integral number of seconds. UTC is kept within 0.9 sec-onds of UT1 by the introduction of one-second steps to UTC, the "leap second." To date these steps have always been positive [2].

Name	Epoch – origin of the timescale	Coinciding Timescale	Leap Seconds	Used by
Unix time	0:0:0 of January 1, 1970	UTC	yes	Unix family operating systems
NTP time	0:0:0 of January 1, 1900	UTC	yes	NTP protocol
PTP time	0:0:0 of January 1, 1970	TAI	no	PTP (IEEE 1588) protocol

Table 2.1: Timescales used by computer and/or network protocols

Definition of second As a unit of time changed in history, corresponding stability of clocks:

- fraction of solar day: second is 1/86400 of mean solar day
- fraction of year: 1/31556925.9747 of the tropical year for 1900 January 0 at 12 hours ephemeris time (note: January 0 refers to the day before January 1)
- SI second: The duration of 9,192,631,770 cycles of radiation corresponding to the transition between two hyperfine levels of the ground state of cesium 133.

Atomic clocks Currently widely used accurate time and frequency references are atomic clocks utilizing very small changes of energy (that correspond to microwave radiation) in atoms of Cesium, Hydrogen or Rubidium, elements of the first group (alkali metals) of Periodic table. The Cesium clock this way directly realizes the SI second. Among these three elements, Cesium clocks have the best long-term stability, while Hydrogen masers have better short-term stability. Rubidium clocks are typically significantly less stable. Atomic time scales TAI and UTC are "coordinated" what means that they are constructed weighted average on large number (few thousands) of atomic clocks, mainly Cesium clocks and Hydrogen masers. These clocks are distributed around the world, each of them is monitored and its stability is calculated. Based on this observation, clock is evaluated and receives individual weigh. In recent years were built so called optical clock. The word "optical" means that they produce frequency in optical bandwidth (hundreds of terahertz) rather than in microwave bandwidth. Optical clocks have excellent stability (1E-17 and better) but until now they do not contribute to TAI or UTC as they are not able to run many months without interruption.

2.2 Network Time Protocols Overview

An idea of time distribution across a computer network is not new. There were many time protocol implementations in early era of internet. The well-known example of time protocol is NTP [3].

2.2.1 Daytime Protocol

This protocol is the oldest and obsolete one. It is defined in RFC 867 [4] and it was intended for measurement and testing purposes.

One daytime service is defined as a connection based application on TCP. A server listens for TCP connections on TCP port 13. Once a connection is established the current date and time is sent out the connection as a ASCII character string (and any data received is thrown away). The service closes the connection after sending the quote [4].

Another daytime service is defined as a datagram based application on UDP. A server listens for UDP datagrams on UDP port 13. When a datagram is received, an answering datagram is sent containing the current date and time as a ASCII character string (the data in the received datagram is ignored) [4].

There is no specific syntax for the daytime. It is recommended that it be limited to the ASCII printing characters, space, carriage return, and line feed. The daytime should be just one line. Example syntax: Weekday, Month Day, Year Time-Zone. The message is then transmitted as “Tuesday, February 22, 1982 17:37:43-PST” [4].

2.2.2 Time Protocol

Time protocol is another example of an obsolete time synchronization approach (defined as RFC 868 [5]). It is very simple protocol with resolution of one second and without mechanism for a path delay compensation.

Basically, the server listens for a datagram on port 37. When a datagram arrives, the server returns a datagram containing the 32-bit time value. If the server is unable to determine the time at its site, it should discard the arriving datagram and make no reply. The time is the number of seconds since 00:00 (midnight) 1 January 1900 GMT, such that the time 1 is 12:00:01 am on 1 January 1900 GMT; this base will serve until the year 2036. For example: the time 2,208,988,800 corresponds to 00:00 1 Jan 1970 GMT [5].

2.2.3 Network Time Protocol

Since 1985 it is one of the oldest TCP/IP family protocol still in use. It is the most common time synchronization protocol as well. The original RFC number is 958 [3].

NTP protocol is intended for synchronizing a set of network clocks using a set of distributed clients and servers. NTP is built on the User Datagram Protocol [6], which provides a connectionless transport mechanism. It is evolved from the Time Protocol [5] and the ICMP Timestamp message [7] and is a suitable replacement for both [3].

2. BACKGROUND AND STATE-OF-THE-ART

In the NTP model a number of primary reference sources, synchronized by wire or radio to national standards, are connected to widely accessible resources, such as backbone gateways, and operated as primary time servers. The purpose of NTP is to convey timekeeping information from these servers to other time servers via the Internet and also to cross-check clocks and mitigate errors due to equipment or propagation failures. Some number of local-net hosts or gateways, acting as secondary time servers, run NTP with one or more of the primary servers. In order to reduce the protocol overhead, the secondary servers distribute time via NTP to the remaining local-net hosts [8].

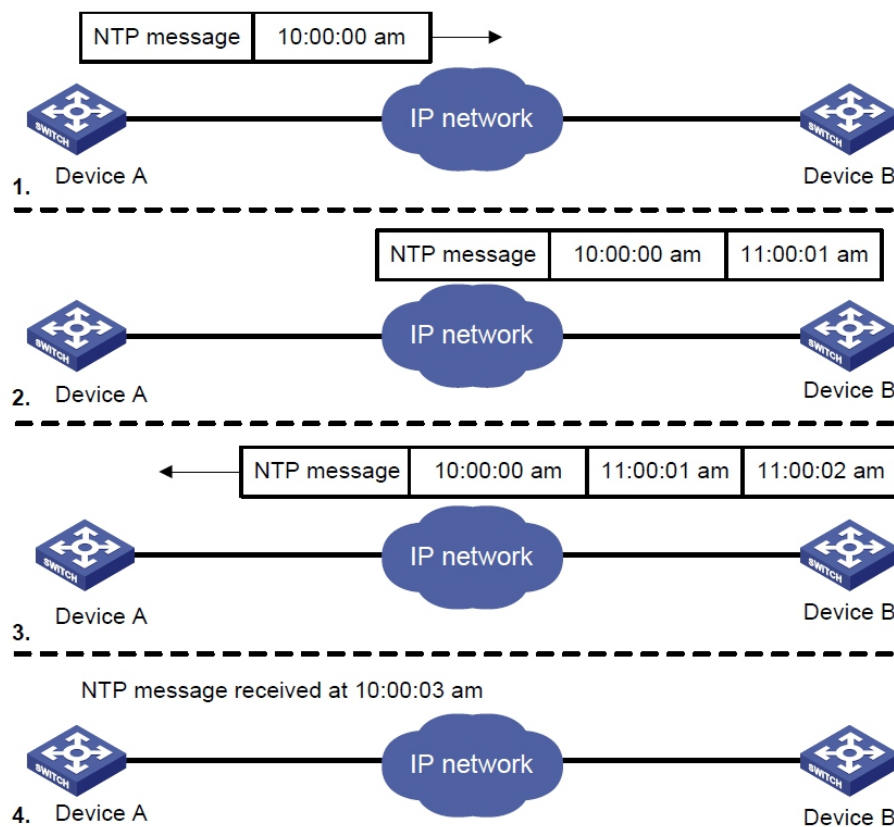


Figure 2.1: NTP operation principle [9].

In what may be the most common client/server model a client sends an NTP message to one or more servers and processes the replies as received. The server interchanges addresses and ports, overwrites certain fields in the message, recalculates the checksum and returns the message immediately. Information included in the NTP message allows the client to determine the server time with respect to local time and adjust the local clock accordingly. In addition, the message includes information to calculate the expected timekeeping accuracy and reliability, as well as select the best from possibly several servers [8].

The NTP operation principle is described in figure 2.1. The packet is timestamped in such a way:

- t_0 is the client's timestamp of the request packet transmission
- t_1 is the server's timestamp of the request packet reception
- t_2 is the server's timestamp of the response packet transmission
- t_3 is the client's timestamp of the response packet reception

The round-trip delay δ is computed as:

$$\delta = (t_3 - t_0) - (t_2 - t_1) \quad (2.1)$$

The offset θ is calculated as:

$$\theta = \frac{(t_1 - t_0) + (t_2 - t_3)}{2} \quad (2.2)$$

The frequency of the host system is then adjusted to minimize the offset. The synchronization is correct when both the incoming and outgoing routes between the client and the server have symmetrical nominal delay. If the routes do not have a common nominal delay, the synchronization has a systematic bias of half the difference between the forward and backward travel times [10].

2.2.4 IEEE 1588

The IEEE 1588 (also known as the “Precise Time Protocol” – PTP) protocol was developed for the need of precise time distribution with more precise synchronization (in comparison to the NTP protocol [3]). It is not a RFC by IETF but it is standardized under IEEE supervision. This protocol is intended to be a standard for devices connected via switched IP networks. The accuracy of synchronization is intended to be beyond one microsecond.

The current version of PTP protocol is number 2 from the year 2008. Here is an overview of the protocol and its features [11].

2.2.4.1 PTP Message Types

PTP messages are encapsulated in the standard TCP/IP hierarchy PDUs (mostly in UDP datagrams). The message format is in the figure 2.2

There are two classes of messages: event and general. Event messages are timed messages in that an accurate timestamp is generated at both transmission and receipt. General messages do not require accurate timestamps [13].

Event messages:

- Sync

2. BACKGROUND AND STATE-OF-THE-ART

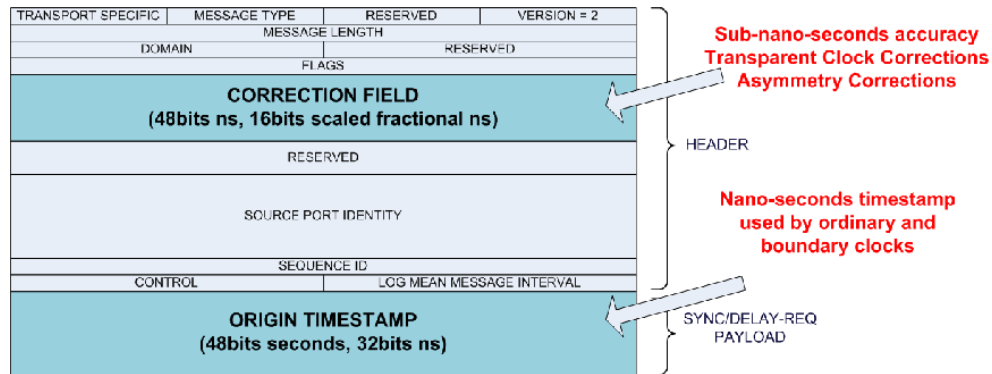


Figure 2.2: PTP packet format. [12]

- Delay_Req
- Pdelay_Req
- Pdelay_Resp

General messages:

- Announce
- Follow_Up
- Delay_Resp
- Pdelay_Resp_Follow_Up
- Management
- Signaling

2.2.4.2 Clock Synchronization

According to [11], the synchronization mechanism is based on a master/slave protocol (in figure 2.3). PTP instances exchange messages in order to determine the offset between master and slave clocks but also the message transit delay through the network.

The first task, called syntonization, is responsible to run the slave clock at the same speed as the master. This is achieved by sending a continuous flow of Sync messages from master to slave. Send time t_1 and receive time t_2 of these Sync messages are measured with the local clocks and processed by the slave. The slave's clock has to be adjusted until time intervals are equal on both clocks. There are two options to transport timestamp t_1 to the slave: two-step clocks use the separate Follow_Up message, while one-step clocks deliver t_1 with the Sync message itself. This option requires that the master is capable to insert t_1 into the Sync message on the fly. Because oscillators are susceptible to environmental

changes, Sync messages are sent continuously at a constant rate of typically one or a few messages per second.

The second task determines the slave's offset from the master, i.e. the difference of time of day between master and slave. This is achieved by measuring the two-way delay (round trip time). For the downlink, t_1 and t_2 are available from the last Sync. The uplink is measured with a Delay_Req message providing timestamps t_3 and t_4 . The Delay_Resp message is used to bring t_4 to the slave. Under the assumption of a symmetric transmission path for Sync and Delay_Req, one-way delay and offset from master are computed according to:

$$\begin{aligned} Delay &= [(t_2 - t_1) + (t_4 - t_3)]/2 \\ Offset &= [(t_2 - t_1) - (t_4 - t_3)]/2 \end{aligned} \quad (2.3)$$

Since environmental conditions can change, a continuous correction of the slave clock is required. For this purpose the slave is controlled by a servo loop (typically a PI loop minimizing the slave's deviation).

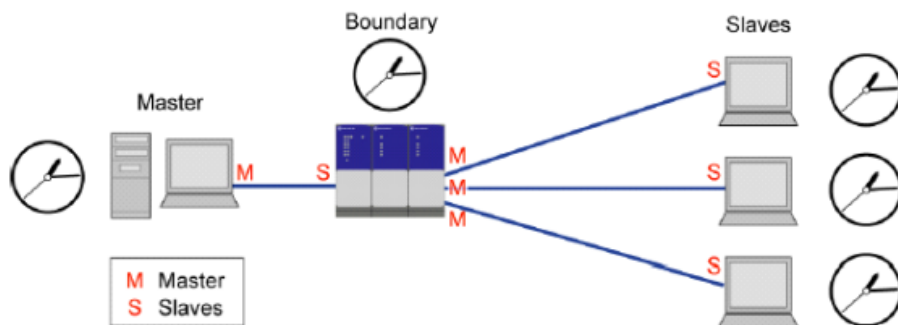


Figure 2.3: Master-Slave PTP hierarchy [14].

2.2.4.3 PTP network

PTP messages are sent to reserved multicast addresses (see table 2.2). Therefore PTP clocks do not need an individual IP configuration. When the delay of the Sync path varies due to queuing in bridges, the individual measurement results are not very useful. One approach to overcome this problem is the concept of PTP-aware bridges, known as Boundary Clocks (BC). A BC is a bridge equipped with a PTP clock synchronized by the master over one of its ports. Over the other ports, the BC synchronizes slave clocks attached to it. Such a configuration represents a synchronization hierarchy which is established automatically by the so called Best Master Clock (BMC) algorithm. This algorithm takes clock quality and priority settings into account and guarantees that the best available master, the Grandmaster, is the root of the synchronization tree [11].

2. BACKGROUND AND STATE-OF-THE-ART

Messages	IPv4	IPv6
All except peer delay messages	224.0.1.129	FF0x::181
Peer delay messages: Pdelay_Req, Pdelay_Resp and Pdelay_Resp_Follow_Up	224.0.0.107	FF02::6B

Table 2.2: PTP multicast group IP addresses

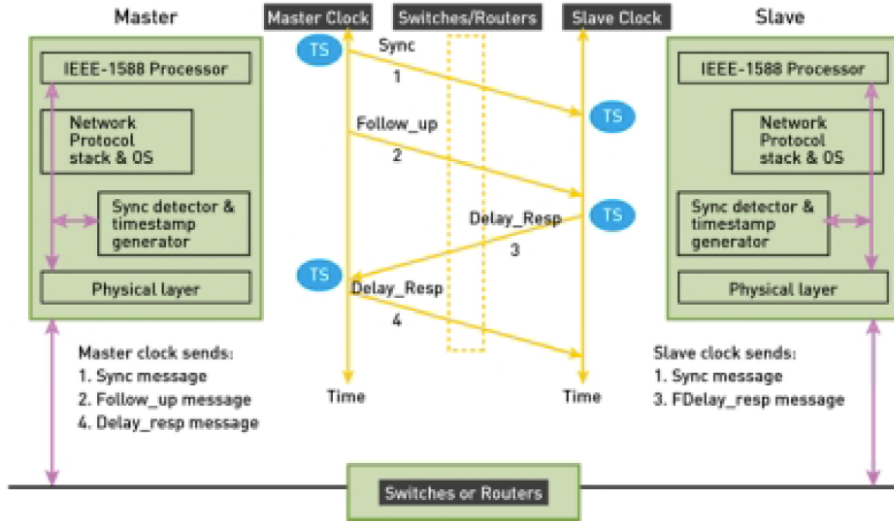


Figure 2.4: Latency origin in a host operating system [12].

2.2.4.4 Transparent Clock

The physical layout of a machine determines the topology of an automation network, which is in many cases a daisy chain. When such a topology is built up with BCs, the result is a chain of control loops which is susceptible to error accumulation. That's why the automation community has proposed the new clock type TC. This is an Ethernet bridge which is capable to measure the residence time of PTP event messages, i.e. the time the message has spent in the bridge during transit. Because the residence time is the difference of two timestamps, the TC does not need to be synchronized. It is sufficient if it can measure short time intervals with reasonable accuracy. Synchronization of the local timer improves accuracy. The residence time of the traversed TCs is summed up in the correction field of the Sync message, if the TC is capable to modify the correction field on the fly, or in the respective Follow_Up message [11].

End-to-End Transparent Clock In the case of end-to-end (e2e) transparent clock (see figure 2.8), the slave measures the delay to the master with an end-to-end delay request/response message exchange 2.7.

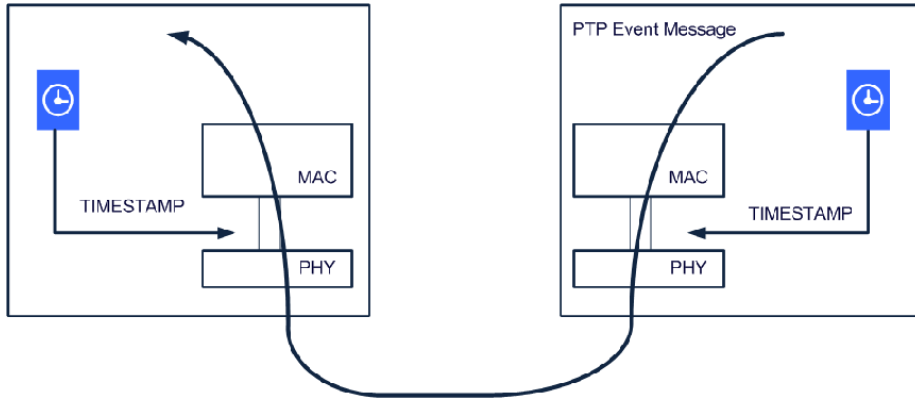


Figure 2.5: Timestamping of the PTP packet [12].

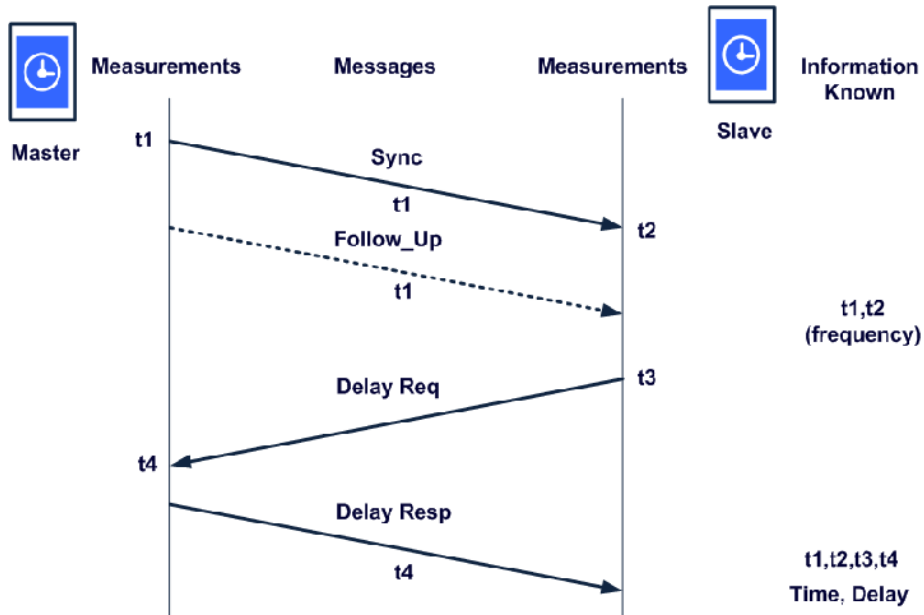


Figure 2.6: Transfer of PTP messages [12].

Peer-to-Peer Transparent Clock The peer-to-peer (p2p) transparent clock measure the link delay to all neighboring clocks with Pdelay_Req/Pdelay_Resp messages. A third message type may be required for this purpose, the Pdelay_Follow_Up. When a Sync traverses a p2p transparent clock, not only the residence time is added to the correction field but also the uplink delay, i.e. the delay of the link over which the Sync has been received [11].

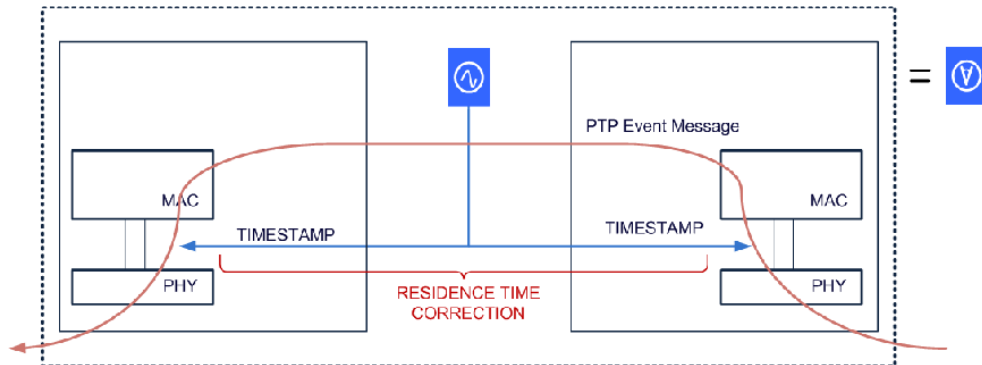


Figure 2.7: End-to-end transparent clock [12].

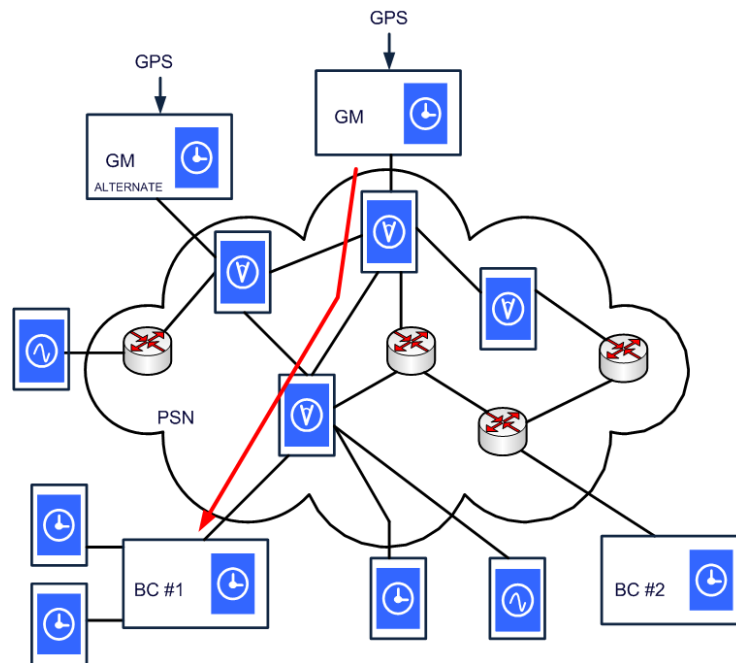


Figure 2.8: End-to-end transparent clock network [12].

2.2.5 White Rabbit

White Rabbit [15] [16] [17] [18] is a name for a project and technology with sub-nanosecond accuracy for time synchronization on a large LAN. It was primarily developed to be used at CERN for synchronization in particle colliders and other scientific applications. The main idea was to utilize existing technologies to build up such a system with desired parameters. Two main building blocks are:

- **Synchronous Ethernet (SyncE)** for frequency distribution (synchronization).
- **IEEE 1588** for precise time synchronization, operating on Layer 2 (Ethernet frame

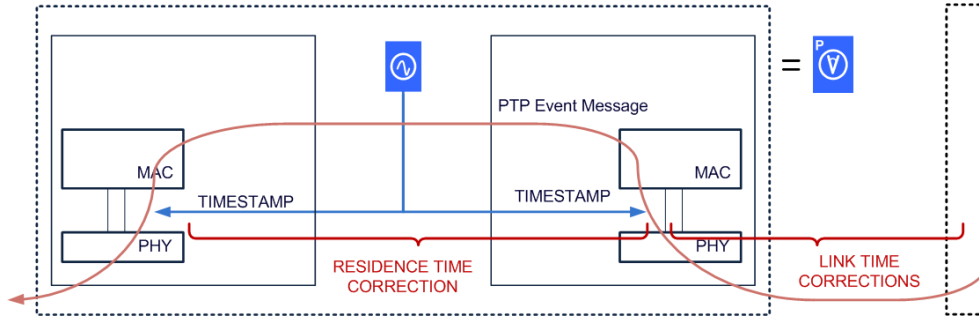


Figure 2.9: Peer-to-peer transparent clock [12].

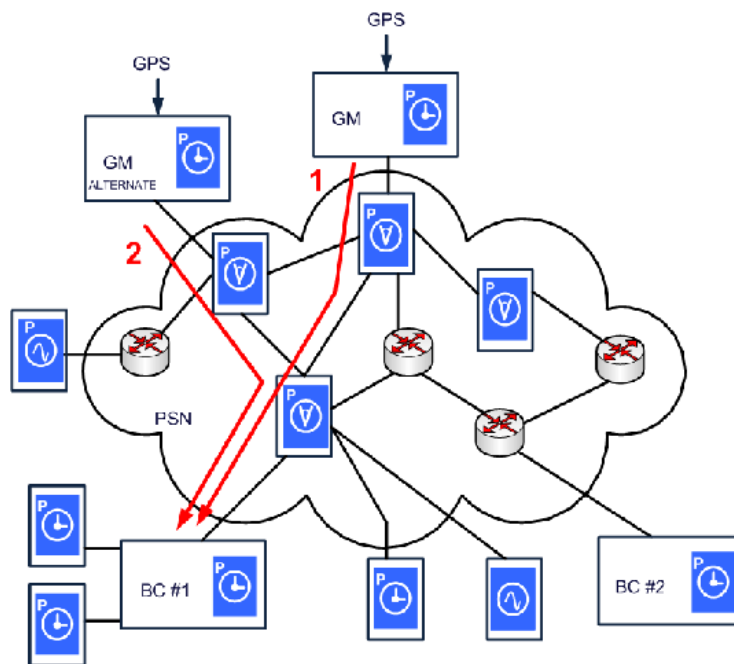


Figure 2.10: Peer-to-peer transparent clock network [12].

payload).

In [16] is described performance testing with transfer over 2km long fiber. The 80 ns jitter was measured during 10 minutes testing which included multiple severe changes of fiber temperature.

2.3 Time Transfer in Optical Fiber

Optical fibers are currently used for precise time and frequency transfer. In case of frequency transfer, the transmitting laser can be unmodulated when its wavelength is directly bound to the frequency source, so the frequency is directly recovered in the receiver. In

2. BACKGROUND AND STATE-OF-THE-ART

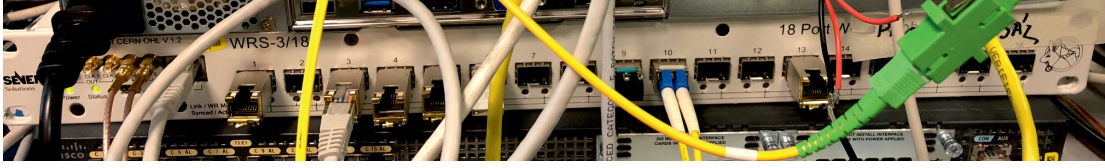


Figure 2.11: White Rabbit switch testing in our laboratory.

case of time transfer, modulation of optical signal is necessary. Several basic approaches for time transfer in optical fibers exist [19]:

1. Unidirectional fiber-optic time/frequency transfer (In Figure 2.12)
2. Bidirectional time/frequency transfer arranged for comparison of two clocks (In Figure 2.13)
3. Bidirectional time/frequency transfer arranged for constant-delay distribution of one clock (In Figure 2.13)

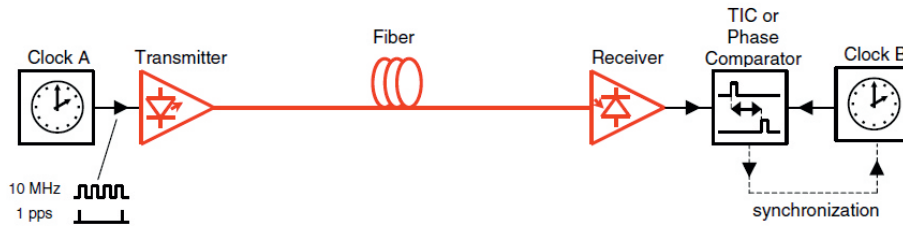


Figure 2.12: Unidirectional fiber-optic time/frequency transfer [19].

Unidirectional transfer In Figure 2.12, there is a principle of unidirectional time/frequency transfer. The source signal of 1PPS or 10 MHz is directly converted from electrical to optical domain and transmitted over a fiber. The received signal is then back converted to the electrical domain and may be compared by the TIC with the local clock or can be used directly for clock synchronization. This method is suitable only for a short distance or less precise transfer.

Bidirectional Transfer – Comparison For the more precise transfer is the unidirectional method inconvenient so there is a need for bidirectional transfer. In Figure 2.13(a), you can see a principle of this method for two clocks comparison. The time or frequency signals generated by the clocks are simultaneously transmitted over the same single fiber that is ended with optical multiplexers/filters. After receiving the signals on each side the both TICs compare the local clock with remote one. The assumption is that both

propagation delays are the same in each direction. We utilize this method in our research because it is suitable for comparison of local and remote timescale.

Bidirectional transfer – Constant Delay In Figure 2.13(b), you can see a principle of constant-delay distribution. It is similar to the method (a), but the signal on the remote side is turned back into the fiber and the round-trip time is measured. The transmitting side is equipped with a variable delay unit in which is any variation of propagation delay compensated.

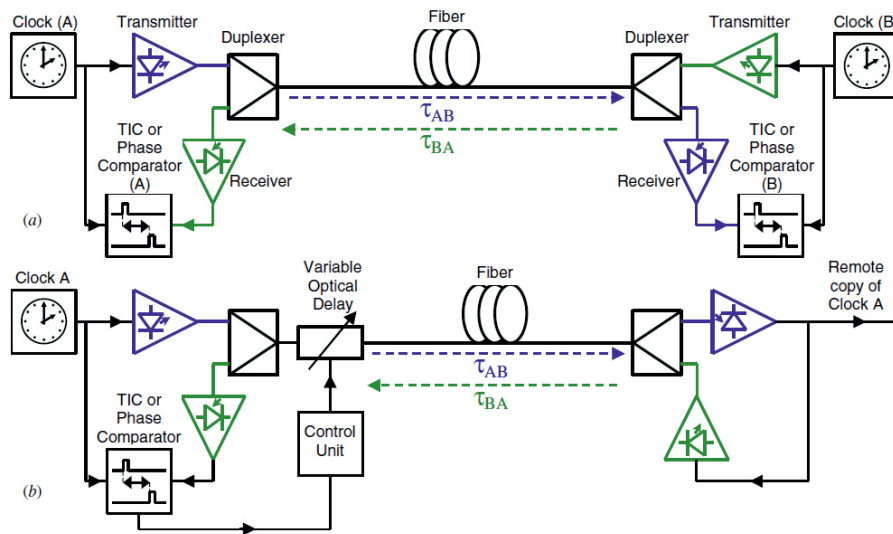


Figure 2.13: Bidirectional time/frequency transfer arranged for (a) comparison of two clocks, (b) constant-delay distribution of one clock [19].

2.4 Time Interval Measurement

Coarse time interval measurement is the fundamental method. In this method we have the incremental counter which is driven by the reference clock with frequency f_{ref} with the

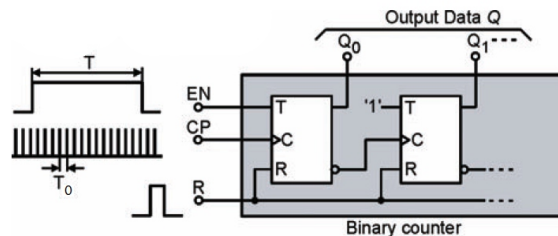


Figure 2.14: Coarse counter principle [20].

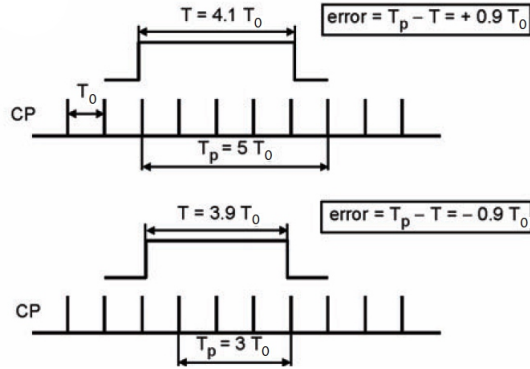


Figure 2.15: Coarse counter errors [20].

appropriate period (this is also the resolution):

$$T_{ref} = 1/f_{ref} \quad (2.4)$$

The coarse measurement is simply done by sampling the incremental counter by the START and STOP events which determines the duration of the measured time interval (see figure 2.14). The samples are two integer numbers n_{START} and n_{STOP} hence the number of periods is:

$$n = n_{STOP} - n_{START} \quad (2.5)$$

The result of the coarse time interval measurement is equal to:

$$n \cdot T_{ref} \quad (2.6)$$

START and STOP signals are asynchronous to the f_{ref} time domain so the maximum quantization error of single measurement is $\pm T_{ref}$ (see figure 2.15). For the more precise (less than T_{ref}) measurement we have to use some of interpolation methods.

Tapped delay line method is based on sampling of the START event propagation in the delay line (In Figure 2.17). The delay line is composed of serial connected delay elements with delay τ and the each output of the delay element is connected to a D flip-flop. All of the flip-flops creates a catch register which is driven by the STOP signal. The delay line is feeded by the START signal which after the START event propagates trough the delay elements. After the STOP event the state of the delay is sampled into the catch register as a series of ones and zeros. If the number of ones is n_{ones} we can calculate the value of time interval between the START and STOP events as:

$$n_{ones} \cdot \tau \quad (2.7)$$

Vernier interpolator – in this method there are two delay lines DL_1 and DL_2 consisted with delay elements with slightly different delay τ_1 and τ_2 . delay elements outputs are

connected to a catch register (D flip-flops), delay line 1 is connected to the data inputs and delay line 2 to the clock inputs of the individual D flip-flop. Delay line 1 is fed by START signal while delay line 2 is fed by STOP signal. The result is stored in the catch register and then is decoded to the corresponding time value (In Figure 2.16).

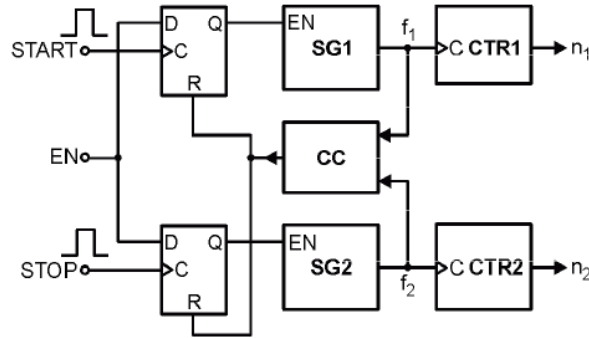


Figure 2.16: Vernier method [20].

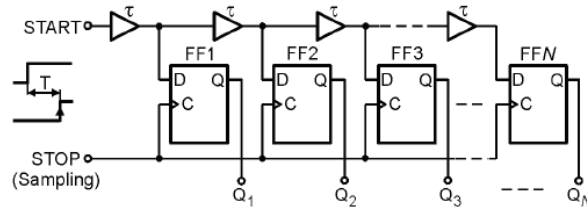


Figure 2.17: Tapped delay line method [20].

For implementation in FPGA both methods are suitable but the *Vernier method* is slightly more complex than the tapped delay line. It is difficult to implement two delay lines with similar delay propagation in the FPGA technology. The catch D flip-flops also have to be manually placed into correct position. An implementation of *the tapped delay line* in FPGA is much more easier. We can with advantage utilize carry chain entities which have regular structure and are connected physically to a one row so the delay propagation is uniform.

Overview of Our Approach

3.1 Time Transfer over Optical Network

There is a basic scheme of two atomic clock systems comparison in figure 3.1.

The time transfer method relies on symmetrical transport delay in both directions.

Two systems are connected by a bidirectional optical link. Each system is provided by a 1PPS signal from local clock and each systems has two outputs: T_r is a 1PPS signal received via optical interface from the other system and T_s represents epoch in which was sent out the encoded 1PPS signal.

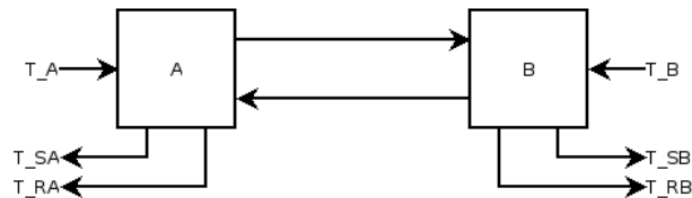


Figure 3.1: Time transfer method [A.3].

Both t_S and t_R signals are connected to STOP inputs of two time interval counters (TIC). The first TIC measures interval x between local 1PPS and t_R (i.e. difference between local second and received second from remote site) and the second TIC measures delay ϵ of processed 1PPS inside the transmitting part of the system.

1PPS pulse from a local clock arrives to system A in time t_A . It is transmitted by system A through the optical fiber to the remote site in time t_{SA} and the reception is signalized by system B in time t_{RB} . Analogically 1PPS pulse from remote clock raised in time t_B is transmitted by system B in time t_{SB} and received by system A in time t_{RA} . Here $\Theta_{AB} = t_B - t_A$ is the clock offset, $\epsilon_{Si} = t_{Si} - t_i, i = A, B$ is the delay of system i and

3. OVERVIEW OF OUR APPROACH

$\delta_{AB} = t_{RB} - t_{SA}$, $\delta_{BA} = t_{RA} - t_{SB}$ is the link delay from site A to site B and from site B to site A respectively.

Using a pair of time interval counters at both sites it possible to measure the system delays and the time intervals:

$$\begin{aligned} x_A &= t_{RA} - t_A = \Theta_{AB} + \epsilon_{SB} + \delta_{BA} \\ x_B &= t_{RB} - t_B = -\Theta_{AB} + \epsilon_{SA} + \delta_{AB} \end{aligned} \quad (3.1)$$

On a symmetrical link where the delay in both directions is $\delta = \delta_{AB} = \delta_{BA}$, the clock offset may be calculated as:

$$\Theta_{AB} = \frac{((x_A - x_B) + (\epsilon_{SA} - \epsilon_{SB}))}{2} \quad (3.2)$$

This work is related to the project “Time Transfer Using Fiber Link” [21]. The main goal is to develop a method to transfer precise time over optical networks for purpose e.g. cesium standard clock comparison instead of GPS-based methods. There were realized three experiments focused on method testing at long optical loop, two atomic clock comparison and demonstration of the system on time transfer between Prague and Vienna.

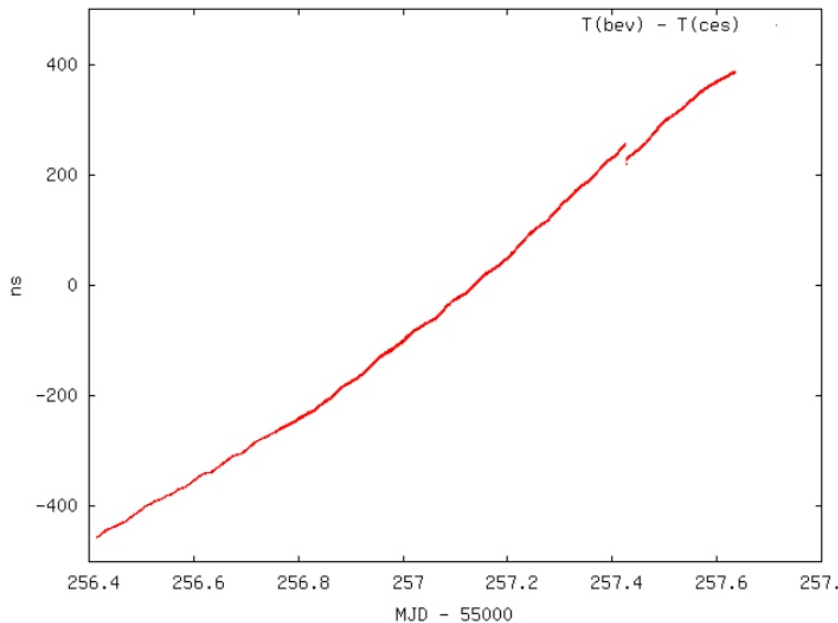


Figure 3.2: Time transfer between clock in Prague and Vienna [21].

The first experiment was aimed at the delay measurement of long optical path in order we can predict influence of the fiber thermal dilatation. A 744 km long bidirectional optical

loop was utilized between cities Prague—Brno—Olomouc—Hradec Kralove—Praha. The loop has been set up as an optical channel in the DWDM system, which included 12 amplifiers EDFA. One segment of the loop is optical cable installed on top of high-voltage poles, so its length is affected by daily changes of temperature. Both end station were located in CESNET laboratory in Prague and were connected the same clock - rubidium clock controlled by GPS. Using two TIC, the delay was measured in both direction [21].

Another experiment was aimed at time transfer between Prague and Vienna. Cesnet operates also a DWDM fiber link from Brno to Vienna, where it ends in the premises of ACONet (Austrian national research and education network) in Vienna university campus. The length of fiber link is 504 km excluding the fiber compensating chromatic dispersion. In Prague, there was used the same Rb clock as in previous experiment. In Vienna was situation complicated by not yet operational fiber link between Vienna University and BEV. Therefore, BEV transported their Rb clock to ACONet premises, where it was free running for the test duration. Figure 3.2 shows time offset between both clocks. As clock in Prague was disciplined by GPS, it can be concluded that the free running Rb clock in Vienna had frequency offset $8.08 \cdot 10^{-12}$ [21].

3.2 Timestamping

The key problem in the area of time transfer and generally timekeeping is timestamping of events. It doesn't matter if the event is incoming/outcoming data packet or a time label in the optical link bitstream. There is also a need of time series capturing, measuring and classification. This can be done using devices called universal counter such a Standford SR620 or Pendulum CNT91 (principle description is out of scope this Dissertation thesis). These universal counters are suitable for laboratory purposes, they have resolution and accuracy in order of tens of picoseconds (SR620 20 ps and CNT91 50 ps). Disadvantages of these devices are high costs and physical dimensions (not supposed to be used in embedded systems). Another time measure device can be a specialised computer extension card e.g. Brilliant Instruments BI200 or Guidetech GT668. These cards are dedicated adapters for time interval measuring and are utilized e.g. by NIST laboratories for checking the farm of very precise atomic clocks.

Because of previous reasons, we designed, developed and verified a specialized FPGA-based embedded timestamper and interval counters suitable for implementation in devices and adapters. The design does not depend on particular family of FPGA chips and therefore might be tailored to already utilized FPGA in nearly any device.

IEEE 1588 Timestamper

IEEE 1588 is a relatively new protocol standard for a precision clock synchronization. It operates mostly over TCP/IP networks and Ethernet. The protocol is also known as the Precision Time Protocol – PTP. Synchronization architecture is a master–slave model with nodes communicating primarily by multicast. The main difference between PTP and its predecessor NTP is that the PTP enabled nodes have to be equipped by some HW support to precisely measure the delay. You can find more about this protocol in [13].

The IEEE 1588 timestamper is a device which creates timestamps of incoming/outgoing packets in network interface hardware and the timestamp is further used in the PTP functionality. The timestamper is placed between the PHY and MAC layer on a MII interface and listens to the traffic. Every PTP packet is timestamped so if there is a lag in the network hardware between the reception and sending, we will know the correct time of physical transmission of the packet. The correct timestamp of a Sync message is sent as a Follow up message.

In figure 4.1, there is a simplified block diagram of the FPGA counter with carry chain interpolation. It utilizes a coarse free-running counter driven by reference frequency. The

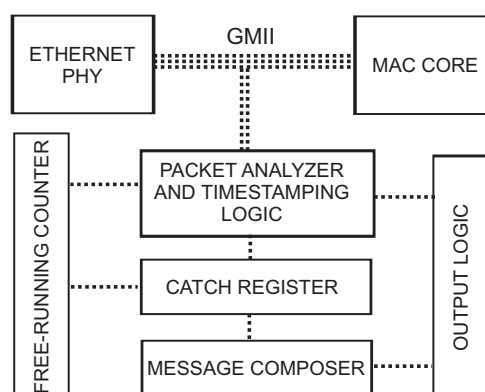


Figure 4.1: Simplified block diagram of the IEEE 1588 timestamper [A.12].

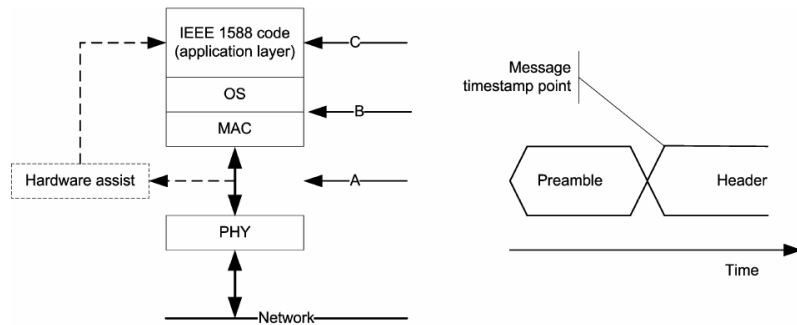


Figure 4.2: PTP timestamp generation model. The timestamping event is generated after start-of-frame delimiter [13].

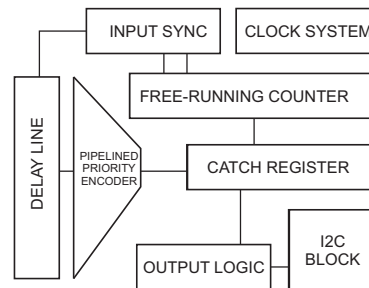


Figure 4.3: Simplified block diagram of the FPGA counter with carry chain interpolation [A.12].

intervals within one clock period is measured by the tapped delay line interpolator (carry chain implementation). The propagation rate of the delay line is computed in the pipelined priority encoder. Measured values are stored in the catch registers and sent by the output logic. The I2C block reads service information from transceivers and manage the frequency synthesis on the daughter card. The timestamper is implemented as IP core for FPGA in VHDL language. The timestamping core is configurable and can operate on Layer 2, 3 and 4 of the ISO/OSI model and also the reference frequency of the free-running counter. You can see a timestamp generation model in figure 4.2.

We did an evaluation of timestamper. In figure 4.4, you can see measured data. There is one issue of an accidental glitch which is the subject of a further research.

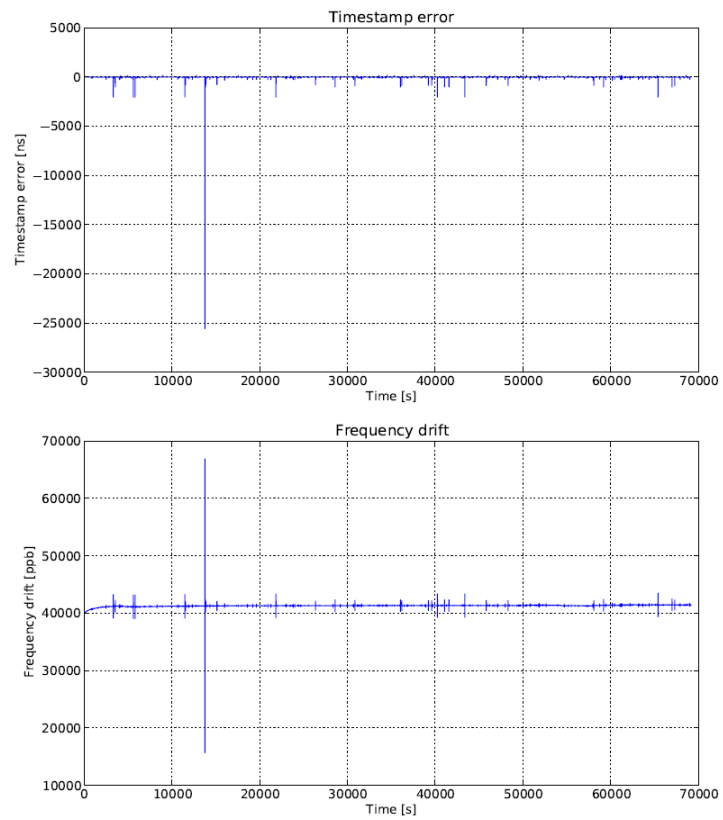


Figure 4.4: IEEE 1588 timestamper evaluation.

Atomic Clock Comparison

Nowadays, an accurate time signal is mostly acquired from the global positioning system (GPS). The Common-View GPS satellite method [1] is used for the atomic clock's time-scale calibration as well. Since the GPS time transfer is prone to the accuracy degradation at distances over 1000 km or there is a problem with a GPS antenna or receiver installation, an alternative method has been developed – the precise time and frequency transfer in optical networks. A fiber-optic-cable-based network can carry a signal up to 2000 km utilizing an optical amplification only.

We have recently designed and deployed a system of adapters for accurate time and frequency transfer [22]. The aim was to compare atomic clocks in geographically distant localities. The system utilizes an optical network consisting of various types of fiber links including channels in DWDM transmission system, which is currently a standard in optical communication. We have developed the second generation of adapters (Figure 5.1) having significant improvement: a newly designed embedded time interval counter in the FPGA structure.

In this section, we present an adapter that significantly simplifies a setup of atomic clock comparison. Measurement results and the embedded counter evaluation was presented in [A.4].

5.1 Adapter

Figure 5.2 shows a block diagram of the first generation adapter. Hardware of this adapter is simple, it consists of two main parts: an optical transceiver and electronics based on a FPGA chip. The time transfer method (described in [23]) is out of scope of this paper. We will focus on the Time Interval Counter (shortly TIC) blocks whose aim is to measure time interval between two events with resolution of few tens of picoseconds. In the first generation of the adapter, an expensive standalone device called *universal counter* is used for this purpose and each adapter requires two of them. In the second generation adapters described in this paper, these standalone devices are replaced with a digital circuit implemented inside the structure of the FPGA chip that is already present in the adapter.



Figure 5.1: A physical view of the adapter that utilizes the embedded TIC.

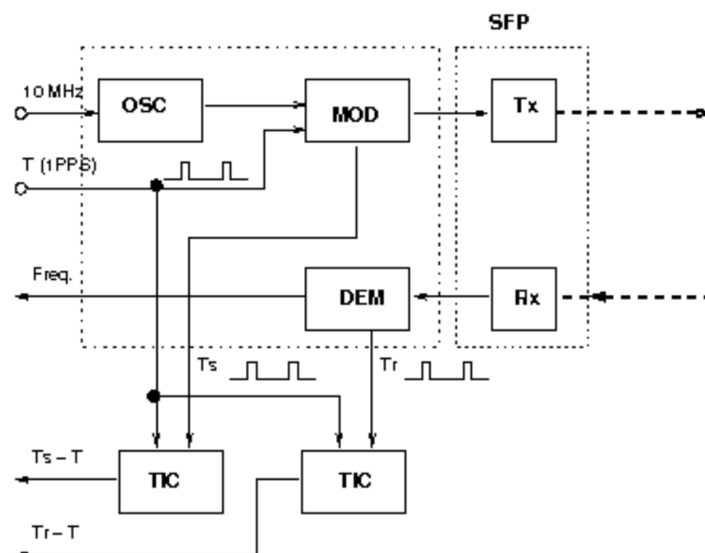


Figure 5.2: The first generation adapter scheme [21], TIC blocks are embedded in the second generation.

5.2 Time Interval Counter

The TIC has been implemented as an embedded intellectual property core (IP core) in the Virtex-5 FPGA (Figure 5.4 shows the block diagram) which is a part of the adapter (the implementation was described in [A.5]). Also, the TIC has two independent input triggered channels START/STOP, reference clock input, measured data output, and a management

I/O. The TIC consists of two base blocks, a *coarse counter* and an *interpolator*. Supporting blocks are a synchronizer, a clock management, an I²C management, and output logic.

5.2.1 Coarse counter

The coarse counter is the main part of the TIC and measures the time interval between START/STOP events on both channels. It is implemented as a simple incremental synchronous counter operating at the reference clock frequency. If f_{ref} is the referential frequency, the appropriate period $T_{ref} = 1/f_{ref}$ is the resolution of coarse measure as well. The coarse counter is sampled by START/STOP events. Samples are two integer numbers n_{START} and n_{STOP} . Hence, the number of periods is $n = n_{STOP} - n_{START}$ and the result of the coarse time interval measurement is equal to $n \cdot T_{ref}$. START/STOP signals are synchronized with respect to the 400 MHz main clock domain in the synchronizer modules, so the absolute resolution of the coarse counter measurement is 2.5 ns.

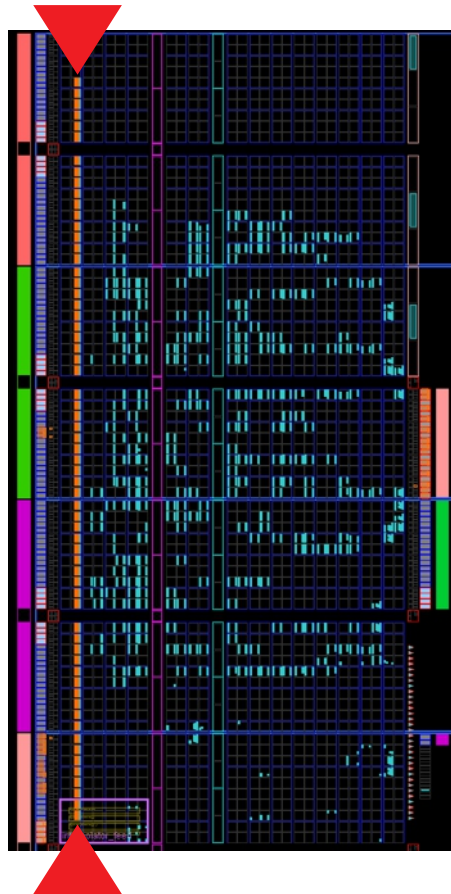


Figure 5.3: A section of the Virtex-5 XC5VLX50T FPGA floorplan with delay line structure (utilizing CARRY4 elements) between arrows [A.4].

5.2.2 Interpolator

In order to achieve better resolution than the coarse counter, an interpolator is implemented as the second main part of the TIC. The interpolation is based on sampling the START event propagation in the delay line within the period of reference clock T_{ref} .

The delay line is composed of serially connected delay elements with delay τ . It is fed by the START signal which propagates after the START event through delay elements. Following the STOP event, the state of the delay line is sampled. If we know the number $n_{elements}$ of active elements at that moment, we can calculate the value of time interval between the START and STOP events as $n_{elements} \cdot \tau$.

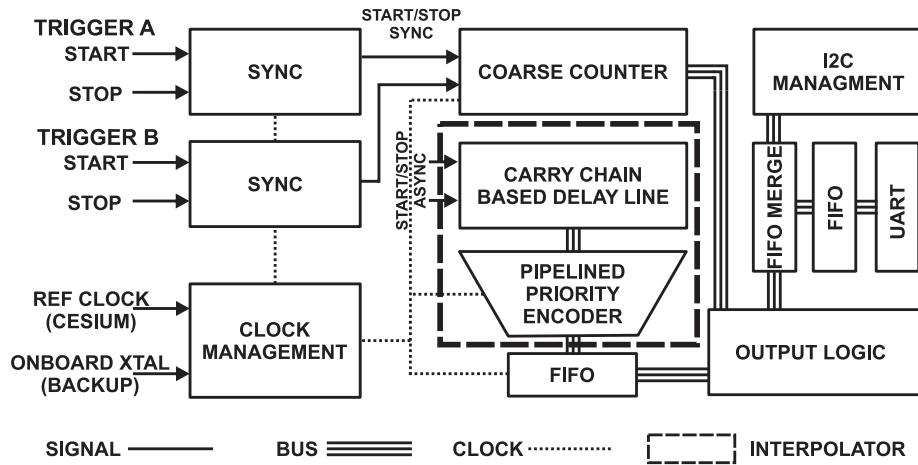


Figure 5.4: A simplified block diagram of the FPGA counter with carry chain based interpolator depicted in the dashed box [A.4].

The delay line is based on a carry chain logic with individual delay elements (actually multiplexers) interconnected in one row. It has input logic that injects logic 0 after propagating every trigger event – so that an event is always propagating as a change from logic 0 to logic 1 and guarantees the same starting conditions for repetitive measurements. Because of the unknown physical structure of carry chain elements, the propagation of the events in the delay line to the delay element outputs is not uniform, but the number of activated outputs is increased after similar portion of time. For that reason there is a pipelined priority encoder that computes the number of activated delay elements every rising edge of the reference clock.

The total delay of the delay line must be higher than T_{ref} period in order each event can be caught. The real speed of delay line (which might be affected for example by temperature, power supply voltage or FPGA speed grade) is expressed in number of elements corresponding to T_{ref} period. Whenever the particular input event is observed again in the next period before it reaches the end of delay line, we know real number of passed elements. Averaging of these numbers provides automatic self-calibration of the delay line. We can say that the resolution of this interpolator is less than 20 ps.

There are one START and two STOP events that share one interpolator. Due to a possibility of incoming trigger signals interference, we inserted some deterministic portion of delay for consecutive events. We attached a logic for enabling trigger signals as well.

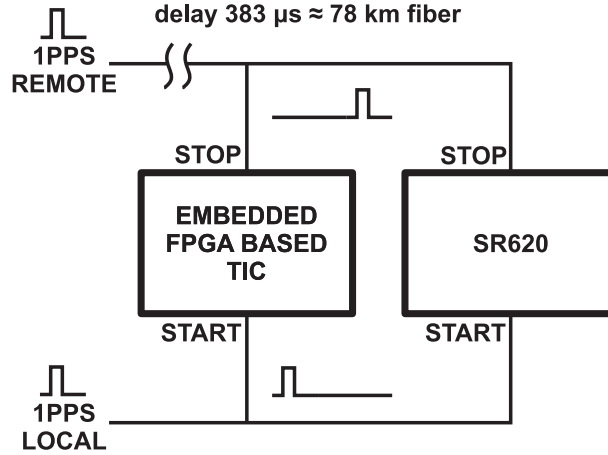


Figure 5.5: TIC evaluation, the interval between remote and local 1PPS events is measured by the embedded TIC and SR620 A.3 simultaneously [A.4].

5.3 Time scale comparison

Generally, atomic clocks generate a time scale which is the subject of comparison. An elementary method is depicted in Figure 5.5. We have local and remote system consisting of an atomic clock and an adapter. Both systems are interconnected via an optical network and the objective is to measure a difference between local and remote one pulse per second (1PPS) signals (the 1PPS signal is commonly used in a timekeeping – there is a rising-edge every second). Some variation of this difference is expected and has two origins: a temperature expansion of the optical fiber and a mutual shift of both atomic clocks time scales. As described in [23], for the optical path with equal delay in both directions, we can apply formula:

$$\Theta_{AB} = \frac{((x_A - x_B) + (\epsilon_{SA} - \epsilon_{SB}))}{2} \quad (5.1)$$

Θ_{AB} stands for clock offset, x for receive and ϵ for send delay. Indexes A and B determine the measuring site. The prefix S indicates that the value is sent from the system. The formula is valid only for a symmetric link.

Adapters are located in two different localities: Prague (CESNET) and Geodetic observatory Pecný (VUGTK). Localities are connected via 78 km long optical fiber. We utilize a bidirectional time transfer in single fiber that is implemented using two DWDM neighbor channels in ITU grid (wavelength difference 0.8 nm).

The graph in Figure 5.6 presents comparison of our time transfer method with standard GPS based methods: Common-View (CV) and Precise Point Positioning (PPP). The

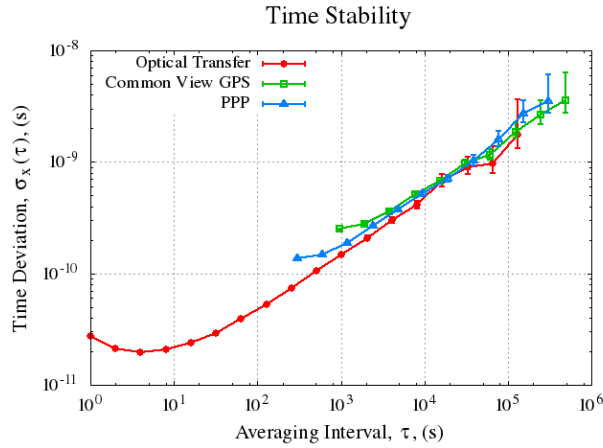


Figure 5.6: Time stability comparison [A.4].

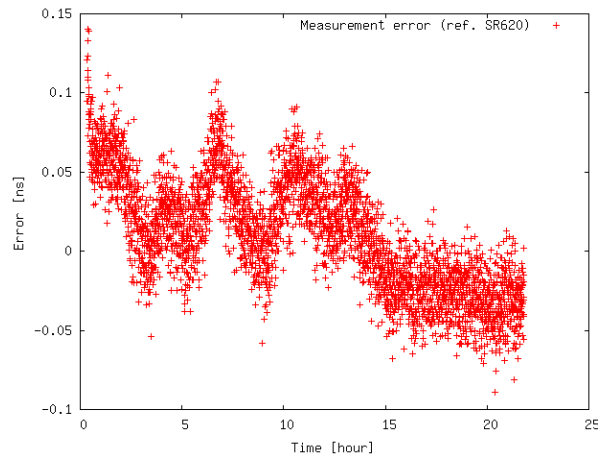


Figure 5.7: Absolute error of individual measurements [A.4].

optical transfer utilizes our new generation adapter with FPGA based counters. As can be seen, our method provides better resolution and stability and lower noise (in terms of TDEV) compared to the GPS based method. The CV method uses data in the format CGGTTS [24], where the granularity is 960s. The PPP has computation period of 300s. This is the reason that there are no data for CV and PPP methods at the beginning of the graph (until time interval 300s or 960s). The lowest observed noise is about 20 ps using averaging interval of 4s. Detailed description of TDEV statistics and GPS based methods is out of scope this paper. We can conclude that the FPGA based counter is suitable for the atomic clock comparison as it is not worse than standard GPS based methods.

5.4 Embedded counter evaluation

As the second generation of adapters with the embedded TIC has been recently developed, we have to evaluate a proper function of time interval measurement. It is done by adding a universal external time interval counter SR620, the situation is depicted in Figure 5.5. Both embedded and external counters are measuring same intervals (difference between local and remote 1PPS) and collected data are statistically evaluated.

The FPGA implementation of the TIC interpolator carries some potential issues. We observed that the signal propagation is not uniform through the delay line because of the obvious logical vs. real FPGA blocks structure difference. We also found out that the correct placement of the delay line is crucial for the correct function. In some cases, the placement affected the delay propagation uniformity by adding an undesirable amount of delay between some delay elements. The reason of this behavior is that the delay line crosses several FPGA clock regions and some of these crossings add a significant delay. The possible placement is different in various FPGAs within the Virtex-5 family, so the delay line placement has to be done by an experiment (physical view of FPGA placement in Figure 5.3).

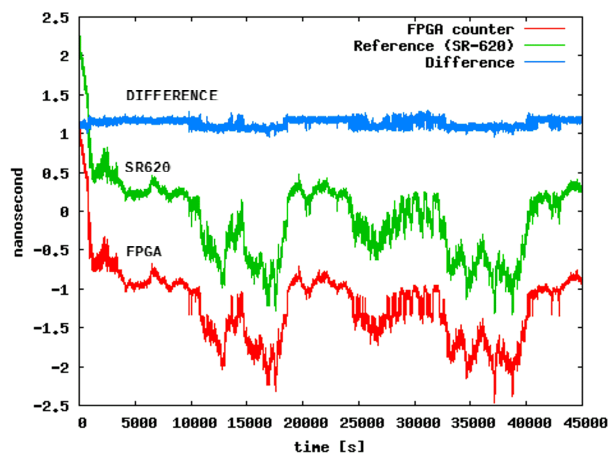


Figure 5.8: Comparison between embedded FPGA counter and SR620 [A.4].

Figure 5.7 shows differences between FPGA based counter and SR620 A.3 as reference. We can see some basic noise whose origin is resolution of our counter and fluctuations between both devices. We identified the interpolator as source of these fluctuations. Figure 5.8 shows another view of both counters comparison. The measurement took a couple of days, the graph shows typical progress in a period of 12 hours (we chose the time frame arbitrarily, this behavior is not repetitive). The red line represents measured interval by the embedded interpolator and the green line represents measured interval by the SR620 universal counter. The difference of both measurements is represented by the blue line. We observe occasional differences with variation under 100 ps. The source of this difference

is still subject of study. We intend to provide a precise calibration of the interpolator as well.

5.5 Summary

We have developed the embedded FPGA based interpolating time interval counter for the second generation of adapters. Pair of these adapters has been deployed and used for comparison of two atomic clocks over 78km long optical fiber. We proved that our method of time comparison provides better results than GPS based time transfer methods. We also evaluated the proper function of this counter and compared it in respect to the SR620 universal counter. We identified differences between both counters which is currently subject of study. The performance and accuracy of our FPGA based counter can be improved and it is goal of our future work.

Dual Interpolating Counter Architecture

An evolved dual interpolating counter architecture for the clock comparison over an optical network, especially utilizing dense wavelength division multiplexing (DWDM) as we presented our results on EWDTs 2014 conference [A.3].

A basic TIC structure was described in [A.5]. The TIC has been implemented as an embedded intellectual property core (IP core) in the Virtex-5 FPGA (Figure 6.1 shows the block diagram) which is a part of the adapter. The TIC consists of two base blocks, a *coarse counter* and an *interpolator*.

The *coarse counter* is the main part of the TIC and measures the time interval between START/STOP events on both channels. It is implemented as a simple incremental synchronous counter operating at the reference clock frequency. START/STOP signals are synchronized with respect to the 400 MHz main clock domain in the synchronizer modules, so the absolute resolution of the coarse counter measurement is 2.5 ns.

In order to achieve better resolution than the coarse counter, an *interpolator* is implemented as the second main part of the TIC. The interpolation is based on sampling the START event propagation in the delay line within the period of reference clock T_{ref} . The delay line is composed of serially connected delay elements with delay τ . It is fed by the START signal which propagates after the START event through delay elements. Following the STOP event, the state of the delay line is sampled. If the number $n_{elements}$ of active elements is known at that moment, the value of time interval between the START and STOP events can be calculated as $n_{elements} \cdot \tau$.

Overall structure and implementation is more complex and out of scope this paper. Details are described in the previous work [A.4]. The architecture of TIC has been improved and the second interpolator has been added. Resulting dual channel TIC architecture is crucial for the mutual time scale comparison.

6.1 Interpolator Feed

There are three inputs to the interpolator for the measurement purpose – START triggered by the 1PPS signal from atomic clock, STOP1 for a local 1PPS transmission and STOP2

6. DUAL INTERPOLATING COUNTER ARCHITECTURE

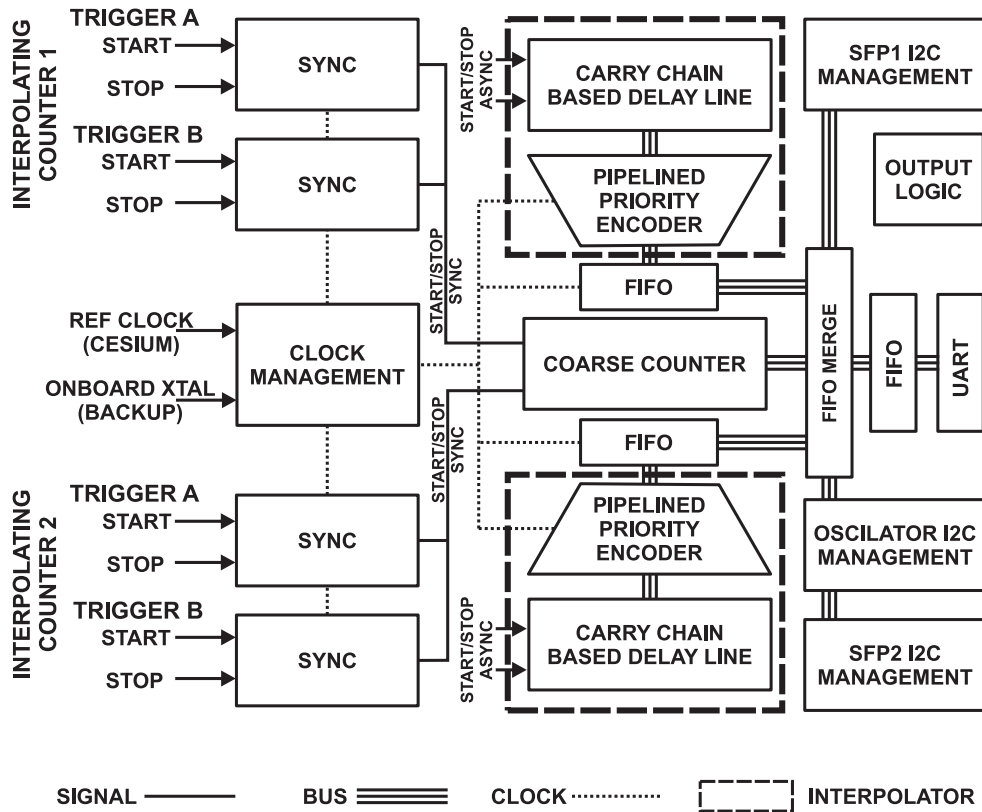


Figure 6.1: Simplified block diagram of the dual FPGA counter with carry chain interpolation. Each interpolating counter has two START/STOP measuring channels with common START signal [A.3].

for a remote 1PPS detection. It was necessary to connect these three signals to the interpolator. There were three edge detectors in the original design that were connected to the interpolator through the XOR gate (Figure 6.2). Edge detector switches the logical output every detected rising edge on input. The consecutive time events were measured as logical level change in the interpolator's delay line. It was observed that the signal propagation is different in a case of switching from logic level "0" to "1" and vice versa. This was a problem for the accurate measurements of the consecutive START-STOP1-STOP2 time events. The problem has been solved as described in the Figure 6.2. Another edge detector was added for the purpose of resetting the interpolator input to the logic level "0" before measuring next time event.

As described, the interpolator feeds are composed of four edge detectors and the XOR gate. You can see the feeds on the floorplan in Figure 6.3.

6.2 Dual Interpolator Design

In the previous work, always only one delay-line based interpolator was utilized for measuring the local and remote time events. It has been proved that it is possible to add the second interpolator into the design. There are more applications for the two interpolators design e.g. the second interpolator can be employed in a case of the time signal distribution over two optical lines. Another practical application is laboratory testing. Having two interpolators, two optical interfaces can be simply interconnected as a loopback. Each line has the same signal and that can be used with advantage for a large delay testing or for the evaluation of the calibration.

The placement of the second interpolator is crucial for a proper function. We have determined the placement (depicted in Figure 6.3) after many attempts and tests. There were some issues in the first interpolator placement (described in [A.4]) such an adding undesirable amount of delay between several delay elements of the interpolator delay line.

6.3 Calibration Method

The delay line is based on a carry chain logic with individual delay elements (actually multiplexers) interconnected in one row. The number of delay elements is 256 with average delay 15 ps depending on the physical placement, temperature and voltage. The total delay of delay line is about 3.8 ns. With the reference clock period of 2.5 ns it is possible to observe in the delay line some time events twice. Whenever the particular time event is observed

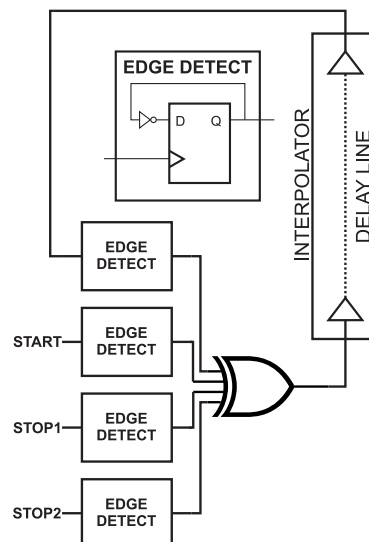


Figure 6.2: Interpolator feed input logic. In the original design, there were three edge detectors. We added another edge detector connected to the output of the delayline to reset delayline feed always to zero [A.3].

again in the next period before it reaches the end of delay line, we know real number of passed elements per reference clock period. We call the number of passed elements the *interpolator rate*. Averaging of these rates provides automatic self-calibration of the delay line. The typical interpolator rate is about 160 in current implementation.

Also some non-uniform delay distribution has been discovered in the delay line beginning (the first ten delay elements). For the correct measurements, it is necessary to discard these low values and use only the second pass values.

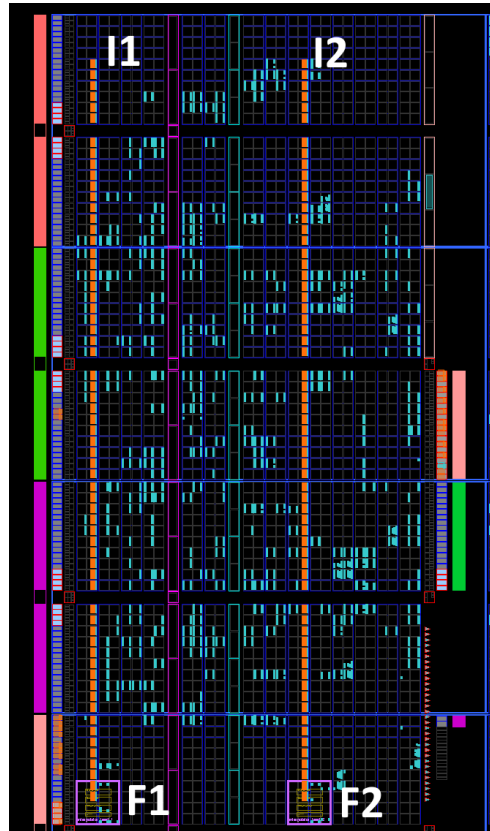


Figure 6.3: Dual interpolating counter floorplan in Virtex-5 FPGA. F1 and F2 are the areas of the interpolator feeds. I1 and I2 denotes the interpolators (the delay line is located between points I and F) [A.3].

6.4 Summary

The significant improvements has been done in the hardware architecture for atomic clock comparison. Second interpolator was added in the design to extend the functionality of our device in order to support a multiple optical lines time signal distribution or calibration algorithm testing. Six devices with the supporting hardware are now deployed on several sites for long-term testing and four other are ready for deployment. Preliminary results

indicates that our architecture is suitable for the purpose of atomic clock time scales comparison. In the previous work, it was proved that our method of time scale comparison provides better results than standard GPS based time transfer methods.

Architecture for Precise Time Measurements

This chapter deals with a new generation of our adapters for atomic clock timescale comparison and other time measurements. Our currently operated adapters are based on Virtex 5 FPGA and the further development on this platform is obsolete. We describe our experience with FPGA based time measurement system consisting of one or two channel interpolating counter. We proposed the new design based on the Zynq All Programmable SoC platform and presented it on [A.2].

7.1 Timescale comparison method overview

We described our method in more detail in our previous contributions [A.4], [A.5]. We have two interconnected systems consisting of an adapter and an adjacent atomic clock. The basic method scheme is in Figure 3.1. Both systems are connected via an optical line and we are measuring difference between local and remote time – more precisely the difference between two 1PPS signals. For optical link with the same delay we can calculate the clock offset using formula:

Θ_{AB} stands for clock offset, x for receive and ϵ for send delay. Indexes A and B determine the measuring site. The prefix S indicates that the value is sent from the system. The formula is valid only for a symmetric link.

A and B stands for the particular location. TA and TB are 1PPS inputs from on-site atomic clock, Tr stands for the received signal and Ts stands for sent signal respectively. Both tS and tR signals are connected to STOP inputs of two time interval counters (TIC). The first TIC measures interval x between local 1PPS and tR (the difference between local second and received second from remote site) and the second TIC measures delay ϵ of processed 1PPS inside the transmitting part of the system.

7.2 Timestamping

For the purpose of time difference measurements we have developed the FPGA embedded Time Interval Counter. The TIC was implemented in Virtex-5 FPGA platform and the design is described in our previous work [A.4]. We implemented TIC using a single carry logic based delay line with 256 delay elements that can interpolate time event twice during one 400 MHz clock period. Similar more complex designs exist utilizing multiple delay lines with multi-phase clocks. After system evaluation we found out that the uncertainty of measurements is below 100 ps which is suitable for the purpose of atomic clock time scales comparison.

7.2.1 Multiple Channels

The single channel TIC is not suitable for more complex measurements e.g. in a network node with several remote atomic clocks. We also plan to evaluate performance of IEEE 1588 protocol where the comparison to the atomic timescale is desired. The next step was to implement a dual channel TIC that deals with these issues. The implementation was challenging because we use 400 MHz clock signal for the interpolation. Finally, we were able to place two delay lines into Virtex-5 FPGA but with the difficulties to insert another complex logic e.g. a CPU for measurements processing.

7.2.2 Priority Encoder

In general, priority encoder is a piece of logic that transforms wider input to a narrower output. We use this encoder to count the number of activated delay elements for the interpolation. The fundamental limitation for design that is utilizing 400 MHz clock is a complex combinatorial logic. We avoided this logic using short register-to-register paths and pipelining. We also observed that the physical and logical structure of delay line is different and the signal propagation in the delay line is non-monotonic. This finding complicates the priority encoder logic that is normally based on detection of the leading 1 position. Similar existing designs use various compensation methods, one of them is to swap the outputs of the delay line (based on simulation) to achieve monotonous signal propagation. We made our design in another way and we simply calculate the count of active elements in a pipelined multistage adders block. You can see diagram of the developed encoder in Figure 7.1.

7.3 Zynq SoC platform

Since the Virtex-5 platform is becoming obsolete, we decided to migrate our current design to the new Zynq platform. There are two main reasons for such a decision: the development on Zynq platform is more sustainable and we can also benefit with the system on chip layout.

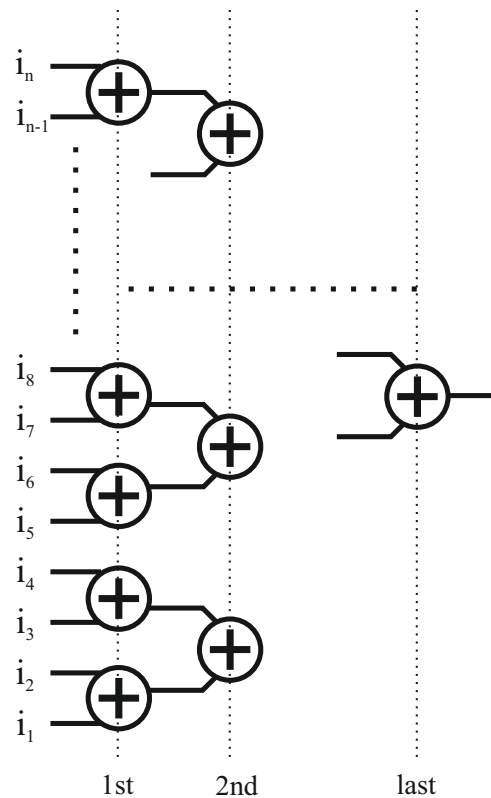


Figure 7.1: Pipelined priority encoder

We made some modification to our design to meet the new technology requirements. The preliminary results of time analysis revealed that the timing is not an issue even with multiple channel TIC architecture. Having this first crucial step, we can move further add new functionality to our system. We want to integrate our design with on chip CPU to eliminate the adapter peripherals. We can also add our IEEE 1588 timestamp core to make a system for a performance evaluation of time network protocols.

7.4 Summary

We developed several generations of FPGA based adapters with TIC for our precise time and frequency transfer infrastructure. Our previous work proved that our adapter design is suitable for the purpose of atomic clock time scales comparison. As we developed the dual channel TIC architecture we faced timing and placement issues because of a design complexity and the obsolete Virtex-5 platform. We migrated our design to the new Zynq All Programmable SoC platform for the purpose of the development sustainability and further system extension. We plan to add another pieces of design that deals with precise time acquisition and transfer to build a complex system for atomic clock timescales comparison and for time network protocols performance evaluation.

7. ARCHITECTURE FOR PRECISE TIME MEASUREMENTS

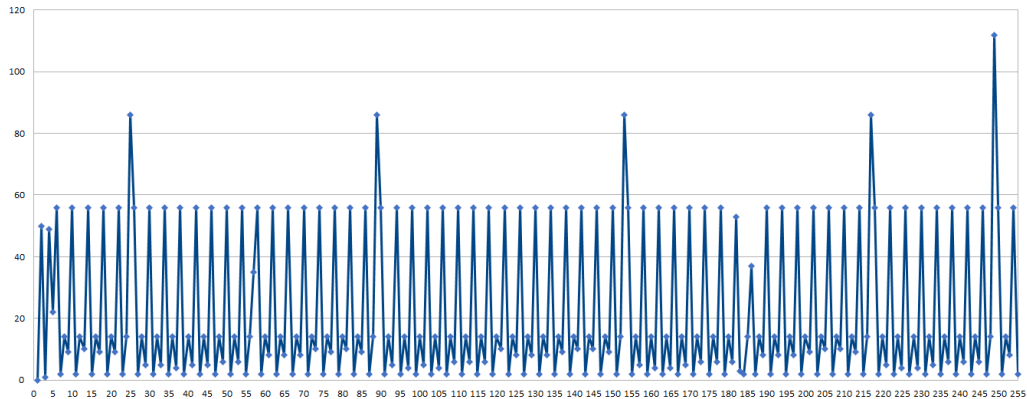


Figure 7.2: Signal propagation in the delay line. Horizontal axis represents delayline output number and vertical axis propagation difference between adjacent outputs in ps. It is visible that signal propagation is not monotonous. Several methods exist mostly based on static time analysis and generating offline sorting function to ensure that the outputs of delayline are activated monotonously.

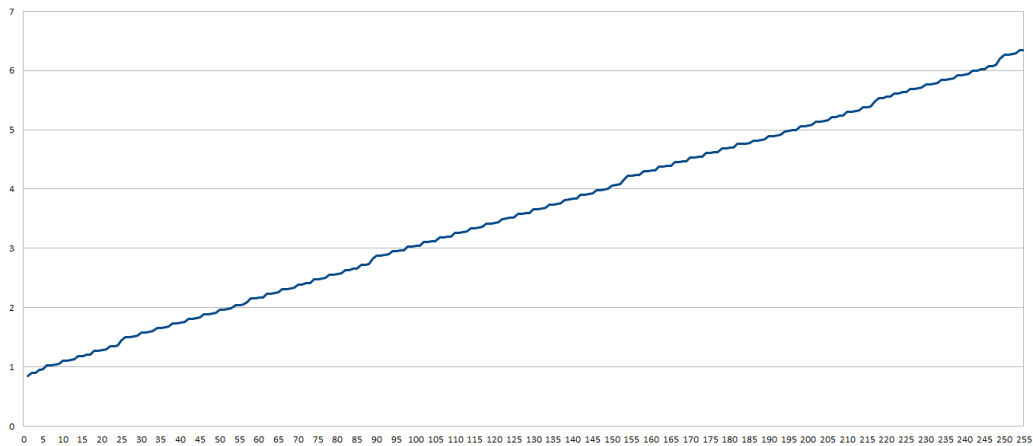


Figure 7.3: Signal propagation in the delay line using our approach – online pipelined priority encoder. Horizontal axis represents delayline output number and vertical axis propagation time events on outputs in ns. It is visible that our encoder sorts the propagation of time events on delayline outputs. The number of active outputs is generated online each clock cycle. It is possible to sample the 400 MHz clock twice during propagation in the delayline.

Running Processor on External Frequency Source

As described in 10.2, an OCXO oscillator is a vital part responsible to run the Linux based NTP stack on MiniATX PC platform with desired stability. The connection of external frequency source (quartz oscillator or OCXO) in the original platform is not a big issue as there is a external clocking IC that is capable to accept frequency from external source. Another advantage of the original platform is that there exists a PLL within the clocking IC so it produces stable frequency output while the input is switched from XO to OCXO (the need of such a switching is that the OCXO need ca. 5 minutes of warm-up time to run on stable frequency so the system starts on ordinary quartz oscillator and after the warming up the OCXO switches to the more stable oscillator).

Nowadays, the clocking circuitry of the processor is very different. The only external part of this system is a crystal with blocking capacitors and the rest of the oscillator and subsequent PLLs and another frequencies synthesis is integrated directly on chip. This architecture change makes very difficult to connect the external frequency source into the processor clocking circuitry.

Generating clock signal for the processor is a main task when designing a microprocessor

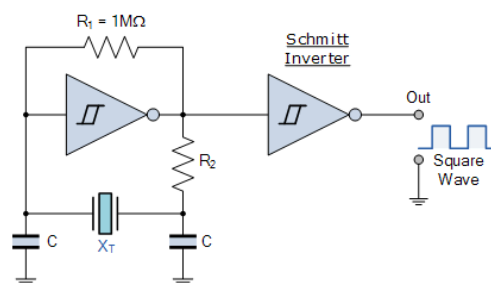


Figure 8.1: Pierce CMOS oscillator. Crystal, load capacitors and feedback resistor are the only external electronic components [25].

8. RUNNING PROCESSOR ON EXTERNAL FREQUENCY SOURCE

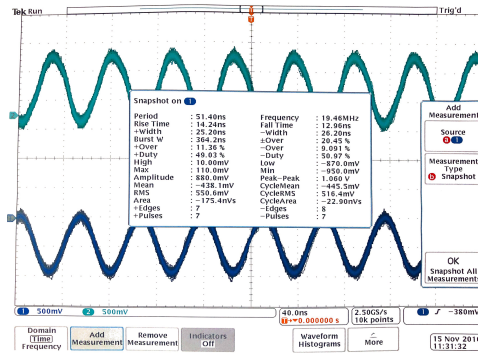


Figure 8.2: Measuring parametres of XTAL pins.

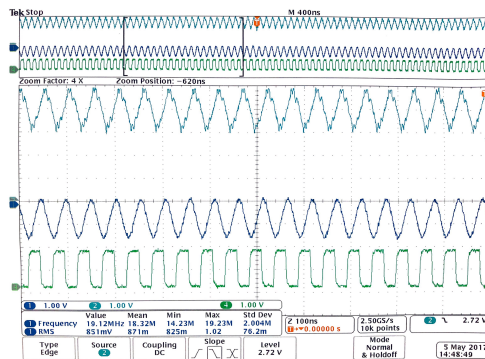


Figure 8.3: Measurements on external source PCB with IDT multiplexer. Channel 1: processor XTAL input pin, Channel 2: OCXO output, Channel 4: TCXO output.

system. More specifically, there are three typical frequency source options when we want to run e.g. a microcontroller:

1. internal RC circuit
2. external crystal
3. external frequency

We will omit the first option which is not interesting and take a look on the rest. The second option is to stabilize the frequency of the internal oscillator with connecting an external crystal. This is typically done using pins XTAL1 and XTAL2 (or pins with similar name like OSCn...) to connect the crystal with proper blocking capacitors and these pins are internally connected to the input and output of the oscillator. So having option 3, we can forget about external crystal and connect an external frequency source with desired parameters directly to the pin XTALn which is internally connected to the oscillator input.

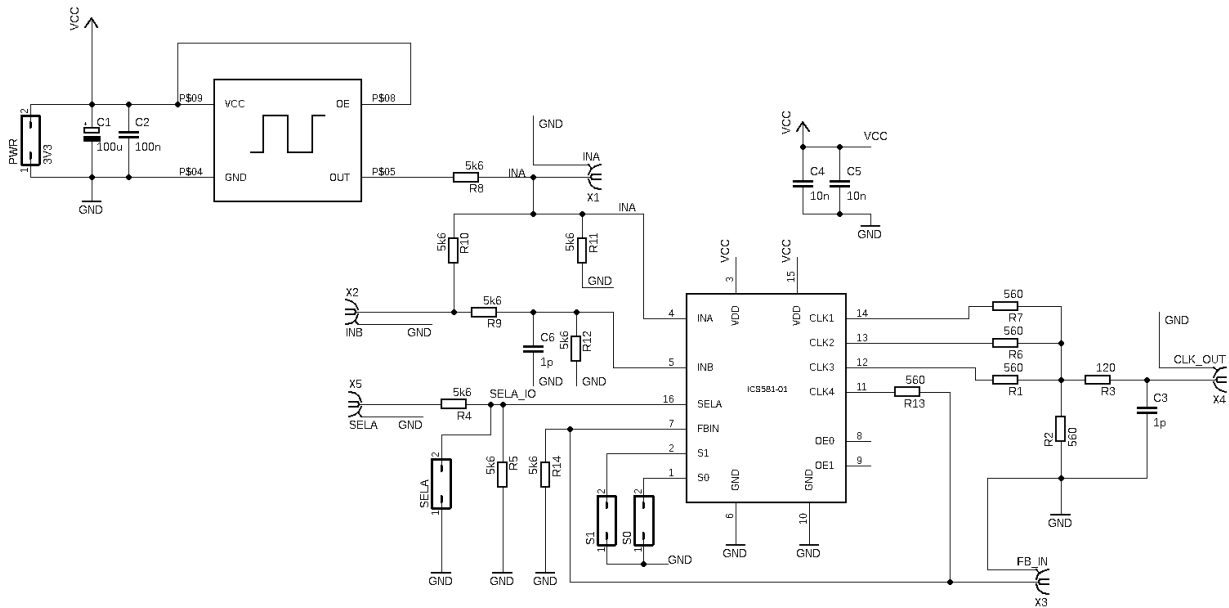


Figure 8.4: Version 2 of external processor clock source with multiplexer based on IDT IC. You can find the original version of our design in C.1

In Figure 8.2 you can see an oscilloscope screenshot of two probes placed directly on XTAL pins of Intel processor. The amplitude of signal on channel 2 has slightly higher voltage so we can assume this pin is internally connected to the output of oscillator and vice versa. our first try to simply connect the external frequency with similar signal parameters was unsuccessful. After this, we have designed a PCB to help us with measuring and tuning the input signal parameters C.1.

Another prerequisite was to identify or guess the type of internal oscillator – knowing only the connection of external passive electronic parts (quartz crystal, two load capacitors and 1 M feedback resistor) we assume that the type of oscillator is a CMOS Pierce oscillator.

Due to the future development of all external frequency source solution, we had to included a clock multiplexer for switching from basic XO to OCXO. We quickly realized that the internal clocking circuits are prone to sudden phase and frequency change leading to an unstable or halting state of the PC platform. So we decided to test two different clock multiplexers in order to choose the best one for the final design. We have selected IDT(ICS)581-01 and Si53323 (you can find the appropriate schematics in Figures C.1 and C.2).

We were dealing with evaluation of the multiplexers and tuning the analog part of feeding circuit to match the electrical parameters of the internal oscillator. Preliminary results showed that the IDT oscillator has better switching capabilities. In Figure 8.3 you can see the captured signals. Testing platform run on external frequency source for several months without any hang-up.

Long Distance IEEE 1588 Evaluation

IEEE 1588 protocol is intended to be used LAN environment. Original specification of this protocol [26] supposed only Data-link Layer number 2 for PTP payload encapsulation thus made routing outside LAN segment nearly impossible. Not surprisingly, the authors of the protocol were targeting to deploy precise time syncing within local network e.g. in laboratory or on institution with its primary time source. In the IEEE 1588 version 2 [27] the layer 3 support were added and the routing of PDUs were then possible.

A cooperation between CESNET and FAU (Friedrich-Alexander-Universitaet Erlangen-Nuernberg) started in order to test the behavior of PTP protocol between distant location (ca. 500 km, round-trip time about 20 ms). PTP server in Erlangen acted as grandmaster and we synchronised our local PTP server as slave.

Initial phase of PTP testing between Prague and Erlangen included:

- o PTP slave and master modes tested

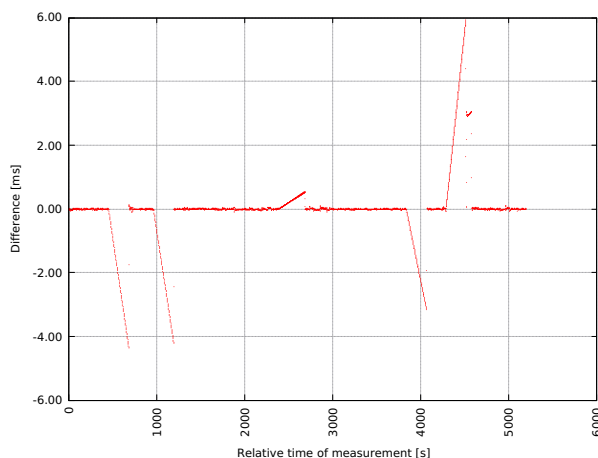


Figure 9.1: Unicast communication PTP test between Prague (slave) and Erlangen (master).

9. LONG DISTANCE IEEE 1588 EVALUATION

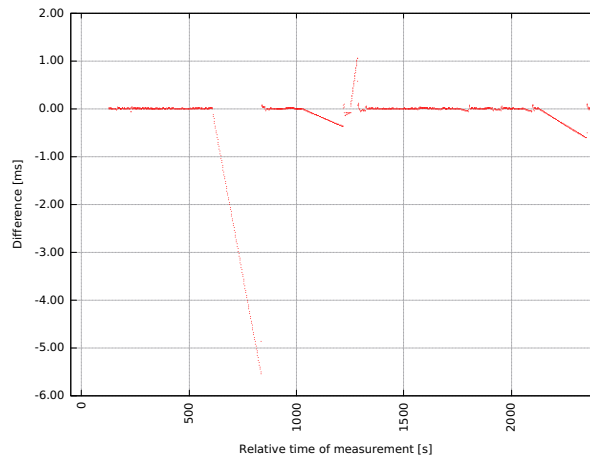


Figure 9.2: Unicast communication PTP test between Prague data center (master) and Prague laboratory (slave).

- Unicast network communication
- CESNET site:
 - 2x Meinberg M600 Grandmaster Clock (M/S)
 - 1x Meinberg PTP270EX PCIe Card (slave only)
 - 5071A Atomic Clock
- FAU site:
 - 1x Meinberg M600 Grandmaster Clock GPS
- Successful connection between sites, CESNET slave connected to the FAU master
- CESNET test: 1PPS comparison (local from 5071A vs. PTP slave output)

On the FAU site, the M600 device was set as the grandmaster with primary time source from the GPS receiver. On the CESNET site, the PTP270EX acted as slave and the resulting 1PPS signal was being compared on Pendulum TIC [28] with the local cesium standard atomic clock [29]. As the network connection was over the L3 routed network, there were routers that contributed with non-deterministic delay due to their input/output buffers and packets processing. In Figure 9.1 is plotted the difference between obtained 1PPS signal and local 1PPS signal from cesium standard. In the table 9.1 is calculated average and standard deviation of 1PPS difference. We were observing peaks with unclear origin. We assume that there were some propagation delays or packet loss and the local PLL was not locked on the PTP data processing thus the difference started to deviate.

For the sake of comparison, we also tested the performance of PTP transfer in our laboratory. We altered the previous scenario with setting our M600 PTP server as the

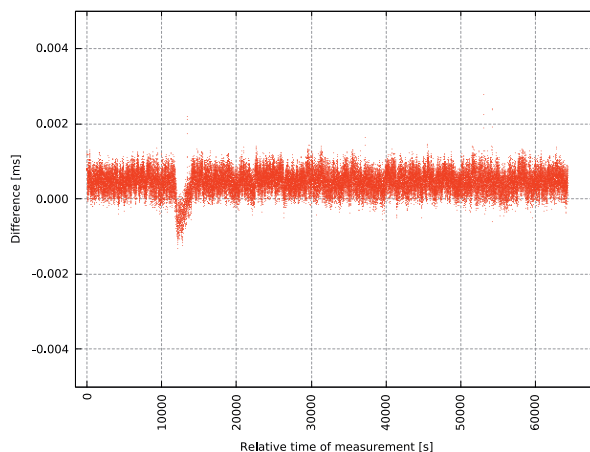


Figure 9.3: Multicast communication PTP test between Prague data center (master) and Prague laboratory (slave).

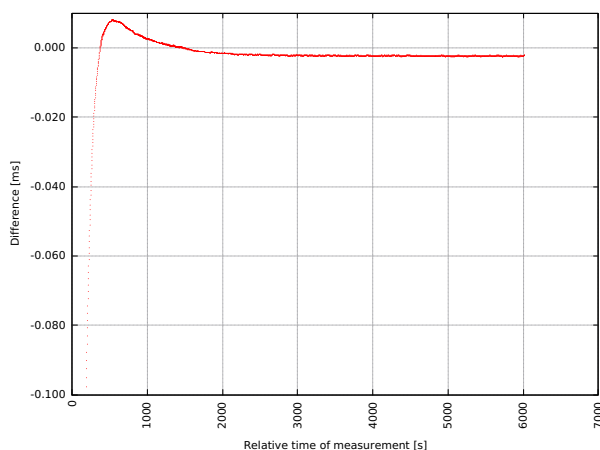


Figure 9.4: SolarFlare card SfPTPd multicast communication PTP test between Prague data center (master) and Prague laboratory (slave).

grandmaster. Another difference was that our grandmaster used the cesium atomic clock as the source of the precise time. Then we run a measurements on the same PTP slave card. In the first measurement (In Figure 9.2), the PTP transfer was unicast and in the second measurement (In Figure 9.3), the PTP protocol packets were transferred by a multicast method. It was observed that the unicast transfer has the same spikes in 1PPS difference while the multicast transfer is free of these artefacts. We assume that the cause is the same as in the distant scenario. Another possible cause is the fact that the unicast transfer is not preferred as a primary transfer method (original PTP specification has only L2 multicast while the unicast messaging was added in version 2 [26]) an there may be design flaws or undocumented behavior of the PTP network devices.

9. LONG DISTANCE IEEE 1588 EVALUATION

Scenario	Average	Standard deviation
Distant Unicast	-0.39E-06	0.60E-06
Lab Unicast	3.41E-06	0.43E-06
Lab Multicast	0.46E-06	0.33E-06

Table 9.1: IEEE 1588 measurements evaluation

We also tested (In Figure 9.4) network card from the vendor SolarFlare which is PTP enabled with 1PPS output. The card and its PTP stack called `sfptpd` was be able to process only the multicast traffic (at the time of measurement). In the figure is visible the locking of the local clock and then generation of stable 1PPS signal.

It is clear from the previous graphs that the measured values for unicast transfer apparently do not have normal distribution an therefore calculated values of standard distributions can not be used for evaluation. Our experiment showed that the unicast communication between IEEE 1588 master and slave can be used also for long distance time transfer with uncertainty in microseconds order.

Time Services in CESNET Network

Provisioning of accurate time (and frequency) is important type network services. Although being often considered by network operators as something on the edge of their interest, CESNET network has long tradition in operating network services of the highest possible quality. In this section a current portfolio of time services, their main parameters and plans for future are described as we presented it on TNC 15 conference [A.9].

10.1 Accurate time and frequency source

The most important part of time and frequency network infrastructure are accurate, stable and reliable sources. As we have fortunately in CESNET free access to flat roof, it was no problem to install GPS receivers. During last 15 years, we checked products of several companies: GARMIN (GPS 35 and GPS 18), MOTOROLA (Oncore) and TRIMBLE (Acutime), the last one is used until now as a primary GPS reference. Later was unavoidable to use atomic clock – we started with Rubidium clock (PRS 10 from Stanford Research systems) that served as time keeper in case of GPS fault. And finally, we installed primary Cesium reference (5071A from Symmetricom). In order to know exact time offset of the Cesium clock from UTC, we use specialized calibration GPS system GTR-50 from Dicom.

As we operate also optical time transfer (as described in section 6), we have connected our Cesium clock to the Czech national time and frequency laboratory in UFE (Institute of Photonics and Electronics of Czech Academy of Sciences). This way, we are ready to report our Cesium clock to BIPM (Bureau International des Poids et Mesure), where is the UTC time scale constructed based of time scales of participating laboratories. This way we will also contribute to UTC.

10.2 NTP – Network Time Protocol

CESNET operates primary (Stratum-1) NTP servers since year 2000. Until 2012, they were controlled by GPS receiver, now we use the Cesium clock. Our NTP servers were

upgraded several times in last 15 years and we reached significant improvement of time accuracy. We observed the best clock stability and performance using simple and cheap main boards without any proprietary improvement (adaptable power consumption, proprietary monitoring, improvements in BIOS...). Until now, we were not able to find a modern, rack-mountable computer suitable for next generation of NTP servers. All tested platforms had problem to achieve acceptable clock stability due to rapid changes of the ambient temperature and significant variation of the interrupt latency degrading stability of the external time signal. Thus, our goal was to develop own generic NTP server with following parameters:

- best available internal clock stability,
- effective processing of 1 PPS (pulse per second) signal,
- high performance,
- authentication support,
- rack-mountable 1U case,
- low power consumption,
- acceptable cost,
- optional front panel display showing server time and status.

10.2.1 Hardware

10.2.1.1 Mainboard

We focused on finding of a suitable mini-ITX format mainboard in order to avoid large heating mainboards. Our latest generation of NTP servers uses Intel boards, unfortunately Intel discontinued production of such category of boards. The performance of the mainboard equipped by 1+ GHz CPU is good enough even for heavy loaded NTP server.

10.2.1.2 Oscillator

All boards (i386 family and later) have 14.318 MHz quartz oscillator from which both the main clock frequency and the CPU clock are derived. Although the NTP package controls the system clock by a digital, kernel implemented PLL (Phase-Locked Loop), the short stability of the clock is done solely by the quartz oscillator stability. We can significantly improve the clock stability by replacement of the original crystal by another one with lower temperature dependence or by an ovenized oscillator (OCX). We have designed and manufactured an oscillator module (In Figure 10.1) which is directly connected to the mainboard instead of the removed crystal. As the oscillator module contains a programmable frequency multiplier/divider, it can be controlled by OCX block having nearly any

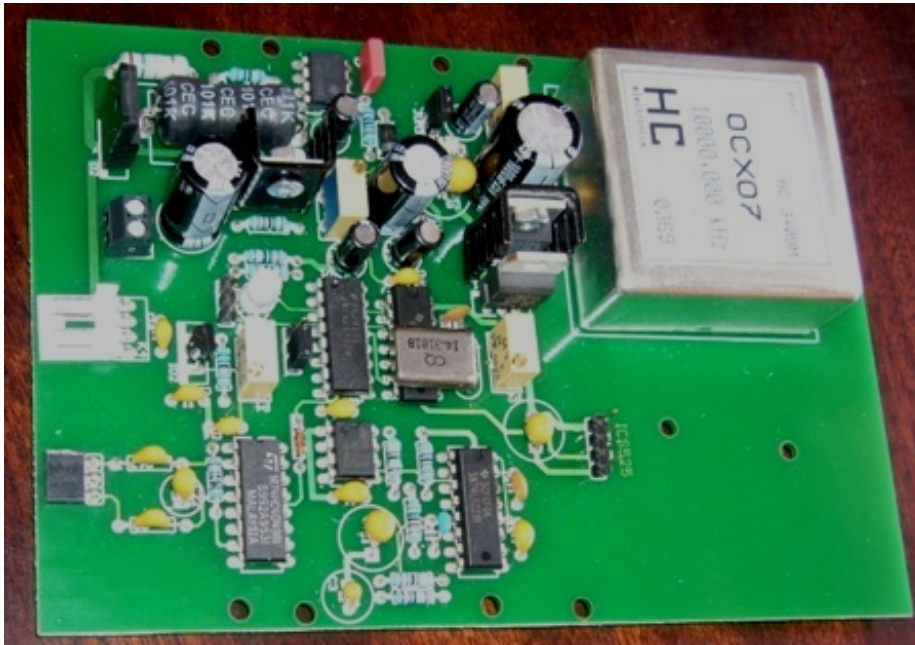


Figure 10.1: OCXO module [A.9].

frequency, for instance 10 MHz. The oscillator module has the same size and position of mounting holes as the 3.5" hard disk and therefore it can be easily installed in a free disk slot. Detailed description of the OCX module and its parameters is out of scope of this document. Figures 10.3 and 10.4 provide comparison of internal clock with and without OCXO module.

10.2.1.3 PPS Capture Card

The 1 PPS signal is traditionally connected to the DCD pin of a serial port. An edge of the signal invokes an interrupt and the interrupt routine assigns timestamp to this event. Unfortunately, the interrupt latency is not constant and its statistical distribution depends on the CPU load. An elimination of the interrupt latency requires some hardware support - there exist processors equipped by the programmable timers but utilizing a specialized PCI card is more generic and reproducible solution.

We have designed such an adapter – its brief schema is shown in Figure 10.2. The adapter contains an oscillator with frequency f , a counter, and a control logic. The active edge of the 1 PPS signal starts counting of pulses of the on-board oscillator and causes a CPU interrupt. In the interrupt routine, the timestamp T_0 is evaluated and the counter register is read – its value n corresponds to the time elapsed since the last 1 PPS pulse. Knowing the actual time T_0 and the counter value n , the exact timestamp T of incoming signal is calculated:

$$T = T_0 - n/f$$

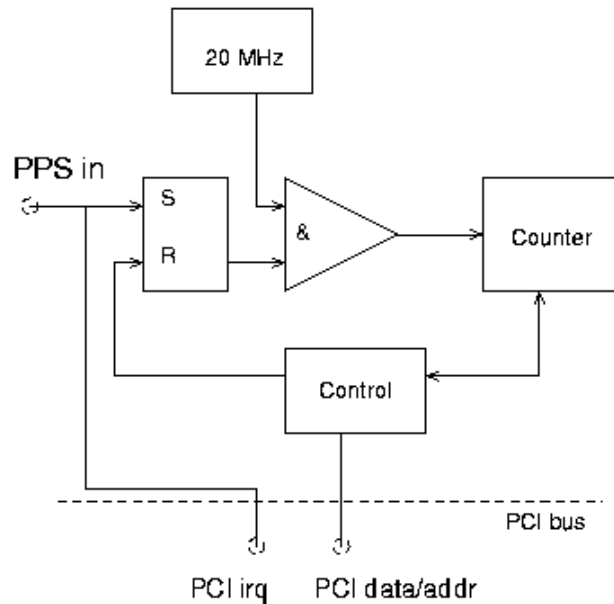


Figure 10.2: PPS capture card diagram [A.9].

While the typical interrupt latency of a serial port ranges from 8 to 50 microseconds, the PPS capture card allows decrease it down to 50 nanoseconds with 20 MHz oscillator. Further reduction of uncertainty is not possible as counter reading requires transfer of at least 3 bytes on the PCI bus. The designed PPS capture card has been implemented in cooperation with Tedia company as a customized FPGA program for their PCI cards PCD-7004 and PCT-7424. Device driver has been written by us and is available under the GNU GPL license.

10.2.1.4 Case

There are available many desktop mini-ITX cases but only a few of them are rack-mountable. The case EM-161LB, made by the company EMKO Case perfectly fits for our NTP server. This case is 1U high and allows installation of two 3.5 inch disks and one PCI card. The position of the second disk is occupied by the OCX module. The case with all components is shown in Figure 10.5.

10.2.2 Software

Operating System We run all our NTP servers under Linux. Some server still use kernel version 2.4.33 which was approved as very stable for that purpose. Never servers run 2.6.x kernel, however we should be careful in system upgrade as not all kernels of 2.6 family are suitable for the 1 PPS signal processing.

10.2.2.1 NTP Package

We use the latest NTP stable version (currently 4.2.8) that we have compiled from source. Versions included in distribution packages usually lack support for 1 PPS signal and therefore are not useable for primary NTP server.

10.2.3 Performance

We run currently four primary NTP servers in two localities. Our best known and therefore mostly utilized NTP server tik.cesnet.cz responds 500–1000 queries per second. Following two graphs demonstrate the influence of ovenized oscillator on internal clock stability.

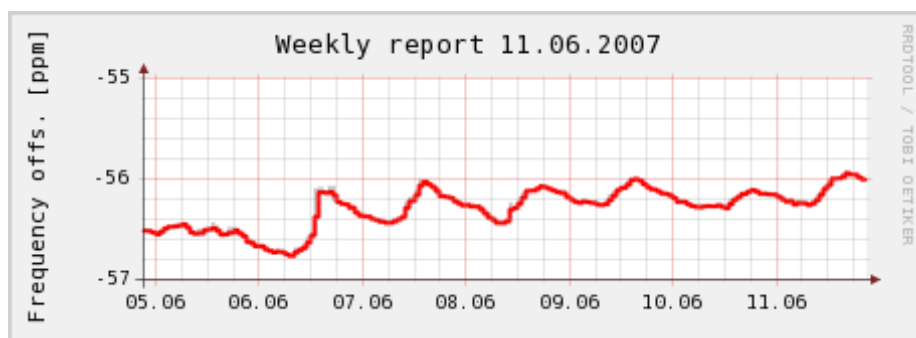


Figure 10.3: Crystal frequency offset [A.9].

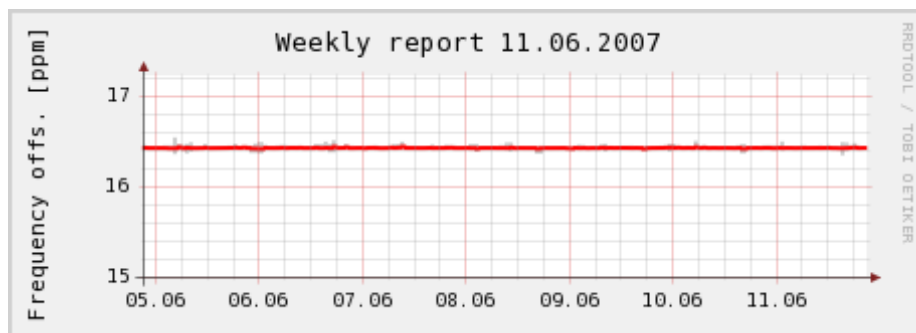


Figure 10.4: OCXO frequency offset [A.9].

10.3 TSA – Timestamp Authority

Time-stamp authority (TSA) is a service providing electronic message containing the trustworthy time of issuance. The time-stamp protocol (TSP) specified in the RFC 3161 defines both the communication protocol and the time-stamp format. The time information must

be provided with a resolution of 1 second or finer. The TSP is based on a request message sent by a client (Time Stamp Query – TSQ) and a response message (Time Stamp Reply – TSR) sent by the server. The TSQ can contain arbitrary information (e.g., one-way hash of a file), which is returned back after being signed in the TSR. This way, the information is bound with the time stamp. The TSA clock must be synchronized by a trustworthy method to a time system that ensures traceability to UTC. According to the RFC3161, the TSP can be implemented either on the 3rd network level (TCP) or on the 4th network level (e.g., HTTP or SMTP).

Although there exists an open source implementation (originally the OpenTSA project, later it was merged into OpenSSL package), many of operating TSAs are commercial proprietary solutions and therefore it is very difficult even for the operator to give an evidence of TSA time service accuracy. In some countries, the law forces every licensed TSA operator to calibrate its TSA system.

10.3.1 TSA Hardware

In CESNET, we have built own TSA servers. The TSA runs at standard server with 64-bit Intel processor. The hardware includes specialized cards that support data encryption and system clock synchronization. CPU performance is not critical, however the data encryption card requires 64-bit operating system.

10.3.1.1 Hardware Security Module

In our application, the hardware security module serves mainly for safe keeping of the private key. We decided to use SCA6000 card from Sun Microsystems – we already successfully utilized it in the CESNET Certification Authority. This card also accelerates cryptographic operations (up to 13000 RSA operations per second) therefore time stamp signing is not a bottleneck of our TSA. However, more important is the system immunity from service compromising as the private key is neither stored in the system memory nor used by the CPU. If the intruder hacks the operating system or even steals the computer, he has no way how to read the private key.

10.3.1.2 Clock

The system is also equipped with the specialised card, which processes the incoming PPS (pulse per second) signal without interrupt latency. The PPS signal represents external time source which is used by the NTP daemon for system clock synchronization. The clock stability is further improved by the ovenized oscillator which replaces the motherboard crystal 14.318 MHz. These two hardware hacks are the same as in NTP servers.

10.3.1.3 TSA Software

The card SCA6000 is the Sun Microsystems proprietary hardware and the manufacturer provides driver only for Solaris and Linux. We decided to use Linux but only limited



Figure 10.5: NTP server internals [A.9].

number of distributions and kernels is supported – the newest one is Red Hat 6.0 with x86-64 kernel version 2.6.32. The OpenTSA package unfortunately lacks the SCA6000 card support and we were forced to program this interface ourselves. Another important system program is the NTP daemon (we have installed recent version 4.2.8) which is responsible for TSA clock synchronization.

10.3.2 Clock Synchronization

Basic assumption of every time stamp is that the issuing TSA guarantees declared level of internal clock accuracy. The TSA operator must arrange the clock synchronization by a trustworthy method to any time system that ensures traceability to UTC. Our TSA are provided by 1 PPS signal from Cesium clock. In order to reach the best possible internal clock accuracy, we developed a simple method that allows to calibrate the system clock: the utility `gen_pps` transmits $T(\text{TSA})$ to a serial port in the form of generated PPS signal and this signal is compared with reference clock. Long-time measurement proved that we synchronize time scale of our TSA prototype with uncertainty less than 2 microseconds. Figure 10.6 shows example of such one-day measurement. Described method of system clock synchronization was adopted from our design of CESNET primary NTP servers.

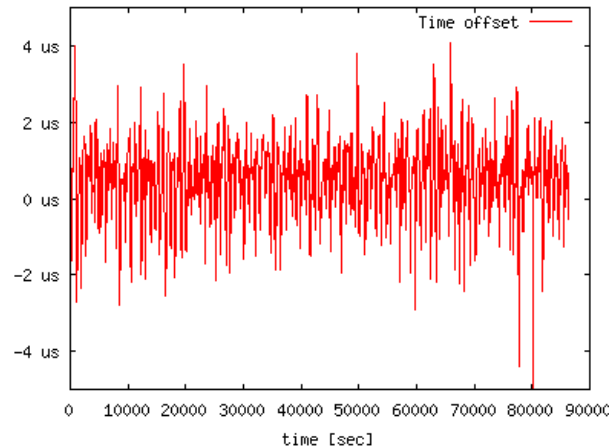


Figure 10.6: TSA clock accuracy [A.9].

10.4 IEEE 1588

The IEEE 1588 (also known as the “Precise Time Protocol” PTP) protocol was developed for the need of precise time distribution with more precise synchronization (in comparison to the NTP protocol). It is not a RFC by IETF but it is standardized under IEEE supervision. This protocol is intended to be a standard for devices connected via switched IP networks. The accuracy of synchronization is intended to be beyond one microsecond. The current version of PTP protocol is number 2 from the year 2008. You can find the overview of this protocol in chapter 2.2.4.

In the CESNET network, we are operating various PTP enabled devices mostly for testing and research purposes:

- Meinberg LanTime M600 Grandmaster connected to the atomic clock as a source of precise time
- Meinberg LanTime M600 Grandmaster connected to the GPS receiver as a source of precise time
- Meinberg PTP207PEX PCI card, PTP slave operation.
- White Rabbit Switch, PTP enabled.
- White Rabbit PCI Card, PTP enabled.
- Hirschmann industrial Ethernet grade 10 Mbps switch

10.5 Time and Frequency Transfer Infrastructure

Network time services are oriented to ordinary network users. However, there exists also demand for transfer of time with sub-nanoseconds accuracy and frequency having stability

better than 10⁻¹³, i.e. with parameters similar to atomic clocks. CESNET has been building the national infrastructure for accurate time and stable frequency distribution (TF infrastructure) with possible reach to neighbouring countries. The major goals of this TF infrastructure are:

- To transfer the accurate time from existing Cesium primary standard to the national TF laboratory located in Institute of Photonics and Electronics (UFE) in order to improve accuracy and stability of the national approximation of Coordinated Universal Time (UTC).
- To compare national approximation of UTC with national approximations in the neighbouring countries (e.g. Austria, Slovakia and Poland).
- To distribute accurate time and stable frequency to users from both academia and R&D spheres.

Figure 10.7 shows current PoP of the TF/infrastructure. The first and longest link (550 km) between national time and frequency laboratories UFE (Prague) and BEV (Vienna) is in operation since 2011. We also set up new single-fiber bidirectional links CESNET – UFE and VUGTK – UFE (Figure 10.7). The time stability of link VUGTK – UFE is shown in Figure 11. The uncertainty is about 30 ps.

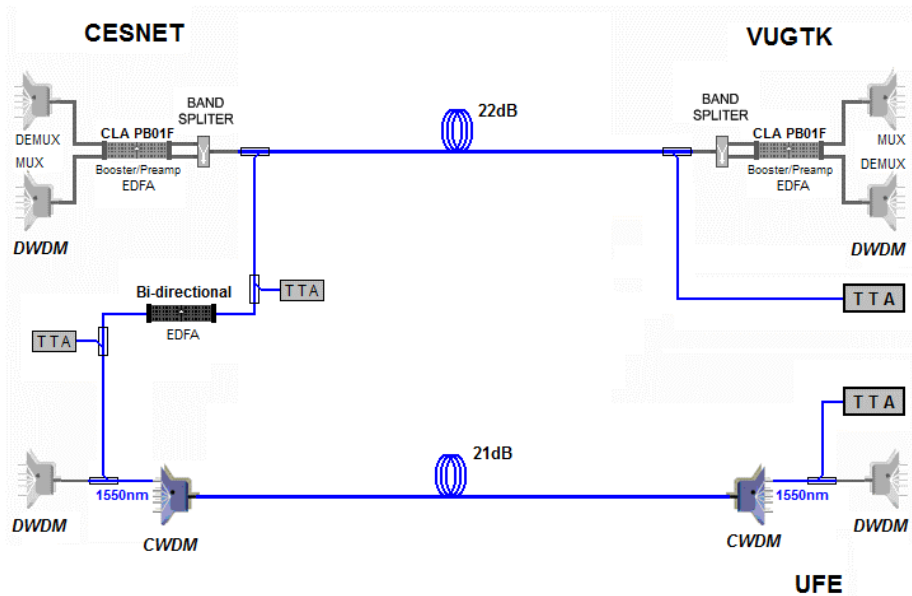


Figure 10.7: Single-fiber lines UFE–CESNET–VUGTK Pecny [A.9].

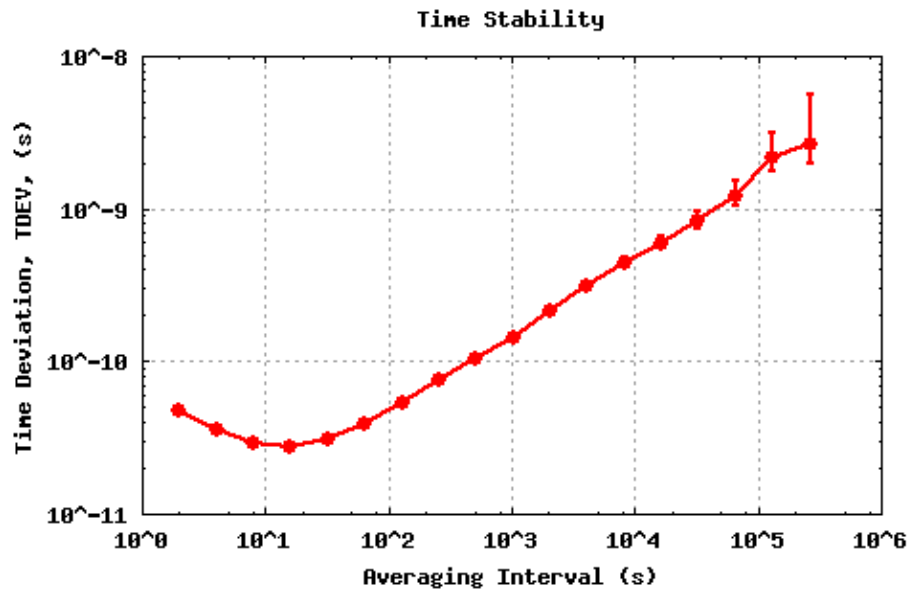


Figure 10.8: Time stability [A.9].

10.5.1 Summary

CESNET is providing variety of time services for an ordinary network operation. We are also developing and operating the time and frequency transfer infrastructure. We will continue our research to improve accuracy and quality of network time services.

Future Work

11.1 New Generation of NTP Servers Platform

As it was partially described in sections 8 and 10, the architecture of main CESNET NTP servers `tik.cesnet.cz` and `tak.cesnet.cz` is outdated and at the end of life. We are working on the main issue how to feed external frequency into Intel processor. We were able to run Intel platform fed by custom build external frequency source thus we have a proof of concept. We are dealing with glitch-free frequency switching from TCXO to OCXO and designing the final PCB with tuned analog part now. After overwhelming these issues we will have ready-to-run new platform with 10 years life cycle.

One could say that designing a custom build precise frequency HW platform for Linux `ntpd` is a useless work because there exists commercially available platforms with similar functionality (one of these products is e.g. Meinberg Lantime M600 [30] which is both NP and PTP capable). That is partially true but our approach lead to HW platform with known parameters running widely used NTP stack on Linux OS so we can at least compare and evaluate performance of such a system.

11.2 Transparent Clock with Deterministic Delay

One feature of IEEE 1588 is the transparent clock (described in chapter 2.2.4.4). TC have been added to version 2 of the IEEE 1588 as an improved method of forming cascaded topologies. Ethernet switches are presently implemented mainly using a store-and-forward architecture (in case of an “intelligent” or “higher-layer” switch the architecture is more complex) and the frame processing time varies due to I/O queues (FIFOs) passing. Transparent clocks compensate the varying switch occupancy time with modifying timestamps in PTP packet.

This functionality is an advantage in a simple network topology in time of the normal operation. When we have a dedicated network topology for testing and IEEE 1588 evaluation this behavior may be undesirable (e.g. while measuring another parameters than

TC performance). Our approach is to develop (with the knowledge of timestamping) the PTP transparent clock with deterministic delay that can be added to the existing standard Ethernet switch.

11.3 Expanding the System on Chip Design

We have successfully migrated our previous designs from obsolete Xilinx Virtex-5 FPGA 7. We have partially integrated our previously developed TIC cores and proposed a new design in our paper [A.1] which take advantage of Zynq SoC both FPGA and ARM parts.

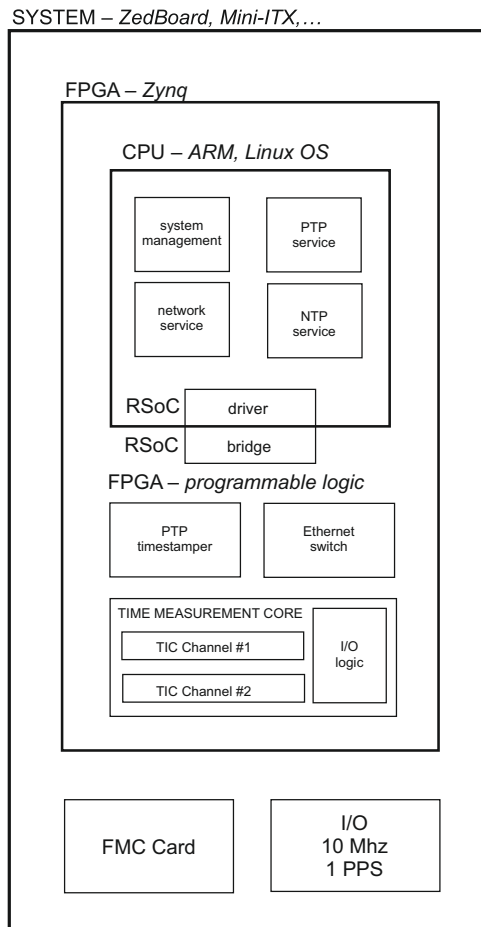


Figure 11.1: Block diagram of the SoC [A.1].

The precise time measurement system must be capable of the following functionality:

- Time Interval Counter with accuracy better than 1 ns.
- Management and evaluation of measurements.

- Network interface.
- IEEE 1588 timestamper.
- Multiple fast I/O capability for SFP+ transceivers.
- IEEE 1588 timestamper.
- Ability to implement our another previously designed units.
- Simple interface between software and hardware.

All these requirements are satisfied by the proposed SoC for Comparison of Precise Time Sources. There is the block diagram of the proposed SoC in Figure11.1.

The Zynq FPGA includes two Cortex-A9 cores for user programs. We intend to run a Linux OS on this CPU with several services. The first one is the collection of measurements data and their evaluation – we have software for this functionality from previous project [A.4]. The second one is the network service for communication over the standard TCP/IP network. Another services can be operated on the system such a PTP or NTP stack.

RSoC Framework acts as a middleware and hides underlying hardware platform. Framework has two main components: RSoC Driver and RSoC Bridge. The Linux OS RSoC Driver enables programs to exchange data with the FPGA design. It uses the interface such a mmap, write/read and ioctl. The RSoC Bridge is a configurable IP core and integrates the rest of the FPGA user logic with existing AXI bus.

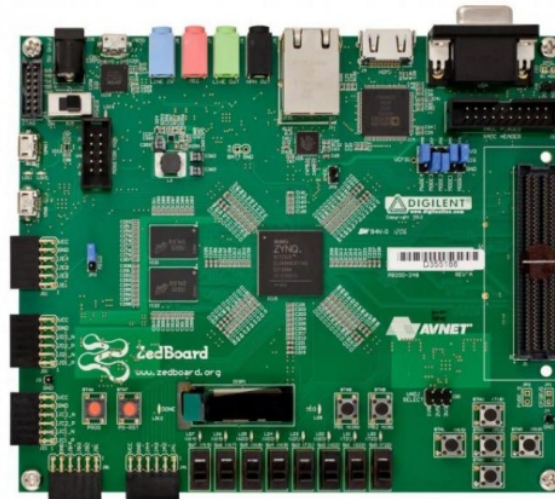


Figure 11.2: ZedBoard platform for basic Zynq SoC evaluation [31].

The Time Measurement Core is a multipurpose unit for precise time interval measurements. More exactly, it is the proven Dual Channel Time Interval Counter from our

previous work on atomic clock timescales comparison [A.3] that is fitted in the new FPGA structure with modified supportive logic.

The core was originally implemented in the Virtex-5 FPGA. Its time interval counter operates on 400 MHz clock signal and the time events are interpolated within the corresponding period of 2.5 ns. The interpolation is done by the sampling of the time signal progress in a carry logic based delay line. As the time interval counter is in the 400 MHz clock domain, the digital design is challenging due to timing violations. Moreover, the part of the design is asynchronous and requires some specific type of design. Similar designs of the FPGA based interval counter exists, some are more complex (e.g. with 4 channels and multiphase clock [32][33][34]). Our experience is that the interpolating counter with one long delay line (with corresponding number of delay elements) is sufficient for our applications. We reach resolution about 17 ps.

On the Zynq FPGA, there is a space for more than two channels – that number was the limitation of the Virtex-5 based design. Also the timing and routing is better. Having more Time Interval Counters, we can make more complex time measurements e.g. compare several remote clocks.

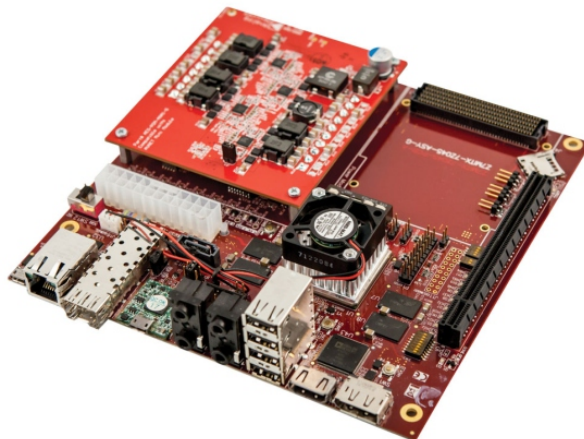


Figure 11.3: Mini-ITX is an advanced Zynq platform intended to operate as a built-in board in rack-mounted device [35].

We chose two existing platforms. The ZedBoard [31] (in Fig. 11.2) is primarily designed as a testing and evaluation board for the Zynq projects. It can be also deployed in a laboratory setting where the absence of the case and other external components is not critical. For the production deployment we want to utilize the Mini-ITX [35] (in Fig. 11.3) platform that is designed to work within some device e.g. rack-mounted time transfer adapter.

Conclusions

This thesis is dealing with precise time transfer in local networks – local does not mean the size of topology but the used architecture and technology on Layer 1 and 2 of OSI model. The objective is achieved by two different approaches: precise time and frequency transfer in optical fibers and network time protocols. In the chapter 2 is described theoretical background and state-of-the-art: time and clocks, overview of network time protocols, time and frequency transfer in optical fibers and measurements of time intervals. After describing our approach in chapter 3 are presented the main results of our research: IEEE 1588 timestamper, atomic clock comparison, architectures for precise time measurements, running processor on external frequency, long distance evaluation of IEEE 1588 performance and time services in CESNET network. At the end is mentioned the ongoing and proposed work.

12.1 Comparison of Goals and Achieved Results

1. Distant atomic clock time scale comparison over optical link

The optical network adapter with an embedded counter was developed, described in chapters 5, 6 and 7. The development of embedded time interval counter in FPGA enabled creation of a compact adapter – its original version demanded external time interval counters and other devices that make utilization of these adapters nearly impossible outside the laboratory environment. The performance of our system is comparable with previously used Pendulum CNT-91 counters (measurement uncertainty of time intervals about 50 ps) with lower costs. Currently are 3 pairs of these adapters successfully deployed and operated in a standard optical network environment.

2. IEEE 1588 evaluation

Chapter 9 contains the description of our tests on long distance unicast/transport layer PTP synchronization performance which is not standard and neglected mode of operation. We compared 1PPS signal obtained from a remote PTP grandmaster and

local 1PPS signal from our primary frequency standard – cesium atomic clock. This goal is also covered in chapter 4 where our FPGA based IEEE 1588 timestamper is described. A PTP infrastructure monitoring tool was also developed [A.16].

3. Contribution to CESNET network time services

A new time services were deployed in the CESNET network and the existing ones were improved (covered in chapter 10). Every device with a possibility to get external time is now connected to our atomic timescale. The new testing IEEE 1588 infrastructure was deployed including White Rabbit devices. This infrastructure is used in another research and is also accessible to the Czech technical university students and academic staff, several thesis was completed utilizing this infrastructure ([A.14] or [A.16]).

Another non standard issue is sourcing the modern processor with an external stable frequency. This approach is utilized in our new generation of NTP servers described in chapters 8 and 10.2.

12.2 Original Results

- Design of dual interpolating counter architecture 6 7.
- Design and implementation of new adapters for atomic timescale comparison 5.
- IEEE 1588 performance testing 9.
- Clocking CPU with external stable frequency 8.
- Design of the new generation of NTP servers, implementation of IEEE 1588 time service in CESNET 10.

Bibliography

- [1] Allan, D. W. and Weiss, M. A., *Accurate Time and Frequency Transfer During Common-View of a GPS Satellite*, 34th Annual Frequency Control Symposium, pp. 334–346, May 1980.
- [2] *Systems of Time*, <https://tycho.usno.navy.mil/systime.html>
- [3] Mills, D., *Network Time Protocol (NTP)*, RFC 958, Linkabit, September 1985.
- [4] Postel, J., *Daytime Protocol*, STD 25, RFC 867, May 1983.
- [5] Postel, J. and Harrenstien, K., *Time Protocol*, STD 26, RFC 868, May 1983.
- [6] Postel, J., *User Datagram Protocol*, STD 6, RFC 768, August 1980.
- [7] Postel, J., *Internet Control Message Protocol*, STD 5, RFC 792, September 1981.
- [8] Mills, D., *Network Time Protocol (Version 3) Specification, Implementation and Analysis*, RFC 1305, March 1992.
- [9] H3C Technologies, *H3C S3100-52P Ethernet Switch Operation Manual-Release 1500*, V1.02.
- [10] Gotoh, T., Imamura, K. and Kaneko, A., *Improvement of NTP time offset under the asymmetric network with double packets method*, Conference on Precision Electromagnetic Measurements 2002, pp. 448–449.
- [11] Weibel, H., *Technology Update on IEEE 1588: The Second Edition of the High Precision Clock Synchronization Protocol*, 2009.
- [12] Cohen, R., *Precision Time Protocol: IEEE1588v2*, TICTOC BOF IETF Prague 2007.
- [13] *IEEE standard for a precision clock synchronization protocol for networked measurement and control systems*. New York, 2008. ISBN 978-073-8154-008.

- [14] Dreher, A. and Mohl, D., *Precision Clock Synchronization – IEEE 1588*, Rev. 1.2, Hirschmann Automation and Control GmbH, 2010.
- [15] Lipinski, M., Wlostowski, T., Serrano, J. and Alvarez, P., *White rabbit: A PTP application for robust sub-nanosecond synchronization*, IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication, ISPCS, 2011, pp. 25-30.
- [16] Serrano, J., Alvarez, P., Cattin, M., Cota, E. G., Lewis, P. M. J. H., Wlostowski, T. et al., *The White Rabbit Project* in Proceedings of ICALEPCS TUC004, Kobe, Japan, 2009.
- [17] *White Rabbit*, <https://www.ohwr.org/projects/white-rabbit>
- [18] *White Rabbit Specification: Draft for Comments*, version 2.0 (06-07-2011), <https://www.ohwr.org/documents/160>
- [19] Sliwczynski, L., Krehlik, P. and Lipinski, M., *Optical fibers in time and frequency transfer*, Measurement Science and Technology, 21 (7), 2010, <http://iopscience.iop.org/article/10.1088/0957-0233/21/7/075302/pdf>
- [20] Kalisz, J., *Review of methods for time interval measurements with picosecond resolution*, Metrologia, Vol.41, No.1, pp. 17–32, 2004.
- [21] Smotlacha, V., Kuna, A., and Mache, W., *Time Transfer Using Fiber Links*, in Proceedings of the EFTF 2010.
- [22] Smotlacha, V., Kuna, A., and Vojtech, J., *Optical Infrastructure for Time and Frequency Transfer*, in Proceedings of the EFTF 2013.
- [23] Smotlacha, V., Kuna, A., and Mache, W., *Time Transfer in Optical Network*, in Proceedings of the 42nd Annual Precise Time and Time Interval (PTTI) Systems and Applications Meeting, Reston, Virginia, USA, 2010, pp. 427-436.
- [24] US Naval Observatory, *About the CGGTTS data format*
- [25] *Quartz Crystal Oscillators*, <https://www.electronics-tutorials.ws/oscillator/crystal.html>
- [26] *IEEE standard for a precision clock synchronization protocol for networked measurement and control systems*. New York, 2002.
- [27] *IEEE standard for a precision clock synchronization protocol for networked measurement and control systems*. New York, 2008.
- [28] *CNT-91 Frequency Timer/Counter/Analyzer/Calibrator* <https://pendulum.se/products/frequency-counters-analyzers/cnt-91-frequency-timercounteranalyzer/calibrator/>

-
- [29] *5071a Cesium Primary Frequency Standard* <https://www.microsemi.com/product-directory/cesium-frequency-references/4115-5071a-cesium-primary-frequency-standard>
- [30] *Meinberg M600 Grandmaster PTP*, <https://www.meinbergglobal.com/english/archive/ptp-time-server-gps.htm>
- [31] *ZedBoard*, <http://zedboard.org/product/zedboard>
- [32] Aloisio, A., Branchini, P., Cicalese, R., Giordano, R., Izzo, V., Loffredo, S. and Lomoro, R., *High-resolution time-to-digital converter in field programmable gate array*, in Proceedings of Topical Workshop on Electronics for Particle physics (TWEPP), 2008.
- [33] Favi, C. and Carbon, E., *A 17 ps Time-to-Digital Converter Implemented in 65nm FPGA Technology*, 2009.
- [34] Pedersen, K., *Low cost, high performance frequency/interval counters*, 2008.
- [35] *Mini-ITX Board*, <http://zedboard.org/product/mini-itx-board>
- [36] *SR620 – Time interval/frequency counter*, <https://www.thinksrs.com/products/sr620.html>
- [37] *PTP270PEX: IEEE1588-2008 slot card for computers* <https://www.meinbergglobal.com/english/archive/ptp270pex.htm>
- [38] Smotlacha, V., and Kuna, A., *Two-way Optical Time and Frequency Time Transfer between IPE and BEV*, in Proceedings of 26th European Frequency and Time Forum, Gothenburg, Sweden, 2012.
- [39] Vojtech, J., Smotlacha, V., Skoda, P., Kuna, A., Hula, M., and Sima, S., *Photonic services, their enablers and applications*, SPIE Optics and Photonics, San Diego, USA, 2012.
- [40] Loffredo, S., *Design, construction and tests of a high resolution, high dynamic range Time to Digital Converter*, 2010.
- [41] Kalisz, J. and Szplet, R., *A PC-based time interval counter with 200 ps resolution*, 2003.
- [42] Wu, J. and Shi, Z., *The 10-ps Wave Union TDC: Improving FPGA TDC Resolution beyond Its Cell Delay*
- [43] Gordon, C., *Introduction to IEEE 1588 & Transparent Clocks*, Tekron International, 2009.
- [44] *White Rabbit Technology*, <http://sevensols.com/index.php/projects/white-rabbit-technology/>

Reviewed Publications of the Author Relevant to the Thesis

- [A.1] Dostal, J. and Smotlacha, V., *System on Chip for Comparison of Precise Time Sources*, In Proceedings of 2016 IEEE East-West Design & Test Symposium (EWDTS). IEEE, 2017. ISBN 978-1-5090-0693-9.
- [A.2] Dostal, J. and Smotlacha, V., *Next generation of architecture for precise time measurements*, In Proceedings of 2015 IEEE East-West Design & Test Symposium (EWDTS). IEEE, 2015. ISBN 978-1-4673-7775-1.
- [A.3] Dostal, J. and Smotlacha, V., *Dual interpolating counter architecture for atomic clock comparison*, In Proceedings of IEEE East-West Design & Test Symposium. IEEE, 2014. pp. 32-35. ISBN 978-1-4799-7630-0.
- [A.4] Dostal, J. and Smotlacha, V., *Atomic Clock Comparison over Optical Network*, In Proceedings of the IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS), Abu Dhabi, United Arab Emirates, IEEE, 2013. pp. 682-685. ISBN 978-1-4799-2452-3.
- [A.5] Dostal, J. and Smotlacha, V., *The hardware architecture and device for accurate time signal processing*, In Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2013). IEEE, 2013, ISBN 978-1-4799-2095-2.

Remaining Publications of the Author Relevant to the Thesis

- [A.6] Krehlik, P., Sliwczynski, L., Dostal, J., Radil, J., Smotlacha, V., Velc, R., Vojtech, J., Campanella, M., Calonico, D., Clivati, C., Levi, F., Cip, O., Rerucha, S., Holzwarth, O., Lessing, M., Camargo, F., Desruelle, B., Lautier-Gaud, J., English, E.L., Kronjager, J., Whibberley, P., Pottie, P.-E., Tavares, R., Tuckey, P., John, F., Snajder, M., Stefl, J., Nogas, P., Urbaniak, R., Binczewski, A., Bogacki, W., Turza, K., Grosche, G., Schnatz, H., Camisard, E., Quintin, N., Diaz, J., Ros, E., Galardini, A., Seeds, A., Yang, Z., Amy-Klein, A., *CLONETS – Clock network services: Strategy and innovation for clock services over optical-fibre networks*, 2017, International Conference on Transparent Optical Networks, art. no. 8024939, .
- [A.7] Krehlik, P., Sliwczynski, L., Dostal, J., Radil, J., Smotlacha, V., Velc, R., Vojtech, J., Campanella, M., Calonico, D., Clivati, C., Levi, F., Cip, O., Rerucha, S., Holzwarth, R., Lessing, M., Camargo, F., Desruelle, B., Lautier-Gaud, J., English, E.L., Kronjager, J., Whibberley, P., Pottie, P.-E., Tavares, R., Tuckey, P., John, F., Snajder, M., Stefl, J., Nogas, P., Urbaniak, R., Binczewski, A., Bogacki, W., Turza, K., Grosche, G., Schnatz, H., Camisard, E., Quintin, N., Diaz, J., Garcia, T., Ros, E., Galardini, A., Seeds, A., Yang, Z., Amy-Klein, A., *CLONETS - Clock network services strategy and innovation for clock services over optical-fibre networks*, 2017, Joint Conference of the European Frequency and Time Forum and IEEE International Frequency Control Symposium, EFTF/IFC 2017 – Proceedings, art. no. 8089004, pp. 697–698
- [A.8] Calonico, D., Clivati, C., Levi, F., Krehlik, P., Śliwczyński, L., Dostal, J., Radil, J., Smotlacha, V., Velc, R., Vojtech, J., Campanella, M., Číp, O., Rerucha, S., Holzwarth, R., Lessing, M., Saint-Jalm, S., Camargo, F., Desruelle, B., Lautier-Gaud, J., English, E.L., Kronjäger, J., Whibberley, P., John, F., Šnajder, M., Štefl, J., Nogaś, P., Urbaniak, R., Binczewski, A., Bogacki, W., Turza, K., Grosche, G., Schnatz, H., Camisard, E., Quintin, N., Diaz, J., García, T., Ros, E., Galardini,

- A., Seeds, A., Yang, Z., Amy-Klein, A., Bookjans, E., Pottie, P.-E., Tuckey, P., *The H2020 European project CLONETS: Clock services over optical-fibre networks in Europe*, European Frequency and Time Forum, EFTF 2018, pp. 285–289.
- [A.9] Dostal, J. and Smotlacha, V., *Time Services in CESNET Network*, TNC15 Networking Conference (TNC), 2015
- [A.10] Dostal, J., *Precision Time and Frequency Distribution*, Počítačové architektury a diagnostika (PAD), 2013, pp. 99-103, ISBN 978-80-261-0270-0.
- [A.11] Dostal, J., *Přenos času a frekvence v lokálních sítích*, Počítačové architektury a diagnostika (PAD), 2012, pp. 43-48, ISBN 978-80-01-05106-1.
- [A.12] Dostal, J., *Hardware Support For Precise Time and Frequency Distribution*, Embedded Systems Workshop (EWS), 2013.
- [A.13] Dostal, J., *Time and Frequency Transfer in Local Networks*, Doctoral Study Report, 2013.

Supervised Publications Relevant to the Thesis

- [A.14] Slabihoudek, M. and Dostal, J., *Systém synchronizace času pomocí protokolu PTP*, Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.
- [A.15] Hulle, R. and Dostal, J., *NTP server na platformě ARM*, Bakalářská práce, Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2012.
- [A.16] Hanousek, V. and Dostal, J., *Monitorovací systém protokolu IEEE 1588*, 2016.

Laboratory Equipment



Figure A.1: Meinberg M600 PTP grandmaster clock [30].



Figure A.2: 5071A Cesium Primary Frequency Standard [30].

A. LABORATORY EQUIPMENT



Figure A.3: SR620 Time interval/frequency counter [36].

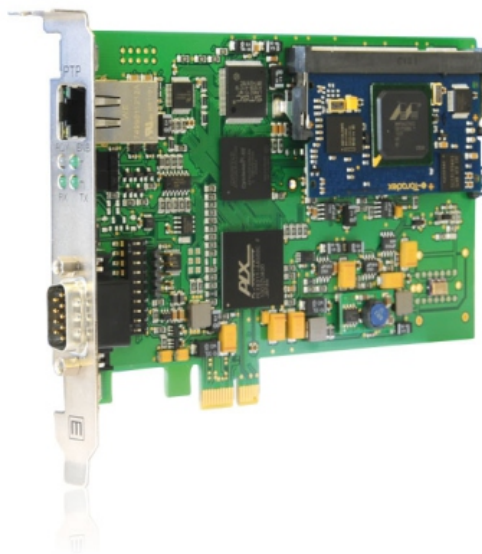


Figure A.4: Meinberg PTP207PEX PCI card [37].

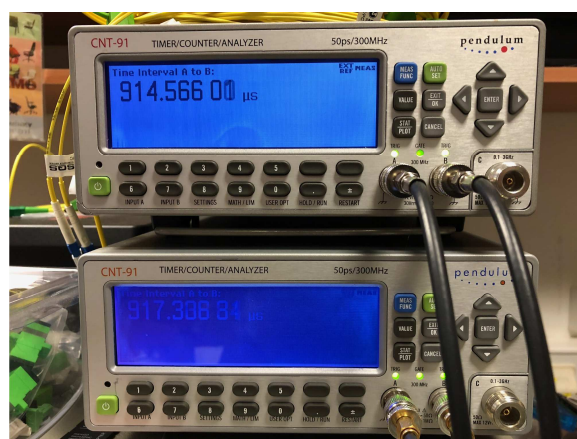


Figure A.5: Pendulum CNT-91 Time Interval Counters.

Working with Socket Options

One of [A.14] outcomes is the review of using the socket options regarding the hardware enabled timestamping.

Setting socket values:

```
int setsockopt (int sockfd, int level, int optname, const void * optval,
               socklen_t optlen)
```

Validating socket values:

```
int getsockopt (int sockfd, int level, int optname, void * optval,
               socklen_t * optlen)
```

SO_TIMESTAMP

Enables or disables generating a control message for each incoming packet in system time. This message is sent at the SOL_SOCKET level, and the `cmsg_data` field is the `(time)` structure of the last packet based on the system time passed to the user in this function call. To get `timeval` structure with timestamp, you need to use the `recvmsg ()` function.

Settings:

```
int enabled = 1;
int sock;
sock = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP);
setsockopt (sock, SOL_SOCKET, SO_TIMESTAMP, & enabled, sizeof (enabled))
```

Returns data in the form of:

```
struct timeval {
time_t tv_sec /* seconds */
suseconds_t tv_usec /* microseconds */
}
```

Obtaining data from the structure:

```
struct msghdr msg;
/ *
```

Preparing the `msghdr` structure data (`msg` variable)

B. WORKING WITH SOCKET OPTIONS

```
...
*/
res = recvmsg (sock, & msg, recvmsg_flags | MSG_DONTWAIT);
struct cmsghdr * cmsg;
for (cmsg = CMSG_FIRSTHDR (& msg); cmsg; cmsg = CMSG_NXTHDR (& msg, cmsg)
) {
struct timeval * stamp = (timeframe struct *) CMSG_DATA (cmsg);
/* Now you can work with the values of the timeval structure described
above.
* printf ("%ld.%06ld", (long) stamp-> tv_sec, (long) stamp-> tv_usec);
*/
}
```

SO_TIMESTAMPNS

The same mechanism as SO_TIMESTAMP, timestamp is written with higher precision (nano seconds, SO_TIMESTAMP - microseconds) using another structure (struct) timespec.

Settings:

```
int enabled = 1;
int sock;
sock = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP);
setsockopt (sock, SOL_SOCKET, SO_TIMESTAMPNS, & enabled, sizeof (enabled)
)
```

Returns data in the form of:

```
struct timespec {
time_t tv_sec / * seconds * /
long tv_nsec / * nanoseconds * /
}
```

Obtaining data from the structure:

```
struct msghdr msg;
/ *
Preparing the msghdr structure data (msg variable)
...
* /
res = recvmsg (sock, & msg, recvmsg_flags | MSG_DONTWAIT);
struct cmsghdr * cmsg;
for (cmsg = CMSG_FIRSTHDR (& msg); cmsg; cmsg = CMSG_NXTHDR (& msg, cmsg)
) {
struct timespec * stamp = (struct timespec *) CMSG_DATA (cmsg);
/ * You can now work with the values of the timespec structure described
above.
* printf ("%ld.%06ld", (long) stamp-> tv_sec, (long) stamp-> tv_nsec);
* /
}
```

SO_TIMESTAMPING

Enables or disables the timestamp generation when received, sent, or both. Supports multiple sources generating time stamps, including hardware. Supports the generation of time sockets for stream sockets. Supports multiple types of time stamp requests. As a result, this socket setting accepts a bit map of the signals (flags).

Defined constants for setting options:

Generating time stamps:

SOF_TIMESTAMPING_RX_HARDWARE: Request to generate timestamps for incoming datagrams with a network card.

SOF_TIMESTAMPING_RX_SOFTWARE: Request to generate time stamps for incoming datagrams at a time,

when the driver takes the data to the kernel input tray.

SOF_TIMESTAMPING_TX_HARDWARE: Request to generate timestamps for outgoing datagrams by a network card.

SOF_TIMESTAMPING_TX_SOFTWARE: Request to generate time stamps for outgoing datagrams at a time,

when the data leaves the system kernel and enters the network interface.

SOF_TIMESTAMPING_TX_SCHED: Request to generate time stamps for outgoing datagrams before entering the scheduler.

The difference between this timestamp and the tag obtained with SOF_TIMESTAMPING_TX_SOFTWARE will show a protocol-independent delay.

The delay resulting from protocol processing can be computed by the difference of the user time tag generated before calling the send () function and this.

In virtual machines where the packet wanders over multiple devices, the time tag is generated on each layer, allowing for accurate measurement of the delay in the queues.

SOF_TIMESTAMPING_TX_ACK: Time tag generation request if all sent packets are confirmed.

(It only makes sense for scrambled protocols)

Timestamps notification: (which timestamps will be reported in the generated control report)

SOF_TIMESTAMPING_SOFTWARE: Any SW timestamp will be announced if available.

SOF_TIMESTAMPING_SYS_HARDWARE: This option is not used and is ignored.

SOF_TIMESTAMPING_RAW_HARDWARE: The HW time tag generated by SOF_TIMESTAMPING_TX_HARDWARE will be announced if available.

Timestamp options:

SOF_TIMESTAMPING_OPT_ID: Generates a unique identifier with each packet. The process may have multiple time stamped requests. The packets can be re-arranged in the broadcasting way (eg in the packet scheduler). In this case, the time stamps are assigned to the error queue outside of the original send () call.

B. WORKING WITH SOCKET OPTIONS

This option associates each packet with a unique identifier when calling `recv` and returns it with the time stamp. The identifier is derived from the u32 counter (counter counter only) of the data type.

Counter is incremented with each sent packet. Its original value is 0 and is reset whenever this option is enabled after it has been disabled. The values already used in the packets remain. This option is only implemented for outbound packets.

`SOF_TIMESTAMPING_OPT_CMSG`: Recover `()` `cmsg` support for all time stamped (stamped) packets.

The control messages are supported on all designated incoming packets, and in the case of IPv6 and on the outbound packets. This option (option) extends support to IPv4 tagged outbound packets.

`SOF_TIMESTAMPING_OPT_TSONLY`: Applies only to outbound timestamps. This causes the kernel to return `cmsg` along with the empty packet instead of the original one. This reduces the amount of memory allocated to the socket inbound budget and delivers the time stamp despite the fact that `sysctl net.core.timestamp_allow_data` is 0.

This option prohibits `SOF_TIMESTAMPING_OPT_CMSG`.

Settings:

```
int so_timestamping_flags = 0;
/* Assign the value of the active flag to the variable by the OR
   operator
so_timestamping_flags = | SOF_TIMESTAMPING_KONSTANTA * /
int sock;
sock = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP);
setsockopt (sock, SOL_SOCKET, SO_TIMESTAMPING, & so_timestamping_flags,
           sizeof (so_timestamping_flags))
```

Returns data in the form of:

```
struct scm_timestamping {
struct timespec ts [3];
}
```

Most time stamps are passed in `ts [0]`.

Hardware time stamps are passed in `ts [2]`.

`ts [1]` is used for hardware time tags converted to system time, instead of displaying the network interface card (NIC) hardware clock directly as a HW PTP clock source. Conversion is done to allow the conversion of time in the user environment and optionally to synchronize system time in a user environment running a PTP stack such as `linuxptp`

Obtaining data from the structure:

```
struct msghdr msg;
/*
Preparing the msghdr structure data (msg variable)
...
*/
```

```

res = recvmsg (sock, & msg, recvmsg_flags | MSG_DONTWAIT);
struct cmsghdr * cmsg;
for (cmsg = CMSG_FIRSTHDR (& msg); cmsg; cmsg = CMSG_NXTHDR (& msg, cmsg)
) {
struct timespec * stamp = (struct timespec *) CMSG_DATA (cmsg);
/* Now we have a pointer to the first element of the returned structure
(timespec [0])
* printf ("SW stamp:%ld.%06ld", (long) stamp-> tv_sec, (long) stamp->
tv_nsec);
* /
stamp ++;
/* Now we have a pointer to the second element of the returned structure
(timespec [1]) * /
stamp ++;
/* Now we have a pointer to the third element of the returned structure
(timespec [2])
* printf ("HW-raw stamp:%ld.%06ld", (long) stamp-> tv_sec, (long) stamp
-> tv_nsec);
}

```

The data is sent with the socket in the additional msghdr data structure.
We obtain the necessary data using macros

CMSG_FIRSTHDR ():

Returns the pointer to the first data structure cmsghdr in the stack of
additional data associated with the forwarded msghdr structure.

CMSG_NXTHDR ():

Returns the pointer to another valid cmsghdr structure after the already
passed cmsghdr structure or NULL if the extra data stack is empty.

CMSG_DATA ():

Returns the pointer to the data portion of the cmsghdr structure, which
makes us most interested in the time stamps.

The data structure cmsghdr looks like this:

```

struct cmsghdr {
socklen_t cmsg_len; /* sum of data bytes, including header */
int cmsg_level; /* Outgoing protocol */
int cmsg_type; /* Protocol Specified Type */
/* followed by unsigned char cmsg_data [] */
}

```

The msshdr structure (msg variable) data structure contains:

```

char data [256];
struct iovec entry;
struct sockaddr_in from_addr;
struct {
struct cmsghdr cm;
char control [512];
} control;
int res;

```

B. WORKING WITH SOCKET OPTIONS

```
memset (& msg, 0, sizeof (msg));
msg.msg_iov = & entry;
msg.msg_iovlen = 1;
entry.iov_base = data;
entry.iov_len = sizeof (data);
msg.msg_name = (caddr_t) & from_addr;
msg.msg_namelen = sizeof (from_addr);
msg.msg_control = & control;
msg.msg_controllen = sizeof (control);
```


C. PROCESSOR EXTERNAL FREQUENCY SOURCE

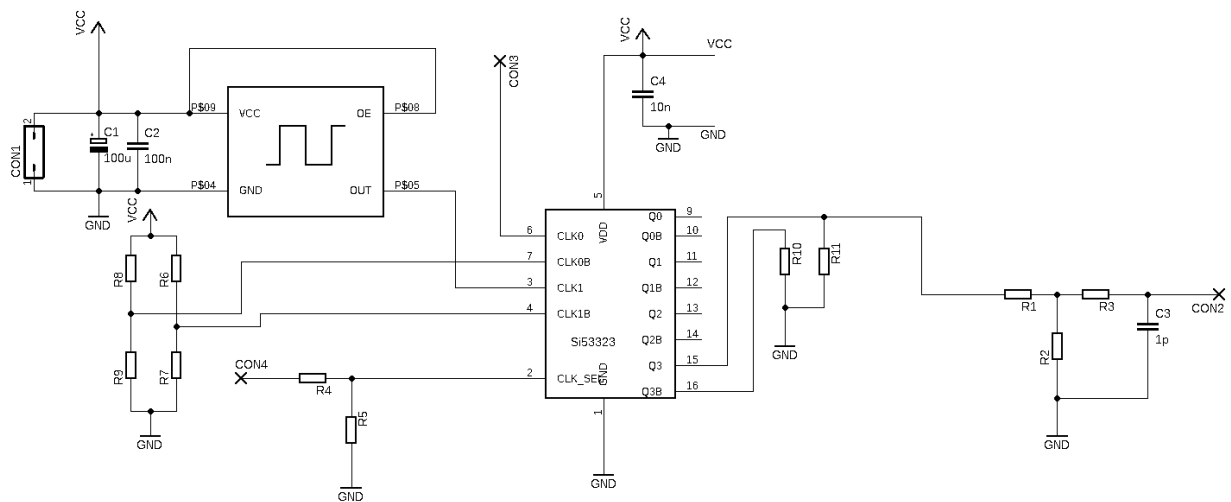


Figure C.2: External processor clock source with multiplexer based on Silicon Labs IC.

IEEE 1588 PTPmon Monitoring Tool

We have developed a PTP monitoring tool [A.16] that can be used online (live network capture) or offline (file with captured traffic). Tool builds a database of IEEE 1588 enabled nodes and show the list of devices with PTP related parameters.

```
https://gitlab.fit.cvut.cz/hanouvi1/PTP_Monitoring/tree/master
```

Install Qt framework v5.5 +. Installers for Win, Mac, Ubuntu OS are on the site.

There is no need for a root when installing. Ubuntu needed to install a few libraries:

```
https://wiki.qt.io/Install_Qt_5_on_Ubuntu
sudo apt-get install mesa-common-dev
sudo apt-get install libglu1-mesa-dev -y
```

To generate the Makefile, use qmake from the installation, for example:

```
user @ linux ~ $ locate bin / qmake
/home/user/Qt/5.5/gcc_64/bin/qmake
/ usr / bin / qmake
/ usr / bin / qmake-qt4
/ usr / lib / x86_64-linux-gnu / qt4 / bin / qmake
/ usr / share / qt4 / bin / qmake
```

```
Used: /home/user/Qt/5.5/gcc_64/bin/qmake
```

To generate a binary should be enough: make -j

```
./PTP_Monitoring -f
The default "./config.ini" path is used
The default parameter values are defined by the ConstParams class in
const_params.h
```

Content Example:

```
[default]
```

D. IEEE 1588 PTPMON MONITORING TOOL

```
; log with lower priority will be ignored.
min_log_level = info
; log_file_path
; filter =

; should live sniff or file sniff.
live_sniffing = false

[offline]
; absolute path to the pcap file.
path_to_pcap = / home / vitek / PTPD / PCAP / trace.pcap

[live]
; sniffing adapter, permissions required.
device = eth0
```

Pipelined Priority Encoder Source Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity full_ones_cnt is
    port ( bits_in : in  STD_LOGIC_VECTOR (255 downto 0);
          ones_cnt_out : out  STD_LOGIC_VECTOR (8 downto 0);
          enable : in std_logic;
          clk : in  STD_LOGIC;
          rst : in  STD_LOGIC
        );
end full_ones_cnt;

architecture rtl of full_ones_cnt is

    SIGNAL reg_temp_in, reg_in, reg_1, reg_1_in : std_logic_vector
        (255 DOWNTO 0);
    SIGNAL reg_2, reg_2_in : std_logic_vector(191 DOWNTO 0);
    SIGNAL reg_3, reg_3_in : std_logic_vector(127 DOWNTO 0);
    SIGNAL reg_4, reg_4_in : std_logic_vector(79 DOWNTO 0);
    SIGNAL reg_5, reg_5_in : std_logic_vector(47 DOWNTO 0);
    SIGNAL reg_6, reg_6_in : std_logic_vector(27 DOWNTO 0);
    SIGNAL reg_7, reg_7_in : std_logic_vector(15 DOWNTO 0);
    SIGNAL reg_8, reg_8_in : std_logic_vector(8 DOWNTO 0);

begin

    stage_1 : for I in 0 to 127 generate
        reg_1_in(2*(I+1)-1 DOWNTO 2*I) <= unsigned('0' &
            reg_in(2*I+1 downto 2*I+1)) + unsigned('0' &
            reg_in(2*I downto 2*I));
    end generate stage_1;
```

E. PIPELINED PRIORITY ENCODER SOURCE CODE

```
stage_2 : for I in 0 to 63 generate
    reg_2_in(3*(I+1)-1 DOWNT0 3*I) <= unsigned('0' &
        reg_1(4*(I+1)-1 DOWNT0 4*I+2)) + unsigned('0'
        & reg_1(4*(I+1)-1-2 DOWNT0 4*I));
end generate stage_2;

stage_3 : for I in 0 to 31 generate
    reg_3_in(4*(I+1)-1 DOWNT0 4*I) <= unsigned('0' &
        reg_2(6*(I+1)-1 DOWNT0 6*I+3)) + unsigned
        ('0' & reg_2(6*(I+1)-1-3 DOWNT0 6*I));
end generate stage_3;

stage_4 : for I in 0 to 15 generate
    reg_4_in(5*(I+1)-1 DOWNT0 5*I) <= unsigned('0' &
        reg_3(8*(I+1)-1 DOWNT0 8*I+4)) + unsigned
        ('0' & reg_3(8*(I+1)-1-4 DOWNT0 8*I));
end generate stage_4;

stage_5 : for I in 0 to 7 generate
    reg_5_in(6*(I+1)-1 DOWNT0 6*I) <= unsigned('0' &
        reg_4(10*(I+1)-1 DOWNT0 10*I+5)) + unsigned
        ('0' & reg_4(10*(I+1)-1-5 DOWNT0 10*I));
end generate stage_5;

stage_6 : for I in 0 to 3 generate
    reg_6_in(7*(I+1)-1 DOWNT0 7*I) <= unsigned('0' &
        reg_5(12*(I+1)-1 DOWNT0 12*I+6)) + unsigned
        ('0' & reg_5(12*(I+1)-1-6 DOWNT0 12*I));
end generate stage_6;

stage_7 : for I in 0 to 1 generate
    reg_7_in(8*(I+1)-1 DOWNT0 8*I) <= unsigned('0' &
        reg_6(14*(I+1)-1 DOWNT0 14*I+7)) + unsigned
        ('0' & reg_6(14*(I+1)-1-7 DOWNT0 14*I));
end generate stage_7;

reg_8_in <= unsigned('0' & reg_7(15 DOWNT0 8)) + unsigned('0' &
    reg_7(7 DOWNT0 0));
reg_8_in <= unsigned('0' & reg_7(15 DOWNT0 8)) + unsigned('0' &
    reg_7(7 DOWNT0 0));

reg_proc : PROCESS (clk)
BEGIN
    IF rising_edge(clk) THEN
        IF rst = '1' THEN
            reg_in <= (OTHERS => '0');
            reg_1 <= (OTHERS => '0');
            reg_2 <= (OTHERS => '0');
            reg_3 <= (OTHERS => '0');
            reg_4 <= (OTHERS => '0');
            reg_5 <= (OTHERS => '0');
```

```

reg_6 <= (OTHERS => '0');
reg_7 <= (OTHERS => '0');
reg_8 <= (OTHERS => '0');
ELSE
    -- when asserted enable, start
    computation
IF enable = '1' THEN
    reg_in <= reg_temp_in;
ELSE
    reg_in <= reg_in;
END IF;

reg_1 <= reg_1_in;
reg_2 <= reg_2_in;
reg_3 <= reg_3_in;
reg_4 <= reg_4_in;
reg_5 <= reg_5_in;
reg_6 <= reg_6_in;
reg_7 <= reg_7_in;
reg_8 <= reg_8_in;
END IF;
END IF;
END PROCESS reg_proc;

reg_temp_in <= bits_in;

ones_cnt_out <= reg_8;

end rtl;

```